4 SUBMANIFOLDS AND SIMPLICIAL COMPLEXES

Having nearly exhausted the theory of one-dimensional geometry, we are ready to step up to higher-dimensional objects. This is where we can find most (but not all) of the interesting action: In computer vision, graphics, and machine learning most of our effort is not dedicated to characterizing the geometry of shoelaces but rather to higher-dimensional objects like surfaces, volumes, and manifolds. That said, many of the themes we explored in the one-dimensional case will persist. We will need extremely careful constructions to apply the tools of calculus to shapes embedded in $\mathbb{R}^n$ rather than functions, and a careful transition from smooth to discrete ensures that the algorithms we use to process shape capture the quantities about which we are reasoning.

In this chapter, we take inspiration from the discussion in §3.1 and focus on defining what geometry *is* in dimensions larger than one. Lifting the idea of a curve to higher dimensions, we will study *submanifolds* of $\mathbb{R}^n$, which encapsulate curves, surfaces, volumes, and higher-dimensional pieces of geometry that sit in Euclidean space. Submanifolds of $\mathbb{R}^n$ comprise the lion's share of geometric objects encountered in computational applications, and thanks to the famous Nash Embedding Theorem even certain abstract geometric objects can be embedded as submanifolds of $\mathbb{R}^n$. After defining smooth submanifolds, we will discuss discrete representations of manifolds, focusing on simplicial complexes. While we will start with general definitions, we will focus on the surface case, or two-dimensional submanifolds of $\mathbb{R}^3$, leading to a data structure that will take a key role in future chapters: the *triangle mesh*.

4.1 SUBMANIFOLDS OF $\mathbb{R}^n$

The discipline of differential geometry seeks to isolate the key properties of *smooth* shapes. The term "shape" here is quite broad and encapsulates many special cases:

- A curve embedded in $\mathbb{R}^2$, as considered in the previous chapter.
- The outer surface of an object in $\mathbb{R}^3$, e.g. the surface of a body of water or the skin of a human.
- A volume of mass in $\mathbb{R}^3$, e.g. the white matter of a subject's brain.
- The $(n-1)$-dimensional unit sphere in $\mathbb{R}^n$.

Each of these examples of geometric objects has properties in common as well as distinguishing characteristics. Each is smooth in the sense that there are no isolated kinks or singularities. They also have dimensions in which they are *embedded*, e.g. $\mathbb{R}^2$ for the curve and $\mathbb{R}^3$ for the surface or volume. At the same time, each example has a different *intrinsic* dimension: The curve is one-dimensional if we zoom close enough, the surface is locally planar (two-dimensional), the volume is three-dimensional, and so on. Some of these examples might also have *boundaries*, for example the endpoints of a curve or the outer surface of a volume.

All of these examples are encapsulated by the definition of a *submanifold* in $\mathbb{R}^n$. Define the m-dimensional half space as:

$$\mathcal{H}_m := \{(x^1, \dots, x^m) \in \mathbb{R}^m : x^m \geq 0\}. \tag{4.1}$$

Then, we can define a submanifold as follows:

Definition 4.1 (Submanifold of $\mathbb{R}^n$, with and without boundary)**.** *A set $\mathcal{M} \subseteq \mathbb{R}^n$ is an m-dimensional submanifold of $\mathbb{R}^n$ if for each $\mathbf{p} \in \mathcal{M}$ there exist open sets $U \subseteq \mathbb{R}^m, W \subseteq \mathbb{R}^n$ and a function $g : U \cap \mathcal{H}_m \to \mathcal{M} \cap W$ such that $\mathbf{p} \in W$ and g is a one-to-one and smooth map whose Jacobian is rank-m and admitting a continuous inverse $g^{-1} : W \cap M \to U$.*

JS: define Jacobian somewhere We call g a *local parameterization* or *coordinate chart* for a neighborhood around $\mathbf{p}$. A point $\mathbf{p}$ is in the *boundary* $\partial\mathcal{M}$ of $\mathcal{M}$ if in any local parameterization $\mathbf{p} = g(\mathbf{u})$ for some $\mathbf{u} \in U$ with at least one coordinate equal to zero. If $\partial\mathcal{M} = \varnothing$, we call M a *manifold without boundary* and can replace $\mathcal{H}_m$ in the definition above with $\mathbb{R}^m$. We owe our phrasing of this definition to [17]; it is illustrated in Figure REF.

In our illustrations, we will focus on a few key examples. A *curve* is a one-dimensional ($m = 1$) submanifold of $\mathbb{R}^2$ or $\mathbb{R}^3$, a *surface* is a two-dimensional ($m = 2$) submanifold of $\mathbb{R}^3$, and a *volume* is a three-dimensional ($m = 3$) submanifold of $\mathbb{R}^3$. For the curve case, it is important to double-check that our definition agrees with Definition 3.1.

Example 4.1 (Pathological cases)**.** *Similar to curves, there exist many functions $f : \mathbb{R}^m \to \mathbb{R}^n$ that do* not *parameterize m-dimensional manifolds. For instance, some examples illustrated in Figure REF are listed below:*

- *Point:* $f(u, v) \equiv (0, 0, 0)$
- *Curve in disguise:* $f(u, v) = (u, u^2, \cos u)$
- *Cusp:* $f(u, v) = (u, v^3, v^2)$

In each of these cases, the Jacobian Df is not full-rank at $(u, v) = (0, 0)$, contradicting Definition 4.1.

