Preface

This volume documents the full day course *Discrete Differential Geometry: An Applied Introduction* presented at SIGGRAPH '05 on 31 July 2005. These notes supplement the lectures given by Mathieu Desbrun, Eitan Grinspun, and Peter Schröder, compiling contributions from: Pierre Alliez, Alexander Bobenko, David Cohen-Steiner, Sharif Elcott, Eva Kanso, Liliya Kharevych, Adrian Secord, John M. Sullivan, Yiying Tong, Mariette Yvinec.

The behavior of physical systems is typically described by a set of continuous equations using tools such as geometric mechanics and differential geometry to analyze and capture their properties. For purposes of computation one must derive discrete (in space and time) representations of the underlying equations. Researchers in a variety of areas have discovered that theories, which are discrete from the start, and have key geometric properties built into their discrete description can often more readily yield robust numerical simulations which are true to the underlying continuous systems: they exactly preserve invariants of the continuous systems in the discrete computational realm.

A chapter-by-chapter synopsis The course notes are organized similarly to the lectures. Chapter 1 presents an introduction to discrete differential geometry in the context of a discussion of curves and curvature. The overarching themes introduced there, *convergence* and *structure preservation*, make repeated appearances throughout the entire volume. Chapter 2 addresses the question of which quantities one should measure on a discrete object such as a triangle mesh, and how one should define such measurements. This exploration yields a host of measurements such as length, area, mean curvature, etc., and these in turn form the basis for various applications described later on. Chapter 3 gives a concise summarization of curvature measures for discrete surfaces, paving the way for the discrete treatment of thin shell mechanics developed in Chapter 4. Continuing with the theme of discrete surfaces, Chapter 5 describes a discrete Willmore energy for fairing applications, this time preferring a discrete surface made up of linked circles instead of triangles. Such circle patterns are also key to a discrete formulation of conformal parameterization, explored in Chapter 6. At this point we shift down to explore the low-level approach of discrete exterior calculus: Chapter 7 overviews this exciting field, and Chapter 8 details a layman's approach to implementing DEC. With this in place, numerically robust and efficient simulations of the Navier-Stokes equations of fluids become possible, as described in Chapter 9. Unlike many graphics simulations of fluids which require regular grids, these fluid simulations are adept for arbitrary meshes around boundaries with complex shapes. The generation of such meshes is the subject of Chapter 10.

Chapter 1:
Introduction to Discrete Differential Geometry:
The Geometry of Plane Curves

Eitan Grinspun
Columbia University

Adrian Secord
New York University

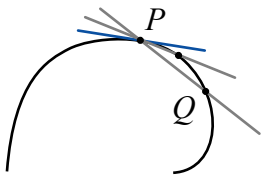
1 Introduction

The nascent field of discrete differential geometry deals with discrete geometric objects (such as polygons) which act as analogues to continuous geometric objects (such as curves). The discrete objects can be measured (length, area) and can interact with other discrete objects (collision/response). From a computational standpoint, the discrete objects are attractive, because they have been designed from the ground up with data-structures and algorithms in mind. From a mathematical standpoint, they present a great challenge: the discrete objects should have properties which are analogues of the properties of continuous objects. One important property of curves and surfaces is their curvature, which plays a significant role in many application areas (see, *e.g.*, Chapters 4 and 5). In the continuous domain there are remarkable theorems dealing with curvature; a key requirement for a discrete curve with discrete curvature is that it satisfies analogous theorems. In this chapter we examine the curvature of continuous and discrete curves on the plane.

The notes in this chapter draw from a lecture given by John Sullivan in May 2004 at Oberwolfach, and from the writings of David Hilbert in his book *Geometry and the Imagination*.

2 Geometry of the Plane Curve

Consider a plane curve, in particular a small piece of curve which does not cross itself (a *simple* curve). Choose two points, P and Q, on this curve and connect them with a straight line: a *secant*. Fixing P as the "hinge," rotate the secant about P so that Q slides along the curve toward Q. If the curve is sufficiently smooth ("*tangent-continuous* at P") then the secant approaches a definite line: the *tangent*. Of all the straight lines passing through P, the tangent is the best approximation to the curve. Consequently we define the *direction* of the curve at P to be the direction of the tangent, so that if two curves intersect at a point P their angle of intersection is given by the angle formed by their tangents at P. If both curves have identical tangents at P then we say "the curves are tangent at P." Returning to our single curve, the line perpendicular to the tangent and passing through P is called the *normal* to the curve at P. Together the tangent and normal form the axes of a local rectangular coordinate system. In addition, the tangent can be thought of as a local approximation to the curve at P.



A better approximation than the tangent is the *circle of curvature*: consider a circle through P and two neighboring points on the curve, and slide the neighboring points towards P. If the curve is sufficiently smooth ("*curvature-continuous* at P") then the circle thus approaches a definite position known as the circle of curvature or *osculating circle*; the center and radius of the osculating circle are the *center of curvature* and *radius of curvature* associated to point P on the curve. The inverse of the radius is κ, the *curvature* of the curve at P.

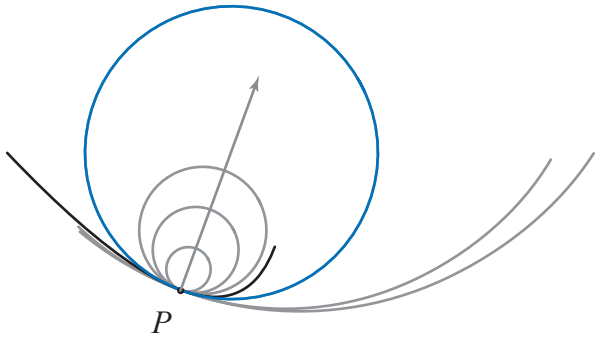


Figure 1: The family of tangent circles to the curve at point P. The circle of curvature is the only one crossing the curve at P.

If we also consider a sense of traversal along the curve segment (think of adding an arrowhead at one end of the segment) then we may measure the *signed* curvature, identical in magnitude to the curvature, but negative in sign whenever the curve is turning clockwise (think of riding a bicycle along the curve: when we turn to the right, it is because the center of curvature lies to the right, and the curvature is negative).

Another way to define the circle of curvature is by considering the infinite family of circles which are tangent to the curve at P (see Figure 1). Every point on the normal to the curve at P serves as the center for one circle in this family. In a small neighborhood around P the curve divides the plane into two sides. Every circle (but one!) in our family lies entirely in one side or the other. Only the circle of curvature however spans both sides, crossing the curve at P. It divides the family of tangent circles into two sets: those with radius smaller than the radius of curvature lying on one side, and those with greater radius lying on the other side. There may exist special points on the curve at which the circle of curvature does not locally cross the curve, and in general these are finite and isolated points where the curve has a (local) axis of symmetry (there are four such points on an ellipse). However on a circle, or a circular arc, the special points are infinitely many and not isolated.

That the circle of curvature crosses the curve may be reasoned by various arguments. As we traverse the curve past

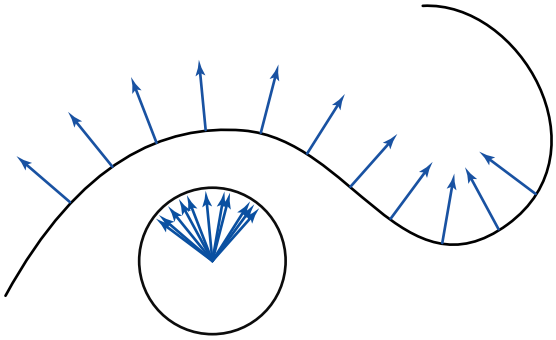


Figure 2: The Gauss map assigns to every point on the curve a corresponding point on the unit circle.

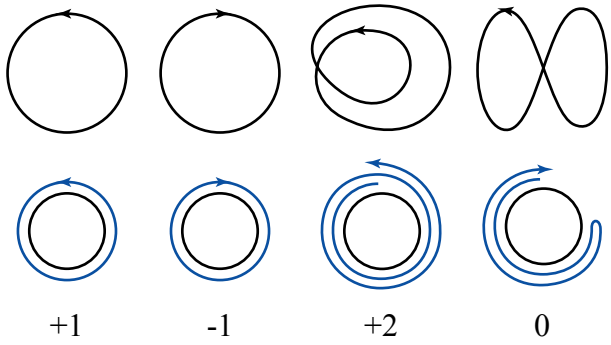


Figure 3: Turning numbers of various closed curves. *Top row:* Two simple curves with opposite sense of traversal, and two self-intersecting curves, one of which "undoes" the turn. *Bottom row:* Gaussian image of the curves, and the associated turning numbers.

point P, the curvature is typically either increasing or decreasing, so that in the local neighborhood of P, so that the osculating circle in comparison to the curve will have a higher curvature on one side and lower on the other. An alternative argument considers our three point construction. Trace along a circle passing through three consecutive points on the curve to observe that the circle must pass from side A to side B on the first point, B to A on the second, and A to B on the third. Similar reasoning of our two-point construction shows that in general the tangent does not cross the curve—the isolated exceptions are the *points of inflection*, where the radius of curvature is infinite and the circle of curvature is identical to the tangent.

Informally we say that P, the tangent at P, and the osculating circle at P have one, two, and three coincident points in common with the curve, respectively. Each construction in sequence considers an additional approaching point in the neighborhood of P and the so-called *order of approximation* (0, 1, and 2 respectively) is identical to the number of additional points.

In 1825 Karl F. Gauss introduced a new tool for thinking about the shape of curves and surfaces. Begin by fixing a sense of traversal for the curve, naturally inducing for every point on the curve a direction for the tangent. By convention, the normal points a quarter turn counterclockwise from tangent direction. Gauss's idea is to draw a unit circle on the plane of the curve, and for any point on the curve, to represent the normal by the radius of the circle parallel to the normal and having the same sense as the normal. To any point P on the curve, the *Gauss map* assigns a point Q on the unit circle, namely the point where the *radius* meets the circle (here, radius means the line segment from the center of the circle to a point on the circumference). Observe that the normal at P is parallel to the radius of the circle, and the tangent to the curve at P is parallel to the tangent to the circle at Q. That the tangent at P and Q are parallel is used to simplify important definitions in differential geometry (see, *e.g.*, the definition of the shape operator in the chapter on discrete shells). While the Gauss map assigns exactly one point on the unit circle to any point on the curve, there may be multiple points on the curve that map to the same point on the circle, *i.e.*the map is not one-to-one.

Consider the image of the curve under the Gauss map: the *Gaussian image* of a curve is the union of all points on the unit circle corresponding to all points on the given curve. For an open curve, the Gaussian image may be an arc or may be the unit circle. Consider a closed simple plane curve: the image is always the unit circle. If we allow the closed curve to intersect itself, we can count how many times the image completely "wraps around" the unit circle (and in which sense): this is the *turning number* or the *index of rotation*, denoted k. It is unity for a simple closed curve traversed counterclockwise. It is zero or ± 2 for curve that self-intersects once, depending on the sense of traversal and on whether or not the winding is "undone."

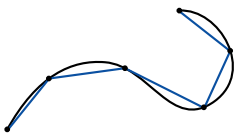
Turning Number Theorem. An old and well-known fact about curves is that the integral of signed curvature over a closed curve, Ω, is dependent *only* on the turning number:

$$\int_{\Omega} \kappa \, ds = 2\pi k \ .$$

No matter how much we wiggle and bend the curve, if we do not change its turning number we do not change its *total signed curvature*[1]. To change the total signed curvature of Ω we are forced to alter its turning number by adjusting the curve to introduce (or rearrange) self-intersecting loops. This theorem about the significance of the turning number is a piece of mathematical *structure*: together all the structure we discover embodies our understanding of differential geometry. Consequently, our computational algorithms will take advantage of this structure. In computing with discrete approximations of continuous geometry, we will strive to keep key pieces of structure intact.

3 Geometry of the Discrete Plane Curve

Given a curve, r, approximate it drawing an *inscribed polygon* p: a finite sequence of (point) *vertices*, $V_1, V_2, \ldots V_n$, ordered by a traversal of the curve, and line segments connecting successive vertices[2].



[1] Beware that in the context of space curves, the phrase "total curvature" is occasionally used to denote the Pythagorean sum of torsion and curvature—a pointwise quantity like curvature. In contrast, here we mean the integral of curvature over the curve.

[2] While we concern ourselves here only with plane curves, this treatment may be extended to curves in a higher-dimensional am-

The length of the inscribed polygon is given by

$$len(p) = \sum_{i=0}^{n} d(V_i, V_{i+1}) \ ,$$

where $d(\cdot, \cdot)$ measures the euclidean distance[3] between two points. We find the length of the continuous curve by taking the supremum over all possible inscriptions:

$$len(r) = \sup_{p \text{ inscribed in } r} len(p) \ .$$

Next, choose a sense of traversal along the curve, naturally inducing a sense for the inscribed polygon. The (discrete) *total signed curvature* of the inscribed polygon is given by

$$tsc(p) = \sum_{i=0}^{n} \alpha_i \ ,$$

where α_i is the signed *turning angle* at vertex V_i, measured in the sense that a clockwise turn has negative sign; if p is open then $\alpha_0 = \alpha_n = 0$. (N.B.: the turning *angle* is a local quantity at each vertex, whereas the turning *number* is a global quantity of a curve—these are two distinct concepts). Again, we may express the total signed curvature of the continuous curve by taking the supremum over all possible inscribed polygons:

$$tsc(r) = \sup_{p \text{ inscribed in } r} tsc(p) \ .$$

A definition based on suprema serves as an elegant foundation for defining the (integral quantities) length and total curvature of a smooth curve using only very simple polygonal geometry; however suprema are typically is not well suited for computation. For an equivalent, computationally meaningful definition, we construct an infinite sequence of inscribed polygons, p_1, p_2, p_3, ..., that approaches the position of r; analogous definitions of $len(r)$ and $tsc(r)$ are formulated as limits of measurements over elements of the sequence.

To clarify what we mean by "the inscribed polygon p approaches the position of r," define the *geometric mesh size* of p by the length of its longest line segment:

$$h(p) = \max_{0 \le i < n} d(V_i, V_{i+1}) \ .$$

Suppose that r is a smooth simple curve. By smooth we mean that every point on the curve has a unique well-defined tangent[4]. Then one can show that given a sequence p_1, p_2, p_3, ... such that $h(p_i)$ vanishes in the limit of the sequence, then $len(p_i)$ approaches $len(r)$. An analogous statement holds for total curvature, as summarized by the following statement:[5]

bient space, $M^m \subseteq \mathbb{R}^d$, by replacing line segments with shortest geodesics in this definition, and straight-line distance by length of geodesic in subsequent definitions.

[3]It measures distance using the metric of the ambient space, in our case $\mathbb{R}^2$.

[4]Observe that smoothness here is in a purely geometric sense—the notion of parametric smoothness in the context of parameterized curves is a different matter altogether.

[5]Note that there are sequences of pathological polygons whose mesh size vanishes yet the limit of the sequence does not approach the curve. For example, if the curve is a circle, consider a polygon whose vertices all cluster about a single point of the circle.

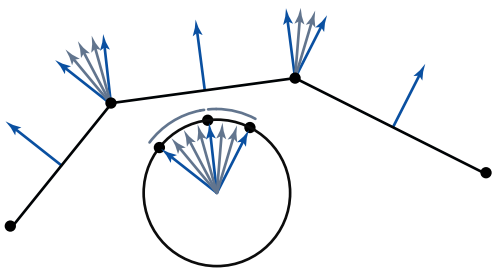


Figure 4: The discrete Gauss map assigns to every edge of the polygon a corresponding point on the unit circle, and to every vertex of the polygon a corresponding arc on the unit circle.

Convergence. A key recurring theme in discrete differential geometry is the *convergence* of a measurement taken over a sequence of discrete objects each better approximating a particular smooth object. In the case of a plane curve, a sequence of inscribed polygons, each closer in position to the curve, generates a sequence of measurements that approach that of the curve:

$$len(r) = \lim_{h(p_i) \to 0} len(p_i) \ ,$$

$$tsc(r) = \lim_{h(p_i) \to 0} tsc(p_i) \ .$$

Establishing convergence is a key step towards numerical computations which use discrete objects as approximations to continuous counterparts. Indeed, one might argue that the notion of continuous *counterpart* is only meaningful in the context of established convergence. Put simply, if we choose an inscribed polygon as our discrete analogue of a curve, then as the position of the approximating polygon approaches the curve, the measurements taken on the approximant should approach those of the underlying curve.

Next, consider the tangents, normals, and Gaussian image of a closed polygon p. Repeating the two-point limiting process we used to define the tangent for a point on the curve, we observe that every vertex of the polygon has two limiting tangents (thus two normals), depending on the direction from which the limit is taken (see Figure 4). Define the Gaussian image of p by assigning to every vertex V_i the arc on the unit circle whose endpoints are the two limiting normals and whose signed angle equals the signed turning angle α_i, *i.e.*, as if one "smoothly interpolated" the two normals in the Gaussian image. Every point on the polygon away from the vertices has a unique normal which corresponds in the Gaussian image to the meeting point of consecutive arcs. The sense of traversal along the polygon induces a natural sense of traversal along the arcs of the Gaussian image. With this construction in place, our definition of turning number for a smooth plane curve carries over naturally to the setting of closed polygons. Not that for for open polygons, the Gaussian image of vertices at the endpoints is a point on the unit circle (a degenerate arc).

As long as the length of the longest line segment shrinks, *i.e.* the polygon clusters more tightly around the point, then this sequence of polygons will satisfy our definition but will clearly not converge to a circle. One may introduce stronger requirements on the polygon sequence to exclude such pathological sequences.

Structure preservation. Does the Turning Number Theorem hold for discrete curves? Yes. Recall that the sum of exterior angles of a simple closed polygon is 2π. This observation may be generalized to show that $tsc(p) = 2\pi k$ where k is the turning number of the polygon. We stress a key point: the Turning Number Theorem is *not* a claim that the total signed curvature converges to a multiple of 2π in the limit of a finely refined inscribed polygon. The Turning Number Theorem is preserved *exactly* and it holds for *any* (arbitrarily coarse) closed polygon. Note, however, that the turning number of an inscribed polygon may not match that of the smooth curve, at least until sufficiently many vertices are added (in the right places) to capture the topology of the curve.

4 Parameterization of the Plane Curve

So far in our exploration of curves our arguments have never explicitly made reference to a system of coordinates. This was to stress the point that the geometry (or *shape*) of the curve can be described without reference to coordinates. Nevertheless, the idea of *parameterizing* a curve occurs throughout applied mathematics. Unfortunately, parameterization can sometimes obscure geometric insight. At the same time, it is an exceedingly useful computational tool, and as such we complete our exploration of curves with this topic.

In working with curves it is useful to be able to indicate particular points and their neighborhoods on the curve. To that end we *parameterize* a curve over a real interval mapping each parameter point, $t \in [0, a]$, to a point $R(t)$ on the plane:

$$R : [0, a] \to \mathbb{R}^2 \ .$$

Thus the endpoints of finite open curve are $R(0)$ and $R(a)$; for closed curves we require $R(0) = R(a)$.

The parameterization of a curve is not unique. Besides the geometric information encoded in the image of R, the parameterization also encodes a parameterization-dependent *velocity*. To visualize this, observe that moving the parameter at unit velocity slides a point $R(t)$ along the curve: the rate of change of $R(t)$, or velocity, is the vector $\vec{v}(t) = \frac{d}{dt}R(t)$. Indeed, given any strictly increasing function $t(s) : [0, b] \to [0, a]$ we *reparameterize* the curve as $R(t(s))$ so that moving along $s \in [0, b]$ generates the same points along the curve; the geometry remains the same, but by chain rule of the calculus the velocity is now $\vec{v}(s) = \frac{d}{ds}R(t(s)) = \frac{d}{dt}R(t(s))\frac{d}{ds}t(s)$: at every point the $R(t(s))$ reparameterization scales the velocity by $\frac{d}{ds}t(s)$.

Given a parameterized curve there is a unique reparameterization, $\hat{R}(s) = R(t(s))$, with the property that $\|\vec{v}(s)\| = 1,\ s \in [0, b]$. In *arc-length* parameterization of a curve, unit motion along the parameter s corresponds to unit motion along the length of the curve. Consequently, s is the length traveled along the curve walking from $\hat{R}(0)$ to $\hat{R}(s)$, therefore b is the length of the entire curve.

In the special setting of an arc-length parameterization the curvature at a point $R(s)$ is identical to the second derivative $\frac{d^2}{ds^2}R(s)$. It is a grave error to identify curvatures with second derivatives in general. The former is a geometric quantity only, and we defined it without reference to a parameterization; the latter encodes both geometry and velocity, and is parameterization-dependent. Here a spaceship analogy is helpful. If a spaceship travels at unit speed along a curved path, the curvature give the acceleration of the spaceship. Now if the spaceship travels at a nonuniform velocity along the path, then part of the acceleration is due to curvature, and part is due to speeding up and slowing down. A parameterization encodes velocity—this can be extremely useful for some applications.

Parameterization enables us to reformulate our statement of convergence. Given a sequence of parameter values, $0 = t_1 \leq t_2 \ldots \leq t_{n-1} \leq t_n = b$, for a "sufficiently well-behaved" parameterization of a "sufficiently well-behaved" curve[6], we may form an inscribed polygon taking $V_i = R(t_i)$. Then the *parametric mesh size* of the inscribed polygon is the greatest of all parameter intervals $[t_i, t_i + 1]$:

$$h_R(p) = \max_i (t_{i+1} - t_i) \ .$$

Unlike *geometric* mesh size, *parametric* mesh size is dependent on the chosen parameterization.

As before, consider a sequence of inscribed polygons, each sampling the curve at more parameter points, and in the limit sampling the curve at all parameter points: the associated sequences of discrete measurements approach their continuous analogs:

$$len(r) = \lim_{h_R(p_i) \to 0} len(p_i) \ ,$$

$$tsc(r) = \lim_{h_R(p_i) \to 0} tsc(p_i) \ .$$

5 Conclusion and Overview

So far we have looked at the geometry of a plane curve and demonstrated that it is possible to define its discrete analogue. The formulas for length and curvature of a discrete curve (a polygon) are immediately amenable to computation. *Convergence* guarantees that in the presence of abundant computational resources we may refine our discrete curve until the measurements we take match to arbitrary precision their counterparts on a smooth curve. We discussed an example of *structure preservation*, namely that the Turning Number Theorem holds exactly for discrete curves, even for coarse mesh sizes. If we wrote an algorithm whose correctness relied on the Turning Number Theorem, then the algorithm could be applied to our discrete curve.

The following chapters will extend our exploration of discrete analogues of the objects of differential geometry to the settings of surfaces and volumes and to application areas spanning physical simulation (thin shells and fluids) and geometric modeling (remeshing and parameterization). In each application area algorithms make use of mathematical structures that are carried over from the continuous to the discrete realm. We are *not* interested in preserving structure just for mathematical elegance—each application demonstrates that by carrying over the right structures from the continuous to the discrete setting, the resulting algorithms exhibit impressive computational and numerical performance.

[6]Indeed, the following theorems depend on the parameterization being *Lipschitz*, meaning that small changes in parameter value lead to small motions along the curve:

$$d(R(a), R(b)) \leq C|a - b| \ ,$$

for some constant C. The existence of a Lipschitz parameterization is equivalent to the curve being *rectifiable*, or having finite arclength. Further care must be taken in allowing non-continuous curves with finitely many isolated jump points.

Chapter 2:
What Can We Measure?

Peter Schröder
Caltech

1 Introduction

When characterizing a shape or changes in shape we must first ask, what can we measure about a shape? For example, for a region in $\mathbb{R}^3$ we may ask for its volume or its surface area. If the object at hand undergoes deformation due to forces acting on it we may need to formulate the laws governing the change in shape in terms of measurable quantities and their change over time. Usually such measurable quantities for a shape are defined with the help of integral calculus and often require some amount of smoothness on the object to be well defined. In this chapter we will take a more abstract approach to the question of measurable quantities which will allow us to define notions such as mean curvature integrals and the curvature tensor for piecewise linear meshes without having to worry about the meaning of second derivatives in settings in which they do not exist. In fact in this chapter we will give an account of a classical result due to Hadwiger, which shows that for a convex, compact set in $\mathbb{R}^n$ there are only $n+1$ unique measurements if we require that the measurements be invariant under Euclidian motions (and satisfy certain "sanity" conditions). We will see how these measurements are constructed in a very straightforward and elementary manner and that they can be read off from a characteristic polynomial due to Steiner. This polynomial describes the volume of a family of shapes which arise when we "grow" a given shape. As a practical tool arising from these consideration we will see that there is a well defined notion of the curvature tensor for piecewise linear meshes and we will see very simple formulas for quantities needed in physical simulation with piecewise linear meshes. Much of the treatment here will initially be limited to convex bodies to keep things simple. This limitation that will be removed at the very end.

The treatment in this chapter draws heavily upon work by Gian-Carlo Rota and Daniel Klein, Hadwigers pioneering work, and some recent work by David Cohen-Steiner and colleagues.

2 Geometric Measures

To begin with let us define what we mean by a measure. A measure is a function μ defined on a family of subsets of some set S, and it takes on real values: $\mu : L \to \mathbb{R}$. Here L denotes this family of subsets and we require of L that it is closed under finite set union and intersection as well as that it contains the empty set, $\emptyset \in L$. The measure μ must satisfy two axioms: (1) $\mu(\emptyset) = 0$; and (2) $\mu(A \cup B) = \mu(A) + \mu(B) - \mu(A \cap B)$ whenever A and B are measureable. The first axiom is required to get anything that has a hope of being well defined. If $\mu(\emptyset)$ was not equal to zero the measure of some set $\mu(A) = \mu(A \cup \emptyset) = \mu(A) + \mu(\emptyset)$ could not be defined. The second axiom captures the idea that the measure of the union of two sets should be the sum of the measures minus the measure of their overlap. For example, consider the volume of the union of two sets which clearly has this property. It will also turn that the additivity property is the key to reducing measurements for complicated sets to measurements on simple sets. We will furthermore require that all measures we consider be invariant under Euclidian motions, *i.e.*, translations and rotations. This is so that our measurements do not depend on where we place the coordinate origin and how we orient the coordinate axes. A measure which depended on these wouldn't be very useful.

Let's see some examples. A well known example of such a measure is the volume of bodies in $\mathbb{R}^3$. Clearly the volume of the empty body is zero and the volume satisfies the additivity axiom. The volume also does not depend on where the coordinate origin is placed and how the coordinate frame is rotated. To uniquely tie down the volume there is one final ambiguity due to the units of measurement being used, which we must remove. To this end we enforce a normalization which states that the volume of the unit, coordinate axis aligned parallelipiped in $\mathbb{R}^n$ be one. With this we get

$$\mu_n^n(x_1, \ldots, x_n) = x_1 \cdot \ldots \cdot x_n$$

for x_1 to x_n the side lengths of a given axis aligned parallelipiped. The superscript n denotes this as a measure on $\mathbb{R}^n$, while the subscript denotes the type of measurement being taken. Clearly the definition of μ_n^n is translation invariant. It also does not depend on how we number our coordinate axes, *i.e.*, it is invariant under permutations of the coordinate axes. Finally if we rotate the global coordinate frame none of the side lengths of our parallelipiped change so neither does μ_n^n. Notice that we have only defined the meaning of μ_n^n for axis aligned parallelipipeds as well as finite unions and intersections of such parallelipipeds. The definition can be extended to more general bodies through a limiting process akin to how Riemann integration fills the domain with ever smaller boxes to approach the entire domain in the limit. There is nothing here that prevents us from performing the same limit process. In fact we will see later that once we add this final requirement, that the measure is continuous in the limit, the class of such measures is completely tied down. This is Hadwiger's famous theorem. But, more on that later.

Of course the next question is, are there other such invariant measures? Here is a proposal:

$$\mu_{n-1}^n(x_1, \ldots, x_n) = \\ x_1x_2 + x_1x_3 + \ldots + x_1x_n + x_2x_3 + \ldots + x_2x_n \ldots$$

For an axis aligned parallelipiped in $\mathbb{R}^3$ we'd get

$$\mu_2^3(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3x_1$$

which is just half the surface area of the paralellipiped with sides x_1, x_2, and x_3. Since we have the additivity property we can certainly extend this definition to more general bodies

through a limiting process and find that we get, up to normalization, the surface area.

Continuing in this fashion we are next led to consider

$$\mu_1^3(x_1, x_2, x_3) = x_1 + x_2 + x_3$$

(and similarly for μ_1^n). For a parallelipiped this function measures one quarter the sum of lengths of its bounding edges. Once again this new measure is clearly rigid motion invariant. What we need to check is whether it satisfies the additivity theorem. Indeed it does, which is easily checked for the union of two paralellipipeds. What is less clear is what this measure represents if we extend it to more general shapes where the notion of "sum of edge lengths" is not clear. The resulting continuous measure is sometimes referred to as the *mean width*.

From these simple examples we can see a pattern. For Euclidian n-space we can use the elementary symmetric polynomials in edge lengths to define n invariant measures

$$\mu_k^n(x_1, \ldots, x_n) = \sum_{1 \le i_1 < i_2 < \ldots < i_k \le n} x_{i_1} x_{i_2} \ldots x_{i_k}$$

for $k = 1, \ldots, n$ for parallelipipeds. To extend this definition to more general bodies we'll follow ideas from geometric probability. In particular we will extend these measures to the ring of compact convex bodies, *i.e.*, finite unions and intersections of compact convex sets in $\mathbb{R}^n$.

3 How Many Points, Lines, Planes,... Hit a Body?

Consider a compact convex set, a *convex body*, in $\mathbb{R}^n$ and surround it by a box. One way to measure its volume is to count the number of points which, when randomly thrown into the box, hit the body versus those that hit empty space inside the box. To generalize this idea we consider *affine subspaces* of dimension $k < n$ in $\mathbb{R}^n$. Recall that an affine subspace of dimension k is spanned by $k+1$ points $p_i \in \mathbb{R}^n$ (in general position), *i.e.*, the space consists of all points q which can be written as affine combinations $q = \sum_i \alpha_i p_i$, $\sum_i \alpha_i = 1$. Such an affine subspace is simply a linear subspace translated, *i.e.*, it does not necessarily go through the origin. For example, for $k = 1$, $n = 3$ we will consider all lines—a line being the set of points one can generate as affine combinations of two points on the line—in three space. Let $\lambda_1^3(R)$ be the measure of all lines going through a rectangle in $\mathbb{R}^3$. Then

$$\lambda_1^3(R) = c\mu_2^3(R),$$

i.e., the measure of all lines which meet the rectangle is proportional to the area of the rectangle. To see this, note that a given line (in general position) either meets the rectangle once or not at all. Conversely for a given point in the rectangle there is a whole set of lines—a sphere's worth—which "pierce" the rectangle in the given point. The measure of those lines is proportional to the area of the unit sphere. Since this is true for all points in the rectangle we see that the total measure of all such lines must be proportional to the area of the rectangle with a constant of proportionality depending on the measure of the sphere. For now such constants are irrelevant for our considerations so we will just set it to unity. Given a more complicated shape C in a plane nothing prevents from performaing a limiting process and we see that the measure of lines meeting C is

$$\lambda_1^3(C) = \mu_2^3(C),$$

i.e., it is proportional to the area of the region C. Given a union of rectangles $D = \cup_i R_i$, each living in a different plane, we get

$$\int X_D(\omega)\, d\lambda_1^3(\omega) = \sum_i \mu_2^3(R_i).$$

Here $X_D(\omega)$ counts the number of times a line ω meets the set D and the integration is performed over all lines. Going to the limit we find for some convex body E a measure proportional to its surface area

$$\int X_E(\omega)\, d\lambda_1^3(\omega) = \mu_2^3(E).$$

Using planes ($k = 2$) we can now generalize the mean width. For a straight line $c \in \mathbb{R}^3$ we find $\lambda_2^3(c) = \mu_1^3(c) = l(c)$, *i.e.*, the measure of all planes that meet the straight line is proportional—as before we set the constant of proportionality to unity—to the length of the line. The argument mimics what we said above: a plane either meets the line once or not at all. For a given point on the line there is once again a whole set of planes going through that point. Considering the normals to such planes we see that this set of planes is proportional in measure to the unit sphere without being more precise about the actual constant of proportionality. Once again this can be generalized with a limiting process giving us the measure of all planes hitting an arbitrary curve in space as proportional to its length

$$\int X_F(\omega)\, d\lambda_2^3(\omega) = \mu_1^3(F).$$

Here the integration is performed over all planes $\omega \in \mathbb{R}^3$, and X_F counts the number of times a given plane touches the curve F.

It is easy to see that this way of measuring recovers the perimeter of a parallelipiped as we had defined it before

$$\lambda_2^3(P) = \mu_1^3(P).$$

To see this consider the integration over all planes but taken in groups. With the parallelipiped having one corner at the origin—and being axis aligned—first consider all planes whose normal (n_x, n_y, n_z) has either all non-negative or non-positive entries (*i.e.*, the normal, or its negative, points into the first octant). Any such plane, if it meets the parallelipiped, meets it in a point along either x_1, x_2 or x_3 giving us the desired $\mu_1^3(P) = x_1 + x_2 + x_3$ as the measure of all such planes. The same argument holds for the remaining seven octants giving us the desired result up to a constant. We can now see that $\mu_1^3(E)$ for some convex body E can be written as

$$\int X_E(\omega)\, d\lambda_2^3(\omega) = \mu_1^3(E),$$

i.e., the measure of all planes which meet E. With this we have generalized the notion of perimeter to more general sets.

All this can be summarized as follows. Let μ be a measure which is Euclidean motion invariant. Then it can be written, up to normalization, as a linear combination of the measures $\mu_k^n(C)$ of all affine subspaces of dimension $n-k$ meeting $C \subset \mathbb{R}^n$ for $k = 1, \ldots, n$. These measures are called the *intrinsic volumes*.

Are these all such measures? It turns out there is one measure missing, which corresponds to the elementary symmetric function of order zero

$$\mu_0(x_1, \ldots, x_n) = \begin{cases} 1 & n > 0 \\ 0 & n = 0 \end{cases}$$

This very special measure is the Euler characteristic of a convex body. It takes the value 1 on all non-empty convex bodies. The main trick is to prove that μ_0 is indeed well defined. This can be done by induction. In dimension $n = 1$ we consider closed intervals $[a, b]$, $a < b$. Instead of working with the set directly we consider a functional on the characteristic function $f_{[a,b]}$ of the set which does the trick

$$\chi_1(f) = \int_{\mathbb{R}} f(\omega) - f(\omega+)\, d\omega.$$

Here $f(\omega+)$ denotes the right limiting value of f at ω: $\lim_{\epsilon \to 0} f(\omega + \epsilon)$, $\epsilon > 0$. For the set $[a, b]$, $f(\omega) - f(\omega+)$ is zero for all $\omega \in \mathbb{R}$ except b since $f(b) = 1$ and $f(b+) = 0$. For higher dimensions we proceed by induction. In $\mathbb{R}^n$ take a straight line L and consider the affine subspaces A_ω of dimension $n - 1$ which are orthogonal to L and parameterized by ω along L. Letting f be the characteristic function of a convex body in $\mathbb{R}^n$ we get

$$\chi_n(f) = \int_{\mathbb{R}} \chi_{n-1}(f_\omega) - \chi_{n-1}(f_{\omega+})\, d\omega.$$

Here f_ω is the restriction of f to the affine space A_ω or alternatively the characteristic function of the intersection of A_ω and the convex body of interest. With this we define $\mu_0^n(G) = \chi_n(f)$ for any finite union of convex bodies G and f the characteristic function of the set $G \in \mathbb{R}^n$.

That this definition of μ_0^n amounts to the Euler characteristic is not immediately clear, but it is easy to show, if we convince ourselves that for any non empty convex body $C \in \mathbb{R}^n$

$$\mu_0^n(\text{Int}(C)) = (-1)^n.$$

For $n = 1$, *i.e.*, the case of open intervals on the real line, this statement is obviously correct. We can now apply the recursive definition to the characteristic function of the interior of C and get

$$\mu_0^n(\text{Int}(C)) = \int_\omega \chi_{n-1}(f_\omega) - \chi_{n-1}(f_{\omega+})\, d\omega.$$

By induction the right hand side is zero except for the first ω at which $A_\omega \cap C$ is non-empty. There $\chi_{n-1}(f_{\omega+}) = (-1)^{n-1}$, thus proving our assertion for all n.

The Euler-Poincaré formula for a polyhedron

$$|F| - |E| - |V| = 2(1 - g)$$

which relates the number of faces, edges, and vertices to the genus now follows easily. Given a polyhedron simply write it as the non-overlapping union of the interiors of all its cells from dimension n down to dimension 0, where the interior of a vertex (0-cell) is the vertex itself. Then

$$\mu_0^n(P) = \sum_{c \in P} \mu_0^n(\text{Int}(c)) = c_0 - c_1 + c_2 - \ldots$$

where c_i equals the number of cells of dimension i. For the case of a polyhedron in $\mathbb{R}^3$ this is exactly the Euler-Poincaré formula.

4 The Intrinsic Volumes and Hadwiger's Theorem

The above machinery can now be used to define the intrinsic volumes as functions of the Euler characteristic alone for all finite unions of convex bodies G

$$\mu_k^n(G) = \int \mu_0^n(G \cap \omega)\, d\lambda_{n-k}^n(\omega).$$

Here $\mu_0^n(G \cap \omega)$ plays the role of $X_G(\omega)$ we used earlier to count the number of times ω hits G.

There is one final ingredient missing, continuity in the limit. Suppose C_n is a sequence of convex bodies which converges to C in the limit as $n \to \infty$. Hadwiger's theorem says that if a Euclidean motion invariant measure μ of convex bodies in $\mathbb{R}^n$ is continuous in the sense that

$$\lim_{C_n \to C} \mu(C_n) = \mu(C)$$

then μ must be a linear combination of the intrinsic volumes μ_k^n, $k = 0, \ldots, n$. In other words, the intrinsic volumes, under the additional assumption of continuity, are the only linearly independent, Euclidean motion invariant, additivie measures on finite unions and intersections of convex bodies in $\mathbb{R}^n$.

What does all of this have to do with the applications we have in mind? A consequence of Hadwiger's theorem assures us that if we want to take measurements of piecewise linear geometry (surface or volume meshes, for example) such measurements should be functions of the intrinsic volumes. This assumes of course that we are looking for additive measurements which are Euclidean motion invariant and continuous in the limit. For a triangle for example this would be area, edge length, and Euler characteristic. Similarly for a tetrahedron with its volume, surface area, mean width, and Euler characteristic. As the name suggests all of these measurements are intrinsic. For a 2-manifold mesh which is the boundary of a solid one of these measurements is an *extrinsic* quantity corresponding to the dihedral angle between triangles meeting at an edge (see below).

5 Steiner's Formula

We return now to questions of discrete differential geometry by showing that the intrinsic volumes are intricately linked to curvature integrals and represent their generalization to the non-smooth setting. This connection is established by Steiner's formula.

Consider a non-empty convex body $K \in \mathbb{R}^n$ together with its parallel bodies

$$K_\epsilon = \{x \in \mathbb{R}^n : d(x, K) \leq \epsilon\}$$

where $d(x, K)$ denotes the Euclidean distance from x to the set K. In effect K_ϵ is the body K thickened by ϵ. Steiner's formula gives the volume of K_ϵ as a polynomial in ϵ

$$V(K_\epsilon) = \sum_{j=0}^{n} V(\mathbb{B}_{n-j}) V_j(K) \epsilon^{n-j}.$$

Here the $V_j(K)$ are the measures μ_k^n we have seen earlier. For this formula to be correct the $V_j(K)$ are normalized so that they compute the j-dimensional volume when restricted to a j-dimensional subspace of $\mathbb{R}^n$. $V(\mathbb{B}_{n-j}) = \pi^{n/2}/\Gamma(1 +$

$1/2n$) denotes the $(n-j)$-volume of the $(n-j)$-unit ball. In particular we have $V(\mathbb{B}_0) = 1$, $V(\mathbb{B}_1) = 2$, $V(\mathbb{B}_2) = \pi$, and $V(\mathbb{B}_3) = 4\pi/3$.

In the case of a polyhedron we can verify Steiner's formula "by hand." Consider a tetrahedron in $T \in \mathbb{R}^3$ and the volume of its parallel bodies T_ϵ. For $\epsilon = 0$ we have the volume of T itself ($V_3(T)$). The first order term in ϵ, $2V_2(T)$, is controlled by area measures: above each triangle a displacement along the normal creates additional volume proportional to ϵ and the area of the triangle. The second order term in ϵ, $\pi V_1(T)$, corresponds to edge lengths. Above each edge the parallel bodies form a wedge with opening angle θ which is the exterior angle of the faces meeting at that edge and radius ϵ (this is the extrinsic measurement alluded to above). The volume of such as wedge is proportional to edge length, exterior angle, and ϵ^2. Finally the third order term in ϵ, $4\pi/3V_0(T)$, corresponds to the volume of the parallel bodies formed over vertices. Each vertex gives rise to additional volume spanned by the vertex and a spherical cap above it. The spherical cap corresponds to a spherical triangle formed by the three incident triangle normals. The volume of such a spherical wedge is proportional to its solid angle and ϵ^3.

If we have a convex body with a boundary which is C^2 we can give a different representation of Steiner's formula. Consider such a convex $M \in \mathbb{R}^n$ and define the offset function

$$g(p) = p + t\vec{n}(p)$$

for $0 \leq t \leq \epsilon$, $p \in \partial M$ and $\vec{n}(p)$ the outward normal to M in p. We can now directly compute the volume of M_ϵ as the sum of $V_n(M)$ and the volume between the surfaces ∂M and ∂M_ϵ. The latter can be written as an integral of the determinant of the Jacobian of g

$$\int_{\partial M} \left(\int_0^\epsilon \left| \frac{\partial g(p)}{\partial p} \right| dt \right) dp.$$

Since we have a choice of coordinate frame in which to do this integration we may assume wlog that we use principal curvature coordinates on ∂M, *i.e.*, a set of orthogonal directions in which the curvature tensor diagonalises. In that case

$$\begin{aligned} \left| \frac{\partial g(p)}{\partial p} \right| &= |\mathbb{I} + t\mathbb{K}(p)| \\ &= \prod_{i=1}^{n-1} (1 + \kappa_i(p)t) \\ &= \sum_{i=0}^{n-1} \mu_i^{n-1}(\kappa_1(p), \ldots, \kappa_{n-1}(p)) t^i. \end{aligned}$$

In other words, the determinant of the Jacobian is a polynomial in t whose coefficients are the elementary symmetric functions in the principal curvatures. With this subsitution we can trivially integrate over the variable t and get

$$V(M_\epsilon) = V_n(M) + \sum_{i=0}^{n-1} \frac{\epsilon^{i+1}}{i+1} \int_{\partial M} \mu_i^{n-1}(\kappa_1(p), \ldots, \kappa_{n-1}(p))\, dp.$$

Comparing the two versions of Steiner's formula we see that the intrinsic volumes generalize curvature integrals. For example, for $n = 3$ and some arbitrary convex body K we get

$$V(K_\epsilon) = 1V_3(K) + 2V_2(K)\epsilon + \pi V_1(K)\epsilon^2 + \frac{4\pi}{3} V_0(K)\epsilon^3$$

while for a convex body M with C^2 smooth boundary the formula reads as

$$\begin{aligned} V(M_\epsilon) = V_3(M) + & \underbrace{\left(\int_{\partial M} \underbrace{\mu_0^2(\kappa_1(p), \kappa_2(p))}_{=1}\, dp \right)}_{=A} \epsilon + \\ & \left(\int_{\partial M} \underbrace{\mu_1^2(\kappa_1(p), \kappa_2(p))}_{=2H}\, dp \right) \frac{\epsilon^2}{2} + \\ & \underbrace{\left(\int_{\partial M} \underbrace{\mu_2^2(\kappa_1(p), \kappa_2(p))}_{=K}\, dp \right)}_{=4\pi} \frac{\epsilon^2}{3}. \end{aligned}$$

6 What All This Machinery Tells Us

We began this section by considering the question of what additive, continuous, rigid motion invariant measurements there are for convex bodies in $\mathbb{R}^n$ and learned that the $n+1$ intrinsic volumes are the only ones and any such measure must be a linear combination of these. We have also seen that the intrinsic volumes in a natural way extend the idea of curvature integrals over the boundary of a smooth body to general convex bodies without regard to a differentiable structure. These considerations become one possible basis on which to claim that integrals of Gaussian curvature on a triangle mesh become sums over excess angle at vertices and that integrals of mean curvature can be identified with sums over edges of dihedral angle weighted by edge length. These quantities are always *integrals*. Consequently *they do not make sense as pointwise quantities*. In the case of smooth geometry we can define quantities such as mean and Gaussian curvature as pointwise quantities. On a simplicial mesh they are only defined as integral quantities.

All this machinery was developed for convex bodies. If a given mesh is not convex the additivity property allows us to compute the quantities no less by writing the mesh as a finite union and intersection of convex bodies and then tracking the corresponding sums and differences of measures. For example, $V(K_\epsilon)$ is well defined for an individual triangle K and we know how to identify the coefficients involving intrinsic volumes with the integrals of elementary polynomials in the principal curvatures. Glueing two triangles together we can perform a similar identification carefully teasing apart the intrinsic volumes of the union of the two triangles. In this way the convexity requirement is relaxed so long as the shape of interest can be decomposed into a finite union of convex bodies.

This machinery was used by Cohen-Steiner and Morvan to give formulas for integrals of a discrete curvature tensor. We give these here together with some fairly straightforward intuition regarding the underlying geometry.

Let P be a polyhedron with vertex set V and edge set E and B a ball in $\mathbb{R}^3$ then we can define integrated Gaussian and mean curvature measures as

$$\phi_P^G(B) = \sum_{v \in V \cap B} K_v \quad \text{and} \quad \phi_P^H(B) = \sum_{e \in E} l(e \cap B)\theta_e,$$

where $K_v = 2\pi - \sum_j \alpha_j$ is the excess angle sum at vertex v defined through all the incident triangle angles at v, while $l(.)$ denotes the length and θ_e is the signed dihedral angle at e made between the incident triangle normals. Its sign is positive for convex edges and negative for concave edges (note that this requires an orientation on the polyhedron). In essence this is simply a restatement of the Steiner polynomial coefficients restricted to the intersection of the ball B and the polyhedron P. To talk about the second fundamental form II_p at some point p in the surface, it is convenient to first extend it to all of $\mathbb{R}^3$. This is done by setting it to zero if one of its arguments is parallel to the normal p. With this one may define

$$\bar{II}_P(B) = \sum_{e \in E} l(e \cap B)\theta_e e_n \otimes e_n, \quad e_n = e/\|e\|.$$

The dyad $e_n \otimes e_n(u, v) = \langle u, e_n \rangle \langle v, e_n \rangle$ projects given vectors u and v along the normalized edge. What is the geometric interpretation of the summands? Consider a single edge and the associated dyad. The curvature along this edge is zero while it is θ orthogonal to the edge. A vector aligned with the edge is mapped to θ_e while one orthogonal to the edge is mapped to zero. These are the principal curvatures *except they are reversed*. Hence $\bar{II}_P(B)$ is an integral measure of the curvature tensor *with the principal curvature values exchanged*. For example we can assign each vertex a three by three tensor by summing the edge terms for each incident edge. As a tangent plane at the vertex, which we need to project the three by three tensor to the expected two by two tensor in the tangent plane, we may take a vector parallel to the area gradient at the vertex. Alternatively we could defined $\bar{II}_P(B)$ for balls containing a single triangle and its three edges each. In that case the natural choice for the tangent plane is the support plane of the triangle. In practice one often finds that noise in the mesh vertex positions makes these discrete computations noisy. It is then a simple matter of enlarging B to stabilize the computations.

Cohen-Steiner and Morvan show that this definition can be rigorously derived from considering the coefficients of the Steiner polynomial in particular in the presence of non-convexities (which requires some fancy footwork...). They also show that if the polyhedron is a sufficiently fine sample of a smooth surface the discrete curvature tensor integrals have linear precision with regards to continuous curvature tensor integrals. They also provide a formula for a discrete curvature tensor which does not have the principal curvatures swapped.

7 Further Reading

The material in this section only gives the rough outlines of what is a very fundamental theory in probability and geometric measure theory. In particular there are many other consequences which follow from relationships between intrinsic volumes which we have not touched upon. A rigorous derivation of the results of Hadwiger, but much shorter than the original can be found in [Klain 1995]. A complete and rigorous account of the derivation of intrinsic volumes from first principles in geometric probability can be found in the short book by Klain and Rota [Klain and Rota 1997], while the details of the discrete curvature tensor integrals can be found in [Cohen-Steiner and Morvan 2003]. Approximation results which discuss the accuracy of these measure vis-a-vis an underlying smooth surface are treated by Cohen-Steiner and Morvan in a series of tech reports available at http://www-sop.inria.fr/geometrica/publications/.

Acknowledgments This work was supported in part by NSF (DMS-0220905, DMS-0138458, ACI-0219979), DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Alias, and Pixar.

References

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted Delaunay Triangulations and Normal Cycle. In *Proceedings of the 19th Annual Symposium on Computational Geometry*, 312–321.

KLAIN, D. A., AND ROTA, G.-C. 1997. *Introduction to Geometric Probability*. Cambridge University Press.

KLAIN, D. A. 1995. A Short Proof of Hadwiger's Characterization Theorem. *Mathematika 42*, 84, 329–339.

Chapter 3:
Curvature Measures for Discrete Surfaces

John M. Sullivan
sullivan@math.tu-berlin.de
Institut für Mathematik, TU Berlin MA 3–2
Str. des 17. Juni 136, 10623 Berlin

The curvatures of a smooth curve or surface are local measures of its shape. Here we consider analogous measures for discrete curves and surfaces, meaning polygonal curves and triangulated polyhedral surfaces. We find that the most useful analogs are those which preserve integral relations for curvature, like the Gauß–Bonnet theorem. For simplicity, we usually restrict our attention to curves and surfaces in euclidean space $\mathbb{R}^3$, although many of the results would easily generalize to other ambient manifolds of arbitrary dimension.

These notes are based on work by many people, but unfortunately do not include proper citations to the literature.

1 Smooth Curves and Surfaces

Before discussing discrete analogs, we briefly review the usual theory of curvatures for smooth curves and surfaces in space.

1.1 Smooth curves

The curvatures of a smooth curve γ are the local properties of its shape invariant under Euclidean motions. The only first-order information is the tangent line; since all lines in space are equivalent, there are no first-order invariants. Second-order information is given by the osculating circle, and the invariant is its curvature $\kappa = 1/r$.

For a plane curve given as a graph $y = f(x)$ let us contrast the notions of curvature and second derivative. At a point p on the curve, we can find either one by translating p to the origin, transforming so the curve is horizontal there, and then comparing to a standard set of reference curves. The difference is that for curvature, the transformation is a Euclidean rotation, while for second derivative, it is a shear $(x, y) \mapsto (x, y - ax)$. A parabola has constant second derivative f'' because it looks the same at any two points after a shear. A circle, on the other hand, has constant curvature because it looks the same at any two points after a rotation.

A plane curve is completely determined (up to rigid motion) by its (signed) curvature $\kappa(s)$ as a function of arclength s. For a space curve, however, we need to look at the third-order invariants, which are the torsion τ and the derivative κ' (which of course gives no new information). These are now a complete set of invariants: a space curve is determined by $\kappa(s)$ and $\tau(s)$. (Generally, for higher codimension, higher-order invariants are needed. For curves in $\mathbb{R}^n$, we need $n - 1$ curvatures, of order up to n, to characterize the shape.)

A smooth space curve γ is often described by its orthonormal *Frenet frame* (T, N, B). With respect to an arclength parameter s, the defining equations are $T := \gamma'$ and

$$\begin{pmatrix} T \\ N \\ B \end{pmatrix}' = \begin{pmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{pmatrix} \begin{pmatrix} T \\ N \\ B \end{pmatrix}.$$

For a curve γ lying on a surface M, it is often more useful to consider the *Darboux frame* (T, η, ν), adapted to this situation. This orthonormal frame includes the tangent vector T to γ and the normal vector ν to M. Its third element is thus $\eta := \nu \times T$, called the *cornormal*. The curvature vector of γ decomposes into parts tangent and normal to M as $T' = \kappa N = \kappa_g \eta + \kappa_n \nu$. Here in fact, κ_n measures the normal curvature of M in the direction T, and is independent of γ.

1.2 Smooth surfaces

Given a (two-dimensional, oriented) surface M (immersed) in $\mathbb{R}^3$, we understand its local shape by looking at the Gauß map $\nu : M \to \mathbb{S}^2$ given by the unit normal vector $\nu = \nu_p$ at each point $p \in M$. We can view its derivative at p as a linear endomorphism $-S_p : T_pM \to T_pM$, since T_pM and $T_{\nu_p}\mathbb{S}^2$ are naturally identified, being parallel planes in $\mathbb{R}^3$. The map S_p is called the shape operator (or Weingarten map).

The shape operator is the second-order invariant (or curvature) which completely determines the original surface M. However, it is usually more convenient to work with scalar quantities. The eigenvalues κ_1 and κ_2 of S_p are called principal curvatures, and (since they cannot be globally distinguished) it is their symmetric functions which have geometric meaning.

We define the *Gauß curvature* $K := \kappa_1\kappa_2$ as the determinant of S_p and the *mean curvature* $H := \kappa_1 + \kappa_2$ as its trace. Note that the sign of H depends on the choice of unit normal ν, and so often it is more natural to work with the *vector mean curvature* (or mean curvature vector) $\mathbf{H} := H\nu$. Note furthermore that some authors use the opposite sign on S_p and thus H, and many use $H = (\kappa_1 + \kappa_2)/2$, justifying the name *mean* curvature. Our conventions mean that the mean curvature vector for a convex surface points inwards (like the curvature vector for a circle). For a unit sphere oriented with inward normal, the Gauß map ν is the antipodal map, $S_p = I$, and $H = 2$.

The Gauß curvature is an intrinsic notion, depending only on the pullback metric on the surface M, and not on the immersion into space. That is, K is unchanged by bending the surface without stretching it. For instance, a developable surface like a cylinder or cone has $K = 0$ because it is obtained by bending a flat plane. One intrinsic definition of $K(p)$ is obtained by considering the circumferences C_ε of (intrinsic) ε-balls around p. We get

$$\frac{C_\varepsilon}{2\pi\varepsilon} = 1 - \frac{\varepsilon^2}{6}K + \mathcal{O}(\varepsilon^3).$$

Mean curvature is certainly not intrinsic, but it has a nice variational interpretation. Consider a variation vectorfield V on M, compactly supported away from any boundary. Then $H = -\delta \text{ Area} / \delta \text{ Vol}$ in the sense that

$$\delta_V \text{ Vol} = \int V \cdot \nu \, dA, \quad \delta_V \text{ Area} = -\int V \cdot H\nu \, dA.$$

With respect to the L^2 inner product $\langle U, V \rangle := \int U_p \cdot V_p \, dA$ on vectorfields, the vector mean curvature is the negative gradient of the area functional, often called the first variation of area: $\mathbf{H} = -\nabla \text{ Area}$. (Similarly, the negative gradient of length for a curve is κN.)

Just as κ is the geometric version of second derivative for curves, mean curvature is the geometric version of the Laplacian Δ. Indeed, if a surface M is written locally as the graph of a height function f over its tangent plane T_pM then $H(p) = \Delta f$. Alternatively, we can write $\mathbf{H} = \nabla_M \cdot \nu = \Delta_M \mathbf{x}$, where $\mathbf{x}$ is the position vector in $\mathbb{R}^3$ and Δ_M is Beltrami's surface Laplacian.

If we flow a curve or surface to reduce its length or area, by following these gradients κN and $H\nu$, the resulting parabolic heat flow is slightly nonlinear in a natural geometric way. This so-called *mean-curvature flow* has been extensively studied as a geometric smoothing flow.

1.3 Integral curvature relations for curves

The *total curvature* of a curve is $\int \kappa \, ds$. For closed curves, the total curvature is at least 2π (Fenchel) and for knotted space curves the total curvature is at least 4π (Fáry/Milnor). For plane curves, we can consider instead the signed curvature, and find that $\int \kappa \, ds$ is always an integral multiple of 2π.

Suppose we define (following Milnor) the total curvature of a polygonal curve simply to be the sum of the turning angles at the vertices. Then all the theorems for smooth curves mentioned in the previous paragraph remain true for polygonal curves. Our goal, when defining curvatures for polyhedral surfaces, will be to again ensure that integral relations for these curvatures remain exactly true.

1.4 Integral curvature relations for surfaces

For surfaces, the integral curvature relations we want to consider relate area integrals over a region $D \subset M$ to arclength integrals over the boundary $\gamma = \partial D$. The Gauß–Bonnet theorem says, when D is a disk,

$$2\pi - \iint_D K \, dA = \oint_\gamma \kappa_g \, ds = \oint_\gamma T' \cdot \eta \, ds = -\oint \eta' \cdot \, d\mathbf{x},$$

where $d\mathbf{x} = T \, ds$ is the vector line element along γ. This implies that the total Gauß curvature of D depends only on a collar neighborhood of γ: if we make any modification to D supported away from the boundary, the total curvature is unchanged (as long as D remains topologically a disk). We will extend the notion of Gauß curvature from smooth surfaces to more general surfaces (in particular polyhedral surfaces) by requiring this property to remain true.

The other relations are all proved by Stokes' Theorem, and thus only depend on γ being the boundary of D in a homological sense; for these D is not required to be a disk. First consider the vector area

$$\mathbf{A}_\gamma := \tfrac{1}{2} \oint_\gamma \mathbf{x} \times d\mathbf{x} = \iint_D \nu \, dA.$$

The right-hand side represents the total vector area of any surface spanning γ, and the relation shows this to depend only on γ (and this time not even on a collar neighborhood). The integrand on the left-hand side depends on a choice of origin for the coordinates, but because we integrate over a closed loop, the integral is independent of this choice. Both sides of this vector area formula can be interpreted directly for a polyhedral surface, and the equation remains true in that case.

A simple integral for curve γ from p to q says that

$$T(q) - T(p) = \int_p^q T'(s) \, ds = \int \kappa N \, ds.$$

This can be viewed as a balance between tension forces trying to shrink the curve, and sideways forces holding it in place. It is the relation used in proving that κ is the first variation of length.

The analog for a surface patch D is the mean curvature force balance equation

$$\oint_\gamma \eta \, ds = -\oint_\gamma \nu \times d\mathbf{x} = \iint_D H\nu \, dA = \iint_D \mathbf{H} \, dA.$$

Again this represents a balance between surface tension forces acting in the conormal direction along the boundary of D and what can be considered as pressure forces (espcially in the case of constant H) acting normally across D. We will use this equation to develop the analog of mean curvature for discrete surfaces.

Two other similar relations that we will not need later are the torque balance

$$\oint_\gamma \mathbf{x} \times \eta \, ds = \oint_\gamma \mathbf{x} \times (\nu \times d\mathbf{x}) = \iint_D H(\mathbf{x} \times \nu) \, dA$$

and the area relation

$$\oint_\gamma \mathbf{x} \cdot \eta \, ds = \oint_\gamma \mathbf{x} \cdot (\nu \times d\mathbf{x}) = \iint_D (\mathbf{H} \cdot \mathbf{x} - 2) \, dA.$$

2 Discrete Surfaces

For us, a discrete or polyhedral surface $M \subset \mathbb{R}^3$ will mean a triangulated surface with a PL map into space. In more detail, we start with an abstract combinatorial triangulation—a simplicial complex—representing a 2-manifold with boundary. We then pick positions $p \in \mathbb{R}^3$ for every vertex, which uniquely determine a linear map on each triangle; these fit together to form the PL map.

2.1 Gauß curvature

It is well known how the notion of Gauß curvature extends to such discrete surfaces M. Any two adjacent triangles (or, more generally, any simply connected region in M not including any vertices) can be flattened—developed isometrically into the plane. Thus the Gauß curvature is supported on the vertices $p \in M$. In fact, to keep the Gauß–Bonnet theorem true, we must take

$$\iint_D K \, dA := \sum_{p \in D} K_p; \qquad K_p := 2\pi - \sum_i \theta_i.$$

Here, the angles θ_i are the interior angles at p of the triangles meeting there, and K_p is often known as the angle defect

at p. If D is any neighborhood of p contained in $\mathrm{Star}(p)$, then $\oint_{\partial D} \eta\, ds = \sum \theta_i$; when the triangles are acute, this is most easily seen by letting ∂D be the path connecting their circumcenters and crossing each edge perpendicularly.

(Similar arguments lead to a notion of Gauß curvature—defined as a measure—for any rectifiable surface. For our polyhedral surface, this measure consists of point masses at vertices. Surfaces can also be built from intrinsically flat pieces joined along curved edges. Their Gauß curvature is spread out with a linear density along these edges. This technique is often used in designing clothes, where corners would be undesirable.)

Note that K_p is clearly an intrinsic notion, as it should be, depending only on the angles of each triangle and not on the precise embedding into $\mathbb{R}^3$. Sometimes it is useful to have a notion of combinatorial curvature, independent of all geometric information. Given just a combinatorial triangulation, we can pretend that each triangle is equilateral with angles $\theta = 60^\circ$, whether or not that geometry could be embedded in space. The resulting *combinatorial curvature* is $K_p = \frac{\pi}{3}(6 - \deg p)$. In this context, the global form $\sum K_p = 2\pi\chi(M)$ of Gauß–Bonnet amounts to nothing more than the definition of the Euler characteristic χ.

2.2 Vector area

The vector area formula

$$\mathbf{A}_\gamma := \tfrac{1}{2} \oint_\gamma \mathbf{x} \times d\mathbf{x} = \iint_D \nu\, dA$$

needs no special interpretation for discrete surfaces: both sides of the equation make sense directly, since the surface normal ν is well-defined almost everywhere. However, it is worth interpreting this formula for the case when D is the star of a vertex p. More generally, suppose γ is any closed curve (smooth or polygonal), and D is the cone from p to γ (the union of all line segments pq for $q \in \gamma$). Fixing γ and letting p vary, we find that the volume enclosed by this cone is a linear function of p, and $\mathbf{A}_p := \nabla_p \operatorname{Vol} D = \mathbf{A}/3 = \frac{1}{6} \oint_\gamma \mathbf{x} \times d\mathbf{x}$. We also note that any such cone D is intrinsically flat except at the cone point p, and that $2\pi - K_p$ is the cone angle at p.

2.3 Mean curvature

The mean curvature of a discrete surface M is supported along the edges. If e is an edge, and $e \subset D \subset \mathrm{Star}(e) = T_1 \cup T_2$, then

$$\mathbf{H}_e := \iint_D \mathbf{H}\, dA = \oint_{\partial D} \eta\, ds = e \times \nu_1 - e \times \nu_2 = J_1 e - J_2 e.$$

Here ν_i is the normal vector to the triangle T_i, and J_i is rotation by 90° in the plane of that triangle. Note that $|\mathbf{H}_e| = 2|e| \sin \frac{\theta_e}{2}$ where θ_e is the exterior dihedral angle along the edge, defined by $\cos\theta_e = \nu_1 \cdot \nu_2$.