For those who have not seen it before, our terminology admits a curious choice of prefixes; we have defined a *sub*manifold but not a manifold. This is because technically submanifolds of $\mathbb{R}^n$ are embedded in another geometric space: Euclidean space $\mathbb{R}^n$. The more general definition of a *manifold* is similar to Definition 4.1 but with the embedding criterion removed:

Definition 4.2 (Manifold)**.** *An m-dimensional (topological)* manifold $\mathcal{M}$ *is a Hausdorff space for which each $\mathbf{p} \in \mathcal{M}$ admits open sets $U \subseteq \mathbb{R}^m, W \subseteq \mathcal{M}$ and a homeomorphism (continuous map with continuous inverse) $g : U \to W$.*

Given our informal discussion it is reasonable to ignore the details of being a Hausdorff space; the key aspect of the definition is that every point $\mathbf{p} \in \mathcal{M}$ has a neighborhood that "looks like" $\mathbb{R}^m$. For simplicity we have defined manifolds without boundary; the extension to manifolds with boundary is similar to Definition 4.1. The basic difference is that a manifold $\mathcal{M}$ is allowed to be an abstract space rather than explicitly being embedded in $\mathbb{R}^n$.

Example 4.2 (Manifold that is not a submanifold of $\mathbb{R}^n$)**.** *JS: Write me.*

Why work with submanifolds? In a sense, a manifold is a *topological* object; Definition 4.2 determines how points are connected together locally but does not give a mechanism for measuring geometric quantities like distance. On the other hand, a submanifold of $\mathbb{R}^n$ inherits geometry from the structure of the space $\mathbb{R}^n$ in which it is sitting; for example, we can measure the length of a curve along a submanifold $\mathcal{M} \subseteq \mathbb{R}^n$ using the arc length formula (3.2) in the previous chapter.

In future chapters, we will discuss alternative ways to put geometric structure onto a manifold. A key example is a *Riemannian manifold*, which is a manifold equipped with a dot product function at each point. For example, as illustrated in Figure REF, a map of the world unrolled onto the plane is a manifold, but if we think of the map as a piece of the plane then we need a different dot product between vectors at each base point to account for the curvature of the earth. One surprising fact is the Nash Embedding Theorem (dating to the 1950s), which states that every m-dimensional Riemannian manifold can be embedded isometrically—that is, while preserving distances—into $2m$-dimensional Euclidean space $\mathbb{R}^{2m}$.

4.2 TANGENTS AND NORMALS

In §2.1, we put some serious effort into defining notions from linear algebra, only to in this chapter define a key object of study—an m-dimensional submanifold $\mathcal{M}$ in $\mathbb{R}^n$—that is potentially

nonlinear. Thankfully we can address this gap by recalling that the *local* structure of a manifold resembles $\mathbb{R}^m$. This reasoning leads us to the definition of the *tangent space* $T_{\mathbf{p}}\mathcal{M}$ at a point $\mathbf{p} \in \mathcal{M}$. The basic idea of a tangent space is fairly intuitive and illustrated in Figure REF; at each $\mathbf{p} \in \mathcal{M}$ we attach a copy of $\mathbb{R}^m$ that just barely touches $\mathcal{M}$ at $\mathbf{p}$.

There are many possible ways to formalize this notion mathematically; one possibility is to use machinery developed in the previous chapter:

Definition 4.3 (Tangent space)**.** *The* tangent space $T_{\mathbf{p}}\mathcal{M}$ *of an m-dimensional submanifold $\mathcal{M} \subseteq \mathbb{R}^n$ is the set of velocity vectors $\gamma'(0)$, where $\gamma : (-\varepsilon, \varepsilon) \to \mathcal{M}$ with $\gamma(0) = \mathbf{p}$.*

Figure REF illustrates this definition. A car driving along $\mathcal{M}$—whose position is encoded in $\gamma(t)$ at time t—cannot experience all possible velocities when driving through $\mathbf{p} \in \mathcal{M}$; in particular, the car never experiences the impulse to jump *off* the manifold. $T_{\mathbf{p}}\mathcal{M}$ codifies the set of velocities the car can experience.

Our first task is to double-check that $T_{\mathbf{p}}\mathcal{M}$ actually inherits linear structure:

Proposition 4.1. *For an m-dimensional submanifold $\mathcal{M} \subseteq \mathbb{R}^n$, $T_{\mathbf{p}}\mathcal{M}$ is an m-dimensional linear subspace of $\mathbb{R}^n$ for all $\mathbf{p} \in \mathcal{M} \backslash \partial M$.*

Proof. **JS: Write me. Basically the image of Dg for any local parameterization $g : \mathbb{R}^m \to \mathcal{M}$.** □

This proposition verifies our intuition that we can add and scale tangent vectors while keeping them in $T_{\mathbf{p}}\mathcal{M}$.

The tangent space at $\mathbf{p} \in \mathcal{M}$ provides all the directions from $\mathbf{p}$ along with we can stay on $\mathcal{M}$ to first order. The orthogonal complement of the tangent space gives us all the ways we can "jump away" from $\mathcal{M}$:

Definition 4.4 (Normal space)**.** *The* normal space $N_{\mathbf{p}}\mathcal{M}$ *of an m-dimensional submanifold $\mathcal{M} \subseteq \mathbb{R}^n$ is the set of vectors $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{v} \cdot \mathbf{w} = 0$ for all $\mathbf{w} \in T_{\mathbf{p}}\mathcal{M}$.*

Figure REF illustrates the normal space for a curve and a surface.