No nonplanar discrete surface has $\mathbf{H}_e = 0$ along every edge. But this discrete mean curvature can cancel out around the vertices. We set

$$2\mathbf{H}_p := \sum_{e \ni p} \mathbf{H}_e = \iint_{\mathrm{Star}(p)} \mathbf{H}\, dA = \oint_{\mathrm{Link}(p)} \eta\, ds.$$

The area of the discrete surface is a function of the vertex positions; if we vary only one vertex p, we find that $\nabla_p \operatorname{Area}(M) = -\mathbf{H}_p$.

Suppose that vertices adjacent to p are $p_1, \ldots, p_n$. Then we have

$$\begin{aligned} 3\mathbf{A}_p = 3\nabla_p \operatorname{Vol} &= \iint_{\mathrm{Star}\, p} \nu\, dA \\ &= \tfrac{1}{2} \oint_{\mathrm{Link}\, p} \mathbf{x} \times d\mathbf{x} = \tfrac{1}{2} \sum_i p_i \times p_{i+1} \end{aligned}$$

and similarly

$$\begin{aligned} 2\mathbf{H}_p = \sum \mathbf{H}_{pp_i} = -2\nabla_p \operatorname{Area} &= \sum J_i(p_{i+1} - p_i) \\ &= \sum_i (\cot\alpha_i + \cot\beta_i)(p - p_i), \end{aligned}$$

where α_i and β_i are the angles opposite edge pp_i in the two incident triangles.

Note that if we change the combinatorics of a discrete surface M by introducing a new vertex p along an existing edge e, and subdividing the two incident triangles, then $\mathbf{H}_p$ in the new surface equals the original $\mathbf{H}_e$, independent of where along e we place p. This allows a variational interpretation of $\mathbf{H}_e$.

2.4 Minkowski mixed volumes

A somewhat different interpretation of mean curvature for convex polyhedra is suggested by Minkowski's theory of mixed volumes (which actually dates in this form well earlier). If X is a smooth convex body in $\mathbb{R}^3$ and $B_t(X)$ denotes its t-neighborhood, then

$$\operatorname{Vol}(B_t(X)) = \operatorname{Vol} X + t \operatorname{Area} X + \frac{t^2}{2} \int_X H\, dA + \frac{t^3}{3} \int_X K\, dA.$$

Here, the last integral is always 4π.

When X is instead a convex polyhedron, the only term that needs a new interpretation is $\int_X H\, dA$. The correct replacement for this term is then $\sum_e |e|\, \theta_e$. This suggests $H_e := |e|\, \theta_e$ as a notion of total mean curvature for the edge e.

We note the difference between this formula and our earlier $|\mathbf{H}_e| = 2|e| \sin\theta_e/2$. Either one can be derived by replacing the edge e with a sector of a cylinder of length $|e|$ and arbitrary (small) radius r. We find then

$$\iint \mathbf{H}\, dA = \mathbf{H}_e, \qquad \iint H\, dA = H_e.$$

The difference is explained by the fact that one formula integrates the scalar mean curvature while the other integrates the vector mean curvature.

2.5 CMC surfaces and Willmore surfaces

A smooth surface which minimizes area under a volume constraint has constant mean curvature; the constant H can be understood as the Lagrange multiplier for the constrained minimization problem. A discrete surface which minimizes area among surfaces of fixed combinatorial type and fixed volume will have constant discrete mean curvature H in the sense that at every vertex, $\mathbf{H}_p = H\mathbf{A}_p$, or equivalently $\nabla_p \operatorname{Area} = -H\nabla_p \operatorname{Vol}$. In general, of course, the vectors $\mathbf{H}_p$ and $\mathbf{A}_p$ are not even parallel: they give two competing notions of a normal vector at p.

Still,

$$h_p := \frac{|\nabla_p \operatorname{Area}|}{|\nabla_p \operatorname{Vol}|} = \frac{|\mathbf{H}_p|}{|\mathbf{A}_p|} = \frac{|\iint_{\operatorname{Star} p} \mathbf{H}\, dA|}{|\iint_{\operatorname{Star} p} \nu\, dA|}$$

gives a better notion of mean curvature near p than, say, the smaller quantity $3|\mathbf{H}_p|/\operatorname{Area}(\operatorname{Star}(p)) = |\iint \mathbf{H}\, dA|/\iint 1\, dA$.

For this reason, a good discretization of the Willmore elastic energy $\iint H^2\, dA$ is given by $\sum_p h_p^2 \frac{1}{3} \operatorname{Area}(\operatorname{Star}(p))$.

2.6 Relation to discrete harmonic maps

Discrete minimal surfaces minimize area, but also have other properties similar to those of smooth minimal surfaces. For instance, in a conformal parameterization, their coordinate functions are harmonic. We don't know when in general a discrete map should be considered conformal, but the identity map is certainly conformal. We have that M is discrete minimal if and only if $\mathrm{Id} : M \to \mathbb{R}^3$ is discrete harmonic. Here a PL map $f : M \to N$ is called discrete harmonic if it is a critical point for the Dirichlet energy $E(f) := \sum_T |\nabla f_T|^2 \operatorname{Area}_M(T)$. We find that $E(f) - \operatorname{Area} N$ is a measure of nonconformality. For the identity map, $E(\mathrm{Id}_M) = \operatorname{Area}(M)$ and $\nabla_p E(\mathrm{Id}_M) = \nabla_p \operatorname{Area}(M)$ confirming that M is minimal if and only if Id_M is harmonic.

Chapter 4:
A Discrete Model of Thin Shells

Eitan Grinspun
Columbia University

Figure 1: *A simulation of a tumbling thin shell.*

Abstract *We present a discrete model for the behavior of thin flexible structures, such as hats, leaves, and aluminum cans, which are characterized by a curved undeformed configuration. Previously such models required complex continuum mechanics formulations and correspondingly complex algorithms. We show that a simple shell model can be derived geometrically for triangle meshes and implemented quickly by modifying a standard cloth simulator. Our technique convincingly simulates a variety of curved objects with materials ranging from paper to metal, as we demonstrate with several examples including a comparison of a real and simulated falling hat.*

This chapter is based on the paper by Grinspun, Hirani, Desbrun, and Schröder which appeared in the Proceedings of the Symposium for Computer Animation 2003 [Grinspun et al. 2003].

1 Introduction

Thin shells are thin flexible structures with a high ratio of width to thickness (> 100) [Ciarlet 2000]. While their well-known counterparts, thin *plates*, relax to a *flat* shape when unstressed, thin *shells* are characterized by a *curved undeformed configuration*. Cloth, recently studied in the computer animation literature, may be modeled as a thin plate, since garments are typically constructed from flat textiles. In stark contrast, thin-walled objects which are naturally curved (*e.g.*, leaves, fingernails), or put into that shape through plastic deformation (*e.g.*, hats, cans, carton boxes, pans, car bodies, detergent bottles) are thin shells and cannot be modeled using plate formulations.

Thin shells are remarkably difficult to simulate. Because of their degeneracy in one dimension, shells do not admit to straightforward tessellation and treatment as three-dimensional solids; indeed, the numerics of such approaches become catastrophically ill-conditioned, foiling numerical convergence and/or accuracy. Robust finite element methods for thin shell equations continue to be an active and challenging research area.

In this chapter we develop a simple model for thin shells with applications to computer animation. Our *discrete model* of shells captures the same characteristic behaviors as more complex models, with a surprisingly simple implementation. We demonstrate the realism of our approach through various examples including comparisons with real world footage (see Figure 3).

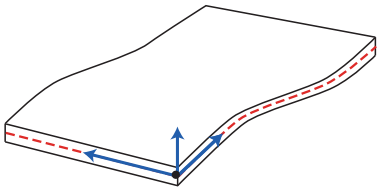


Figure 2: *The local coordinate frame in a small neighborhood of a thin shell: two axes span the* middle surface*, and the normal* shell director *spans the thickness.*

2 Kinematics

We begin by describing the geometry of a thin shell, as well as its *kinematics*—the geometric description of a shell's deformation. Since it is thin, the shell is well described by its *middle-surface* (see Figure 2). At any point on the middle surface the local tangent plane and surface normal induce a coordinate frame in which to describe "motion along the surface" and "motion along thickness."

In the discrete setting, we describe the middle surface with an arbitrary 2-manifold triangle mesh, $M = \{v, e, f\}$, where v, e, f are sets of vertices, edges and faces repectively. We denote a specific mesh vertex by v_i, $i \in \{1, 2, \dots n\}$, likewise for edges, e_j, and faces, f_k. Consider that the embedding of the mesh into ambient space can uniquely be specified by the discrete *configuration function*, $C(v_i) : v \to \mathbb{R}^3$, mapping every vertex into ambient space (here chosen as $\mathbb{R}^3$, without loss of generality). Such a discrete function may be represented as an element of the *configuration space* $\Omega = (\mathbb{R}^3)^n$, *i.e.*, as a vector, $x = [x_1, x_2, \dots x_n]$, $x_i \in \mathbb{R}^3$, of points in ambient space corresponding to the positions of mesh vertices, v_i. Furthermore, we will use the correspondence between subsets of the mesh and lower-dimensional subspaces of the configuration space, *e.g.*, consider that the embedding of a single triangle, f_i, is specified uniquely by the embedding of its three vertices, hence this subset of $\{f_i\} \subset M$ has a corresponding configuration subspace $C(f_i) \in (\mathbb{R}^3)^3$ where, in an abuse of notation, we say that the configuration function, C, maps subsets of the mesh onto lower-dimensional subspaces of configuration space.

With a shell representation in hand, we want to quantify the ways in which a shell deforms: the geometry of deformation is called the shell *kinematics*. Which kind of quantities make *physical* sense? Observe that coordinate systems are an imposed artifact: the physical behavior of our world is independent of the chosen (global or local) coordinate system. Therefore, in building a physical model we should measure

the mesh only in ways which are invariant to changes in coordinates, *i.e.*, to rigid body motions. We have seen in the previous chapters (in particular Chapter 2) that such considerations lead to a limited palette of fundamental measures for which (by some aggregation) any compound measure is constructed. In our setting, we will consider only measurements which are functions of edge length, face area, and signed dihedral angle.

Suppose that the mesh is deformed from its original configuration. We can express this as a piecewise-affine *deformation map*, $\varphi : \Omega \to \Omega$, from the configuration space to the configuration space, mapping every mesh face (resp. edge, vertex) from its undeformed to its deformed configuration. In general, this map encodes both local changes in shape as well as a global translation and rotation, *i.e.*, both *shape* and *rigid-body* deformations. A natural question is "how much has the geometry deformed?" Consider that we are modeling the mechanics of materials whose stored energy is *local* in nature, such that the deformation in one section of the surface only affects the stored energy associated to that section. Thus we ask "how much has this very local piece of geometry deformed?"

Consider a small, localized region, $M_x \subset M$, of the surface, and the associated subset $\Omega_x \subset \Omega$ of configuration space. We want a function $s(x_0, x_1) : \Omega_x \times \Omega_x \to \mathbb{R}$, quantifying the amount of change moving from configuration x_0 to x_1. In particular, we define the local generalized strain (here onward "strain") as $s(x_0, \varphi(x_0))$, *i.e.*, a (signed) amount of deformation from the undeformed to the deformed configuration. Strain should be a local measure of change in shape, and *not* a measure of rigid-body deformation. Furthermore, it might make sense for strain to be antisymmetric in its arguments: $s(x_0, x_1) = -s(x_1, x_0)$, *e.g.*, the strain of a locally curved sheet deformed to a locally flattened shape is the opposite of that of a locally flat sheet deformed to a locally curved shape (note however in general neither x_0 nor x_1 need represent a flat configuration).

We express our strain in terms of the fundamental measures evaluated (separately) over the undeformed and deformed configurations (and then aggregated in some antisymmetric way). As a consequence our strain formulation will be invariant under rigid body motions. Strain must be zero if there is no change in shape, and should grow with the change in shape. A simple functional form which achieves this is the difference between evaluations of a fundamental measure over the undeformed and deformed configurations. We adopt the simplest such terms: the difference in edge length, s^e, evaluated over a single edge; the difference in area, s^f, evaluate over a single face; and the difference in dihedral angles, s^θ, evaluated over a single edge incident on two faces. While these are perhaps the simplest possible constituent terms of a shape deformation metric, other more complicated options can be very interesting. Recent research in discrete shell models has focused on functions evaluated over mesh faces which aggregate in one term the configuration of all the incident edge lengths and dihedral angles [Gingold et al. 2004].

Observe that s^e and s^f are invariant under *locally*-isometric deformations, *i.e.*, the deformation may be arbitrary but in the neighborhood of the edge or face in question the deformation is length-preserving. Analogously, s^θ is invariant under deformations which locally-preserve discrete mean curvature. Furthermore, s^f is invariant under deformations which locally-preserve area. Consequently, we say that s^e and s^f "see" the stretching but not the bending component of a deformations, while s^θ sees only bending but not stretching component. Finally, s^f sees the *area-preserving* (considering shell thickness, also volume preserving) component of a stretching deformation, whereas s^e sees all of the stretching deformation. With these deformation pseudometrics in hand, we now proceed in proposing a constitutive elastic model for the stored energy.

3 Constitutive Model

Having defined the *geometry* of thin shells, we turn our attention to the governing *physical* equations. The *stored elastic energy* of a thin shell is at the heart of the equations which govern its response to elastic deformations. The stored energy, $W(x_0, x_1) = f(s(x_0, x_1))$, is a function of the local strain. It is defined over small mesh neighborhoods via the evaluation, $W(x_0, \varphi(x_0))$ of a pseudometric, $W(x_0, x_1) : \Omega_x \times \Omega_x \to \mathbb{R}$. Recall that a pseudometric is a nonnegative function that measures the distance between points for a given set, in this case a configuration subspace; a metric is symmetric, satisfies the triangle inequality, and evaluates to zero for identical points; a *pseudo*metric may *also* evaluate to zero for *non*identical points. Here the distinction of *pseudo* is physically significant: from a given initial configuration, there is more than one deformation that yields the same (local) stored energy: consider, *e.g.*, taking a sheet of paper and bending it up versus bending it down. Since the stored energy is a function of *local* shape deformation, *i.e.*, energy contributions are directly attributable to some portion of the mesh, then $W(x_0, x_1)$ should satisfy the usual inclusion/exclusion principle: given two mesh subsets, $M_x, M_y \subset M$, we require that $W(M_x \cup M_y) = W(M_x) + W(M_y) - W(M_x \cap M_y)$, where $W(M_x)$ is shorthand for the energy, $W(C(M_x), \varphi(C(M_x)))$, associated to the deformation of mesh subset M_x (we first saw the inclusion/exclusion principle in Chapter 2). Because strain is invariant under rigid body transformations of the undeformed and/or deformed configurations, a key theorem by Nöther guarantees that the internal forces derived from the stored energy obey the conservation laws for (linear as well as angular) momentum.

We choose the simplest expression for energy that is consistent with Hookean mechanics. In 1676 Robert Hooke stated

> The power [*sic.*] of any springy body is in the same proportion with the extension.

This statement was the birth of modern elasticity, which states that a first order approximation for the response of a material is a force proportional to strain, and consequently (by the definition of work as force over distance) that the first approximation of stored energy is quadratic in strain. We propose:

$$W(M_x) = \sum_{e_i \in M_x} W^e(e_i) + \sum_{f_j \in M_x} W^f(f_i) + \sum_{e_k \in M_x} W^\theta(e_k) \ ,$$

$$\begin{aligned} W^e(e_i) &= k_i^e (s^e(e_i))^2 \\ &= k_i^e (\text{change in length of edge } e_i)^2 \ , \\ W^f(f_j) &= k_j^f (s^f(f_j))^2 \\ &= k_j^f (\text{change in area of face } f_j)^2 \ , \\ W^\theta(e_k) &= k_k^\theta (s^\theta(e_k))^2 \\ &= k_k^\theta (\text{change in dihedral angle at edge } e_k)^2 \ . \end{aligned}$$

using the same shorthand

$$\begin{aligned} W(M_x) &= W(C(M_x), \varphi(C(M_x))) \ , \\ s(M_x) &= s(C(M_x), \varphi(C(M_x))) \ . \end{aligned}$$

Our constitutive model for thin shells is governed by nonlinear[1] *membrane* and *flexural* energy terms. We examine these terms in turn, and discuss the coefficients k_e, k_f, and k_θ.

Membrane Elastic surfaces resist stretching (local change in area) and (local change in length).

While some materials such as rubber sheets may undergo significant deformations in the stretching or shearing (*membrane*) modes, we focus on inextensible shells which are characterized by nearly *isometric* deformations, i.e., possibly significant deformations in bending but unnoticeable deformation in the membrane modes. Works on cloth simulation similarly focus on inextensible plates [Baraff and Witkin 1998; Bridson et al. 2002]. Most membrane models for triangle meshes satisfy this small-membrane-strain assumption with choice of suitably large membrane stiffness coefficients.

The only missing link in our definitions of $W^e(e_i)$ and $W^f(f_j)$ are the coefficients k_i^e, k_j^f. We can learn more about these coefficients by considering the following scaling argument. Place in front of you three patches of the same elastic material, with the third patch having the combined area of the two other patches. Stretch all the patches by 30%. Then the energy stored in the third (large) patch should be the same as the combined energy stored in the other two patches (think of the third patch as two smaller patches, attached, each stretched by 30%). Substituting this logic into our stored energy formulation yields the requirement that k_j^f is inversely proportional to the area of the undeformed face f_j. With the same calculation we find that k_i^e must be independent of local area or length. Finally, both k_j^f and k_i^e should be proportional to the *stiffness* of the material, *i.e.*, its resistance to stretching deformations. Thus we have $k_i^e = K^e$ and $k_j^f = K^f/|\bar{f}_j|$, where K^e and K^f are the material parameters determining stiffness, and $|\bar{f}_j|$ is the area of face j. Barred quantities ($\bar{f}_j$) denote measurements taken in the undeformed configuration. Putting this all together the membrane terms are given by

$$\begin{aligned} W^e(e_i) &= K^e \left(|e_i| - |\bar{e}_i|\right)^2 = K^e |\bar{e}_i|^2 \left(\frac{|e_i|}{|\bar{e}_i|} - 1\right)^2 \ , \\ W^f(f_j) &= \frac{K^f}{|\bar{f}_j|} \left(|f_j| - |\bar{f}_j|\right)^2 = K^f |\bar{f}_j| \left(\frac{|f_j|}{|\bar{f}_j|} - 1\right)^2 \ , \end{aligned}$$

where $|e_i|$ is the length of edge i. This is a unitless strain measurement, squared, and then integrated over the area of the local neighborhood, and multiplied by the material-dependent parameters. Observe that under regular refinement of a triangle mesh, the local area indeed scales as $|\bar{f}_i|$ and as $|\bar{e}_j|^2$, both of which have units of area. The units of the material parameters are energy per unit area, *i.e.*, surface energy density. In engineering models of shells, the material parameter is given as a volume energy density, and the energy is integrated over shell thickness yielding a surface energy density. Note the opportunity to precompute quantities that depend only on the undeformed configuration, in this case $K^e|\bar{e}_i|^2$ and $K^f|\bar{f}_i|$.

[1]Observe that while the restoring forces are proportional to strain, strain is *not* linear in the displacements of the mesh vertices.

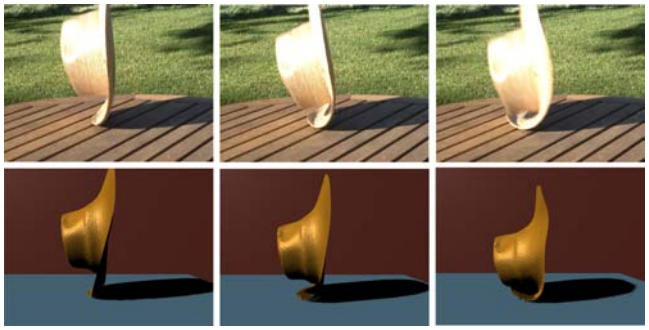


Figure 3: *Real footage vs. Simulation: left, a real hat is dropped on a table; right, our shell simulation captures the bending of the brim. Notice that volumetric-elasticity, plate, or cloth simulations could not capture this behavior, while earlier work on shell simulation required significant implementation and expertise (see also the color plate).*

So far we have only discussed energies that measure membrane (*intrinsic*) deformations. However, when a surface bends—an *extrinsic* [Gray 1998] deformation—*flexural energy* comes into play.

Flexure In this section we discuss our discrete bending energy in relation to its continuous analogues. Models in mechanics are based on invariant measures, *i.e.*, quantities which are not affected by rigid-body transformations of the coordinate frame. Typically, this has led to formulations based on tensors. For example, shell models use the difference of the second fundamental forms [Gray 1998] in the deformed and undeformed configurations (pulling back the deformed tensor onto the undeformed configuration). In the graphics literature, the work of Terzopoulos *et al.* introduced such tensorial treatments [Terzopoulos et al. 1987]. These treatments derive tensorial expressions over smooth manifolds, and as a final step discretize to carry out the numerics. In contrast, we define a *discrete* constitutive model by applying geometric operators over piecewise-linear surfaces. In both earlier treatments and our discrete treatment *the underlying geometry is the same.* However, the resulting expressions are simpler in the discrete approach.

The shape operator [Gray 1998] is the derivative of the Gauss map[2]: geometrically, it measures the local curvature at a point on a smooth surface. Our bending energy is an extrinsic measure of the difference between the shape operator evaluated on the deformed and undeformed surfaces. We express this difference as the *squared difference of mean curvature*:

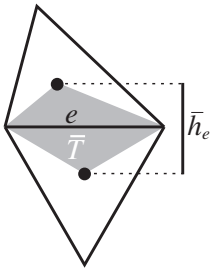
$$\left[\mathrm{Tr}(\varphi^* \mathrm{S}) - \mathrm{Tr}(\bar{\mathrm{S}})\right]^2 = 4(H \circ \varphi - \bar{H})^2 \ , \qquad (1)$$

where $\bar{\mathrm{S}}$ and S are the shape operators evaluated over the undeformed and deformed configurations respectively; likewise $\bar{H}$ and H are the mean curvatures; $\varphi^*\mathrm{S}$ is the pull-back of S onto $\bar{\Omega}$, and we use $\mathrm{Tr}(\varphi^* S) = \varphi^* \mathrm{Tr}(S) = \mathrm{Tr}(S) \circ \varphi = H \circ \varphi$ for a diffeomorphism φ. This measure is *extrinsic*: it sees only changes in the *embedding* of the surface in $\mathbb{R}^3$. This measure is *invariant under rigid-body transformations*: this ensures conservation of linear and angular momentum. Integrating (1) over the reference domain we find the continuous flexural energy $\int_{\bar{\Omega}} 4(H \circ \varphi - \bar{H})^2 d\bar{A}$. Next, we discretize this

[2]This is the map from the surface to the unit sphere, mapping each surface point to its unit surface normal.

integral over the *piecewise linear mesh* that represents the shell.

We derive the discrete, integral *mean-curvature squared* operator as follows. We first partition the undeformed surface into disjoint union of diamond-shaped tiles, $\bar{T}$, associated to each mesh edge, e, as indicated on the side figure (following [Meyer et al. 2003], one can use the barycenter of each triangle to define these regions—or alternatively, the circumcenters). Over such a diamond, the mean curvature integral is $\int_{\bar{T}} \bar{H} d\bar{A} = \bar{\theta}|\bar{e}|$ (for a proof see [Cohen-Steiner and Morvan 2003]). A similar argument leads to: $\int_{\bar{T}} (H \circ \varphi - \bar{H}) d\bar{A} = (\theta - \bar{\theta})|\bar{e}|$. Using the notion of area-averaged value from [Meyer et al. 2003], we deduce that $(H \circ \varphi - \bar{H})|_{\bar{T}} = (\theta - \bar{\theta})/\bar{h}_e$, where $\bar{h}_e$ is the span of the undeformed tile, which is one sixth of the sum of the heights of the two triangles sharing $\bar{e}$. For a sufficiently fine, non-degenerate tessellation approximating a smooth surface, the average over a tile (converging pointwise to its continuous counterpart) *squared* is equal to the squared average, leading to: $\int_{\bar{T}} (H \circ \varphi - \bar{H})^2 d\bar{A} = (\theta - \bar{\theta})^2 |\bar{e}|/\bar{h}_e$.

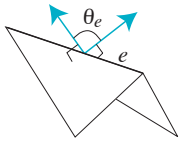


We might instead consider a formula of the form $(\theta - \bar{\theta})^2 |\bar{e}|$. Here the energy functional becomes dependent only on its piecewise planar geometry *not* on the underlying triangulation. An attractive claim, this is appealing in that a material's physical energy should depend on its shape, *not* on the discretization of the shape. Unfortunately, there is no discretization of (1) that simultaneously is (a) dependent only on the geometry *not* its triangulation, and (b) converges to its continuous equivalent under refinement. Indeed, the area integral of (1) is in general unbounded for a piecewise planar geometry! A discrete energy satisfying both (a) and (b) may exist for smoother surfaces, but our focus is piecewise planar (triangle mesh) geometry.

Following the argument found in [Meyer et al. 2003], there may be numerical advantages in using circumcenters instead of barycenters for the definition of the diamond tiles (except in triangles with obtuse angles). This affects the definition of $\bar{h}_e$ and of the lumped mass. Since we only need to compute these values for the undeformed shape, the implementation and performance of only initialization code would be affected. As noted in [Bobenko 2004], when circumcenters are used the derivation of discrete shells present here coincides, in the case of a flat undeformed configuration, with the derivation of the discrete Willmore energy based on circle packing (see Chapter 5).

As we have just seen, we can express our *discrete flexural energy* as a summation over mesh edges, where the term for edge e_k is

$$W^{\theta}(e_k) = K^{\theta} \left(\theta_k - \bar{\theta}_k\right)^2 \frac{|\bar{e_k}|}{\bar{h}_k} , \quad (2)$$



where θ_k and $\bar{\theta}_k$ are corresponding complements of the dihedral angle of edge e_k measured in the deformed and undeformed configuration respectively, K^{θ} is the material bending stiffness, and $\bar{h}_k$ is a third of the average of the heights of the two triangles incident to the edge e_k (see the appendix for another possible definition of $\bar{h}_k$). Note that the unit of K^{θ} is energy (not surface energy density). This formulation is consistent with the physical scaling laws of thin shells: if the (deformed and undeformed) geometry of a thin shell is uniformly scaled by λ along each axis, then surface area scales as λ^2 as does the total membrane energy, *however* the total bending energy is *invariant* under uniform scaling. This formulation of bending energy was contemporaneously published in [Bridson et al. 2003; Grinspun et al. 2003].

Following the reasoning for (1), we could have formed a second energy term taking the determinant instead of the trace of S. This would lead to a difference of Gaussian curvatures, but this is always zero under isometric deformations (pure bending). This is not surprising, as Gaussian curvature is an *intrinsic* quantity, *i.e.*, it is independent of the embedding of the two-dimensional surface into its ambient three-dimensional space. In contrast, flexural energy measures *extrinsic* deformations.

4 Dynamics

The treatment of the temporal evolution of a thin shell is beyond the scope of our discrete differential geometry course. In this section we briefly summarize the basic components required to simulate the motion of thin shells.

Our dynamic system is governed by the ordinary differential equation of motion $\ddot{\mathbf{x}} = -\mathbf{M}^{-1} \nabla W(\mathbf{x})$ where $\mathbf{x}$ is the vector of unknown DOFs (*i.e.*, the vertices of the deformed geometry) and $\mathbf{M}$ is the mass matrix. We use the conventional simplifying hypothesis that the mass distribution is lumped at vertices: the matrix M is then diagonal, and the mass assigned to a vertex is a third of the total area of the incident triangles, scaled by the area mass density.

Newmark Time Stepping We adopt the Newmark scheme [Newmark 1959] for ODE integration,

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{x}_i + \Delta t_i \dot{\mathbf{x}}_i + \Delta t_i^2 \left((1/2 - \beta)\ddot{\mathbf{x}}_i + \beta \ddot{\mathbf{x}}_{i+1}\right), \\ \dot{\mathbf{x}}_{i+1} &= \dot{\mathbf{x}}_i + \Delta t_i \left((1 - \gamma)\ddot{\mathbf{x}}_i + \gamma \ddot{\mathbf{x}}_{i+1}\right), \end{aligned}$$

where Δt_i is the duration of the i^{th} timestep, $\dot{\mathbf{x}}_i$ and $\ddot{\mathbf{x}}_i$ are configuration velocity and acceleration at the beginning of the i^{th} timestep, respectively, and β and γ are adjustable parameters linked to the accuracy and stability of the time scheme. Newmark is either an explicit ($\beta = 0$) or implicit ($\beta > 0$) integrator: we used $\beta = 1/4$ for final production, and $\beta = 0$ to aid in debugging. Newmark gives control over numerical damping via its second parameter γ. We obtained the best results by minimizing numerical damping ($\gamma = 1/2$); this matches Baraff and Witkin's observation that numerical damping causes undesirable effects to rigid body motions. See also [West et al. 2000] for a discussion of the numerical advantages of the Newmark scheme.

Dissipation Shells dissipate energy via flexural oscillations. To that end we complete our model with an *optional* damping force proportional to $(\dot{\theta} - \dot{\bar{\theta}})\nabla\theta$ where $\dot{\bar{\theta}} = 0$ for elastic deformations but is in general non-zero for plastoelastic deformations. This is consistent with standard derivations of Rayleigh-type damping forces using the strain rate tensor, as discussed by [Baraff and Witkin 1998].

Discussion Our proposed discrete flexural energy (2) generalizes on published energies for (*flat*) plates both continuous and discrete: (a) [Ge et al. 1996] presented a geometric argument that the stored energy of a continuous inextensible plate has the form $\int_{\Omega} c_H H^2 + c_K K dA$ for material-specific coefficients c_H and c_K; (b) [Haumann 1987] used a discrete hinge energy, similarly [Baraff and Witkin 1998] used a discrete constraint-based energy, of the form $W_B(\mathbf{x}) = \sum_{\bar{e}} \theta_e^2$.

Our approach generalizes both (a) and (b), and produces convincing simulations beyond the regime of thin plate and cloth models (see Section 6).

Our approach can also be viewed within the framework laid out by [Terzopoulos et al. 1987]: we focus on the second fundamental form, choose a computationally convenient and geometrically intuitive norm, and propose a simple, effective discretization.

Our novel formulation has three salient features: (a) the energy is invariant under rigid body transformation of both the undeformed and the deformed shape: our system conserves linear and angular momenta; (b) the piecewise nature of our geometry description is fully captured by the purely intrinsic membrane terms, and the purely extrinsic bending term; most importantly, (c) it is *simple* to implement.

5 Implementation

We encourage readers to implement this novel approach to simulating shells as follows: take working code for a thin plate or cloth simulator (*e.g.*, as presented by [Baraff and Witkin 1998]), and replace the bending energy with (2). From an implementation point of view, this involves minimal work. For example, consider that [Baraff and Witkin 1998] already implemented all the computations relating to θ_e. The key hurdle is that the undeformed configuration, $\bar{\mathbf{x}}$, is represented in $(x, y) \in \mathbb{R}^2$ coordinates, thus kinematically imposing a *flat* shape. One could augment this with explicitly-stored undeformed-angles, $\bar{\theta}_e$, but this would work only for developable surfaces. Any surface which cannot be unfolded into a flat sheet—a surface with intrinsic curvature, such as a hat or a car body—requires a more complete treatment than this. Instead, we express $\bar{\mathbf{x}}$ in coordinates $(x, y, z) \in \mathbb{R}^3$, *i.e.*, *not* restricting ourselves to planar undeformed configuration. Consequently, the undeformed configuration is in general curved, and we must duplicate the code that computes θ_e to also compute $\bar{\theta}_e$. Rereading Baraff and Witkin's paper with these changes in mind, it is immediately clear that these modification require just a few hours of work.

As part of ongoing and future research, our priorities in implementing our simulator are extendibility and ease of implementation. We have made several design choices to aid in numerical robustness and to avoid bugs in implementing our formulas:

Automatic Differentiation The use of an explicit integrator necessitates the evaluation of energy gradients, or forces, with respect to vertex DOFs. Formulae for the gradients of edge-length and area are easily found in the literature [Desbrun et al. 2002]; the gradient of the dihedral angle requires more work, but can still be derived by hand. Since our goal is to ease implementation and debugging of new, experimental energies, we chose to use an automatic differentiation (AD) technique.

The use of an *implicit* integrator necessitates evaluation of *force* gradients with respect to vertex DOFs, *i.e.*, we need formulae for *second* derivatives of energy. Deriving such formulae is cumbersome and error-prone, consequently we used AD, a technique for augmenting software with derivative computations [Corliss et al. 2001]. The technique is based on the observation that every computational algorithm can be written as the composition of simple, easily differentiable, steps to which the chain rule can be applied. AD is not new to graphics [Gleicher 1994; Kass 1992]. Our AD code is available at `http://multires.caltech.edu/software` .

The salient features of our AD implementation are: (a) it differentiates directly with respect to vector (not scalar) unknowns; (b) it uses C++ type-checking to ensure both efficiency and completeness of differentiation. Although there are several good AD libraries publicly available [ADIFOR 2002; Autodiff.org 2002], we opted for implementing this simple set of classes specially for differentiation with respect to vector variables.

We define two classes, Scalar and Vector, representing *independent* scalar and vector values respectively. The related classes DScalar and DVector represent *dependent* quantities; these carry a tuple *(scalar value, vector-valued derivative)* and *(vector value, matrix-valued derivative)* respectively. The standard algebraic operators are overloaded to inter-operate between the classes, with a special restriction on assignment: dependent quantities may not be assigned to independent variables, and vice-versa. This condition ensures both correctness (no dependent quantity is overlooked) and efficiency (independent quantities never compute/store derivatives). More documentation is given in the publicly-available release of our small AD library.

In a production code, we believe that hand-derived formulas would display better performance. As demonstrated in the works of [Baraff and Witkin 1998] and [Bridson et al. 2003], it is reasonable to explicitly take the derivatives by hand. Early in our investigation we converted our code to the automatic technique, in order to facilitate future exploration, and to learn more about the technique; although we did not compare timings of the hand- and automatic-techniques, in our research code they appeared to run at comparable speeds, and we opted for the convenience of AD: the actual performance degradation was well worth the guaranteed consistency between energy, forces, and force gradients for our research purposes.

6 Results

We exercised our implementation on three problems: fixed beams, falling hats, and pinned paper (see `http://multires.caltech.edu/pubs/DS-CDROM`). Computation time, on a 2GHz Pentium 4 CPU, ranged from 0.25s–3.0s per frame. In light of the discussion in Section 5, we expect an optimized implementation of our method to be as efficient as state-of-the-art cloth simulators.

Beams We pinned to a wall one end of a v-beam, and released it under gravity. Figure 4, and the video, demonstrate the effect of varying flexural stiffness on oscillation amplitude and frequency. Higher flexural stiffness gives higher structural rigidity. The curved undeformed shape of a v-beam gives qualitatively and quantitatively different behavior than a flat beam. Compare: hold a simple paper strip by its end; repeat after folding a v-shaped cross-section.

Elastic hats We dropped both real and virtual hats and compared (see Figure 3): the deformation is qualitatively the same, during impact, compression, and rebound. Adjusting the damping parameter, we capture or damp away the brim's vibrations. Adjusting the flexural stiffness, we can make a hat made of hard rubber or textile (see the videos of a nearly-rigid hat and a floppy hat).

Plastoelasticity As discussed in the early work of [Kergosien et al. 1994], a compelling simulation of paper would require a mechanical shell model. Using our simple shell model, we can easily simulate a sheet of paper that is rolled, then creased, then pinned (see Figure 5). Here the physics require *plastic* as well as elastic deformations. We begin

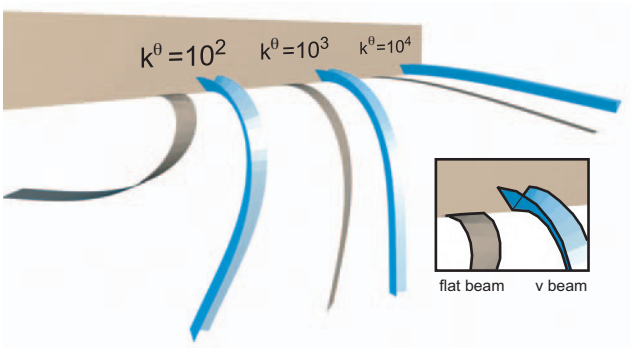


Figure 4: *Three pairs of flat and v-beams with increasing flexural stiffness K^θ (left to right) of 100, 1000, and 10000. The flexural energy coefficient has a high dynamic range; extreme values (from pure-membrane to near-rigid) remain numerically and physically well-behaved. Observe that increasing flexural stiffness augments structural rigidity. Compare the behavior of beams: the non-flat cross section of the v-beam contributes to structural rigidity, especially for low flexural stiffness.*

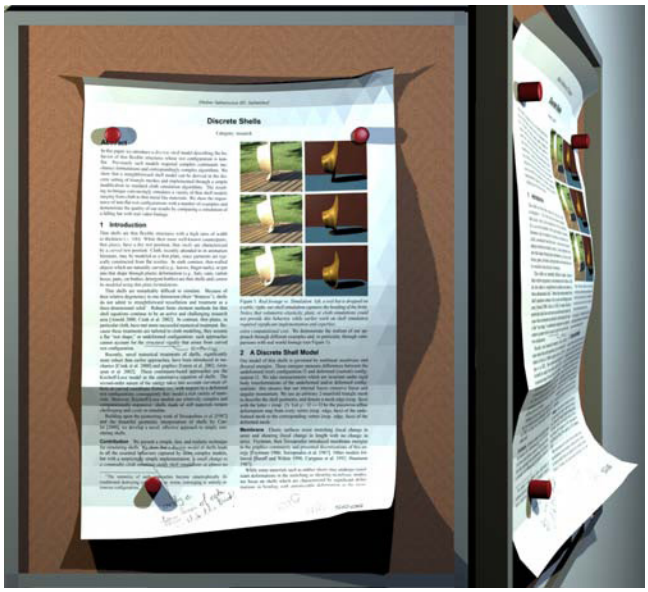


Figure 5: *Modeling a curled, creased, and pinned sheet of paper: by altering dihedral angles of the reference configuration, we effect plastic deformation. While the rendering is texture-mapped we kept flat-shaded triangles to show the underlying mesh structure (see also the color plate).*

with a flat surface, and gradually increase the undeformed angles, $\bar{\theta}_e$. Notice: modifying the *undeformed* configuration effects a *plastic* deformation. The kinematics of changing $\bar{\theta}_e$ span only physically-realizable bending, *i.e.*, *inextensible* plastic deformations. In contrast, directly modifying $\bar{\mathbf{x}}$ could introduce plastic deformations with unwanted membrane modes. We introduced elastic effects by applying three pin constraints to the *deformed* configuration. Observe the half-crease on the left side. The energy of the undeformed state is no longer zero! The (plastically-deformed) left and (untouched) right halves have incompatible undeformed shapes, consequently the undeformed configuration is *not* stress-free.

Figure 6: *A measure of discrete strain is used to fracture a thin shell in this simulation of a shattering lightbulb.*

More recently, Gingold *et al.* demonstrated that simple, discrete models of thin shells can also produce striking examples of shattering glass (see Figure 6) [Gingold et al. 2004].

7 Conclusion and Acknowledgements

We introduced a novel discrete model of thin shells for computer animation, generalizing earlier discrete models of thin plates, while complementing contemporaneous developments in cloth simulation. Our simple model captures the characteristic behaviors of shells, including flexural rigidity and crumpling; visually, animations compare favorably to sophisticated shell models requiring cumbersome high-order constitutive equations and finite-element techniques. Implementing a thin shell simulator for graphics applications is now practical and worthwhile.

The work described here is the fruit of a collaboration with Mathieu Desbrun, Anil Hirani, and Peter Schröder. More recent work on modeling thin shells is part of an active collaboration with Adrian Secord, Yotam Gingold, and Denis Zorin. We are indebted to Jerry Marsden, Tom Duchamp, and Anastasios Vayonakis for insightful discussions. Pierre Alliez, Ilja Friedel, and Steven Schkolne were pivotal in the production of the images shown here.

8 Further Reading

A comprehensive survey of this expansive body of literature is far beyond the scope of this chapter; as a starting point see the recent work of [Arnold 2000; Cirak et al. 2002] and references therein. Here we highlight only a few results from the graphics and engineering literature.

Recently, novel numerical treatments of shells, significantly more robust than earlier approaches, have been introduced in mechanics [Cirak et al. 2000] and graphics [Green et al. 2002; Grinspun et al. 2002]. These continuum-based approaches use the Kirchoff-Love constitutive equations, whose energy captures curvature effects in curved coordinate frames; consequently they model a rich variety of materials. The novel approaches remain relatively complex and computationally expensive: shells made of stiff materials are considered *challenging* and *costly* to simulate.

In contrast, thin *plate* equations tailored to animations of cloth and garmets have seen successful numerical treatment in the computer graphics literature [House and Breen 2000]. Thin plates have also been useful for variational geometric modeling [Celniker and Gossard 1991; Greiner 1994; Welch and Witkin 1992] and intuitive direct manipulation of surfaces [Qin and Terzopoulos 1997; Qin and Terzopoulos 1996; Terzopoulos and Qin 1994]. In graphics, researchers have used two kinds of approaches to modeling plates: finite-elements and mass-spring networks. In the latter resistance

to bending is effected by springs connected to opposite corners of adjacent mesh faces. Unfortunately, this simple approach does not carry over to curved undeformed configurations: the diagonal springs are insensitive to the sign of the dihedral angles between faces. While some applications can use thin plate models, many cannot. Simulations of objects such as car bodies, wine glasses, and hats rely on the *structural rigidity* that arises from a *curved* undeformed configuration, a characteristic captured by a thin shell but not a thin plate model. Earlier work on elastic surfaces includes [Feynman 1986], then [Terzopoulos et al. 1987] and [Baraff and Witkin 1998; Carignan et al. 1992; Haumann 1987].

References

ADIFOR, 2002. Argonne National Laboratory / Rice University. http://www-unix.mcs.anl.gov/autodiff/ADIFOR/.

Arnold, D., 2000. Questions on Shell Theory. Workshop on Elastic Shells: Modeling, Analysis, and Computation. Mathematical Sciences Research Institute, Bekeley.

Autodiff.org, 2002. http://www.autodiff.org.

Baraff, D., and Witkin, A. 1998. Large Steps in Cloth Simulation. In *Proceedings of SIGGRAPH*, 43–54.

Bobenko, A. I. 2004. A Conformal Energy for Simplicial Surfaces. Published online at `http://arxiv.org/abs/math.DG/0406128`, August.

Bridson, R., Fedkiw, R. P., and Anderson, J. 2002. Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. *ACM Trans. on Graphics 21*, 3 (July), 594–603.

Bridson, R., Marino, S., and Fedkiw, R. 2003. Simulation of Clothing with Folds and Wrinkles. In *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, D. Breen and M. Lin, Eds.

Carignan, M., Yang, Y., Thalmann, N. M., and Thalmann, D. 1992. Dressing Animated Synthetic Actors with Complex Deformable Clothes. In *Proceedings of SIGGRAPH*, 99–104.

Celniker, G., and Gossard, D. 1991. Deformable Curve and Surface Finite Elements for Free-Form Shape Design. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4, 257–266.

Ciarlet, P. 2000. *Mathematical Elasticity. Vol. III*, vol. 29 of *Studies in Mathematics and its Applications*. Amsterdam. Theory of shells.

Cirak, F., Ortiz, M., and Schröder, P. 2000. Subdivision Surfaces: A New Paradigm for Thin-Shell Finite-Element Analysis. *Internat. J. Numer. Methods Engrg. 47*, 12, 2039–2072.

Cirak, F., Scott, M., Antonsson, E., Ortiz, M., and Schröder, P. 2002. Integrated Modeling, Finite-Element Analysis, and Engineering Design for Thin-Shell Structures Using Subdivision. *CAD 34*, 2, 137–148.

Cohen-Steiner, D., and Morvan, J.-M. 2003. Restricted Delaunay Triangulations and Normal Cycle. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, 237–246.

Corliss, G., Faure, C., Griewank, A., Hascoët, L., and Naumann, U., Eds. 2001. *Automatic Differentiation of Algorithms: From Simulation to Optimization*. Springer.

Desbrun, M., Meyer, M., and Alliez, P. 2002. Intrinsic Parameterizations of Surface Meshes. In *Proceedings of Eurographics*, 209–218.

Feynman, C. 1986. *Modeling the Appearance of Cloth*. MSc thesis, MIT.

Ge, Z., Kruse, H. P., and Marsden, J. E. 1996. The Limits of Hamiltonian Structures in Three-Dimensional Elasticity, Shells, and Rods. *Journal of Nonlinear Science 6*, 19–57.

Gingold, Y., Secord, A., Han, J. Y., Grinspun, E., and Zorin, D. 2004. Poster: A Discrete Model for Inelastic Deformation of Thin Shells. In *ACM/Eurographics Symposium on Computer Animation '04*.

Gleicher, M. 1994. *A Differential Approach to Graphical Manipulation (Chapter 5)*. PhD thesis.

Gray, A. 1998. *Modern Differential Geometry of Curves and Surfaces*. Second edition. CRC Press.

Green, S., Turkiyyah, G., and Storti, D. 2002. Subdivision-Based Multilevel Methods for Large Scale Engineering Simulation of Thin Shells. In *Proceedings of ACM Solid Modeling*, 265–272.

Greiner, G. 1994. Variational Design and Fairing of Spline Surfaces. *Computer Graphics Forum 13*, 3, 143–154.

Grinspun, E., Krysl, P., and Schröder, P. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Transactions on Graphics 21*, 3 (July), 281–290.

Grinspun, E., Hirani, A., Desbrun, M., and Schröder, P. 2003. Discrete Shells. In *ACM SIGGRAPH Symposium on Computer Animation*. to appear.

Haumann, R. 1987. Modeling the Physical Behavior of Flexible Objects. In *Topics in Physically-based Modeling,* Eds. Barr, Barrel, Haumann, Kass, Platt, Terzopoulos, and Witkin*, SIGGRAPH Course Notes*.

House, D. H., and Breen, D. E., Eds. 2000. *Coth Modeling and Animation*. A.K. Peters.

Kass, M. 1992. CONDOR: Constraint-based Dataflow. In *Proceedings of SIGGRAPH*, 321–330.

Kergosien, Y. L., Gotoda, H., and Kunii, T. L. 1994. Bending and Creasing Virtual Paper. *IEEE Computer Graphics and Applications*, 40–48.

Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H. 2003. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and Mathematics III*, H.-C. Hege and K. Polthier, Eds. Springer-Verlag, Heidelberg, 35–57.

Newmark, N. M. 1959. A Method of Computation for Structural Dynamics. *ASCE J. of the Engineering Mechanics Division 85*, EM 3, 67–94.

Qin, H., and Terzopoulos, D. 1996. D-NURBS: A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics 2*, 1, 85–96.

Qin, H., and Terzopoulos, D. 1997. Triangular NURBS and their dynamic generalizations. *Computer Aided Geometric Design 14*, 4, 325–347.

Terzopoulos, D., and Qin, H. 1994. Dynamic NURBS with Geometric Constraints for Interactive Sculpting. *ACM Transactions on Graphics 13*, 2, 103–136.

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. 1987. Elastically Deformable Models. In *Proceedings of SIGGRAPH*, 205–214.

Welch, W., and Witkin, A. 1992. Variational Surface Modeling. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 157–166.

West, M., Kane, C., Marsden, J. E., and Ortiz, M. 2000. Variational Integrators, the Newmark Scheme, and Dissipative Systems. In *International Conference on Differential Equations 1999*, World Scientific, Berlin, 1009 – 1011.

Chapter 5:
Discrete Willmore Flow

Alexander I. Bobenko
TU Berlin

Peter Schröder
Caltech

Abstract

The Willmore energy of a surface, $\int (H^2 - K)\, dA$, as a function of mean and Gaussian curvature, captures the deviation of a surface from (local) sphericity. As such this energy and its associated gradient flow play an important role in digital geometry processing, geometric modeling, and physical simulation. In this paper we consider a *discrete* Willmore energy and its flow. In contrast to traditional approaches it is not based on a finite element discretization, but rather on an ab initio discrete formulation which preserves the Möbius symmetries of the underlying continuous theory in the discrete setting. We derive the relevant gradient expressions including a linearization (approximation of the Hessian), which are required for non-linear numerical solvers. As examples we demonstrate the utility of our approach for surface restoration, n-sided hole filling, and non-shrinking surface smoothing.

CR Categories: G.1.8 [Numerical Analysis]: Partial Differential Equations—Elliptic equations; Parabolic equations; Finite difference methods; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid and object representations; Geometric algorithms, languages, and systems; Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation; Visual.

Keywords: Geometric Flow; Discrete Differential Geometry; Willmore Energy; Variational Surface Modeling; Digital Geometry Processing.

1 Introduction

The *Willmore* energy of a surface $S \subset \mathbb{R}^3$ is given as

$$E_W(S) = \int_S (H^2 - K)\, dA = 1/4 \int_S (\kappa_1 - \kappa_2)^2\, dA,$$

where κ_1 and κ_2 denote the principal curvatures, $H = 1/2(\kappa_1 + \kappa_2)$ and $K = \kappa_1 \kappa_2$ the mean and Gaussian curvature respectively, and dA the surface area element. Immersions of surfaces which minimize this energy are of great interest in several areas:

- **Theory of surfaces**: the Willmore energy of a surface is conformally invariant [Blaschke 1929] making it an important functional in the study of conformal geometry [Willmore 2000];
- **Geometric modeling**: for compact surfaces with fixed boundary a minimizer of $E_W(S)$ is also a minimizer of total curvature $\int_S {\kappa_1}^2 + {\kappa_2}^2\, dA$ which is a standard functional in variationally optimal surface modeling [Lott and Pullin 1988; Welch and Witkin 1994; Greiner 1994];
- **Physical modeling**: thin flexible structures are governed by a surface energy of the form

  $$E(S) = \int_S \alpha + \beta (H - H_0)^2 - \gamma K\, dA,$$

  the so-called Canham-Helfrich model [Canham 1970; Helfrich 1973] (H_0 denotes the "spontaneous" curvature which plays an important role in thin-shells [Grinspun et al. 2002; Bridson et al. 2003; Grinspun et al. 2003]). For $\alpha = H_0 = 0$, $\beta = \gamma$ the Canham-Helfrich model reduces to the Willmore energy.

In all of these application areas one typically deals with the associated geometric flow

$$\dot{S} = -\nabla E(S),$$

(time derivatives are denoted by an overdot) which drives the surface to a minimum of the potential energy given by $E(S)$. In the theory of surfaces as well as in geometric modeling one is interested in critical points of $E(S)$. In physical modeling the solution shape is characterized by a balance of external and internal forces. In this setting the internal forces are a function of the Willmore gradient.

Contributions In this paper we explore a novel, discrete Willmore energy [Bobenko 2005] and introduce the associated *geometric flow* for piecewise linear, simplicial, 2-manifold meshes. In contrast to earlier approaches the discrete flow is not defined through assemblies of lower level discrete operators, nor does the numerical treatment employ operator splitting approaches. Instead the discrete Willmore energy, defined as a function of the vertices of a triangle mesh, is used directly in a non-linear numerical solver to affect the associated flow as well as solve the static problem. Since the discrete formulation has the same symmetries as the continuous problem, *i.e.*, it is Möbius invariant, the associated properties, such as invariance under scaling, carry over *exactly* to the discrete setting of meshes. To deal effectively with boundaries we introduce appropriate boundary conditions. These include position and tangency constraints as well as a free boundary condition. We demonstrate the method with some examples from digital geometry processing and geometric modeling.

1.1 Related Work

We distinguish here between *discrete* geometric flows, *i.e.*, flows based on discrete analogues of continuous differential geometry quantities, and those based on *discretizations* of continuous systems. The guiding principle in the construction of the former is the preservation of symmetries of the original continuous system, while the latter is based on traditional finite element or finite difference approaches which in general do not preserve the underlying symmetries. There is also a broad body of literature which uses linearized versions of the typically non-linear geometric functionals. Such approaches are not based on intrinsic geometric properties (*e.g.*, replacing curvatures with second derivatives) but rather depend on the particular parameterization chosen. For this reason we will not further consider them here.

Discrete Flows In the context of mesh based geometric modeling a number of discrete flows have been considered. For example, Desbrun *et al.* [1999] used mean curvature flow ($\alpha = 1$, $\beta = \gamma = H_0 = 0$) to achieve denoising of geometry. Pinkall and Polthier [1993] used a related approach, area minimizing flow, to

construct discrete minimal surfaces. Critical points of the area functional also play an important role in the construction of discrete harmonic functions [Duchamp et al. 1997], their use in parameterizations [Eck et al. 1995; Desbrun et al. 2002], and the construction of conformal structures for discrete surfaces [Mercat 2001; Gu and Yau 2003]. Since the underlying "membrane" energy is second order only, it cannot accomodate G^1 continuity conditions at the boundary of the domain. These are important in geometric modeling for the construction of tangent plane continuous surfaces. Fourth order flows on the other hand can accomodate position and tangency conditions at the boundary. Perhaps the simplest fourth order flow is *surface diffusion*, *i.e.*, flow by the Laplace-Beltrami operator of mean curvature, $\dot{S} = -\Delta_S H$. Such discrete flows were studied by Schneider and Kobbelt [2001], Xu *et al.* [2003], and Yoshizawa and Belyaev [2002]. In each case the approach was based on taking the square of a discrete Laplace-Beltrami operator combined with additional simplifications to ease implementation. Unfortunately surface diffusion flow can lead to singularities in finite time [Mayer and Simonett 2000] leading to "pinching off" of surfaces which are too thin. Yoshizawa and Belyaev [2002] demonstrate this behavior and show the comparison with Willmore flow, which leads to much better results in this regard. This difference in behavior between surface diffusion and Willmore flows is due to the additional terms appearing in the Euler-Lagrange (EL) equation of the Willmore flow

$$\Delta_S H + 2H(H^2 - K) = 0.$$

Yoshizawa and Belyaev took the EL equation as their starting point and defined a discrete Willmore flow by assemblying the components from individual, well known discrete operators. Unfortunately in that discrete setting properties such as $H^2 - K \geq 0$ can no longer be guaranteed. In contrast we define our discrete Willmore energy directly using the Möbius invariance of the integrand $(H^2 - K)\,dA$ as the fundamental principle. Among other properties one achieves the $H^2 - K \geq 0$ always, as expected (see Section 2).

Discretized Flows Both surface diffusion and Willmore flows have been treated numerically through a variety of discretizations. For example, Tasdizen *et al.* [2003] and Chopp and Sethian [1999] use a level set formulation for surface diffusion flow, while Mayer [2001] uses finite differences, and Deckelnick *et al.* [2003] use finite elements. For Willmore flow finite element approaches were pursued by Hari *et al.* [2001] and Clarenz *et al.* [2004]. A level set formulation was given by Droske and Rumpf [2004]. In these approaches no attempt is made to preserve the Möbius symmetries. On the other hand they do have the advantage that a rich body of literature applies when it comes to error and convergence analysis. Our approach as of now lacks a complete analysis of this type. Partial results on the convergence of the discrete Willmore energy to the continuous Willmore energy are discussed at the end of Section 2.

2 Discrete Willmore Energy

In this section we recall the definition of the discrete Willmore energy and some of its relevant properties.

The derivation of the discrete Willmore energy is based on the observation that the integrand

$$(H^2 - K)\,dA$$

is invariant under Möbius transformations [Blaschke 1929], *i.e.*, translations, rotations, uniform scale, and inversion. The first two are obvious and the latter two follow from the change of variable formula [Chen 1973]. This immediately implies that $E_W(S)$ itself is a conformal invariant of the surface. Note that for compact closed surfaces we also have $E_H(S) = \int_S H^2\,dA$ as a conformal invariant [White 1973]. However the *integrand* of $E_H(S)$ is *not* Möbius invariant. It is for this reason that we prefer E_W over E_H (the latter is used by some authors as the definition of the Willmore energy).

We are interested in evaluating this energy for discrete surfaces, *i.e.*, surfaces given as topological 2-manifold configurations of simplicies. Such a "mesh" consists of vertices $v_i = (x_i, y_i, z_i)^T$ $(i = 1, \ldots, N)$ and the topological complex is given as a set of edges e_{ij} connecting v_i with v_j and triangles t_{ijl} bounded by vertices v_i, v_j and v_l and edges e_{ij}, e_{jl}, and e_{li} (see Figure 1). For notational simplicity we assume that the surface is closed (*i.e.*, each edge e_{ij} is bounded by exactly two triangles, t_{ijl} and t_{jik}) and that triangles incident on a given edge are consistently oriented (note however that we do not assume global orientability). Boundaries will be discussed in Section 3.3.

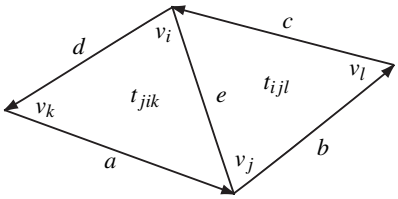


Figure 1: *Notation for vertices, edges and triangles in the vicinity of a given edge* $e = (v_i, v_j)$.

The discrete Willmore energy on a mesh is defined at each vertex v_i as

$$W_i = \sum_{e_{ij}} \beta_j^i - 2\pi,$$

i.e., a sum over the edges incident to v_i of certain angles β_j^i, which measure the angle between the circumcircles defined by the two triangles t_{ijl} and t_{jik} incident to the given edge e_{ij} (see Figure 2). Obviously W_i is Möbius invariant since its definition is based on angles between circles. The Willmore energy of the entire mesh is then simply the sum, $W = \sum_i W_i$. For later use we also recall the definition of discrete Gauss curvature at a vertex v_i

$$K_i = 2\pi - \sum_{t_{ikj}} \alpha_{kj}^i.$$

Here α_{kj}^i denotes the Euclidean angle at i inside the triangle t_{ikj}.

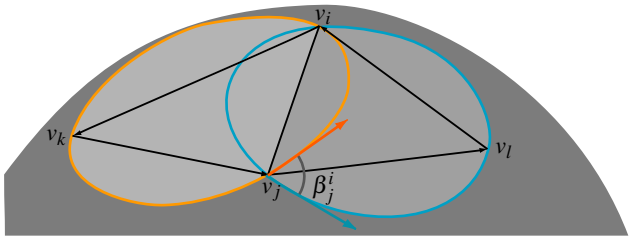


Figure 2: *Geometry of* β_j^i.

The geometric picture is as follows. A given edge has two incident triangles. Each triangle has a circumcircle. Since the four vertices forming the two triangles are (generically) on a common sphere (possibly at infinity) the two circumcircles are also on this sphere. The two circles meet in the vertices v_i and v_j where they intersect. Consider a tangent vector to each circle at v_i. These two tangent vectors make the angle β_j^i which lies in the tangent plane to the

sphere at that point. Note that this geometric setup implies that $\beta_j^i = \beta_i^j$. Suppose now v_i and all its neighbors v_j (*i.e.*, corresponding to edges e_{ij}) lie on a common sphere and that the (embedded) 1-ring of v_i is convex. In that case it is easy to see that the β_j^i neatly add up to exactly 2π in the tangent plane at v_i and hence $W_i = 0$ (see Figure 3) as expected. Now suppose that v_i and its neighboring vertices do not share a common sphere. In that case $W_i > 0$. To see this use the Möbius invariance of the energy and map the central point v_i to infinity by a Möbius transformation. All circles passing through v_i are mapped to straight lines and the energy becomes the sum $\sum_j \beta_j^i$ of the external angles of a non-planar closed polygon in three space. In that interpretation the inequality $\sum_j \beta_j^i \geq 2\pi$ follows easily [Bobenko 2005] (this inequality is a polygonal version of Fenchel's theorem [Fenchel 1929]). With the same argument one also concludes that $W_i + K_i \geq 0$, *i.e.*, $\sum_j \beta_j^i - \sum_{kj} \alpha_{kj}^i \geq 0$, reflecting the fact that $H^2\,dA$ is always non-negative.

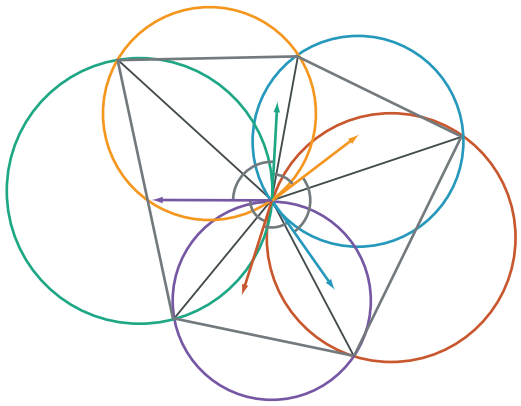


Figure 3: *Geometry of $\sum_{e_{ij}} \beta_j^i$ around a vertex. The angles between subsequent circumcircles—appropriate tangent vectors are indicated with colors corresponding to the circumcircles of each triangle—neatly add up to 2π if all vertices are co-spherical.*

Finally we observe that $W_i \geq 0$ and that it vanishes iff v_i and all its edge neighbors v_j lie on a common sphere and the vertex v_i is convex. These two conditions are equivalent to the condition that the triangles meeting at v_i build a Delaunay triangulation on a sphere.

Smooth Limit The discrete Willmore energy W is not only an analogue of the continuous one. It approximates the continuous Willmore energy $\mathscr{W}$ in a "natural" limit. Let $(u,v) \mapsto f(u,v)$ be a curvature line parameterization of a surface. Without loss of generality consider the vicinity of the origin $(u,v) = (0,0)$ in the tangent plane where we have

$$(u,v) \mapsto (u,v,\frac{1}{2}(\kappa_1 u^2 + \kappa_2 v^2) + o(u^2+v^2)),$$

with κ_1, κ_2 denoting the principal curvatures of the surface at the point $(0,0)$. Now consider a triangular lattice $L_\varepsilon = \{\varepsilon(la+mb+nc) : l,m,n \in \mathbb{Z}\}$ in the parameter plane generated by three vectors a,b,c with $a+b+c=0$. Here ε is a small parameter. Consider the hexagon D_ε in the parameter plane with vertices $p_1 = \varepsilon a$, $p_2 = -\varepsilon c$, $p_3 = \varepsilon b$, $p_4 = -\varepsilon a$, $p_5 = \varepsilon c$, $p_6 = -\varepsilon b$ and its image $f(D_\varepsilon)$ on the surface. Let $\mathscr{W}(D_\varepsilon)$ be the smooth Willmore energy of $f(D_\varepsilon)$. On the other hand, the vertices $f(p_i), i = 1,\ldots,6$ together with $f(0)$ build a simplicial surface with six triangles. Denote by $W(D_\varepsilon)$ the discrete Willmore energy of this surface and consider the quotient of the discrete and smooth Willmore energies of such an infinitesimal hexagon

$$R = \lim_{\varepsilon \to 0} \frac{W(D_\varepsilon)}{\mathscr{W}(D_\varepsilon)}.$$

A direct but rather complicated computation leads to the following conclusions:

1. R is independent of the curvatures κ_1, κ_2,
2. $R \geq 1$, and $R = 1$ iff the lattice L_ε has two of its directions aligned with the curvature lines of the surface (two of the vectors a,b,c are curvature line directions).