By definition, the dimension of the normal space is $n - m$, that is, the difference between the embedding dimension and the intrinsic dimension of $\mathcal{M}$; this difference is known as the *codimension* of $\mathcal{M}$. A key case is in codimension-1 submanifolds of $\mathbb{R}^n$, e.g. curves on the plane and surfaces in $\mathbb{R}^3$. In this case, $n - m = 1$, so for each $\mathbf{p} \in \mathcal{M} \backslash \partial M$ we have $\dim N_{\mathbf{p}}\mathcal{M} = 1$; in other words, every point $\mathbf{p} \in \mathcal{M}$ admits only one normal direction up to scaling. That is, for every $\mathbf{p} \in \mathcal{M}$ with codimension 1, we know $N_{\mathbf{p}}\mathcal{M} = \{c\mathbf{n} : c \in \mathbb{R}\}$ for some vector $\mathbf{n} \in \mathbb{R}^n$ with $\|\mathbf{n}\|_2 = 1$; we call the vector $\mathbf{n}$ a *normal* to $\mathcal{M}$ at $\mathbf{p}$.

The choice of a normal vector $\mathbf{n}_{\mathbf{p}}$ for each $\mathbf{p}$ is nonunique, even when the codimension is one. In particular, if $\mathbf{n}$ is a normal vector, so is $-\mathbf{n}$. We might try to construct a *continuous* function that assigns a normal vector $\mathbf{n}$ to each point in the interior of a submanifold, as illustrated in Figure REF. The existence of such a function, known as a *Gauss map*, is not guaranteed for all submanfiolds; the famous Möbius strip illustrated in Figure REF is a counterexample. We call a submanifold *orientable* if it admits a continuous Gauss map:

Definition 4.5 (Orientation)**.** *A submanifold $\mathcal{M} \subseteq \mathbb{R}^n$ is* orientable *if there exists a continuous function $\mathbf{n}(\mathbf{p}) : \mathcal{M} \backslash \partial M \to \mathcal{S}^{n-1}$ with $\mathbf{n}(\mathbf{p}) \in N_{\mathbf{p}}\mathcal{M}$ for all $\mathbf{p} \in \mathcal{M} \backslash \partial \mathcal{M}$.*

Here, we use $\mathcal{S}^{n-1}$ to denote the *unit sphere*, or set of all unit vectors:

$$\mathcal{S}^{n-1} := \{\mathbf{v} \in \mathbb{R}^n : \|\mathbf{v}\|_2 = 1\}.$$

A function $f : \mathcal{M} \to \mathbb{R}^k$ is considered continuous if the composition $f \circ g$ is continuous for any local parameterization g, as defined in Definition 4.1.

4.3 SIMPLICIES AND SIMPLICIAL COMPLEXES

Our careful definitions of submanifolds and manifolds in some sense give us a reasonable definition of what it means to be a "shape" in $\mathbb{R}^n$. With some theoretical language under our belt, we now transition to structures we could use to represent manifold-like objects in a computational system. Our goal is to extend the definition of a poly-line from §3.7 to higher dimensions; we briefly mention some other shape representations in §4.9.

Discussion in this section is largely *topological* in nature. Here, topology refers to properties like connectivity, which are invariant to geometric transformations. Structures like triangle meshes are convenient because they separate topology and geometry: The geometry is expressed in vertex positions, while the topology is reflected in the way vertices are linked together into triangles.

From a few feet away, the high-level idea of using poly-lines to discretize curves is to build curves from flat, one-dimensional facets. This approach extends elegantly to higher dimensions by building shapes out of *simplices*, which include line segments (1-simplices), triangles (2-simplices), and tetrahedra (3-simplices); see Figure REF for illustration. To define a simplex carefully, we need to make sure it does not collapse; for instance, a line segment whose beginning and ending vertices are the same degenerates to a point. To avoid this condition, we extend the definition of linear independence given in Definition 2.2 from vectors to points:

Definition 4.6 (Affinely independent)**.** *A set of points $\mathbf{p}_0, \ldots, \mathbf{p}_k \in \mathbb{R}^n$ is* affinely independent *if the set of vectors $\{\mathbf{p}_1 - \mathbf{p}_0, \ldots, \mathbf{p}_k - \mathbf{p}_0\}$ is linearly independent.*

Figure REF illustrates this definition; if we think of $\{\mathbf{p}_0, \ldots, \mathbf{p}_k\}$ as defining the vertices of a polyhedron, then affine independence guarantees that vectors along the sides of the polyhedron do not coincide. With this definition in place, we define a simplex as follows:

Definition 4.7 (Simplex)**.** *A k-*simplex *is a set $S \subseteq \mathbb{R}^n$ that can be written*

$$S = \left\{ \sum_{i=0}^{k} a^i \mathbf{p}_i : a_i \geq 0 \; \forall i \text{ and } \sum_{i=0}^{k} a^i = 1 \right\}$$

for an affinely independent collection of points $\{\mathbf{p}_i\}_{i=0}^k \subseteq \mathbb{R}^n$. The $\mathbf{p}_i$'s are the vertices *of S, and k is the* dimension *of S.*

Note a common off-by-one math mistake: A k-simplex has $k+1$ (not k!) vertices. A more succinct way to put Definition 4.7 is that a k-simplex is the convex hull of $k+1$ affinely independent points.