Thus, after sufficiently many $1 \to 4$ refinements of the smooth surface the discrete Willmore energy approximates the smooth one if the curvature line net is triangulated, otherwise the discrete energy is larger.

The question whether the discrete Willmore energy can be used as a variational method for computation of curvature line nets is currently under investigation.

3 Evaluation

For the numerical treatment of discrete Willmore flow and the solution of energy minimization problems we need effective evaluation procedures for the Willmore energy and its derivatives. To simplify the implementation of these functions we begin with a discussion of the definition of the angles β_j^i and some of the consequent symmetries in the expressions.

3.1 Definition of Intersection Angles

Consider edge e_{ij} and its two incident triangles t_{jik} and t_{ijl} with associated vertices v_k, v_j, v_l, and v_i (see Figure 1). Defining the four directed edge vectors

$$A = \frac{a}{|a|} = \frac{v_j - v_k}{|v_j - v_k|} \qquad B = \frac{b}{|b|} = \frac{v_l - v_j}{|v_l - v_j|}$$
$$C = \frac{c}{|c|} = \frac{v_i - v_l}{|v_i - v_l|} \qquad D = \frac{d}{|d|} = \frac{v_k - v_i}{|v_k - v_i|}$$

the angles follow as

$$\begin{aligned} \cos\beta_j^i &= -R(Q) = -R(AB^{-1}CD^{-1}) \\ &= \langle A,C\rangle\langle B,D\rangle - \langle A,B\rangle\langle C,D\rangle - \langle B,C\rangle\langle D,A\rangle, \end{aligned}$$

where $\langle .,. \rangle$ denotes the usual Euclidean dot product and $R(Q)$ the real part of the normalized cross ratio of the four edges bounding the "diamond" formed by the two triangles incident on edge e_{ij}. This cross ratio is defined in terms of quaternion algebra with the standard identification of 3-vectors with imaginary quaternions, $\mathbb{R}^3 \equiv I(\mathbb{H})$, $v \mapsto (0,ix,jy,kz)^T$ ($i^2 = j^2 = k^2 = -1$, $ij = k$, $jk = i$, $ki = j$). The subsequent expression of this quaternion cross ratio in terms of Euclidean inner products follows from the rules of quaternion multiplication and Lagrange's identity for the inner product between two cross products. More details can be found in [Bobenko 2005].

Properties of β_j^i and its Derivatives A number of surprising facts—which we exploit to significantly simplify the expressions needed by the numerical solver—are immediately obvious from the above definition. To clarify these we make all arguments explicit,

$\beta^i_j = \beta(k,j,l,i)$ going around the diamond in counter clockwise order. We already noted earlier that $\beta(k,j,l,i) = \beta(l,i,k,j)$. In fact from the formula for $\cos\beta(k,j,l,i)$ it terms of scalar products it is immediately clear that $\beta(k,j,l,i)$ is invariant under all cyclic permutations and reflections of its arguments. In particular if we flip the edge $e_{ij} \mapsto e_{kl}$ the cosine of the angle remains the same.

From the invariance under cyclic and reflection permutations of its arguments it also follows that all first derivatives can be written as a single function $f_1(.,.,.,.)$ with suitably permuted arguments

$$\beta_{,k} = f_1(k,j,l,i) \quad \beta_{,j} = f_1(j,l,i,k)$$
$$\beta_{,l} = f_1(l,i,k,j) \quad \beta_{,i} = f_1(i,k,j,l).$$

(Here and in what follows we use comma notation to denote partial derivatives with respect to the corresponding argument and write $\beta := \beta^i_j$ to reduce clutter.)

3.2 Energy Gradient

For gradient flow numerical computations we require the gradient of the discrete Willmore energy. A direct calculation readily yields

$$\begin{aligned} -\sin(\beta)\beta_{,k} \quad = \quad & (-\frac{1}{|a|}P_A(C)\langle B,D\rangle + \langle A,C\rangle\frac{1}{|d|}P_D(B) \\ & +\frac{1}{|a|}P_A(B)\langle C,D\rangle - \langle A,B\rangle\frac{1}{|d|}P_D(C) \\ & -\langle B,C\rangle(\frac{1}{|d|}P_D(A) - \frac{1}{|a|}P_A(D))). \end{aligned}$$

Here we used $P_X = I - X \otimes X$ as shorthand for the projection operator into the orthogonal complement of (the unit vector) X.

Remarkably, if we separate out the linear dependence of this expression on a, b, c, and d we arrive at a *scalar* linear combination

$$\begin{aligned} -\sin(\beta)\beta_{,k} \quad = \quad & (\frac{\langle D\times A, B\times C\rangle}{|a|^2} - \frac{\langle B,C\rangle}{|a||d|} - \frac{\langle B,C\rangle\langle D,A\rangle}{|a|^2})a \\ & +(\frac{\langle A,C\rangle}{|b||d|} + \frac{\langle C,D\rangle}{|b||a|})b - (\frac{\langle A,B\rangle}{|c||d|} + \frac{\langle B,D\rangle}{|c||a|})c \\ & -(\frac{\langle D\times A, B\times C\rangle}{|d|^2} - \frac{\langle B,C\rangle}{|a||d|} - \frac{\langle B,C\rangle\langle D,A\rangle}{|d|^2})d. \end{aligned} \tag{1}$$

In a semi-implicit time stepping algorithm this amounts to requiring only the solution of a sparse linear system of size $n \times n$ rather than $(3n) \times (3n)$ for n vertices, a very attractive feature. In fact Equation 1 can serve as a linearized version of the Hessian of the energy. See Section 4 for further comments on this fact.

For the free boundary treatment we also need expressions for the gradient of the angle between two edges. Desbrun and coworkers [Desbrun et al. 2002] (Appendix B) derive these and we will not repeat them here.

Gradient Singularity If v_k, v_j, v_l, and v_i are co-circular then $\beta = 0$ and $\beta_{,k}$ is not defined. For vertices in general positions this does not occur. However, in practice the case that the four vertices of a diamond are nearly co-circular, while rare, does occur. For some inputs it can in fact be a frequent occurance (see for example Figure 10). Consider a quadrangulation of a smooth surface which is turned into a triangle mesh through insertion of diagonals in each quad. In this setting the diagonal edges very often have β nearly equal to zero.

Keeping in mind that in the end we care about the direction of *negative gradient*, *i.e.*, steepest descent, of the discrete Willmore energy we make the following geometric observation. In case $\beta = 0$ there is one direction of varying v_k in which the angle does not change (infinitesimally). This is the tangential direction to the circle C passing through the points v_i, v_j, v_k and v_l. For (infinitesimal) unit motions in all orthogonal directions the angle β increases at equal rate. This property of the gradient is conformal and thus preserved under Möbius transformations. It can be seen more easily in a Möbius transformed picture. Send the point v_i to infinity by the inversion in a sphere centered at v_i. Both circles in Figure 4 become straight lines. Let $\tilde{v}_j, \tilde{v}_k, \tilde{v}_l$ be the images of the vertices v_j, v_k, v_l under this Möbius transformation. For the case of $\beta = 0$ both circles in Figure 4 are coincident—call this common circle C—and the points $\tilde{v}_j$, $\tilde{v}_k$ and $\tilde{v}_l$ become collinear: they lie on the straight line L which is the Möbius image of the circle C. The only direction of varying $\tilde{v}_k$ in which the angle does not change is along the straight line L. Variations in all orthogonal directions increase the angle at equal rate.

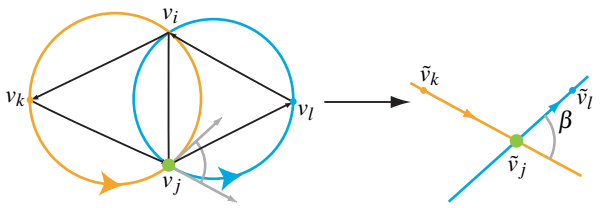


Figure 4: *After sending v_i to infinity, the two circles have been mapped to two lines which intersect with angle β.*

Consider now a given vertex v_i and assume for the moment that only one β contributing to the gradient computation at v_i vanishes. Let $C \ni v_i$ be the corresponding circle with four vertices lying on it. Let all other, well defined, negative gradient directions sum to g. Decompose a variation direction $G = G_o + G_p$ of v_i into the parts orthogonal $G_o \perp C$ and parallel $G_p \parallel C$ to the tangent of the circle in v_i and let $g = g_o + g_p$ be the same decomposition of g. The contribution to the gradient from all "regular" (non-vanishing) β's is $-\langle g, G\rangle$ and the contribution of the vanishing β is $R \mid G_o \mid$ with some $R > 0$. For the whole gradient this implies

$$-G_p g_p + (\mid G_o \mid R - \langle G_o, g_o\rangle).$$

Thus the total negative gradient direction, i.e. the direction in which the energy decreases the most is g_p (parallel to C) if $R > \mid g_o \mid$ and $g_p + g_o(1 - R/\mid g_o \mid)$ if $R < \mid g_o \mid$.

The case of multiple β's in the support of the gradient of W_i with respect to the given vertex v_i vanishing, is more complicated. One can get the negative gradient direction (if it exists) in this case from the following non-linear minimization process. To each of the edges e_n with vanishing $\beta(e_n) = 0$ there corresponds a circle C_n through v_i. For the variation G the contribution to the gradient of this edge is $\mid R_n \times G \mid$ where R_n is a vector tangent to C_n. We define

$$\delta = \min_{|G|=1, \langle G,g\rangle \geq 0} \sum_n |R_n \times G| - \langle G, g\rangle$$

where the sum is taken over all vanishing β from the 1-ring with flaps of v_i. The first term measures the length of the projection of G into the orthogonal complement of R_n, *i.e.*, the amount of (infinitesimal) increase of energy while the second term measures the decrease in energy for the direction G. If $\delta > 0$ no motion exists which decreases the energy and the direction of steepest descent is the zero vector. If $\delta < 0$ the direction G which achieves the minimum is our sought after steepest descent direction with magnitude $|\delta|$.

The case that all β in the support of the gradient of v_i vanish simultaneously, corresponds to a configuration which puts all vertices in the 1-ring with flaps of v_i including v_i itself onto a common circle. In this case no direction decreasing the Willmore energy at v_i exists.

In our implementation we have experimented with the non-linear minimization to find a valid direction of energy decrease (or zero if none exists) but found it to give the same results (numerically) as a far simpler heuristic: if $|\sin\beta| < \varepsilon$ set the corresponding gradient to zero. We found $\varepsilon = 10^{-6}$ to give reliable results in double precision for all our experiments.

3.3 Boundary Conditions

So far we have implemented two types of boundary conditions.

G^1-boundary The variational problem we are dealing with is a fourth order system. To be well posed it requires two independent boundary conditions. The most natural choice here is to fix positions and normals at a boundary. We specify this kind of boundary data on a mesh by fixing positions of the boundary vertices and those vertices within one edge distance from the boundary. The normals of the triangles of this boundary strip can be treated as normals on the boundary. This boundary condition fits perfectly for G^1-gluing of surfaces. Typical applications are surface restoration and smooth filling of a hole (see Figures 8 and 9). Note that the method requires no conditions on the topology of the mesh. In particular one can fix some "islands" of internal vertices (or faces) of the required surface.

Free Boundary Alternatively we have experimented with closing boundary curves by adding a vertex at infinity to each boundary loop. This is an unusual treatment since it actually removes the boundary and adds a Dirichlet condition at infinity. The idea comes from Möbius geometry where the infinity point is not distinguished.

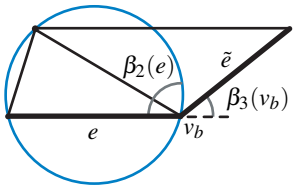


Figure 5: *Free boundary conditions. Boundary edges e and $\tilde{e}$, and a boundary vertex v_b with the angles β_2 and β_3.*

For simplicity consider a surface with one boundary curve. By adding the infinity point and connecting it to each boundary vertex we obtain a closed surface. We distinguish three types of edges of this surface $E = E_i \cup E_b \cup E_\infty$: internal edges E_i, boundary edges E_b of the original surface and new edges E_∞ incident to the infinity point. The circumcircles passing through the infinity point are straight lines. The discrete Willmore energy of the closed surface consists of three terms

$$\sum_{e\in E} \beta(e) = \sum_{e\in E_i} \beta_1(e) + \sum_{e\in E_b} \beta_2(e) + \sum_{e\in E_\infty} \beta_3(e).$$

The first term is just the discrete Willmore energy of the original surface. The angles $\beta_2(e)$ are associated to the boundary edges $e \in E_b$ and are the intersection angles of these edges with the circumcircles of the corresponding boundary triangles. Another interpretation for $\beta_2(e)$ is that this is π minus the angle of the boundary triangle opposite to the edge $e \in E_b$. Finally the angle $\beta_3(e)$ is associated to the additional edge $e \in E_\infty$ connecting ∞ to a boundary vertex v_b. Equivalently it can be associated to the boundary vertex v_b. This is the intersection angle of two circumcircles (which are straight lines in this case) passing through v_b and ∞, *i.e.*, the intersection angle of two boundary edges meeting at v_b (see Figure 5).

The resulting behavior is that of a free boundary (see Figure 7; right column).

4 Numerical Experiments

We have implemented the discrete Willmore gradient flow using linear and non-linear solvers from the excellent PETSc [Balay et al. 2004] and TAO [Benson et al. 2004] libraries, allowing us to experiment with a wide variety of pre-canned solvers, while needing to supply only the gradient, respectively the approximation of the Hessian (Equation 1). For the time discretization we experimented with both the forward and backward Euler method. For the forward Euler method the time step limitation imposed by the Courant condition for fourth order problems—time increments must be of the order of the fourth power of the shortest edge in the mesh—is too severe to be practical except for very simple meshes. The backward Euler method leads to a non-linear problem at each step. These can be solved with a full Newton method requiring evaluation of the Hessian of the energy at each iteration step. We did derive the expressions for the Hessian, but found that the effort was not justified as a function of evaluation cost and numerical behavior. The latter was no better in our experiments than a much simpler approach based on a semi-implicit time discretization using the linearized version of the gradient (Equation 1). In that setup only a *linear* system must be solved at each time step to find the position increment $\Delta x^{(t)}$

$$(\frac{1}{dt}\mathbb{I} + K^{(t)})\Delta x^{(t)} = -\nabla E^{(t)}.$$

Here dt is the time step and a super script $^{(t)}$ denotes quantities evaluated at time t. The matrix K is $n \times n$ where n is the number of (free) vertices and collects the terms of Equation 1. Assuming an average valence of six, each row of the matrix contains (on average) thirteen non-zero entries (1-ring with flaps plus the center vertex). The full Hessian has non-trivial 3×3 blocks instead and results in a linear system of size $(3n) \times (3n)$.

The use of such approximations is well established and works well in practice though the usual convergence guarantees of (trust region) Newton methods are missing. Desbrun and co-workers [Desbrun et al. 1999] used a similar approach when they performed implicit mean curvature flow with a constant matrix per time step. Recall that the coefficients of the "cotan formula" change throughout the time step. Keeping them constant corresponds to a similar linearization of the gradient as we employed. For the particular case of problems involving squared curvature bending energies Hauth and co-workers [Hauth et al. 2003], similarly found that inexact Newton methods using even fairly aggressive linearizations of gradients work very well.

Figures 6 and 7 show some simple examples. The icosahedron is subdivided linearly four times and becomes essentially a perfect sphere within a few minimization steps (Figure 6). After 24 steps convergence was achieved with a final energy of 10^{-7} (a perfect sphere would be zero). The difference between fixed and free boundary conditions is illustrated with the cathead example (Figure 7). First the result of flow with fixed boundaries then the result of keeping the boundaries free (both intermediate and final state

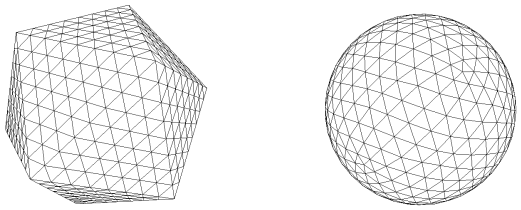


Figure 6: *Subdivided icosahedron rapidly evolves to a sphere.*

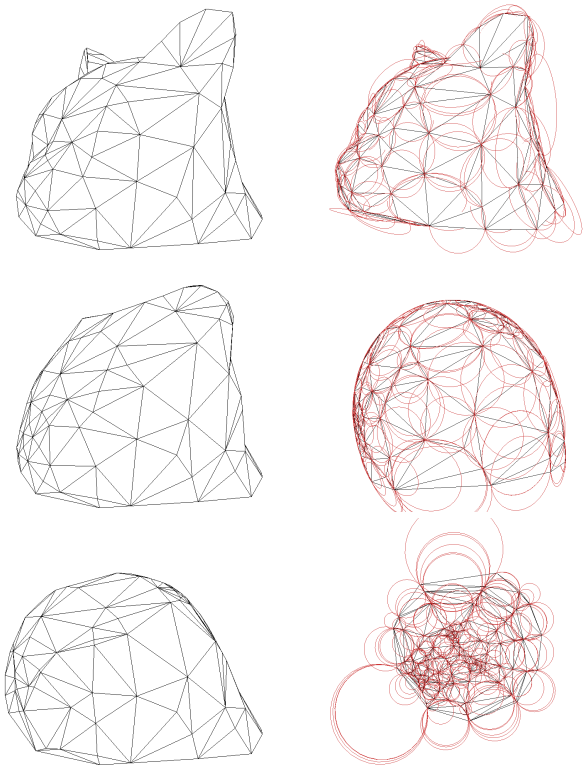


Figure 7: *Cathead evolved with fixed boundaries (left column) and free boundaries (right column). In each case the original mesh is followed by an intermediate state of the evolution and the final state.*

shown). The latter evolves to a planar polygon with convex boundaries. In the latter example we show the circumcircles for all triangles.

Figure 8 shows a standard benchmark example from k-sided (six in this case) hole filling. An initial triangulation with boundaries coming from a Loop subdivision surface is relaxed under the Willmore flow. With the two outermost "rows" of vertices fixed tangent continuity across the boundary is assured. Note that this example starts in a configuration with many edges having $\beta = 0$. Our simple strategy of setting these gradients to zero works quite well in this example. After a few steps all β angles have become sufficiently non-zero (above our threshold of $\varepsilon = 10^{-6}$) that the flow proceeds as expected.

Figure 9 shows an example of mesh restoration. A set of triangles is marked as free while all others are held fixed. The free vertices flow to "repair" the scar with a surface section which smoothly (G^1) interpolates the surrounding fixed surface (compare to the example in [Clarenz et al. 2004]).

Finally Figure 10 shows an example of geometry denoising. The mesh smoothed in this case is the raw result of a light field scanner with typically small amplitude noise due to measurement error.

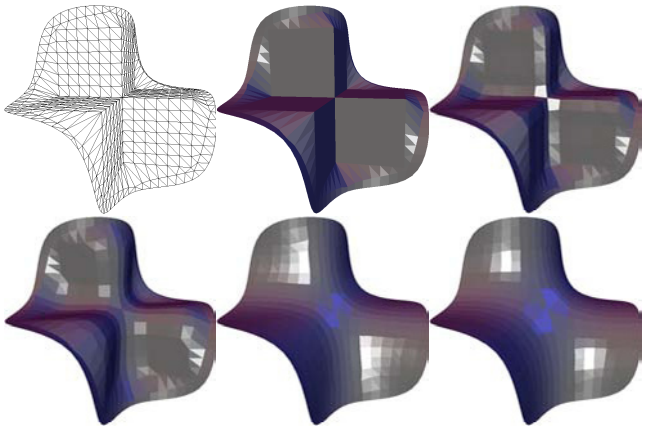


Figure 8: *Smooth filling of a six sided hole. On the upper left the original configuration showing the underlying mesh. The boundary triangles follow a smooth outline and fix position and tangency constraints (all other vertices are unconstrained). Evolution to the energy minimum is illustrated through a number of intermediate steps with the final hole fill in the lower right. All shaded images use triangle normals for shading without interpolation.*

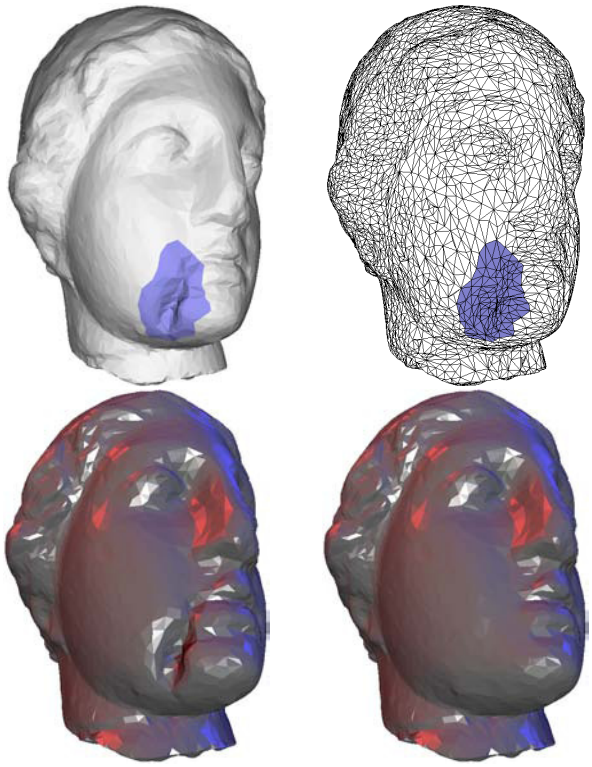


Figure 9: *Surface restoration for the Egea model. A region to be restored is outlined (top). All vertices in the blue triangles are unconstrained with the surrounding vertices providing position and tangency constraints. Results of the energy minimization (before and after; bottom).*

In particular for examples of this type the non-shrinking nature of the Willmore flow (the energy is scale invariant) favors it over standard approaches based on mean curvature flow. The mesh contains over 37k vertices (and 88 boundary loops). This mesh is particularly challenging since it contains many edges with β near zero: the original mesh is a triangulated quadrangulation. It also has many triangles with very high aspect ratio right next to small, round triangles. Figure 10 shows the original mesh followed by the results of 10 respectively 100 smoothing steps.

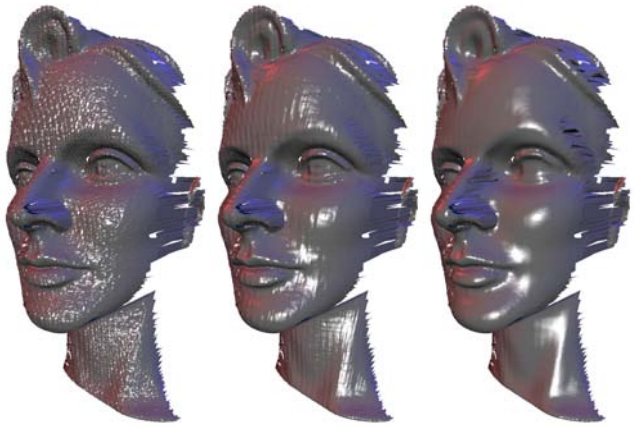


Figure 10: *Denoising of scanned geometry. On the left the original mesh with noise due to an active light stripe based scanner. Followed by the results of 10 and 100 smoothing steps (37k vertices; 88 boundary loops).*

5 Conclusion

In this paper we have considered a discrete Willmore flow. The discrete energy is expressed in terms of circles and the angles they make with one another and therefore Möbius invariant, reproducing the symmetries of the continous energy. The discrete energy approaches the continuous energy in the infinitesimal limit for regular triangulations with two edges aligned with principal curvature directions. We have experimented with a number of different linear and non-linear solvers and found a simple linear approximation of the Hessian to be sufficient in our experiments.

Ongoing investigations are geared towards more powerful numerical methods. Especially for large meshes a multigrid solver for the linear systems arising in the semi-implicit time stepping method may well provide significant speedups over our current (unoptimized) implementation. Possible future directions include the use of Willmore gradient flow for the construction of variational subdivision schemes which would optimize functionals such as $\int \kappa_1{}^2 + \kappa_2{}^2 \, dA$ in a fully non-linear fashion. Another interesting avenue is the use of the Willmore functional to construct curvature line nets. We have observed that the discrete Willmore flow leads to meshes aligned with the curvature lines of the surface. This phenomenon, theoretically partially explained in Section 2, is quite natural since the curvature lines are also a subject of Möbius geometry. A closely related problem, currently under investigation, is the definition of the discrete Willmore energy for quadrilateral meshes, which in a sense would be more natural for curvature line nets.

Acknowledgments This work was supported in part by NSF (DMS-0220905, DMS-0138458, ACI-0219979), DFG (Research Center MATHEON "Mathematics for Key Technologies," Berlin), DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Alias, and Pixar. Special thanks to Kevin Bauer, Oscar Bruno, Mathieu Desbrun, Ilja Friedel, Cici Koenig, Nathan Litke, and Fabio Rossi.

References

BALAY, S., BUSCHELMAN, K., EIJKHOUT, V., GROPP, W. D., KAUSHIK, D., KNEPLEY, M. G., MCINNES, L. C., SMITH, B. F., AND ZHANG, H. 2004. PETSc Users Manual. Tech. Rep. ANL-95/11 - Revision 2.1.5, Mathematics and Computer Science Division, Argonne National Laboratory. Available at http://www-unix.mcs.anl.gov/petsc/petsc-2/.

BENSON, S. J., MCINNES, L. C., MORÉ, J., AND SARICH, J. 2004. TAO User Manual (Revision 1.7). Tech. Rep. ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory. Available at http://www-unix.mcs.anl.gov/tao.

BLASCHKE, W. 1929. *Vorlesungen über Differentialgeometrie III*. Springer.

BOBENKO, A. I. 2005. A Conformal Energy for Simplicial Surfaces. In *Combinatorial and Computational Geometry*, J. E. Goodman, J. Pach, and E. Welzl, Eds., MSRI Publications. Cambridge University Press, 133–143.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, Eurographics Association, 28–36.

CANHAM, P. B. 1970. The Minimum Energy of Bending as a Possible Explanation of the Biconcave Shape of the Human Red Blood Cell. *Journal of Theoretical Biology 26*, 61–81.

CHEN, B.-Y. 1973. An Invariant of Conformal Mappings. *Proceedings of the American Mathematical Society 40*, 2, 563–564.

CHOPP, D. L., AND SETHIAN, J. A. 1999. Motion by Intrinsic Laplacian of Curvature. *Interfaces and Free Boundaries 1*, 1, 107–123.

CLARENZ, U., DIEWALD, U., DZIUK, G., RUMPF, M., AND RUSU, R. 2004. A Finite Element Method for Suface Restoration with Smooth Boundary Conditions. *Computer Aided Geometric Design*. To appear.

DECKELNICK, K., DZUIK, G., AND ELLIOTT, C. M. 2003. Fully Discrete Semi-Implicit Second order Splitting for Anisotropic Surface Diffusion of Graphs. Tech. Rep. 33, University of Magdeburg.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. 1999. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *Computer Graphics (Proceedings of SIGGRAPH)*, 317–324.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic Parameterizations of Surface Meshes. *Computer Graphics Forum (Proceedings of Eurographics 2002) 21*, 3, 209–218.

DROSKE, M., AND RUMPF, M. 2004. A Level Set Formulation for Willmore Flow. *Interfaces and Free Boundaries*. To appear.

DUCHAMP, T., CERTAIN, A., DEROSE, T., AND STUETZLE, W. 1997. Hierarchical Computation of PL Harmonic Embeddings. Tech. rep., University of Washington.

ECK, M., DEROSE, T. D., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution Analysis of Arbitrary Meshes. In *Proceedings of SIGGRAPH*, 173–182.

FENCHEL, W. 1929. Über die Krümmung und Windung geschlossener Raumkurven. *Math. Ann. 101*, 238–252.

GREINER, G. 1994. Variational Design and Fairing of Spline Surfaces. In *Proceedings of EUROGRAPHICS*, vol. 13, 143–154.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Transactions on Graphics 21*, 3, 281–290.

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete Shells. In *Symposium on Computer Animation*, 62–67.

GU, X., AND YAU, S.-T. 2003. Global Conformal Surface Parameterization. In *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 127–137.

HARI, L. P., GIVOLI, D., AND RUBINSTEIN, J. 2001. Computation of Open Willmore-Type Surfaces. *Applied Numerical Mathematics 37*, 257–269.

HAUTH, M., ETZMUSS, O., AND STRASSER, W. 2003. Analysis of Numerical Methods for the Simulation of Deformable Models. *The Visual Computer 19*, 7–8, 581–600.

HELFRICH, W. 1973. Elastic Properties of Lipid Bilayers: Theory and Possible Experiments. *Zeitschrift für Naturforschung Teil C 28*, 693–703.

LOTT, N. J., AND PULLIN, D. I. 1988. Method for Fairing B-Spline Surfaces. *Computer-Aided Design 20*, 10, 597–600.

MAYER, U. F., AND SIMONETT, G. 2000. Self-Intersections for the Surface Diffusion and the Volume Preserving Mean Curvature Flow. *Differential and Integral Equations 13*, 1189–1199.

MAYER, U. F. 2001. Numerical Solution for the Surface Diffusion Flow in Three Space Dimensions. *Computational and Applied Mathematics 20*, 3, 361–379.

MERCAT, C. 2001. Discrete Riemann Surfaces and the Ising Model. *Communications in Mathematical Physics 218*, 1, 177–216.

PINKALL, U., AND POLTHIER, K. 1993. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics 2*, 1, 15–36.

SCHNEIDER, R., AND KOBBELT, L. 2001. Geometric Fairing of Irregular Meshes for Free-From Surface Design. *Computer Aided Geometric Design 18*, 4, 359–379.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2003. Geometric Surface Processing via Normal Maps. *ACM Transactions on Graphics 22*, 4, 1012–1033.

WELCH, W., AND WITKIN, A. 1994. Free-Form Shape Design Using Triangulated Surfaces. *Computer Graphics (Proceedings of SIGGRAPH) 28*, 247–256.

WHITE, J. H. 1973. A Global Invariant of Conformal Mappings in Space. *Proceedings of the American Mathematical Society 38*, 1, 162–164.

WILLMORE, T. J. 2000. Surfaces in Conformal Geometry. *Annals of Global Analysis and Geometry 18*, 3-4, 255–264.

XU, G., PAN, Q., AND BAJAJ, C. L. 2003. Discrete Surface Modeling using Geometric Flows. Tech. rep., University of Texas.

YOSHIZAWA, S., AND BELYAEV, A. G. 2002. Fair Triangle Mesh Generation with Discrete Elastica. In *Geometric Modeling and Processing*, IEEE Computer Society, 119–123.

Chapter 6:
Discrete Conformal Mappings via Circle Patterns

Liliya Kharevych
Caltech

Boris Springborn
TU Berlin

Peter Schröder
Caltech

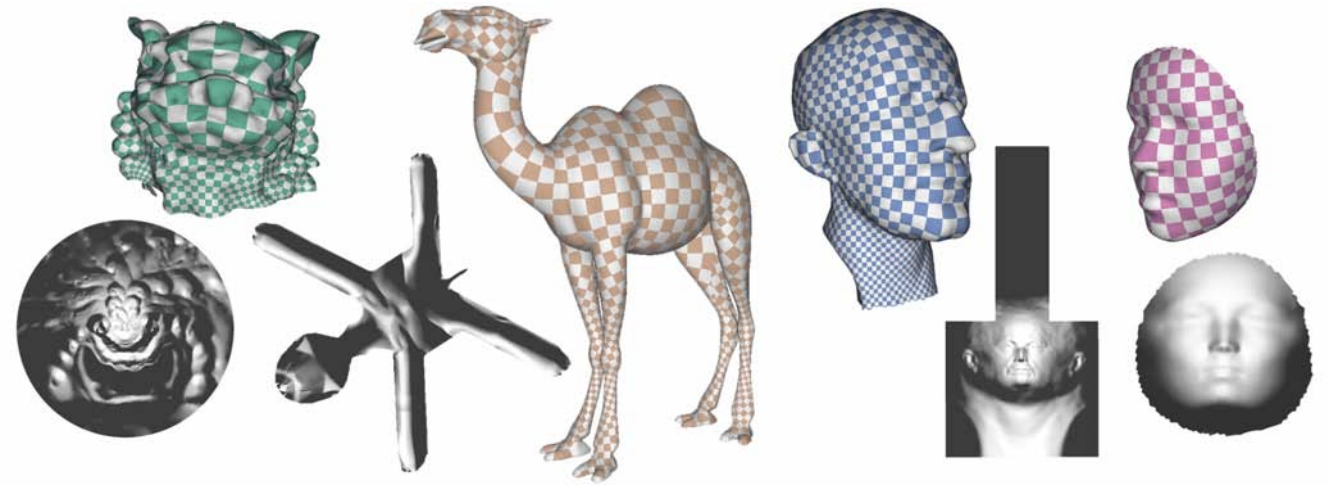


Figure 1: *Examples of discrete conformal maps produced with our method. Next to each 3D texture image is a visualization of the planar region over which the surface is parameterized. The (cut) Camel demonstrates a constrained complex boundary shape; the Max Planck parameterization (also with cuts) shows a straightline bounded parameter region suitable for good texture packing; the face mask demonstrates natural boundary conditions; and the lion head mapping to a disk.*

Abstract

We introduce a novel method for the construction of discrete conformal mappings from (regions of) embedded meshes to the plane. Our approach is based on *circle patterns*, *i.e.*, arrangements of circles—one for each face—with prescribed intersection angles. Given these angles the circle radii follow as the *unique* minimizer of a convex energy. The method has two principal advantages over earlier approaches based on discrete harmonic mappings: (1) it supports very flexible boundary conditions ranging from natural boundaries to control of the boundary shape via prescribed curvatures; (2) the solution is based on a convex energy as a function of *logarithmic* radius variables with simple explicit expressions for gradients and Hessians, greatly facilitating robust and efficient numerical treatment. We demonstrate the versatility and performance of our algorithm with a variety of examples.

CR Categories: G.1.0 [Numerical Analysis]: General—Numerical Algorithms; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve,surface, solid and object representations; Geometric algorithms, languages, and systems.

Keywords: Conformal parameterizations; discrete differential geometry; circle patterns; discrete analytic functions; meshing; texture mapping

1 Introduction

Surfaces are often represented as collections of samples with connectivity, typically in the form of a simplicial mesh. It is natural and convenient to use the implied piecewise linear mesh as the basis for formulating a variety of computational algorithms, such as parameterization problems or the solution of partial differential equations for purposes of simulation. In this paper we argue that when it comes to computing conformal structures, *e.g.*, conformal parameterizations of surfaces, circles can be a far better basis upon which to formulate the underlying relationships and consequent algorithms (see Figure 2). In particular we advocate the formulation of the *discrete conformal mapping*[1] problem in terms of circles and the angles with which they intersect, so called *circle patterns*.

The idea of using circles to capture a discrete notion of conformality goes back to a conjecture of Thurston's [1985] who posited that one may approximate the Riemann mapping[2] from a given region in the plane to the unit disk through a sequence of increasingly fine, regular (hexagonal) *circle packings*. This conjecture was later proven correct by Rodin and Sullivan [1987]. Circle packings assign a circle to each vertex, with pairwise tangency for each edge in the mesh. A numerical algorithm for the construction of such mappings, based on iterative adjustment of circle radii, was pro-

[1] In this article we use the term "discrete conformal map" for maps between meshes that are close to angle preserving.

[2] The Riemann Mapping Theorem asserts that there exists a unique (up to Möbius transformations of the unit disk to itself) conformal map from any region in the plane (open, connected and simply connected, not the whole plane) onto the unit disk.

posed by Thurston [1980] and improved and realized by Collins and Stephenson [2003]. Unfortunately, circle *packings* yield mappings which depend only on the *combinatorics* of the original mesh, while we are seeking methods which depend on the *geometry* of the mesh. One possible avenue to remedy this shortcoming is to use patterns of non-intersecting circles [Bowers and Hurdal 2003]. Unfortunately there exists little theory concerning these patterns and in empirical practice solutions cannot always be found.

In contrast, circle *patterns*, which associate a circle with each *face* in the original mesh provide an opportunity to incorporate the intrinsic geometry of the original mesh: each edge is assigned an angle $\theta \in (0, \pi)$ which corresponds to the intersection angle of the two incident face circles. A recent theorem of Bobenko and Springborn [2004] characterizes such circle patterns as the *unique* minimizer of a convex energy expressed in terms of *logarithmic* radius variables (and the given edge angles). Simple, explicit expressions for the energy, its gradient, and Hessian are available and greatly facilitate an efficient implementation.

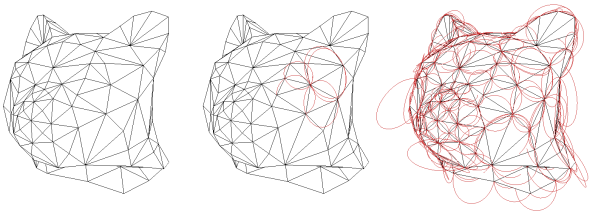


Figure 2: *Typically a triangle mesh is understood as the piecewise linear interpolation of given vertex coordinates induced by the connectivity of the mesh (left). Alternatively we may also think of the vertices as the unique loci where incident triangle circumcircles intersect (middle;right). The latter point of view is more appropriate for formulating relationships of conformal geometry.*

Contributions We describe a robust and efficient numerical procedure for the construction of discrete conformal mappings. In the first stage of the algorithm, we compute edge labels that are close to the circumcircle intersection angles in the original mesh and may serve as circumcircle intersection angles of a planar Delaunay mesh. This requires solving a quadratic programming problem. In the second stage, we use the theory of Bobenko and Springborn [2004] to construct the planar Delaunay mesh with the given intersection angles from the first stage. Here we have to minimize a convex function subject to *no constraints*. The variables are the logarithmic circumcircle radii. We modify the energy of Bobenko and Springborn (without compromising the underlying theory) to provide a simple and uniform treatment of boundary conditions (see Figure 3). With these we can provide both natural boundaries and detailed control over the boundary shape (see Figure 1). The parameter mappings are always locally injective. They may fail to be globally injective due to self-overlap of the boundary of the parameter domain. However, this can be avoided since we can prescribe the boundary curvature κ: if for any sequence of consecutive boundary vertices the sum of κs is larger or equal $-\pi$, then there can be no overlap and the method is guaranteed to produce a global embedding. The resulting flat triangulations are of high quality as they are always triangulations of a (not necessarily convex) planar domain with all interior edges satisfying the local Delaunay criterion.

1.1 Related Work

Most approaches to the construction of conformal mappings for meshes have relied on *discretizations of continuous formulations*. First order finite difference approximations of the Cauchy-Riemann

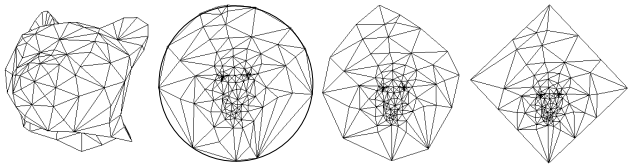


Figure 3: *A simple mesh (left) mapped to the plane using circle patterns. Boundary shape is controlled through appropriate curvature conditions. Examples: disk boundary; natural boundary; and rectangular boundary (left to right).*

equations were used by Levy *et al.* [2002]. The same equations, the so-called "cotan formula" [Pinkall and Polthier 1993], also result when considering a discrete variational Ansatz based on mesh invariants [Desbrun et al. 2002] or when deriving discrete holomorphy [Mercat 2001] from first principles using Discrete Exterior Calculus (DEC) methods [Hirani 2003]. Such approaches, for example, have been used to compute discrete approximations of Riemann structures for general meshes by Gu and co-workers (see [Gu and Yau 2003] and the references therein).

A great advantage of these methods is that they require only the solution of simple linear systems. However, due to negative cotan weights solutions may lack local injectivity. More troublesome is the lack of flexible boundary conditions: Dirichlet conditions introduce non-conformal distortion (see Figure 4), while natural boundaries provide essentially no control over the boundary.

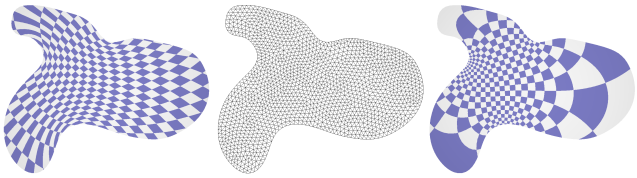


Figure 4: *Harmonic parameterization (left) of a region (middle) to the disk with Dirichlet boundary conditions (vertices are mapped to a k-gon with matching secant edge lengths) compared to our approach which sets appropriate* angle *conditions at the boundary (right).*

A completely different Ansatz to the construction of conformal maps is based on circle *packing*. Continuous conformal mappings can be characterized as mapping infinitesimal circles to infinitesimal circles. Circle packings replace infinitesimal circles with finite circles. In the limit of refinement the continuous conformal maps are recovered [Rodin and Sullivan 1987]. Collins and Stephenson [2003] have implemented these ideas in their software `CirclePack`. The disadvantage of using circle packings (with tangent circles) is that they depend only on the combinatorics of the original mesh. In particular, if one starts with a planar mesh and parameterizes it, the result is not the original mesh. This is in contrast to our approach. Given a triangulation of a region in the plane satisfying the empty circumcircle property, one simply assigns the observed intersection angles of circumcircles (at the boundary one adds infinite circles, *i.e.*, straight lines) to each edge and our variational approach will return the identity map as a conformal map of the region to itself.

An extension of Stephenson's original circle packing scheme that takes the geometry of the original mesh into account is based on patterns of non-intersecting circles, so called *inversive distance circle packings* [Bowers and Hurdal 2003]. (Non-intersecting circles have imaginary "intersection angles.") However, virtually nothing is known regarding the existence and uniqueness of inversive distance packings.

The first variational principle for circle packings was presented in

a seminal paper by Colin de Verdière [1991]. The variables are the circle radii. However, a closed formula is presented only for the derivative of the energy, not for the energy itself. Since then, different variational principles for circle *packings* [Brägger 1992] as well as circle *patterns* [Rivin 1994], [Leibon 2002] were discovered. In these, the variables are the angles of a triangulation, subject to numerous linear constraints (one per edge and one per face for Rivin's energy. Leibon's energy deals with patterns in the hyperbolic plane.) In this paper, we use the most general functional which was given by Bobenko and Springborn [2004]. In their setup, the variables are logarithmic circle radii. Most importantly, they are not subject to any constraints.

Most closely related to our approach is the work of Sheffer and de Sturler [2000]. They flatten a given mesh by formulating a constrained quadratic minimization problem which seeks to find angles at the corners of triangles which are close to desired angles in a weighted L_2 norm. The constraints capture the angle sum conditions at all faces (sum of angles $= \pi$) and vertices (sum of angles $= 2\pi$) as well as a non-linear condition on the product of sines of angles. The resulting minimization problem has local minima and does not have a unique solution. The resulting mappings are of excellent quality but only natural boundary conditions are provided. Their approach is similar in spirit to ours as we also optimize angles. We discuss the similarities and differences of both schemes in Section 2.3.

2 Circle Patterns and Discrete Conformal Maps

For computational problems in Euclidean geometry the use of triangles as a basic primitive is convenient and natural, since triangles are the basic invariant "building blocks" of Euclidean geometry. When one is interested in *conformal* geometry the picture changes. The basic invariants of conformal geometry are circles and the angles they make with one another. In the case of triangle meshes these differing points of view are naturally compatible. For example, the vertices in a triangle mesh are the unique loci where the circumcircles of the incident triangles intersect (see Figure 2). Similarly the empty circumcircle property, which corresponds to non-negative intersection angles between circumcircles incident on an edge, is a defining feature of Delaunay triangulations. While we may use triangles for tasks such as interpolation and rendering, conformal relationships between vertices are better captured through expressions involving the circumcircles they define and the angles these circles make with one another. A benefit of this different point of view is that much mathematical machinery from conformal geometry carries over to the discrete computational setting. For example, existence and uniqueness properties of conformal maps are reflected in the existence and uniqueness properties of circle patterns.

We begin this section by defining circle patterns in the plane and describe their characterization as minimizers of a variational energy. The latter forms the basis for our approach. While we only deal with triangle meshes here, the theory extends to polyhedral meshes and also to circle patterns with cone singularities [Bobenko and Springborn 2004].

2.1 Circle Patterns

Consider a Delaunay triangulation $\mathscr{T} = (V, E, T)$ of finitely many points $P = \{p_i\}$ in the plane. Here $V = \{v_i\}$, $E = \{e_{ij}\}$, and $T = \{t_{ijk}\}$ denote the sets of vertices, edges, and triangles respectively with p_i the point position of vertex v_i. From this Delaunay triangulation, we may read off the following edge weights

$$\forall e_{ij} \in E : \theta_e = \begin{cases} \pi - \alpha_{ij}^k - \alpha_{ij}^l & \text{for interior edges} \\ \pi - \alpha_{ij}^k & \text{for boundary edges} \end{cases}, \quad (1)$$

where α_{ij}^k (and α_{ij}^l) are the angle(s) opposite e_{ij} in the adjacent triangle(s) t_{ijk} (and t_{jil}). The θ-weight of an interior edge is the (exterior) intersection angle of the circumcircles of the incident triangles (see Figure 5). The θ-weight of a boundary edge is the intersection

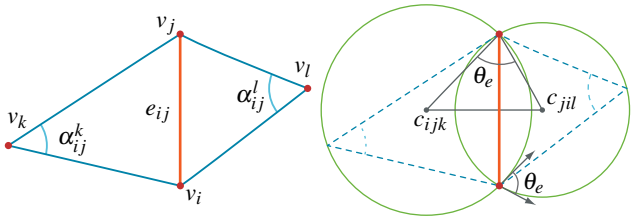


Figure 5: *Notation: The angles α_{ij}^k and α_{ij}^l opposite a given edge e_{ij} and incident on a particular vertex v_k respectively v_l. The edge angle θ_e denotes the exterior intersection angle of the incident circumcircles or equivalently the angle between the two radii at v_i (or v_j).*

angle of the circumcircle of the incident triangle with the straight line containing the boundary edge. (View this line as a circle with infinite radius.)

Assume that the Delaunay triangulation is unique (points are in general position). Then

$$\forall e_{ij} \in E : 0 < \theta_e < \pi. \quad (2)$$

(Otherwise, some θ_e may be 0.) For an interior vertex v_i, the sum of edge weights on the incident edges is 2π:

$$\forall v_i \in V_{\text{int}} : \sum_{e \ni v_i} \theta_e = 2\pi, \quad (3)$$

while for a boundary vertex v_i, the defect

$$\forall v_i \in V_{\text{bdy}} : \kappa_i = 2\pi - \sum_{e \ni v_i} \theta_e \quad (4)$$

is the curvature angle of the polygonal boundary at that vertex (see Figure 6). Since the Delaunay triangulation triangulates the con-

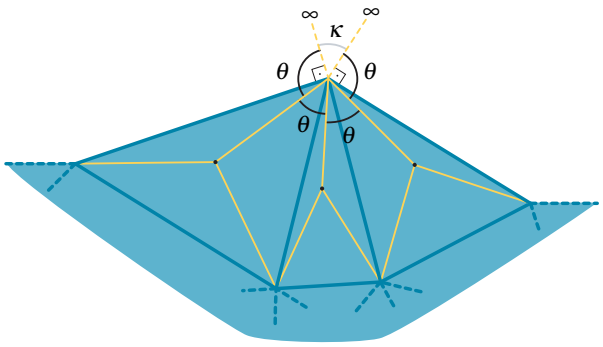


Figure 6: *The θ angle sum at the boundary contains the discrete curvature term κ.*

vex hull of the sites, $0 \leq \kappa_i < \pi$. However, we want to consider a slightly more general setup: *Instead of a Delaunay triangulation,*

we may start with a flat PL-surface that is topologically a disk and that is triangulated in such a way that the edge weights θ satisfy the local Delaunay condition (Equation 2). That is, we allow "Delaunay triangulations" of non-convex regions with polygonal boundary. Hence, κ_i may be negative, and we speak of *circle patterns* instead of Delaunay triangulations.

Now the idea is to reconstruct a circle pattern from its abstract triangulation and the intersection angles.

Circle Pattern Problem *Given* an abstract triangulation $\mathscr{T}$ of a topological disk and a function $\theta \in \mathbb{R}^E$ on the edge set E that satisfies Equation 2 and the angle sum condition for interior vertices (Equation 3), *find* a circle pattern that is combinatorially equivalent to $\mathscr{T}$ and has the given edge weights θ.

The circle pattern problem has a solution (unique up to scale) *if and only if* a coherent angle system exists [Rivin 1994; Bobenko and Springborn 2004]. A *coherent angle system* is an assignment of angles $\hat{\alpha}_{ij}^k$ for all triangles such that

(i) they are all positive
$$\hat{\alpha}_{ij}^k > 0,$$

(ii) they sum to π in each triangle
$$\forall t_{ijk} \in T : \hat{\alpha}_{ij}^k + \hat{\alpha}_{jk}^i + \hat{\alpha}_{ki}^j = \pi,$$

(iii) they satisfy Equations 1 (with $\hat{\alpha}$ instead of α) for the *given* θ-weights.

The solvability question for a circle pattern problem is therefore reduced to a linear feasibility problem with $3|T|$ variables, $3|T|$ inequality constraints and $|T|+|E|$ equality constraints.

Conditions (i) and (ii) imply that all $\hat{\alpha}_{ij}^k < \pi$. Condition (iii) implies that the $\hat{\alpha}_{ij}^k$ sum to 2π around interior vertices and to $\pi - \kappa_i$ for boundary vertices (with κ_i defined by Equation 4). This may give the false impression that finding a coherent angle system is equivalent to solving the circle pattern problem—in the sense that one could construct triangles with angles $\hat{\alpha}_{ij}^k$ and lay them out. This is not so. The angles determine the triangles only up to scale. In general it is not possible to determine the size of each triangle in such a way that they all fit together. This observation is also what lead Sheffer and de Sturler [2000] to add their non-linear constraints on the product of sines of triangle angles.

Conditions (i), (ii) and (iii) combined imply that a necessary condition for the solvability of a circle pattern problem is that

$$\sum_{v_i \in V_{\text{bdy}}} \kappa_i = 2\pi, \tag{5}$$

with κ_i defined by Equation 4.

Local Geometry of an Edge To elucidate the role of terms which make up the variational energy characterizing the solution to the circle pattern problem we consider the local geometry around a given edge (see Figure 7). The basic building blocks of the parameterization are the *kites* formed by the endpoints of an edge (v_i, v_j) and the incident face circumcircle centers c_{ijk} and c_{jil} for triangles t_{ijk} and t_{jil} respectively (at the boundary t_{jil} is missing). Since all relations are scale invariant it is convenient to introduce the *logarithmic* radius variables $\rho_t = \log r_t$ for $t \in T$. With these definitions the angle φ_e^k induced at c_{ijk} by e_{ij} follows as

$$\varphi_e^k = \begin{cases} f_e(x) = \operatorname{atan2}(\sin\theta_e, e^x - \cos\theta_e) & e \in E_{\text{int}} \\ \pi - \theta_e & e \in E_{\text{bdy}} \end{cases} \tag{6}$$

where $x = \rho_{ijk} - \rho_{jil}$ (see Figure 7). The condition that every face

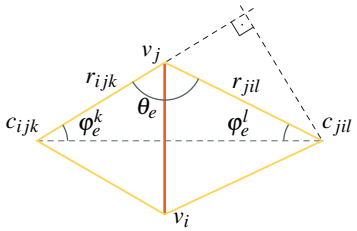


Figure 7: *Geometry around an edge (left). The kite formed by the edge endpoints and the incident face circumcircle centers (c_{ijk}, c_{jil}) allows us to determine φ_e^k as a function of the given θ_e and unknown radii r_{ijk} and r_{jil} (see Equation 6).*

in the parameterization should be flat constrains the ρ_t as functions of the θ_e through the system of non-linear equations

$$\forall t \in T : 0 = 2\pi - \sum_{e \in t} 2\varphi_e^t, \tag{7}$$

i.e., summing around all edges incident on a face the resulting total angle must be 2π.

The basic idea of the variational energy formulation of Bobenko and Springborn now is to give an energy $S(\rho)$ for which the face flatness Equations 7 are equivalent to the vanishing of the energy gradient, $\nabla_\rho S = 0$.

Variational Characterization of Circle Patterns To arrive at the desired energy Bobenko and Springborn used the fact that

$$F_e(x) = \int_{-\infty}^{x} f_e(\xi)\, d\xi = \operatorname{Im}\operatorname{Li}_2(e^{x+i\theta_e}),$$

where

$$\operatorname{Li}_2(x) = -\int_0^x \frac{\log(1-\xi)}{\xi}\, d\xi \overset{|x|\le 1}{=} \sum_{k=1}^{\infty} \frac{x^k}{k^2}$$

denotes the dilogarithm function. The imaginary part of the dilogarithm function of a complex argument can be expressed in terms of a 2π-periodic real function (Clausen's integral) that can be computed efficiently with high accuracy. Using ρ_k and ρ_l as shorthand for ρ_{ijk} (respectively ρ_{jil}) the energy is

$$\begin{aligned} S(\rho) = \sum_{e \in E_{\text{int}}} \big(&\operatorname{Im}\operatorname{Li}_2(e^{\rho_k - \rho_l + i\theta_e}) + \operatorname{Im}\operatorname{Li}_2(e^{\rho_l - \rho_k + i\theta_e}) \\ &- (\pi - \theta_e)(\rho_k + \rho_l) \big) \\ - \sum_{e \in E_{\text{bdy}}} 2(\pi - \theta_e)\rho_k &+ 2\pi \sum_{t \in T} \rho_t. \end{aligned} \tag{8}$$

The gradient of this energy with respect to ρ_k is

$$\frac{\partial S}{\partial \rho_k} = 2\pi - \sum_{\{e \in k\} \cap E_{\text{int}}} 2 f_e(\rho_k - \rho_l) - \sum_{\{e \in k\} \cap E_{\text{bdy}}} 2(\pi - \theta_e), \tag{9}$$

giving us the desired equivalence of $\nabla_\rho S = 0$ and Equation 7. The Hessian of the energy is

$$d\rho^T (\operatorname{Hess} S) d\rho = \sum_{e \in E_{\text{int}}} \frac{\sin\theta_e}{\cosh(d\rho_k - d\rho_l) - \cos\theta_e} (d\rho_k - d\rho_l)^2. \tag{10}$$

In particular from this expression we can see that the energy is convex except along the scaling "direction," *i.e.*, the Hessian has a null space spanned the constant vector $d\rho = (1,1,1,\ldots)$ and is otherwise positive. (This immediately implies the uniqueness of solutions of the circle pattern problem up to scale.)

2.2 Algorithm

There are three basic stages to the algorithm: (1) setting the θ angles; (2) minimizing the energy; and (3) generating the layout. We discuss these in turn.

2.3 Edge Angles

As a first step of the algorithm, θ_e angles need to be assigned to all edges of the mesh. These must satisfy the bounds constraints (Equation 2), sum conditions (Equation 3) at interior vertices, and, in the case of prescribed boundary curvatures, the boundary curvature conditions (Equations 4 and 5). Last but not least, a coherent angle system must exist for them. Of course, the θ angles should also reflect the conformal structure of the original mesh as well as possible. Let α_{ij}^k denote the angles in the original mesh. Ideally, one would like to assign $\theta_e = \pi - \alpha_{ij}^k - \alpha_{ij}^l$, where α_{ij}^k and α_{ij}^l are the angles opposite an interior edge e (and $\theta_e = \pi - \alpha_{ij}^k$ for a boundary edge.) Then the intersection angles in the circle pattern would be the same as the intersection angles of the circumcircles in the mesh. This is too much to ask, though, because the conditions on θ_e will be violated (after all the original mesh is not flat).

Our aim is to find angles $\hat{\alpha}_{ij}^k$ close to α_{ij}^k that can serve as a coherent angle system for the θ-angles that we read off in the usual way. To this end, we minimize the objective function

$$Q(\hat{\alpha}) = \sum |\hat{\alpha}_{ij}^k - \alpha_{ij}^k|^2$$

subject to the following constraints:

- positivity: $\forall \hat{\alpha}_{ij}^k : \hat{\alpha}_{ij}^k > 0$,
- local Delaunay condition: $\forall e_{ij} \in E_{\text{int}} : \hat{\alpha}_{ij}^k + \hat{\alpha}_{ij}^l < \pi$,
- triangle sum condition: $\forall t_{ijk} \in T : \hat{\alpha}_{ij}^k + \hat{\alpha}_{jk}^i + \hat{\alpha}_{ki}^j = \pi$,
- vertex sum condition: $\forall v_k \in V_{\text{int}} : \sum_{t_{ijk} \ni v_k} \hat{\alpha}_{ij}^k = 2\pi$.

If we want natural boundary conditions (see for example the face in Figure 1 and the lion and Max Planck examples in Figures 9, 11), we add the constraint

- natural boundary condition: $\forall v_k \in V_{\text{bdy}} : \sum_{t_{ijk} \ni v_k} \hat{\alpha}_{ij}^k < 2\pi$.

If we want to prescribe the boundary curvature κ at boundary vertices (see the constrained boundary for the camel and the straight-line layout for Max Planck in Figure 1), we add the constraint

- prescribed boundary curvature:
  $\forall v_k \in V_{\text{bdy}} : \sum_{t_{ijk} \ni v_k} \hat{\alpha}_{ij}^k = \pi - \kappa_k$.

We solve this quadratic minimization problem with linear inequality constraints (on the $\hat{\alpha}_{ij}^k$) with the software [Mosek 2005]. Since both the bounds constraints and the objective are convex and the additional constraints linear we have experienced no difficulty finding solutions efficiently even for large meshes (see Section 3).

Then we set $\theta_e = \pi - \hat{\alpha}_{ij}^k - \hat{\alpha}_{ij}^l$ on interior edges and $\theta_e = \pi - \hat{\alpha}_{ij}^k$ on boundary edges and proceed to the next stage of energy minimization.

It is theoretically possible that the constraints are not feasible. In the cases of natural boundary conditions and mapping to the disk this is due to the (counterintuitive) fact that there exist triangulations of the topological sphere that cannot be realized as convex polyhedra with vertices on the unit sphere; see, *e.g.*, Grünbaum [2003]. These triangulations are rather special and we do not expect to encounter them in practice. In any case, it can be shown that 4 to 1 refinement applied once leads to a legal triangulation. In the case of prescribed boundary curvature, it may happen that the constraints become infeasible. We have not encountered any problems, but we have also not put any effort into exploring just how far one can go. (Our most complex example along these lines is the texture boundary for the Camel with numerous prescribed curvatures all around the boundary.)

2.3.1 Mapping to the Disk

To map a mesh to the unit disk, we first pick a vertex at the boundary of the mesh and remove it together with all adjacent faces (see Figure 8). Now we map the resulting mesh to a convex polygon

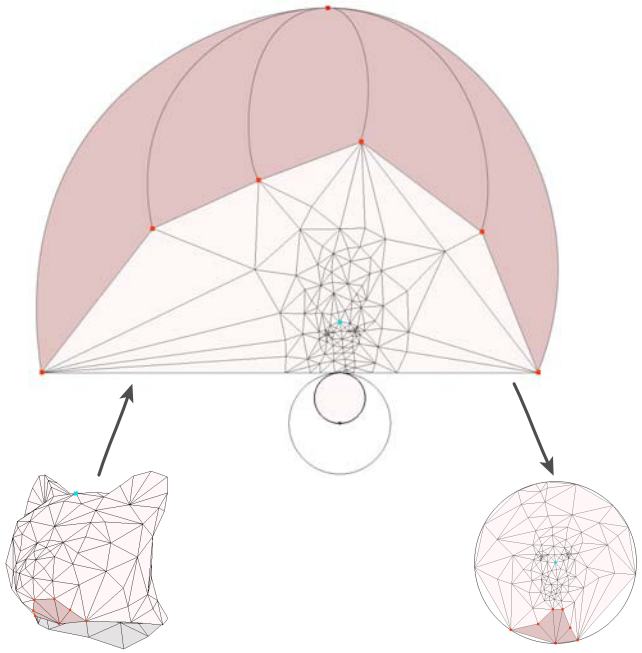


Figure 8: *Mapping a mesh to the disk. (1) Remove a boundary vertex together with all adjacent faces (dark pink). (2) Map the mesh to a convex polygon where all the original boundary edges lie on a straight line (top). The removed vertex (which should be imagined at infinity) and its incident triangles are shown schematically (dark pink). (3) Invert in a circle and reinsert the missing vertex to complete the mesh (bottom right). In this inversion we get to pick a vertex which will map to the center of the disk. In our example this vertex is the marked vertex between the ears of the cat.*

in the plane with prescribed boundary curvature. At boundary vertices that were not adjacent to any of the removed faces, we fix the curvature to be 0. At the others, we bound the curvature to be positive. These conditions ensure that the original mesh boundary will be mapped to a straight line (Figure 8, top). Now we reflect in a circle whose center lies on the other side of that line. This maps

the boundary line to a boundary circle. Finally we reinsert the removed vertex at the center of the reflection circle (it lies on the boundary circle) and complete the mesh (Figure 8, bottom right). Note that the circle reflection maps the other edges of the polygon to the circumcircles of the reinserted triangles. If we want to have a particular interior vertex at the center of the circular map, this can be achieved by an appropriate choice of reflection circle (or equivalently by a Möbius transformation).

In the polygonal mesh (Figure 8, top), the dynamic range of the edge lengths is typically large. This was initially a cause for concern, but it turned out to be harmless. We give a rough argument why this should be so. Suppose the mesh has n vertices. Then the number of boundary edges will be roughly $n^{\frac{1}{2}}$. Suppose that in the final circular map the boundary vertices are evenly spaced on the boundary circle. After inversion on a circle that sends one of these vertices to infinity, one finds that the largest (finite) edge on the boundary line is proportional to $n^{\frac{1}{2}}$ while the smallest edge is proportional to $n^{-\frac{1}{2}}$. Hence the ratio is n. If $n = 10^6$ we would expect to loose about 6 out of 16 double precision digits. Figure 11 shows an example of a larger mesh mapped to the disk. We experienced no numerical difficulties in this procedure.