The boundary of a simplex is itself composed of (lower-dimensional) simplices, known as its *faces*:

Definition 4.8 (Faces)**.** *The* faces *of a simplex with vertices $\mathbf{p}_0, \ldots, \mathbf{p}_k$ are the ℓ-simplices ($\ell \leq k$) whose vertices are subsets of $\{\mathbf{p}_0, \ldots, \mathbf{p}_k\}$. By convention, we include the empty set as a face of any simplex ($\ell = 0$).*

For example, the faces of a tetrahedron include its vertices, edges, and boundary triangles.

Of course, we are unlikely to be able to approximate an arbitrary shape with a single simplex. Hence, we can combine them together into a structure known as a *simplicial complex*:

Definition 4.9 (Simplicial complex)**.** *A* simplicial complex *$\mathcal{K}$ is a set of simplices in $\mathbb{R}^n$ such that every face of simplex in $\mathcal{K}$ is also in $\mathcal{K}$, and the intersection $S \cap S'$ of any two simplices $S, S' \in \mathcal{K}$ is a face of both S and S'.*

Our sneaky inclusion of the empty set as a face of a simplex greatly simplifies the wording of the second half of this definition.

Example 4.3 (Simplicial complexes)**.** *Valid simplicial complexes include the ones shown in Figure REF; note they include not only manifold-like shapes but also ones that combine different dimensionalities, like a line segment glued onto the side of a triangle. Some invalid simplicial complexes in Figure REF include simplices that intersect along points other than their faces, or simplices that do not include their boundaries.*

There are many reasons to include simplicial complexes and their manifold relatives (defined in the next section) in the context of computation. These include typical structures encountered in the computer graphics and vision literatures, including graphs, triangle meshes, and tetrahedral meshes. Since they are composed of many flat facets, they are typically straightforward to render, and algorithms like subdivision can be used to smooth or refine these meshes when more detail is needed. From a geometric perspective, adjacency along a simplicial complex provides a reasonable notion of a local neighborhood around a point, which is a key piece of defining differential quantities that measure bending, stretching, and other phenomena.

4.4 MANIFOLD SIMPLICIAL COMPLEXES

The condition that an object can be represented as a simplicial complex is a much weaker condition than being a submanifold of $\mathbb{R}^n$, even if we relax smoothness/differentiability. We can bring these two worlds closer to one another by defining what it means for a simplicial complex to be *manifold*, a set of conditions intended to mimic Definition 4.1; here we follow the definitions provided in [7]. As a warning, the constructions in this section are somewhat technical and for most of the applications we will consider are either trivial or can be ignored; they are included here for completeness.

We call a simplicial complex $\mathcal{K}$ a *pure k-complex* if every simplex of $\mathcal{K}$ of dimension $< k$ is a face of a simplex of dimension k; this condition prevents "hanging" artifacts like the spurious edge in Figure REF, but it does not prevent our a complex from resembling the bowtie shape in Figure REF.

For any $h \leq k$, we consider two simplices $S, S' \in \mathcal{K}$ *h-adjacent* if the intersection $S \cap S'$ is a simplex of dimension h in $\mathcal{K}$; $\mathcal{K}$ is k-connected if every pair of k-simplices is connected by a path of $(k-1)$-adjacent simplices. This leads us to the definition:

Definition 4.10 (Pseudo-manifold complex)**.** *A pure simplicial k-complex $\mathcal{K}$ is* pseudo-manifold *if it is k-connected and each $(k-1)$-simplex has exactly one or two adjacent k-simplices in $\mathcal{K}$.*

Here we discount complexes composed of multiple disconnected pieces; adjusting our definition to include this case is straightforward.

There is one final boundary case we must account for, illustrated in Figure REF. **JS: Add sentence with description of what goes wrong. Ed's Clifford torus example? David's cone over an annulus?** To fix this case, we need even more definitions, illustrated in Figure REF. The *co-faces* of a simplex $S \in \mathcal{K}$ are the simplices in $\mathcal{K}$ that have S as a face, and the *star* of S is its set of co-faces. Finally, the *link* of S is the set of faces in the star of S that are not incident to S. With these definitions in place, we define:

Definition 4.11 (Combinatorial manifold)**.** *A pseudo-manifold complex is a* combinatorial manifold *if the link of every vertex (0-simplex) is homeomorphic to a $(k-1)$-sphere $\mathcal{S}^{k-1}$ or to a k-ball $\{\mathbf{x} \in \mathbb{R}^k : \|\mathbf{x}\|_2 \leq 1\}$.*

Here, a homeomorphism is a continuous function with a continuous inverse; two collections of simplices in $\mathbb{R}^n$ are homeomorphic if there exists a homeomorphism between their unions in $\mathbb{R}^n$. Figure REF illustrates this definition for a tetrahedral mesh; removing the link of a vertex either takes a crater out of the surface or hollows out a sphere in the interior. **JS: Every 2d pseudomanifold is a combinatorial manifold?**

4.5 ORIENTATION

Our final technical task in this chapter is to extend the notion of *orientation* introduced in Definition 4.5 to simplicial complexes. In doing so, we will also extend it to submanifolds whose codimension is not equal to 1. For example, as illustrated in Figure REF, a loop of string in $\mathbb{R}^3$ somehow can be assigned an orientation by deciding on a canonical direction in which cars can drive along the loop without running into one another; there are two such orientations. The ba-

sic issue is that the concept of a continuous normal vector does not extend easily to simplicial complexes; these complexes have sharp facets between which the normal experiences a discrete jump. Hence, we must come up with an alternative notion of orientability that better extends to this case.