Discussion Angle optimization is also at the heart of the work of Sheffer and de Sturler [2000], who formulate their flattening problem as one which minimizes the (weighted) least squares deviation of the measured α_{ij}^k (scaled to satisfy the angle sum condition around a vertex) from the realizable $\hat{\alpha}_{ij}^k$ with flatness sum conditions for each triangle and each vertex. Unfortunately, when using the $\hat{\alpha}_{ij}^k$ to determine the final shape of the triangles in the flat mesh, one must additionally include non-linear conditions on the quotients of sines of $\hat{\alpha}_{ij}^k$ around each vertex (due to the law of sines). This additional non-linear condition makes the minimization problem numerically much harder since it becomes non-convex (for recent significant progress in the numerical treatment see [Sheffer et al. 2004]). Due to the lack of convexity, it cannot be expected that there is only one local minimum.

Our method works in two stages. First we change the angles of the triangles, but only so much that we can read off valid θ-angles. The deviation from the ideal angles is measured with a quadratic objective just as in the work of Sheffer and de Sturler. The critical observation is that the sum of two α-angles across an edge is a conformal invariant (because it measures the angle of intersection between the circumcircles), while the α-angles themselves are not. In other words, the best one can hope for in a discrete conformal mapping is conservation of θ-angles, not α-angles. Formulating the problem in terms of given θ-angles then leads to a simple convex, and thus *unique*, minimization problem: the final circle pattern in the plane is completely determined by the θ-angles.

An alternative approach for the construction of mappings to the disk in the context of discrete harmonic mappings (cotan formula) was proposed by Jin and co-workers [2004]. They "welded" a mesh with its double at the boundary to create a sphere topology and then proceeded to map to the sphere. Enforcing symmetry in this mapping turns the original boundary into an equator. They had to effectively double the degrees of freedom while in our approach the number of degrees of freedom is (essentially) constant. As they observed, being able to map to a canonical region such as a disk one can effectively map from any region to any other through composition of one map to the disk with the inverse of another map to the disk (though the meshes in general do not match up and some interpolation in the disk is required).

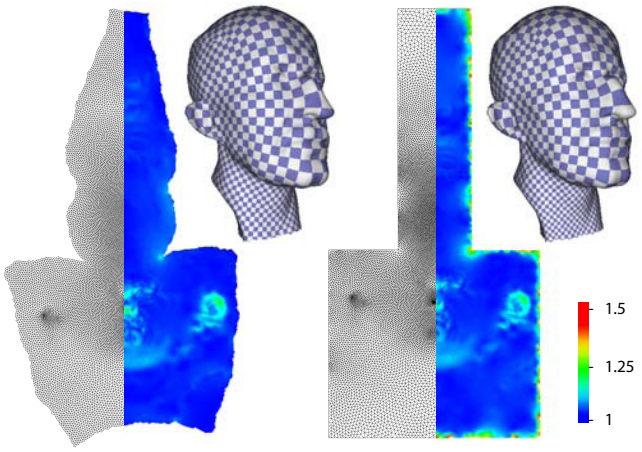


Figure 9: *Comparison of two different mappings of the cut Max Planck model: natural boundaries and a straightline boundary. Error plots indicate the quasiconformal distortion between original and mapped triangles.*

2.4 Energy Minimization

With a given valid assignment of θ_e for all edges the energy minimization is straight forward (using Equations 8, 9, and 10). We have relied on a black box energy minimizer [Mosek 2005] with excellent results (see Section 3).

2.5 Layout Generation

Once all radii have been determined the length of each edge follows easily from a *local* computation

$$|e_{ij}| = 2r_k \sin \varphi_e^k = 2r_l \sin \varphi_e^l$$

(see Figure 7). We begin the layout with some interior edge, starting it at the origin and orienting it along the positive x-axis and push it onto an empty stack. When popping an edge off the stack we lay out any vertices not yet laid out in the incident triangles (at most two). This results in at most four edges being finished (second end point fixed); such edges are pushed onto the stack. The process repeats until the stack is empty. There may be concern that such a procedure might lead to cumulating error as one proceeds. We have found consistently that we achieve accuracies on the order of 10^{-8} with double precision data (for example for the Camel, Max Planck, and Lion datasets). Should accuracy of the layout become an issue we recommend the procedure given in [Sheffer et al. 2004], which solves for all vertex positions simultaneously with a simple (least squares induced) linear system.

3 Results

Figure 1 demonstrates different boundary shapes that can be obtained by our method. The Max Planck head is parameterized by a simple polygonal region. This was achieved by prescribing the boundary curvature of the parameter domain. It was set to zero at all boundary vertices, except for eight designated corner vertices, where it was set to $\pm\pi/2$. The parameterized camel shows that more complicated boundary shapes can also be achieved by prescribing the boundary curvature. Here the parameter domain is essentially a polygonal region with rounded corners. Both the Max

Planck mesh and the camel mesh were cut before parameterization. We do nothing to ensure continuity across the cut and, unsurprisingly, there is none. The face was parameterized with natural boundary conditions and the lion head was mapped to a disk. The following table shows timings for the angle optimization and energy minimization. The layout times are negligible (∼0.5s). The timings were measured on a 3.6GHz Pentium IV Xeon.

Model	Faces	Angles	Energy
Max Planck	37.9K	26s	9s
Camel	77.7K	56s	17s
Face	12.6K	7s	2s
Lion Head	39.6K	28s	8s

We claim that our discrete conformal maps are close to angle preserving. A measure for the conformality of a map is the *quasiconformal distortion:* the ratio of the larger to the smaller eigen value of the Jacobian matrix. It is at least 1 and equal to 1 everywhere only for conformal maps. Figures 9 and 11 show the quasiconformal distortion of our discrete conformal maps for different meshes and different boundary conditions. For the Max Planck head with natural boundary conditions, the average and maximal values are 1.02 and 1.35. With the polygonal boundary conditions the maximum distortion goes up to 4.06, but the high distortion is concentrated at the boundary and spreads very little inwards. Even the example of the highly convoluted lion head (Figure 11) shows very low conformal distortion in most places with a few hot spots of high distortion. Because the natural boundary shape is already fairly round, the difference between natural boundary conditions and disk boundary is not so pronounced.

Figure 10 shows the results of an experiment to see how sensitive our method is to abruptly differing sampling rates. The geometry of the head is symmetric while the sampling rate doubles at the right/left symmetry line. Examining the flattened mesh (using natural boundary conditions) we observe that the left/right symmetry is preserved (see also the resulting texture mapping on the original surface).

4 Conclusion

We have presented a new method to parameterize surface meshes. It is based on the mathematical theory of *circle patterns* and produces discrete conformal maps. The shape of the boundary may be determined by natural boundary conditions or by prescribing the curvature of the boundary. This affords a high degree of control over the boundary shape ranging from disks and simple polygonal outlines to more complex boundary arrangements. The examples show that our method is efficient, provides good parameterizations even for large and complex meshes, and that the result is insensitive to the way the surface is triangulated.

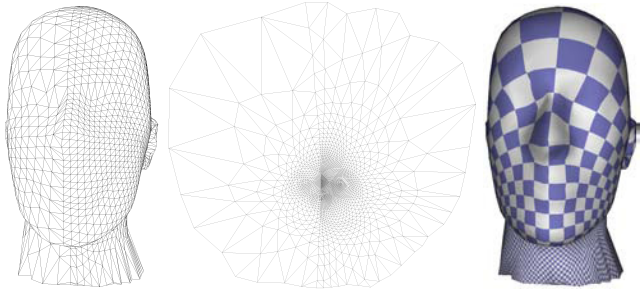


Figure 10: *Our method is robust to varying sampling rates. Here a symmetric (left/right) geometry sampled at different rates. The parameterization (with natural boundary) maintains the symmetry of the geometry.*

Our algorithm works in three stages: first, we solve a quadratic programming problem to obtain intersection angles. These are the input for the second stage that consists in the minimization of a convex energy. The output of this stage (the radii of the circumcircles) and the intersection angles determine the shape of all triangles which are then laid out. In every stage, the solution is unique and depends continuously on the input.

At the moment, we use general purpose solvers in the first two stages, and a very simple layout algorithm in the third. Efficiency could be improved by customizing and tuning the minimizers, and by switching to hierarchical methods [Sheffer et al. 2004]. Their sophisticated layout scheme—it minimizes the global layout error—can also be used without change as the third stage in our algorithm.

When we parameterize a mesh with prescribed boundary curvature, we have to set the curvature at each boundary vertex. At present, we do this manually. A user friendlier graphical interface is desirable. Since it is most likely that the performance of our method can be improved further, it is not unreasonable to envision an interactive interface that lets the user manipulate the parameter domain while the parameterization is incrementally recomputed.

The most exciting avenue for future work concerns the use of variational methods for the construction of discrete conformal mappings to the sphere and for surfaces of higher genus. Energy functionals for circle patterns on the sphere as well as hyperbolic space exist and can be used in a manner similar to our approach presented here. There are both additional difficulties (the sphere functional is *not* convex) and opportunities: the hyperbolic circle pattern case does *not* require a surface with higher genus to be cut open first. Instead one can solve the energy minimization directly and only after the solution has been found (through a layout in the hyperbolic plane) pick a suitable fundamental domain.

Acknowledgments This work was supported in part by NSF (DMS-0220905, DMS-0138458, ACI-0219979), DFG (Research

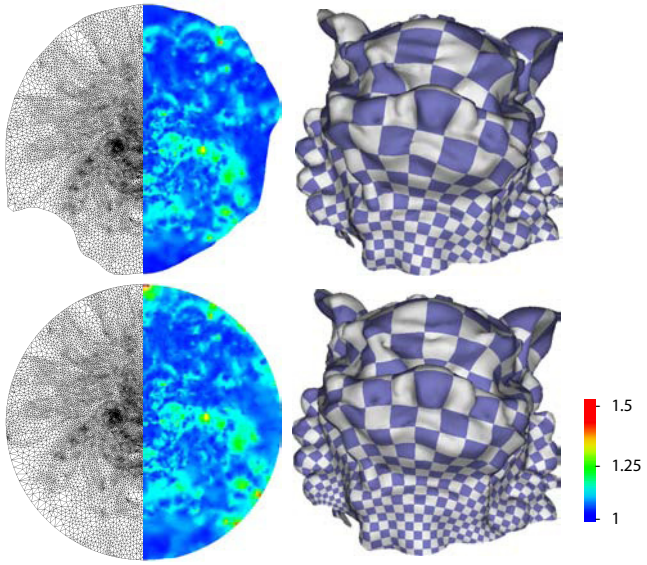


Figure 11: *Comparison of two different mappings of the Lion data set: natural boundaries and disk boundary. Error plots indicate the quasiconformal distortion between original and mapped triangles.*

Center MATHEON "Mathematics for Key Technologies," Berlin), DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Alias, and Pixar. Special thanks to Alexander Bobenko, Mathieu Desbrun, Ilja Friedel, and Cici Koenig.

References

BOBENKO, A. I., AND SPRINGBORN, B. A. 2004. Variational Principles for Circle Patterns and Koebe's Theorem. *Transactions of the American Mathematical Society 356*, 659–689.

BOWERS, P. L., AND HURDAL, M. K. 2003. Planar Conformal Mappings of Piecewise Flat Surfaces. In *Visualization and Mathematics III*, Springer-Verlag, Berlin, H.-C. Hege and K. Polthier, Eds., Mathematics and Visualization, 3–34. Papers from the 3rd International Workshop held in Berlin, May 22–25, 2002.

BRÄGGER, W. 1992. Kreispackungen und Triangulierungen. *Enseign. Math. 38*, 201–217.

COLIN DE VERDIÈRE, Y. 1991. Un principe variationnel pour les empilements de cercles. *Invent. Math. 104*, 655–669.

COLLINS, C., AND STEPHENSON, K. 2003. A Circle Packing Algorithm. *Computational Geometry: Theory and Applications 25*, 233–256.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic Parameterizations of Surface Meshes. *Computer Graphics Forum (Proceedings of Eurographics 2002) 21*, 3, 209–218.

GRÜNBAUM, B. 2003. *Convex polytopes*, second ed., vol. 221 of *Graduate Texts in Mathematics*. Springer-Verlag, New York.

GU, X., AND YAU, S.-T. 2003. Global Conformal Surface Parameterization. In *Symposium on Geometry Processing*, Eurographics/ACM SIGGRAPH, 127–137.

HIRANI, A. N. 2003. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology.

JIN, M., WANG, Y., YAU, S.-T., AND GU, X. 2004. Optimal Global Conformal Surface Parameterizations. In *Proceedings of IEEE Visualization*, IEEE, 267–274.

LEIBON, G. 2002. Characterizing the Delaunay Decompositions of Compact Hyperbolic Surfaces. *Geom. Topol. 6*, 361–391.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Transactions on Graphics 21*, 3, 362–371.

MERCAT, C. 2001. Discrete Riemann Surfaces and the Ising Model. *Communications in Mathematical Physics 218*, 1, 177–216.

MOSEK, 2005. Constrained Quadratic Minimization Software. http://www.mosek.com/. Version 3.1r42.

PINKALL, U., AND POLTHIER, K. 1993. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics 2*, 1, 15–36.

RIVIN, I. 1994. Euclidean Structures on Simplicial Surfaces and Hyperbolic Volume. *Annals of Mathematics 139*, 3, 553–580.

RODIN, B., AND SULLIVAN, D. 1987. The Convergence of Circle Packings to the Riemann Mapping. *J. Differential Geom. 26*, 2, 349–360.

SHEFFER, A., AND DE STURLER, E. 2000. Surface Parameterization for Meshing by Triangulation Flattening. In *Proceedings of the 9th International Meshing Roundtable*, Sandia National Labs, 161–172.

SHEFFER, A., LÉVY, B., MOGILNITSKI, M., AND BOGOMYAKOV, A. 2004. ABF++: Fast and Robust Angle Based Flattening. *ACM Transactions on Graphics*. To appear.

THURSTON, W. P. 1980. The Geometry and Topology of Three-Manifolds. Available at http://www.msri.org/publications/books/gt3m/.

THURSTON, W. P. 1985. The finite Riemann mapping theorem. Invited talk at the symposium on the occasion of the proof of the Bieberbach conjecture held at Purdue University, March 1985.

Chapter 7:
Discrete Differential Forms for Computational Modeling

Mathieu Desbrun Eva Kanso Yiying Tong
Applied Geometry Lab
Caltech*

1 Motivation

The emergence of computers as an essential tool in scientific research has shaken the very foundations of differential modeling. Indeed, the deeply-rooted abstraction of smoothness, or *differentiability*, seems to inherently clash with a computer's ability of storing only finite sets of numbers. While there has been a series of computational techniques that proposed discretizations of differential equations, the geometric structures they are supposed to simulate are often lost in the process.

1.1 The Role of Geometry in Science

Geometry is the study of space and of the properties of shapes in space. Dating back to Euclid, models of our surroundings have been formulated using simple, geometric descriptions, formalizing apparent *symmetries* and experimental *invariants*. Consequently, geometry is at the foundation of many current physical theories: general relativity, electromagnetism (E&M), gauge theory as well as solid and fluid mechanics all have strong underlying geometrical structures. Einstein's theory for instance states that gravitational field strength is directly proportional to the *curvature of space-time*. In other words, the physics of relativity is *directly modelled* by the shape of our 4-dimensional world, just as the behavior of soap bubbles is modeled by their shapes. Differential geometry is thus, de facto, the mother tongue of numerous physical and mathematical models.

Unfortunately, the inherent geometric nature of such models is often obstructed by their formulation in vectorial or tensorial notations: the traditional use of a coordinate system, in which the equations of those models are expressed, often obscures the underlying structures by an overwhelming usage of indices. Moreover, such complex expressions entangle the topological and geometrical content of the model.

1.2 Geometry-based Exterior Calculus

The geometric nature of these models is best expressed and elucidated through the use of the *Exterior Calculus of Differential Forms*, first introduced by Cartan [Cartan 1945]. This geometry-based calculus was further developed and refined over the twentieth century to become the foundation of modern differential geometry. The calculus of exterior forms allows one to express differential and integral equations on smooth and curved spaces in a consistent manner, while revealing the geometrical invariants at play. For example, the classical operations of gradient, divergence, and curl as well as the theorems of Green, Gauss and Stokes can all be expressed concisely in terms of differential forms and an operator on these forms called the exterior derivative—hinting at the generality of this approach.

Compared to classical tensorial calculus, this exterior calculus has several advantages. First, it is often difficult to recognize the coordinate-independent nature of quantities written in tensorial notation: local and global invariants are hard to notice by just staring at the indices. On the other hand, invariants are easily discovered when expressed as differential forms by invoking either Stokes' theorem, Poincaré lemma, or by applying exterior differentiation. Note also that the exterior derivative of differential forms—the antisymmetric part of derivatives—is one of the most important parts of differentiation, since it is invariant under coordinate system change. In fact, Sharpe states in [Sharpe 1997] that every differential equation may be expressed in term of the exterior derivative of differential forms. As a consequence, several recent initiatives have been aimed at formulating the physical laws in terms of differential forms. For recent work along these lines, the reader is invited to refer to [Burke 1985; Abraham et al. 1988; Lovelock and Rund 1993; Flanders 1990; Morita 2001; Carroll 2003; Frankel 2004] for books offering a theoretical treatment of various physical theories using differential forms.

1.3 Differential vs. Discrete Modeling

We have seen that a large amount of our scientific knowledge relies on a deeply-rooted differential (*i.e.*, smooth) apprehension of the world. This abstraction of differentiability allows researchers to model complex physical systems via concise equations. With the sudden advent of the digital age, it was therefore only natural to resort to computations based on such differential equations.

However, since digital computers can only manipulate finite sets of numbers, their capabilities seem to clash with the basic foundations of differential modeling. In order to overcome this hurdle, a first set of computational techniques (*e.g.*, finite difference or particle methods) focused on satisfying the continuous equations at a discrete set of spatial and temporal samples. Unfortunately, focusing on accurately discretizing the local laws often fails to respect important global structures and invariants. Later methods such as Finite Elements (FEM), drawing from developments in the calculus of variations, remedied this inadequacy to some extent by satisfying local conservation laws on average and preserving some important invariants. Coupled with a finer ability to deal with arbitrary boundaries, FEM became the de facto computational tool for engineers. Even with significant advances in error control, convergence, and stability of these finite approximations, the underlying structures of the simulated continuous systems are often destroyed: a moving rigid body may gain or loose momentum; or a cavity may exhibit fictitious eigenmodes in an electromagnetism (E&M) simulation. Such examples illustrate some of the loss of fidelity that can follow from a standard discretization process, failing to preserve some fundamental geometric and topological structures of the underlying continuous models.

The cultural gap between theoretical and applied science communities may be partially responsible for the current lack of proper discrete, computational modeling that could mirror and leverage the rich developments of its differential counterpart. In particular, it is striking that the calculus of differential forms has not yet had an impact on the mainstream computational fields, despite excellent initial results in E&M [Bossavit 1998] or Lagrangian me-

*E-mail: {mathieu|eva|yiying}@caltech.edu

chanics [Marsden and West 2001]. It should also be noticed that some basic tools necessary for the definition of a discrete calculus already exist, probably initiated by Poincaré when he defined his cell decomposition of smooth manifolds. The study of the structure of ordered sets or simplices now belongs to the well-studied branch of mathematics known as *Combinatorial Differential Topology and Geometry*, which is still an active area of research (see, e.g., [Forman 2005] and [Björner and Welker 1995] and references therein).

1.4 Calculus ex Geometrica

Given the overwhelming geometric nature of the most fundamental and successful calculus of these last few centuries, it seems relevant to *approach computations from a geometric standpoint*.

One of the key insights that percolated down from the theory of differential forms is rather simple and intuitive: one needs to recognize that different physical quantities have different properties, and must be treated accordingly. Fluid mechanics or electromagnetism, for instance, make heavy use of line integrals, as well as surface and volume integrals; even physical measurements are performed as specific local integrations or averages (think flux for magnetic field, or current for electricity, or pressure for atoms' collisions). Pointwise evaluations or approximations for such quantities are not the appropriate discrete analogs, since the defining geometric properties of their physical meaning cannot be enforced naturally. Instead, *one should store and manipulate those quantities at their geometrically-meaningful location*: in other words, we should consider values on vertices, edges, faces, and tetrahedra as proper discrete versions of respectively pointwise functions, line integrals, surface integrals, and volume integrals: only then will we be able to manipulate those values without violating the symmetries that the differential modeling supposedly tried to exploit for predictive purposes.

1.5 Similar Endeavors

The need for improved numerics have recently sprung a (still limited) number of interesting related developments in various fields. Although we will not try to be exhaustive, we wish to point the reader to a few of the most successful investigations with the same "flavor" as our discrete geometry-based calculus, albeit their approaches are rarely similar to ours. First, the field of *Mimetic Discretizations of Continuum Mechanics*, led by Shashkov, Steinberg, and Hyman [Hyman and Shashkov 1997], started on the premise that spurious solutions obtained from finite element or finite difference methods often originate from inconsistent discretizations of the operators div, curl, and grad, and that addressing this inconsistency pays off numerically. Similarly, *Computational Electromagnetism* has also identified the issue of field discretization as the main reason for spurious modes in numerical results. An excellent treatment (upon which this paper is inspired) of the discretization of the Maxwell's equations resulted [Bossavit 1998], with a clear relationship to the differential case. Finally, recent developments in *Discrete Lagrangian Mechanics* have demonstrated the efficacy of a proper discretization of the Lagrangian of a dynamical system, rather than the discretization of its derived Euler-Lagrange equations: with a discrete Lagrangian, one can ensure that the integration scheme satisfies an exact discrete least-action principle, preserving all the momenta directly for arbitrary orders of accuracy [Marsden and West 2001]. Respecting the defining geometric properties of both the fields and the governing equations is a common link between all these recent approaches.

1.6 Advantages of Discrete Differential Modeling

The reader will have most probably understood our bias by now: we believe that the systematic construction, inspired by Exterior Calculus, of **differential, yet readily-discretizable computational foundations** is a crucial ingredient for numerical fidelity. Because many of the standard tools used in differential geometry have discrete combinatorial analogues, the *discrete versions of forms or manifolds* will be formally identical to (and should partake of the same properties as) the continuum models. Additionally, such an approach should clearly maintain the separation of the topological (*metric-independent*) and geometrical (*metric-dependent*) components of the quantities involved, keeping the geometric picture (*i.e.*, intrinsic structure) intact.

A *discrete differential modeling approach to computations* will also be often much simpler to define and develop than its continuous counterpart. For example, the discrete notion of a differential form will be implemented simply as values on mesh elements. Likewise, the discrete notion of orientation will be more straightforward than its continuous counterpart: while the differential definition of orientation uses the notion of equivalence class of atlases determined by the sign of the Jacobian, the orientation of a mesh edge will be one of two directions; a triangle will be oriented clockwise or counterclockwise; a volume will have a direction as a right-handed helix or a left-handed one; no notion of atlas (a collection of consistent coordinate charts on a manifold) will be required.

Figure 1: *Typical 2D and 3D meshes: although the David' head appears smooth, its surface is made of a triangle mesh; tetrahedral meshes (such as this mechanical part, with a cutaway view) are some typical examples of irregular meshes on which faithful computations need to be performed. David's head mesh is courtesy of Marc Levoy, Stanford.*

1.7 Goal of This Paper

This paper was written with several purposes in mind. First, we wish to demonstrate that the foundations on which powerful methods of computations can be built are quite approachable—and are not as abstract as the reader may fear: the ideas involved are very intuitive as a side effect of the simplicity of the geometric underlying principles.

Second, we wish to help bridge the gap between applied fields and theoretical fields: we have tried to render the theoretical bases of our exposition accessible to computer scientists, and the concrete implementation insights understandable by non-specialists. This is quite an exercise in style, as one has to be sufficiently clear to not put off theory-oriented minds while not losing the interests of practical minds. For this very reason, the reader should not consider this introductory exposition as a definite source of knowledge: it should instead be considered as a portal to better, more focused work on related subjects. We only hope that we will ease our readers into foundational concepts that can be undoubtedly and fruitfully applied to all sorts of computations—be it for graphics or simulation.

With these goals in mind, we will describe the background needed to develop a principled, geometry-based approach to computational

modeling that circumvents the apparent mismatch between differential and discrete modeling.

2 Relevance of Forms for Integration

The evaluation of differential quantities on a discrete space (mesh) is a nontrivial problem. For instance, consider a piecewise-linear 2-dimensional surface embedded in a three-dimensional Euclidean space, *i.e.*, a triangle mesh. Celebrated quantities such as the Gaussian and mean curvatures are delicate to define on it. More precisely, the Gaussian curvature can be easily proven to be zero everywhere *except* on vertices, where it is a Dirac delta function. Likewise, the mean curvature can only be defined in the distributional sense, as a Dirac delta function on edges. However, through local *integrations*, one can easily manipulate these quantities numerically: if a careful choice of non-overlapping regions is made, the delta functions can be properly integrated, rendering the computations relatively simple as shown, for example, in [Meyer et al. 2002; Hildebrandt and Polthier 2004]. Note that the process of integration to suppress discontinuity is, in spirit, equivalent to the idea of weak form used in the Finite Element method.

This idea of integrated value has predated in some cases the equivalent differential statements: for instance, it was long known that the genus of a surface can be calculated through a cell decomposition of the surface via the Euler characteristic. The actual Gauss-Bonnet theorem was, however, derived later on. Now, if one tries to discretize the Gaussian curvature of a piecewise-linear surface in an arbitrary way, it is less than likely that its integral over the surface equals the desired Euler characteristic, while its discrete version, defined on vertices (or, more precisely, on the dual of each vertex), naturally preserves this topological invariant.

2.1 From Integration to Differential Forms

Integration is obviously a linear operation, since for any disjoint sets A and B,

$$\int_{A\cup B} = \int_A + \int_B .$$

Moreover, the integration over a subset of measure zero is always zero; for example, an area integral of (a lower dimensional object such as) a curve or a point is equal to zero. Finally, integration is *objective* (*i.e.*, relevant) only if its evaluation is invariant under change of coordinate systems. These three properties combined directly imply that the integrand (*i.e.*, the whole expression after the integral sign) has to be *antisymmetric*. That is, the basic building blocks of any type of integration are **differential forms**. Chances are, the reader is already very well acquainted with forms, maybe without even knowing it.

2.1.1 An Intuitive Definition

A differential form (also denoted as exterior differential form) is, informally, an integrand, *i.e.*, a quantity that can be integrated. It is the dx in $\int dx$ and the $dx\ dy$ in $\iint dx\, dy$. More precisely, consider a smooth function $F(x)$ over an interval in $\mathbb{R}$. Now, define $f(x)$ to be its derivative, that is,

$$f(x) \;=\; \frac{dF}{dx} ,$$

A simple rewriting of this last equation leads to $dF = f(x)dx$, which leads (this is known as the Newton-Leibnitz formula):

$$\int_a^b dF = \int_a^b f(x)dx = F(b) - F(a) , \qquad (1)$$

The integrand $f(x)dx$ is called a *1-form*, because it can only be integrated over any 1-dimensional (1D) real interval. Similarly, for a function $G(x, y, z)$, we have:

$$dG = \frac{\partial G}{\partial x}dx + \frac{\partial G}{\partial y}dy + \frac{\partial G}{\partial z}dz ,$$

which can be integrated over any 1D curve in $\mathbb{R}^3$, and is also a 1-form. More generally, *a k-form can be described as an entity ready (or designed, if you prefer) to be integrated on a k-(sub)region*. Note that forms are valued zero on (sub)regions that are of higher or lower order dimension than the original space; for example, 4-forms are zero on $\mathbb{R}^3$. These differential forms are extensively used in mathematics, physics and engineering, as evidenced by the fact that operations like gradient, divergence, and curl can all be expressed in terms of forms only, as well as fundamental theorems like Green's or Stokes.

2.1.2 A Formal Definition

For concreteness, consider the n-dimensional Euclidean space $\mathbb{R}^n$, $n \in \mathbb{N}$ and let $\mathcal{M}$ be an open region $\mathcal{M} \subset \mathbb{R}^n$; $\mathcal{M}$ is also called an *n-manifold*. The vector space $T_x\mathcal{M}$ consists of all the (tangent) vectors at a point $x \in \mathcal{M}$ and can be identified with $\mathbb{R}^n$ itself. A k-form ω^k is a rank-k, anti-symmetric, tensor field over $\mathcal{M}$. That is, at each point $x \in \mathcal{M}$, it is a multi-linear map that takes k tangent vectors as input and returns a real number:

$$\omega^k : T_xM \ldots \times T_xM \longrightarrow \mathbb{R}$$

which *changes sign when you switch two variables* (hence the term antisymmetric). Any k-form naturally induces a k-form on a submanifold, through restriction of the linear map to the domain that is the product of tangent spaces of the submanifold.

Note on Pseudo-forms There is a closely related concept named pseudo-form. Pseudo-forms change sign when we change the orientation of coordinate systems, just like pseudo-vectors. As a result, the integration of a pseudo-form does not change sign when the orientation of the manifold is changed. Unlike k-forms, a pseudo-k-form induces a pseudo-k-form on a submanifold *only* if a transverse direction is given. For example, fluid flux is sometimes called a pseudo-2-form: indeed, given a transverse direction, we know how much flux is going through a piece of surface; it does not depend on the orientation of the surface itself. Vorticity is, however, a true 2-form: given an orientation of the surface, the integration gives us the circulation around that surface boundary induced by the surface orientation. It does *not* depend on the transverse direction of the surface. But if we have an orientation of the ambient space, we can always associate transverse direction with internal orientation of the submanifold. Thus, in our case, we may treat pseudo-forms simply as forms because we can consistently choose a representative from the equivalence class.

2.2 The Differential Structure

Differential forms are the building blocks of a whole calculus. To manipulate these basic blocks, Exterior Calculus defines seven operators:

- d: the exterior derivative, that extends the notion of the differential of a function to differential forms;
- $\star$: the Hodge star, that transforms k-forms into (n-k)-forms;
- $\wedge$: the wedge product, that extends the notion of exterior product to forms;
- $\sharp$ and $\flat$: the sharp and flat operators, that, given a metric, transforms a 1-form into a vector and vice-versa;

- ◇ i_X: the interior product with respect to a vector field X (also called contraction operator), a concept dual to the exterior product;
- ◇ $\mathcal{L}_X$: the Lie derivative with respect to a vector field X, that extends the notion of directional derivative.

In this chapter, we will restrict our discussions to the first three operators, to provide the most basic tools necessary in computational modeling.

2.3 A Taste of Exterior Calculus in R^3

To give the reader of a taste of the relative simplicity of Exterior Calculus, we provide a list of equivalence (in the continuous world!) between traditional operations and their Exterior Calculus counterpart in the special case of $\mathbb{R}^3$. We will suppose that we have the usual Euclidean metric. Then, forms are actually quite simple to conceive:

0-form $\Leftrightarrow$ scalar field
1-form $\Leftrightarrow$ vector
2-form $\Leftrightarrow$ vector
3-form $\Leftrightarrow$ scalar field

To be clear, we will add a superscript on the forms to indicate their rank. Then applying forms to vector fields amounts to:

1-form: $u^1(v) \Leftrightarrow u \cdot v$.
2-form: $u^2(v, w) \Leftrightarrow u \cdot (v \times w)$.
3-form: $f^3(u, v, w) \Leftrightarrow fu \cdot (v \times w)$.