Our approach uses the concept of *handedness* often encountered in physics and other disciplines. As illustrated in Figure REF, suppose we place the base of our right hand at a point $\mathbf{p} \in \mathcal{M}$, where $\mathcal{M}$ is a smooth surface embedded in $\mathbb{R}^3$. Then, looking down at the remaining fingers shows that they curl in a *counterclockwise* direction. If we drag our hand along the surface while keeping the thumb aligned to the normal to an oriented surface, this direction of circulation does not change; along a Möbius strip, however, counterclockwise becomes clockwise if we go around the length of the strip. This rough illustration provides an alternative equivalent notion of orientation for a surface: An assignment of a "counterclockwise" direction to each tangent plane to a surface.

While formalizing this notion in smooth terms is somewhat difficult, it turns out to be relatively straightforward to do so for simplicial complexes. Consider orienting a triangle with vertices $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$, shown in Figure REF. There are three orderings of the vertices ($[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]$, $[\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_1]$, and $[\mathbf{p}_3, \mathbf{p}_1, \mathbf{p}_2]$) that are in counterclockwise order, and three orderings of the vertices ($[\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_2]$, $[\mathbf{p}_2, \mathbf{p}_1, \mathbf{p}_3]$, and $[\mathbf{p}_3, \mathbf{p}_2, \mathbf{p}_1]$) that are in clockwise order. Note that swapping any two vertices in a clockwise ordering creates an ordering that is counterclockwise, and vice versa. This induces an equivalence relation on tuples $[\mathbf{p}_0, \ldots, \mathbf{p}_k]$: We notate $[\mathbf{p}_0, \ldots, \mathbf{p}_k] = [\bar{\mathbf{p}}_0, \ldots, \bar{\mathbf{p}}_k]$ if and only if we can obtain the second tuple by swapping pairs of elements of the first tuple an even number of times. Otherwise, when the sets of points are the same but the orderings are not even numbers of swaps apart, we notate $[\mathbf{p}_0, \ldots, \mathbf{p}_k] = -[\bar{\mathbf{p}}_0, \ldots, \bar{\mathbf{p}}_k]$. This signed notation is suggestive of algebraic properties explored in exercise 4.1..

With this relation in place, we define

Definition 4.12 (Oriented simplex)**.** *An* oriented k-simplex *is an ordered set* $[\mathbf{p}_0, \ldots, \mathbf{p}_k]$ *of affinely independent points* $\mathbf{p}_0, \ldots, \mathbf{p}_k$*, up to the equivalence relation* $=$ *developed above.*

An orientation assigned to a k-simplex *induces* an orientation for its $(k-1)$-faces, by taking ordered subsets of its vertices where one is removed; by convention, we assign opposite sign when we remove odd-indexed vertices. For instance, the oriented line segment faces of an oriented triangle $[\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2]$ are

$$\begin{aligned} &[\mathbf{p}_1, \mathbf{p}_2], && \text{Removed } \mathbf{p}_0 \\ -&[\mathbf{p}_0, \mathbf{p}_2] = [\mathbf{p}_2, \mathbf{p}_0], \text{ and} && \text{Removed } \mathbf{p}_1 \\ &[\mathbf{p}_0, \mathbf{p}_1]. && \text{Removed } \mathbf{p}_2 \end{aligned}$$

Notice our sign convention correctly orients the edges of the triangle in Figure REF. Similarly, the oriented triangular faces of an oriented tetrahedron $[\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]$ are

$$\begin{aligned} &[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3], && \text{Removed } \mathbf{p}_0 \\ -&[\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_3] = [\mathbf{p}_0, \mathbf{p}_3, \mathbf{p}_2], && \text{Removed } \mathbf{p}_1 \\ &[\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_3], \text{ and} && \text{Removed } \mathbf{p}_2 \\ -&[\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2] = [\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_1]. && \text{Removed } \mathbf{p}_3 \end{aligned}$$

Now, consider the two adjacent triangles $[\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_1]$ and $[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]$ shown in Figure REF. The first triangle orients the shared edge as $[\mathbf{p}_2, \mathbf{p}_1] = -[\mathbf{p}_1, \mathbf{p2}]$ and the second triangle orients the shared edge as $+[\mathbf{p}_1, \mathbf{p}_2]$: The shared edge inherits opposite orientations!

This observation (finally) leads to our definition of orientability for a combinatorial manifold:

Definition 4.13 (Orientable combinatorial manifold)**.** *A combinatorial k-manifold $\mathcal{K}$ is* orientable *if there exists an assignment of orientations to its k-simplices so that simplices sharing a $(k-1)$-face induce opposite orientations on that face.*

This definition is a mouthful but for triangle meshes aligns with our original intuition above. The definition asserts that we order the three vertices of each triangle, effectively defining a "clockwise" direction per triangle that is consistent for every adjacent pair.

4.6 THE SMOOTH–DISCRETE INTERFACE

Over the next several chapters, we will develop discrete versions and approximations of calculus that operate on meshes represented as combinatorial manifolds. These techniques will comprise powerful and intuitive ways to apply differential geometry to faceted structures, which classically would not be relevant to this smooth theory. At several junctures, we will suggest *discrete* rather than *discretized* approaches, which define analogs of differential quantities in terms of lengths and angles measurable on the edges and facets of combinatorial structures. These measurements are well-defined for coarse and fine meshes alike, which in some sense is part of their appeal: They apply to the regime of sampled data without relying on a density assumption, and in some cases they even preserve structures from smooth theory exactly.

At the same time, practitioners in this discipline must develop some intuition for the interface between smooth and discrete. Even if our definition of discrete curvature, for example, can be evaluated on the vertices of any manifold triangle mesh, if the mesh is a poor approximation of an underlying smooth surface it easily could be the case that our curvature measurement is no longer meaningful or intuitive. Years of intuition and experience—combined with theories from finite elements (FEM) and other disciplines—have led to science (and some mythology) regarding the difference between "good" and "bad" meshes. For instance, on triangle meshes many tools from FEM work best when the interior angles of triangles are close to 60°; Taylor series based approximations can determine e.g. the edge lengths needed to capture geometric features of different sizes.

While we will point out the role of discretization quality in the context of algorithms presented in future chapters, this section largely serves as a vague warning that manifold simplicial complexes impose much weaker structure on a geometric object than requiring it to be a smooth submanifold of $\mathbb{R}^n$.

Example 4.4 (Meshing a square)**.** *Consider the two meshes of a square shown in Figure* REF*. One is fairly dense and isotropic, while the other has a variety of different-sized elements, interior angles, and edge lengths. In some sense, both meshes capture the same geometric object: A square. But consider, for example, algorithmic pipelines that approximate functions* $f : [0,1] \times [0,1] \to \mathbb{R}$ *with one value per vertex. The class of functions reasonably captured by the evenly-meshed square has some limitations, e.g. it cannot capture features smaller than the size of a single triangle, but the function space is somehow a smoothed-out version of functions on* $[0,1] \times [0,1]$*. Contrastingly, functions on the more contorted mesh can be dense in some places and high-frequency in others, and interpolation along long, slivery triangles can be inaccurate.*

4.7 TRIANGULATED SURFACES

Of particular interest in many applications is the *triangle mesh*, or, in the language developed above, manifold simplicial 2-complex embedded in $\mathbb{R}^3$. Triangle meshes, such as those shown in Figure REF, are the most common ways to store and render geometry in the computer graphics pipeline and are the most common structures for computational shape analysis based on differential equations and related machinery.

While Definition 4.13 is somewhat abstract, the condition of a simplicial 2-complex being a manifold triangle mesh is relatively straightforward:

- Every edge is adjacent to one or two triangles.
- The set of triangles adjacent to every vertex form a closed or open fan.

Here, a *fan* is a collection of triangles sharing a common vertex and chained together by a sequence of shared edges; examples of closed and open fans are shown in Figure REF.

Typical notation for a triangle mesh is to denote it as a tuple (V, E, F). Here, $V = \{\mathbf{v}_i\}_{i=1}^n$ is a collection of vertex positions, $E \subseteq V \times V$ is a collection of edges connecting vertices together, and $F \subseteq V \times V \times V$ is a set of triangles expressed as triplets of vertices; the convention of using F for "triangle" avoids a collision with "tetrahedron" for volumetric meshes. This notation is shorthand rather than any means of communicating how a triangle mesh might be stored on a computer. In particular, this structure would make even simple queries—like listing all the triangles adjacent to a given vertex—take $O(|F|)$ time. Rather, a zoo of potential data structures has been developed for storing triangle meshes on a computer, each with its own trade-off between simplicity, storage space, and efficiency:

Example 4.5 (Triangle soup)**.** *The simplest format to store a triangle mesh is as a sequence of triplets giving triangles one-by-one:*

```
x1 y1 z1 / x2 y2 z2 / x3 y3 z3
x1 y1 z1 / x2 y2 z2 / x3 y3 z3
x1 y1 z1 / x2 y2 z2 / x3 y3 z3
x1 y1 z1 / x2 y2 z2 / x3 y3 z3
x1 y1 z1 / x2 y2 z2 / x3 y3 z3
...
```

This format is optimized for rendering: The graphics processing unit (GPU) has special functionality to render long lists of triangles. On the other hand, even checking if two triangles share a vertex is a relatively challenging task with this structure and subject to numerical imprecision.

Example 4.6 (Shared vertex structure)**.** *The popular* `.obj` *mesh format and its peers improve on triangle soup by giving separate lists of vertices and triangles:*

```
v 0.2 1.5 3.2
v 5.2 4.1 8.9
...
f 1 5 3
f 5 1 2
...
```

Here, every line marked `v` *gives the position of a vertex on a triangle mesh in* $\mathbb{R}^3$*. The lines marked* `f` *give triangles as indices into the array of vertices; this way if two triangles share a vertex, its position is not duplicated in memory.*

Example 4.7 (Halfedge)**.** *Consider the following simple algorithm for smoothing a triangle mesh:*

```
for i=1 to n_iter
     for each vertex v
          v = .5*v + .5*(average of neighbors);
```

This algorithm may seem simplistic, but it actually is quite close to mean curvature flow techniques we will construct in § REF*. Using the structures developed above, finding the neighbors of a given vertex would be extremely challenging, requiring an iteration down the entire list of triangles.*

As an alternative, the halfedge *data structure illustrate in Figure* REF *is designed to facilitate queries about local neighborhoods, e.g. finding the adjacent vertices to an edge or the edges of a triangle. The halfedge structure is a higher-dimensional analog of the linked list data structure, composed of three different data types: vertices, faces, and halfedges. Here, every edge of the triangle mesh is doubled into one forward and one backward halfedge, induced by the orientation of each triangle.*

These elements store the following information, shown in Figure REF*:*

- Vertices *store their positions in* $\mathbb{R}^3$ *as well as an arbitrary outgoing halfedge.*

- *Each* HALFEDGE *store its reverse halfedge, labeled 'flip;' the 'next' halfedge in the triangle moving in counterclockwise order; the triangle 'face' to which the halfedge belongs, and the 'vertex' at its base.*
- *Triangle* FACES *store an arbitrary adjacent halfedge.*

Assembling a halfedge is a challenging exercise in pointer manipulation.

Once a halfedge structure is assembled, walking around the mesh becomes relatively straightforward. For instance, the following pseudocode illustrated in Figure REF *gives a method for iterating over the vertices adjacent to some central vertex:*

```
Iterate(v):
startEdge = v.out;
e = startEdge;
do
     process(e.flip.from)
     e = e.flip.next
while e != startEdge
```

Example 4.8 (Adjacency matrix)**.** *Yet another way to represent an oriented triangle mesh is to leverage data structures for sparse linear algebra. Suppose we number the edges using indices* $\{1,\dots,|V|\}$*. Since each triangle is oriented, we can think of the triangles as triplets* $f_i = (f_{i1}, f_{i2}, f_{i3}) \in \{1,\dots,|V|\}^3$*, where* $i \in \{1,\dots,|F|\}$*. We similarly can write a list of edges* $e_i = (e_{i1}, e_{i2})$*; here each edge appears once (it is* not *a halfedge) and is arbitrarily oriented.*

Then, we can define a set of oriented incidence matrices *as follows:*

$$(D_{01})_{ij} := \begin{cases} 1 & \text{if } e_{i2} = j \\ -1 & \text{if } e_{i1} = j \\ 0 & \text{otherwise.} \end{cases}$$

$$(D_{12})_{ij} := \begin{cases} 1 & \text{if } e_j \in f_i \\ -1 & \text{if } -e_j \in f_i \\ 0 & \text{otherwise.} \end{cases}$$

Note $D_{01} \in \{-1,0,1\}^{|E|\times|V|}$ *and* $D_{12} \in \{-1,0,1\}^{|F|\times|E|}$*. In words, they express the following:*

- $(D_{01})_{ij}$ *is* 1 *if vertex* j *is the second vertex of edge* i*,* -1 *if vertex* j *is the first vertex of edge* i*, and* 0 *otherwise.*
- $(D_{12})_{ij}$ *is* 1 *if edge* j *is in triangle* i *with the same orientation as triangle* i*,* -1 *if edge* j *is in triangle* i *with opposite orientation, and* 0 *otherwise.*

These matrices will become extremely important when we consider constructions in the world of discrete exterior calculus (DEC) in Chapter REF*. They leverage existing machinery for storing and manipulating sparse matrices, and by reading off rows and columns in these sparse structures it is quickly possible to determine adjacency relationships along the mesh.*

4.8 EULER CHARACTERISTIC FOR SURFACES

A reasonable question to ask is how much memory a data structure will take to store a triangle mesh. Some rough rules of thumb can guide estimates when answering this question. Along the way, we will introduce one more key topological property of a simplicial complex appearing in both theory and applications.

Define the *Euler characteristic* χ of a triangle mesh as follows:

$$\chi := |V| - |E| + |F|. \tag{4.2}$$

For higher-dimensional simplicial complexes, the Euler characteristic formula simply extrapolates further as an alternating sum, counting the number of k-simplices with sign $(-1)^k$. χ is a famous *topological invariant* of a simplicial complex: If two simplicial complexes are *homotopy-equivalent*, their Euler characteristics χ must be the same. Without delving into detailed topological constructions, roughly two spaces are homotopic if the can be deformed continuously into one another; see Figure REF for some famous examples.

For surfaces and their triangulations, the Euler characteristic is related to *genus* g, the number of holes, via the relationship $\chi = 2 - 2g$. Figure REF shows some examples; a sphere has $g = 0$ and $\chi = 2$, while a torus has $g = 1$ and $\chi = 0$ and a double torus has $g = 2$ and $\chi = -2$. For our purposes, the key fact is that χ for most surfaces we encounter in practice is an integer close to 0, reflecting the fact that most shapes do not have *too* many handles.

The definition of χ as $2 - 2g$ extends to smooth surfaces. If we faithfully triangulate a surface while preserving its topology, then computing χ on the resulting mesh using (4.2) still gives $2 - 2g$. In fact, this construction is roughly how many theoretical discussions of topology go about computing χ for smooth objects.

Example 4.9 (Euler characteristic). *JS: Show some meshes, counts of vertices/edges/faces, χ*

Now, suppose a manifold triangle mesh has no boundary. From Euler's formula and a few counting arguments we can develop some estimates for the ratios between $|V|$, $|E|$, and $|F|$:

- Every edge is adjacent to two faces, and every face has three edges: $2|E| = 3|F|$
- Plugging into (4.2): $|V| - \frac{1}{2}|F| = \chi$
- Since χ is a small integer and the mesh has many vertices/faces: $|F| \approx 2|V|$

Define the *valence* of a vertex to be its number of outgoing edges. If we sum all the valences of all the vertices, every edge gets double-counted. Hence, the sum of the valences is $2|E| = 3|F| \approx 6|V|$, showing that the average valence of a vertex on a manifold triangle mesh is approximately six.

4.9 OTHER MODELS

Manifold simplicial complexes dovetail elegantly with submanifolds in $\mathbb{R}^n$; their definitions are similar, and constructions on submanifolds tend to adapt nicely to simplicial complex structures. Furthermore, an intensive study of these structures is not a purely theoretical exercise: Triangle and tetrahedral meshes are some of the most common ways to represent shapes in the computational pipeline, appearing in simulation, modeling, rendering, vision, and countless other contexts.

At the same time, we should acknowledge early and often that a simplicial view of the world neglects key expressions of geometry used in practice. These structures will make appearances in future chapters, although our main thread in many of the early chapters will be analysis on meshes. Briefly, some other shape representations each worthy of its own text include the following:

- POINT CLOUDS: A point cloud is simply a collection of points $\{\mathbf{x}_i\}_{i=1}^N \subseteq \mathbb{R}^n$ sampled from a shape embedded in $\mathbb{R}^n$. Point clouds are general and versatile, and thanks to 3D sensing modalities like LiDAR they are coming into prominence as a shape representation rather than an intermediate means of collecting data before assembling into a connected structure like a mesh. Point clouds also elegantly extend to high dimensions, making them a natural modality to consider in machine learning applications that treat datasets as high-dimensional manifold. The primary challenge in working with a point cloud is the complete lack of structure: It is unclear what the "local neighborhood" is of any $\mathbf{x}_i$, and nothing about this structure specifies an intrinsic dimensionality to the data beyond the dimension n of the embedding space.
- POLYGON SOUP: A slightly more informative structure than the point cloud is "polygon soup:" a large, potentially unstructured collection of simplices like that shown in Figure REF. Machine

learning algorithms learning from digital 3D models often have to operate on this modality, since looks matter more than functionality in designing virtual worlds. As a typical example, Figure REF shows a 3D table model from a database of shapes CITE; rather than attaching the legs of the model to the tabletop, the artist has simply modeled the table using intersecting polygons to give the illusion of a table object. Polygon soups are regular occurrences in digital shape data but can cause shape analysis algorithms to break down in subtle ways; for instance, many mesh-based algorithms would fail to analyze the legs and top of the table model as parts of the same object.

- LEVEL SETS: A representation popular in physical simulation and other domains where shapes tend to change connectivity and topology over time is the level set, in which a shape is the level set $\{\mathbf{x} : f(\mathbf{x}) = 0\}$ for some $f : \mathbb{R}^n \to \mathbb{R}^{n-k}$. This *Eulerian* (or *implicit*) representation makes sense for shapes that undergo many physical transformations, like the surface of a splashing fluid, and early pioneering work in level set methods shows that many PDEs governing the evolution of surfaces and other structures over time can be written in terms of level set functions. Typical approaches here include representing values of f on a fixed background grid of $\mathbb{R}^n$ or writing it as a sum of basis functions like Gaussians, centered at a sparse set of points.

- SUBDIVISION SURFACES: At the opposite end of the spectrum from point clouds are computational representations of classes of smooth surfaces. For example, subdivision algorithms show how to obtain a *limit surface* by subdividing meshes and inserting the new vertices at carefully-chosen positions; see Figure REF for an example. Recursively iterating the procedure leads to a truly differentiable surface, making this technique popular for computer-aided design (CAD), animation, and other applications where detail matters. Furthermore, the coarse mesh that gets subdivided provides a more limited set of degrees of freedom that an artist or engineer can use to control geometry. A frontier in the geometry processing literature explores the proper way to extend the constructions we will carry out on simplicial manifolds to their subdivision counterparts CITE. At any fixed level of subdivision, we have a mesh and can carry out the techniques we will discuss in future chapters, but special care is needed e.g. to make these methods approximately commute with subdivision rules or leverage the underlying smooth surface to increase accuracy of computations on the coarse model.

- PARAMETERIZED PATCHES: Applications in engineering often demand design of truly smooth surfaces; after all, a car body composed of lots of sharp triangular patches is far from the aerodynamic, smooth lines typically desired in this industry. Beyond subdivision, an even more straightforward way to obtain smooth surface patches for these applications is to design them explicitly as functions $f(u, v)$, e.g. by having the coordinate functions be given by polynomials in u and v. Tensor product patches, total degree surfaces, and other constructions in the CAD world fall into this category and have the advantage of not needing discrete differential geometry at all for some measurements: True computations from smooth theory apply. In exchange for ease of local computation, however, *global* conditions become harder to enforce. A well-known challenge is to glue two polynomial surface patches together with curvature continuity, and restricting a designer to model a surface or volume out of relatively few such patches can be limiting or lead to artifacts where they join together. Recent algorithms in isogeometric analysis (IGA) bring parameterized patches closer to algorithms traditionally applied to meshes, potentially alleviating the need to tessellate parameterized patches when an analytic solution is unknown for a given problem.

4.10 EXERCISES

4.1. **JS: Something about sign of a permutation from §4.5**

4.2. Verify that the conditions in §4.7 are equivalent to Definition 4.13 for triangle meshes.

4.3. **JS: Something involving walking on a halfedge**

4.4. A *regular vertex* on a triangle mesh has valence 6. Show that a closed surface admitting a manifold triangle mesh with *only* regular vertices must be a torus, that is, must have genus 1.

4.5. **JS: Something about edge collapse etc.**