Furthermore, the usual operations like gradient, curl, divergence and cross product can all be expressed in terms of the basic exterior calculus operators. For example:

$d^0 f = \nabla f,\ d^1 u = \nabla \times u,\ d^2 u = \nabla \cdot u;$

$*^0 f = f,\ *^1 u = u,\ *^2 u = u,\ *^3 f = f;$

$\delta^1 u = \nabla \cdot u,\ \delta^2 u = \nabla \times u,\ \delta^3 f = \nabla f;$

$f^0 \wedge u = fu,\ u^1 \wedge v^1 = u \times v,\ u^1 \wedge v^2 = u^2 \wedge v^1 = u \cdot v;$

$i_v u^1 = u \cdot v,\ i_v u^2 = u \times v,\ i_v f^3 = fv.$

Now that we have established the relevance of differential forms even in the most basic vector operations, time has come to turn our attention to make this concept of forms readily usable for computational purposes.

3 Discrete Differential Forms

Finding a discrete counterpart to the notion of differential forms is a delicate matter. If one was to represent differential forms using their coordinate values and approximate the exterior derivative using finite differences, basic theorems such as Stokes theorem would not hold numerically. The main objective of this section is therefore to present a proper discretization of the forms on simplicial complexes. We will show how this discrete geometric structure, well suited for computational purposes, is designed to preserve all the fundamental differential properties. For simplicity, we restrict the discussion to forms on 2D surfaces or 3D regions embedded in $\mathbb{R}^3$, but the construction is applicable to general manifolds in arbitrary spaces. In fact, the only necessary assumption is that the embedding space must be a vector space, a natural condition in practice.

3.1 Simplicial Complexes and Discrete Manifolds

For the interested reader, the notions we introduce in this section are defined formally in much more details (for the general case of k-dimensional spaces) in references such as [Munkres 1984] or [Hatcher 2004].

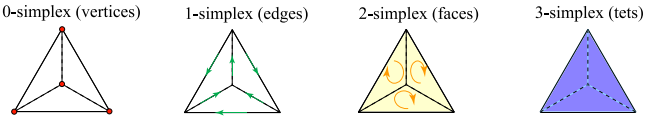


Figure 2: *In 1D, the simplex is the line segment . In 2D, the simplex is the convex hull of the equilateral triangle. In 3D, the simplex is the convex hull of the tetrahedron.*

3.1.1 Notion of Simplex

A **k-simplex** is the generic term to describe the simplest mesh element of dimension k—hence the name. By way of motivation, consider a three-dimensional mesh in space. This mesh is made of a series of adjacent tetrahedra (denoted *tets* for simplicity throughout). The vertices of the tets are said to form a 0-simplex. Similarly, the line segments or edges form a 1-simplex, the triangles or faces form a 2-simplex, and the tets a 3-simplex. Also, the faces (2-simplex) can be thought of as the boundaries of the tets (3-simplex), and the edges (1-simplex) as the boundaries of the faces, etc.

The definition of a simplex can be made more abstract as a series of k-tuples (referring to the vertices they are built upon). However, for the type of applications that we are targeting in this chapter, we will not make any distinction between an abstract simplex and its topological or geometrical realization.

Formally, a k-simplex σ_k is the convex hull of (k+1) geometrically distinct points $v_0, \dots v_k \in \mathbb{R}^n$ with $n \geq k$. In other words, it is the intersection of all convex sets containing $(v_0, \dots v_k)$; namely:

$$\sigma_k = \{x \in \mathbb{R}^n | x = \sum_{i=0}^{k} \alpha^i \, \mathrm{v}_i \ \text{ with } \ \alpha^i \geq 0 \text{ and } \sum_{i=0}^{k} \alpha^i = 1\}.$$

The points $v_0, \dots v_k$ are called the *vertices* and k is called the dimension of the k-simplex., which we will denote as:

$$\sigma_k = \{v_0 v_1 ... v_k\} \ .$$

3.1.2 Orientation of a Simplex

Note that all orderings of the $k+1$ vertices of a k-simplex can be divided into two equivalent classes, *i.e.*, two orderings differing by an even permutation. Such an ordering is called an *orientation*. In the present work, we always assume that local orientations are *given* for each simplex; that is, each element of the mesh has been given a particular orientation. For example, an edge $\sigma_1 = \{v_0 v_1\}$ on Figure 2 has an arrow indicating its default orientation. If the opposite orientation is needed, we will denote it as $\{v_1 v_0\}$, or, equivalently, by $-\{v_0 v_1\}$. For more details and examples, the reader is referred to [Hirani 2003].

3.1.3 Boundary of a Simplex

Any (k-1)-simplex spanned by a strict subset of $\{v_0, \dots v_k\}$ is called a *(k-1)-face* of σ_k. That is, a *(k-1)-face* is simply a $(k-1)$-simplex whose k vertices are all from the $(k+1)$ vertices of the k-simplex. The union of the $(k-1)$-faces is what is called the *boundary* of the k-simplex. One should be careful here: because of the default orientation of the simplices, the formal *signed* sum of the (k-1)-faces defines the boundary of the k-simplex. Therefore, the boundary operator takes a k-simplex and gives the sum of all its $(k-1)$-faces with 1 or -1 as coefficients depending on whether their respective orientations match or not, see Figure 4.

To remove possible mistakes in orientation, we can define the

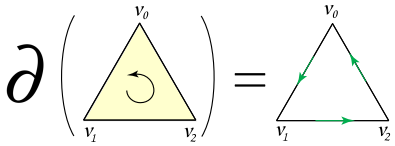


Figure 3: *The boundary operator ∂ applied to a triangle (a 2-simplex) is equal to the signed sum of the edges (i.e., the faces of the 2-simplex).*

boundary operator as follows:

$$\partial\{v_0 v_1 ... v_k\} = \sum_{j=0}^{k} (-1)^j \{v_0, ..., \widehat{v_j}, ..., v_k\}, \qquad (2)$$

where $\widehat{v_j}$ indicates that v_j is missing from the sequence, see Figure 3. Clearly, each k-simplex has k+1 faces. For this statement to be valid even for $k = 0$, the empty set $\emptyset$ is usually defined as a (-1)-simplex face of every 0-simplex. The reader is invited to verify this definition on the triangle $\{v_0, v_1, v_2\}$ in Figure 3:

$$\partial\langle v_0, v_1, v_2\rangle = \langle v_1, v_2\rangle - \langle v_0, v_2\rangle + \langle v_0, v_1\rangle$$

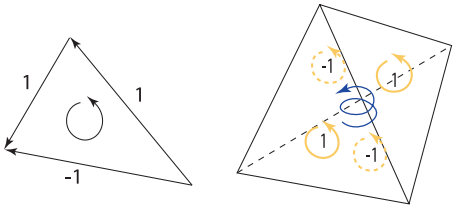
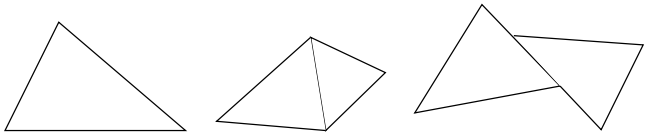


Figure 4: *Boundary operator applied to a triangle (left), and a tetrahedron (right). Orientations of the simplices are indicated with arrows.*

3.1.4 Simplicial Complex

A simplicial complex is a collection $\mathcal{K}$ of simplices, which satisfies the following two simple conditions:

- every face of each simplex in $\mathcal{K}$ is in $\mathcal{K}$;
- the intersection of any two simplices in $\mathcal{K}$ is either empty, or a single common face.



simplicial complexes *not a simplicial complex*

Computer graphics makes heavy use of what is called *realizations* of simplicial complexes. Loosely speaking, a realization of a simplicial complex is an embedding of this complex structure into the underlying space $\mathbb{R}^n$. Triangle meshes in 2D and tet meshes in 3D are examples of such simplicial complexes (see Figure 1). Notice that polygonal meshes can be easily triangulated, thus can be easily turned into simplicial complexes. One can also use the notion of *cell complex* by allowing the elements of $\mathcal{K}$ to be non-simplicial; but we will restrict our explanations to the simpler, yet as general case of simplicial complexes.

3.1.5 Discrete Manifolds

An n-dimensional discrete manifold $\mathcal{M}$ is an n-dimensional simplicial complex that satisfies the following condition: for each simplex, the union of all the adjacent simplices forms an n-dimensional ball (i.e., a disk in 2D, a ball in 3D, etc), or half a ball if the simplex is on the boundary. As a consequence, each (n-1)-simplex has exactly two adjacent n-simplices—or only one if it is on a boundary.

Basically, a discrete manifold is nothing but a *triangulation* of a smooth manifold. For example in 2d, discrete manifolds cannot have isolated edges (also called sticks or hanging edges) or isolated vertices, and each of their edges is adjacent to 2 faces (except for the boundary; in that case, the edge is adjacent to only one face). A surface mesh in 3d cannot have a "fin", *i.e.*, a edge with more than two adjacent triangles. In other words, a small, imaginary *inhabitant* of an n-dimensional discrete manifold would consider itself living in $\mathbb{R}^n$.

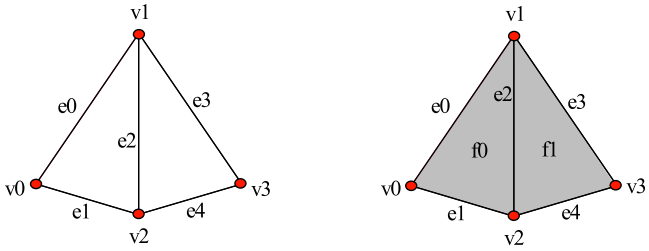


Figure 5: *(a) A simplicial complex consisting of all vertices $\{v0, v1, v2, v3\}$ and edges $\{e0, e1, e2, e3, e4\}$. This simplicial complex is not a discrete manifold because the neighborhoods of the vertices or of any points on an edge are not 1d balls. (b) If we add the faces $f0$ and $f1$ to the simplicial complex, it becomes a 2-manifold with one boundary.*

3.2 Notion of Chains

We have already encountered the notion of chain, without mentioning it. Recall that the boundary operator takes each k-simplex and gives the *signed* sum of all its (k-1)-faces. We say that the boundary of a k-simplex produces a *(k-1)-chain*. The following definition is more precise and general.

3.2.1 Definition

A k-**chain** of an oriented simplicial complex $\mathcal{K}$ is a set of values, one for *each k-simplex* of $\mathcal{K}$. That is, a k-chain c can then be thought of as a linear combination of all the k-simplices in $\mathcal{K}$:

$$c = \sum_{\sigma \in \mathcal{M}} c(\sigma) \cdot \sigma, \qquad (3)$$

where $c(\sigma) \in \mathbb{R}$. More formally, a chain is a free abelian group generated by the k-simplices, *i.e.*, a mapping from the collection of all k-simplices in $\mathcal{K}$ to $\mathbb{R}$. We will denote the space of k-chains as $\mathcal{C}_k$.

3.2.2 Implementation of Chains

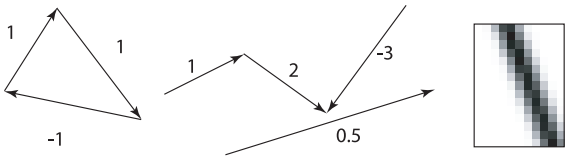
Let the set of all k-simplices in $\mathcal{K}$ be denoted $\mathcal{K}^k$, and let its cardinality be denoted as $|\mathcal{K}^k|$. A k-chain can simply be stored as a *vector* (or array) of dimension $|\mathcal{K}^k|$.

3.2.3 Boundary Operator on Chains

We mentioned that the boundary operator ∂ was returning a particular type of chain, namely, a chain with coefficients equal to either 0, 1, or -1. Therefore, it should not be surprising that we can extend the notion of boundary to act also on k-chains, simply by linearity:

$$\partial \sum_k c_k \sigma_k = \sum_k c_k \partial \sigma_k.$$

That is, from one set of values assigned to all simplices of a com-



(a) (b) (c)

Figure 6: *(a) An example of a* 1*-chain being the boundary of a face (*2*-simplex); (b) a second example of* 1*-chain with 4 non-zero coefficients; (c) an anti-aliased line is, in disguise, a* 2*-chain (*i.e.*, values on pixels)!*

plex, one can deduce another set of values derived by weighting the boundaries of each simplex by the original value stored on it. This operation is very natural, and can thus be implemented extremely easily as explained next.

3.2.4 Implementation of the Boundary Operator

Since the boundary operator is a linear mapping from the space of k-simplices to the space of (k-1)-simplices, it can simply be represented by a *matrix* of dimension $|\mathcal{K}^k| \times |\mathcal{K}^{k-1}|$. The reader can convince herself that this matrix is sparse, as only immediate neighbors are involved in the boundary operator. Similarly, this matrix contains only the values 0, 1, and -1. Notice than in 3D, there are 3 boundary operators ∂^k. however, the operator needed for a particular operation is obvious from the type of the argument. Thanks to this context-dependence, we can simplify the notation and remove the superscript when there is no ambiguity.

3.3 Notion of Cochains

A k-cochain ω is the *dual* of a k-chain, that is to say, ω is a linear mapping that takes k-chains to $\mathbb{R}$. One writes:

$$\omega : \quad \mathcal{C}_k \quad \rightarrow \quad \mathbb{R} \tag{4}$$

$$c \quad \rightarrow \quad \omega(c), \tag{5}$$

which reads as: a k-cochain ω operates on a k-chain c to give a scalar in $\mathbb{R}$.

Clearly, a co-chain also corresponds to one value per simplex (since all the k-simplices form a basis for vector space C_k, and we only need to know the mapping of vectors in basis to determine a linear mapping), and hence the notion of duality of chains and co-chains is appropriate. But contrary to a chain, a k-cochain is *evaluated* on each simplex of the dimension k. In other words, a k-cochain can be thought of as a *field* that can be evaluated on each k-simplex of an oriented simplicial complex $\mathcal{M}$.

3.3.1 Implementation of Cochains

The numerical representation of cochains follows from that of chains by duality. Recall that a k-chain can be represented as a vector c_k of length equal to the number of k-simplices in $\mathcal{M}$. Similarly, one may represent ω by a vector ω^k of the same size as c_k.

Now, remember that ω operates on c to give a scalar in $\mathbb{R}$. The linear operation $\omega(c)$ translates into an inner product $\omega^k \cdot c_k$. More specifically, one may continue to think of c_k as a *column vector* so that the $\mathbb{R}$-valued linear mapping ω can be represented by a *row vector* $(\omega^k)^t$, and $\omega(c)$ becomes simply the matrix multiplication of the row vector $(\omega^k)^t$ with the column vector c_k.

3.4 Discrete Forms as Co-chains

The attentive reader will have noticed by now: *k-cochains are discrete analogs to differential forms*. Indeed, a continuous k-form was defined as a linear mapping from k-dimensional sets to $\mathbb{R}$, as we can only integrate a k-form on a k-(sub)manifold. Note now that a k-d set, when one has only a mesh to work with, is simply a *chain*. And a linear mapping from a chain to a real number is what we called a cochain: *a cochain is therefore a natural discrete counterpart of a form*.

For instance a 0-form can be evaluated at each point, a 1-form can be evaluated on each curve, a 2-form can be evaluated on each surface, etc.

Now if we *restrict* integration to take place only on the k-submanifold which is the sum of k-simplices in the triangulation, we get a k-cochain; thus k-cochains are a discretization of k-forms. One can further map a continuous k-form to a k-cochain. To do this, first integrate the k-form on each k-simplex and assign the resulting value to that simplex to obtain a k-cochain on the k-simplicial complex. This k-cochain is a discrete representation of the original k-form.

3.4.1 Evaluation of a Form on a Chain

We can now naturally extend the notion of evaluation of a differential form ω on an arbitrary chain simply by linearity:

$$\int_{\sum_i c_i \sigma_i} \omega = \sum_i c_i \int_{\sigma_i} \omega. \tag{6}$$

As mentioned above, the integration of ω on each k-simplex σ_k provides a discretization of ω or, in other words, a mapping from the k-form ω to a k-cochain represented by:

$$\omega[i] = \int_{\sigma_i} \omega.$$

However convenient this chain/cochain standpoint is, in practical applications, one often needs a point-wise value for a k-form or to evaluate the integration on a particular k-submanifold. How do we get these values from a k-cochain? We will cover this issue of *form interpolation* in Section 6.

4 Operations on Chains and Cochains

4.1 Discrete Exterior Derivative

In the present discrete setting where the discrete differential forms are defined as cochains, defining a discrete exterior derivative can be done very elegantly: Stokes' theorem, mentioned early on in Section 2, can be used to *define* the exterior derivative d. Traditionally, this theorem states a vector identity equivalent to the well-known curl, divergence, Green's, and Ostrogradsky's theorems. Written in terms of forms, the identity becomes quite simple: it states that d applied to an arbitrary form ω is evaluated on an arbitrary simplex σ as follows:

$$\int_{\sigma} d\omega = \int_{\partial\sigma} \omega. \tag{7}$$

You surely recognize the usual property that an integral over a k-dimensional set is turned into a boundary integral (*i.e.*, over a set of dimension k-1). With this simple equation relating the evaluation of $d\omega$ on a simplex σ to the evaluation of ω on the boundary of this simplex, the exterior derivative is *readily defined*: each time you encounter an exterior derivative of a form, replace any evaluation over a simplex σ by a direct evaluation of the form itself over the boundary of σ. Obviously, Stokes' theorem will be enforced by construction!

4.1.1 Coboundary Operator

The operator d is called the *adjoint* of the boundary operator ∂: if we denote the integral sign as a pairing, *i.e.*, with the convention that $\int_\sigma \omega = [\omega, \sigma]$, then applying d on the left hand side of this operator is equivalent to applying ∂ on the right hand: $[d\omega, \sigma] = [\omega, \partial\sigma]$. For this very reason, d is sometimes called the coboundary operator.

Finally, by linearity of integration, we can write a more general expression of Stokes' theorem, now extended to arbitrary chains as follows:

$$\int_{\sum_i c_i \sigma_i} d\omega = \int_{\partial\left(\sum_i c_i \sigma_i\right)} \omega = \int_{\sum_i c_i \partial\sigma_i} \omega = \sum_i c_i \int_{\partial\sigma_i} \omega$$

Consider the example shown in Figure 7. The discrete exterior derivative of the 1-form, defined as numbers on edges, is a 2-form represented by numbers on oriented faces. The orientation of the 1-forms may be opposite to that induced on the edges by the orientation of the faces. In this case, the values on the edges change sign. For instance, the 2-form associated with the d of the 1-forms surrounding the oriented shaded triangle takes the value $\omega = 2 - 1 - 0.75 = 0.25$.

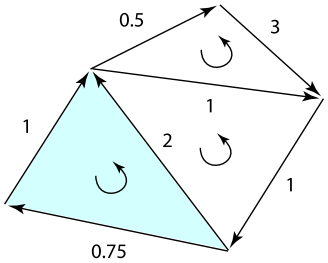


Figure 7: *Given a* 1*-form as numbers on oriented edges, its discrete exterior derivative is a* 2*-form. In particular, this* 2*-form is valued 0.25 on the oriented shaded triangle.*

4.1.2 Implementation of Exterior Derivative

Since we use vectors of dimension $|\mathcal{K}^k|$ to represent a k-cochain, the operator d can be represented by a matrix of dimension $|\mathcal{K}^{k+1}| \times |\mathcal{K}^k|$. Furthermore, this matrix has a trivial expression. Indeed, using the matrix notation introduced earlier, we have:

$$\int_{\partial c} \omega = \omega^t(\partial c) = (\partial^t \omega)^t c = \int_c d\omega.$$

Thus, the matrix d is simply equal to ∂^t. This should not come as a surprise, since we previously discussed that d is simply the adjoint of ∂. Note that extreme care should be used when boundaries are present. However, and without digging too much into the details, it turns out that even for discrete manifolds *with* boundaries, the previous statement is valid. Implementing the exterior derivative while preserving Stokes' theorem is therefore a trivial matter in practice. Notice that just like for the boundary operator, there is actually more than one matrix for the exterior derivative operator: there is one per simplex dimension. But again, the context is sufficient to actually know which matrix is needed. A brute force approach that gets rid of these multiple matrices is to use a notion of super-chain, *i.e.*, a vector storing *all* simplices, ordered from dimension 0 to the dimension of the space: in this case, the exterior derivative can be defined as a single, large sparse matrix that contains these previous matrices as blocks along the diagonal. We will not use this approach, as it makes the exposition more cumbersome in general.

4.2 Exact/Closed Forms and Poincaré Lemma

A k-form ω is called exact if there is a (k-1)-form α such that $\omega = d\alpha$, and it is called closed if $d\omega = 0$.

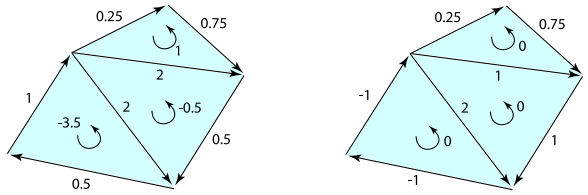


Figure 8: *(a) The 2-form on the oriented shaded triangles defined by the exterior derivative* d *of the 1-form on the oriented edges is called an* exact 2-form*; (b) The 1-form on the oriented edges whose derivative* d *is identically zero is called a* closed 1-form*.*

It is worth noting here that every exact form is closed, as will be seen in Section 4.3. Moreover, it is well-known in the continuous setting that a closed form on a smooth contractible (sub)-manifold is locally exact. This result is called the Poincaré lemma. The discrete analogue to Poincaré lemma can be stated as follows: given a closed k-cochain ω on a logically star-shaped complex, that is to say, $d\omega = 0$, there exits a (k-1)-cochain α such that $\omega = d\alpha$. For a formal statement and proof of this discrete version, see [Desbrun et al. 2004].

4.3 Introducing the deRham Complex

The boundary of a boundary is the empty set. That is, the boundary operator applied twice to a k-simplex is zero. Indeed, it is easy to verify that $\partial\,\partial\sigma_k = 0$, since each ($k$-2)-simplex will appear exactly twice with different signs and, hence, cancel out. From the linearity of ∂, one can readily conclude that the property $\partial\,\partial = 0$ is true for all k-chain since k-simplices are the basis. Similarly, one has that the discrete exterior derivative satisfies $d\;d = \partial^t\partial^t = (\partial\,\partial)^t = 0$, analogously to the exterior derivative of differential forms (notice that this last equality corresponds to the equality of mixed partial derivatives, which in turn is responsible for identities like $\nabla\times\nabla = 0$ and $\nabla\cdot\nabla\times = 0$ in $\mathbb{R}^3$).

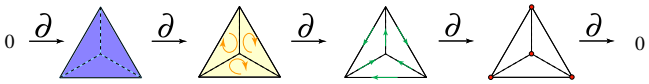


Figure 9: *The chain complex of a tetrahedron with the boundary operator: from the tet, to its faces, to their edges, and to their vertices.*

4.3.1 Chain Complex

In general, a *chain complex* is an array of linear spaces, connected with a linear operator D that satisfies the property $D\,D = 0$. Hence, the boundary operator ∂ (resp., the exterior derivative d) makes the spaces of chains (resp., cochains) into a chain complex, as shown in Figures 9 and 13.

4.3.2 Examples

Consider the 2d simplicial complex in Figure 10(a) and choose the oriented basis of the i-dimensional simplices ($i = 0$ for vertices, $i = 1$ for edges and $i = 2$ for the face) as suggested by the ordering in the figure.

One gets $\partial(f0) = e0 - e4 - e3$, which can be identified with the vector $(1, 0, 0, -1, -1)$. By repeating similar calculations for all simplices, one can readily conclude that the the boundary operator

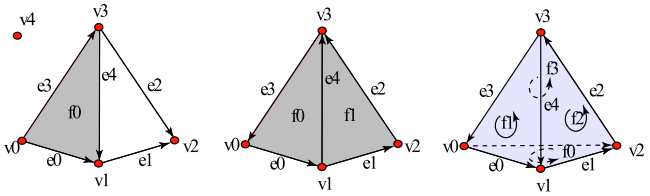


Figure 10: *Three examples of simplicial complexes. The first one is not manifold. The two others are.*

∂ is given by:

$$\partial^2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \\ -1 \end{pmatrix}, \quad \partial^1 = \begin{pmatrix} -1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

That is, the chain complex under the boundary operator ∂ can be written as:

$$0 \longrightarrow \mathcal{C}_2 \xrightarrow{\partial^2} \mathcal{C}_1 \xrightarrow{\partial^1} \mathcal{C}_0 \longrightarrow 0$$

where C_i, $i = 0, 1, 2$, denote the spaces of i-chains.

Consider now the domain to be the mesh shown in Figure 10(b). The exterior derivative operator, or the coboundary operator, can be expressed as:

$$d^0 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 \end{pmatrix}, \qquad d^1 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & -1 \end{pmatrix}.$$

It is worth noting that, since d is adjoint to ∂ by definition, the coboundary operator d induces a cochain complex:

$$0 \longleftarrow \mathcal{C}^2 \xleftarrow{d^1} \mathcal{C}^1 \xleftarrow{d^0} \mathcal{C}^0 \longleftarrow 0$$

where C^i, $i = 0, 1, 2$, denote the spaces of i-cochains.

Finally, suppose the domain is the tetrahedron in Figure 10(c), then the exterior derivative operators are:

$$d^0 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}, d^1 = \begin{pmatrix} 1 & 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}, d^2 = \begin{pmatrix} -1 & 1 & 1 & -1 \end{pmatrix}.$$

4.4 Notion of Homology and Cohomology

Homology is a concept dating back to Poincaré that focuses on studying the topological properties of a space. Loosely speaking, homology does so by counting the number of holes. In our case, since we assume that our space is a simplicial complex (*i.e.*, triangulated), we will only deal with *simplicial homology*, a simpler, more straightforward type of homology that can be seen as a discrete version of the continuous definition (equivalently, it is equivalent to the continuous one is the domain is triangulated). As we are about to see, the notion of discrete forms is intimately linked with these topological notions. In fact, we will see that (co)homology is the study of the relationship between *closed* and *exact* (co)chains.

4.4.1 Simplicial Homology

A fundamental problem in topology is that of determining, for two spaces, whether they are topologically equivalent. That is, we wish to know if one space can be morphed into the other without having to puncture it. For instance, a sphere-shaped tet mesh is not topologically equivalent to a torus-shaped tet mesh as one cannot alter the sphere-shaped mesh (*i.e.*, deform, refine, or coarsen it locally) to make it look like a torus.

The key idea of homology is to define *invariants* (*i.e.*, quantities that cannot change by continuous deformation) that characterize topological spaces. The simplest invariant is the number of connected components that a simplicial complex has: obviously, two simplicial complexes with different numbers of pieces cannot be continuously deformed into each other! Roughly speaking, homology groups are an extension of this idea to define more subtle invariants than the number of connected components.

Cycles and their Equivalence Classes Generalizing the previous example to other invariants is elegantly done using the notion of *cycles*. A cycle is simply a *closed k-chain*; that is, a linear combination of k-simplices so that the boundary of this chain (see Section 3.2) is the empty set. Any set of vertices is such a closed chain; any set of 1D loops are too; etc. Evidently, the mathematical definition of a k-cycle is any k-chain that belongs to $\mathrm{Ker}\ \partial^k$, by definition.

On this set of all k-cycles, one can define *equivalence classes*: we will say that a k-cycle is *homologous* to another k-cycle (*i.e.*, in the same equivalence class than the other) when these two chains differ by a boundary of a (k+1)-chain (*i.e.*, by an *exact chain*). Notice that this exact chain is, by definition (see Section 4.2), in the image of ∂^{k+1}, *i.e.*, $\mathrm{Im}\ \partial^{k+1}$.

4.4.2 Homology Groups

Let us now use these definition on the simple case of the 0^{th} homology group $\mathcal{H}_0$.

Homology Group $\mathcal{H}_0$ The boundary of any vertex is $\emptyset$, so a 0-cycle is any linear combination of vertices. Now if two vertices v_0 and v_1 are connected by an edge, $v_1 - v_0$ (*i.e.*, the difference of two cycles) is the boundary of this edge. Thus, by our previous definition, two vertices linked by an edge are *homologous*. By the same reasoning, any two vertices taken from the *same* connected component are, also, homologous, since there exists a chain of edges in between. Consequently, we can pick only one vertex per connected component to form a basis of this homology group. Its dimension, β_0, is therefore simply the number of connected components. The basis elements of that group are called *generators*, since they generate the whole homology group.

Homology Group $\mathcal{H}_1$ Let us proceed similarly for the 1^{st} homology class: we now have to consider 1-cycles (linear combinations of $1D$ loops). Again, one can easily conceive that there are different *types* of such cycles, and it is therefore possible to separate all possible cycles into different equivalence classes. For instance, the loop L_1 in Figure 11 is topologically distinct from the curve L_2: one is around a hole and the other is not, so the difference between the two is *not* the boundary of a 2-chain. Conversely, L_1 *is* in the same class as curve L_3 one since they differ by one connected area. Thus, in this figure, the 1^{st} homology group is a 2-dimensional group, and L_1 and L_2 (or L_3 and L_2, similarly) are generators. The reader is invited to apply this simple idea on the triangulated torus, to find two loops as generators of $\mathcal{H}_1$. In general, the dimensions of higher order homology will simply become the number of voids (holes) of the same order.

Formal Definition of Homology Groups We are now ready to generalize this construction to all homology groups. Remember that we have a series of k-chain spaces:

$$C_n \xrightarrow{\partial^n} C_{n-1} \ldots \xrightarrow{\partial^2} C_1 \xrightarrow{\partial^0} \mathcal{C}_0$$

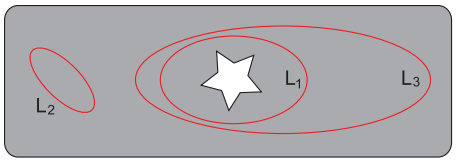


Figure 11: *Example of Homology Classes: the cycles L_1 and L_2 are topologically distinct as one encloses a hole while the other does not; L_2 and L_3 are however in the same equivalence class.*

with the specificity that $\partial\ \partial$ is the empty set. This direct implies that image of $\mathcal{C}_j$ is always in the kernel of ∂^{j+1}—such a series is called a *chain complex*. Now, the homology groups $\{\mathcal{H}_k\}_k$ of a chain complex based on ∂ are defined as the following quotient spaces:

$$\mathcal{H}_k = \text{Ker}\ \partial^k / \text{Im}\ \partial^{k+1}.$$

The reader is invited to check that this definition is *exactly* what we did for the 0^{th} and 1^{st} homology groups—and it is now valid for any order: indeed, we use the fact that closed chains (belonging to Ker ∂) are homologous if their difference is in Im ∂, and this is exactly what this quotient vector space is.

Example Consider the example in Figure $10(a)$. Geometrically, $\mathcal{H}_0$ is nontrivial because the simplicial complex σ is disconnected (it is easy to see $\{v_0, v_4\}$ form a basis for $\mathcal{H}_0$), while $\mathcal{H}_1$ is nontrivial since the cycle $(e_1 - e_2 + e_4)$ is not the boundary of any 2-chain of σ ($\{(e_1 - e_2 + e_4)\}$ is indeed a basis for this 1D space $\mathcal{H}_1$).

Link to Betti Numbers The dimension of the k-th cohomology group is called k-th Betti number; $\beta_k = |\mathcal{H}_k|$. For a 3D simplicial complex embedded in $\mathbb{R}^3$, these numbers have very straightforward meanings. β_0 is the number of connected components, β_1 is number of tunnels, β_2 is the number of voids, while β_4 is the number of empty 4D space inside the domain, which is 0 in the Euclidean (flat 3D) case. Finally, note that $\sum_{k=0,\ldots,n}(-1)^k\beta_k$, where β_k is the k-th Betti number, gives us the well-known Euler characteristics.

4.4.3 Cohomology Groups

The definition of homology groups is much more general than what we just reviewed. In fact, the reader can take the formal definition in the previous section, replace all occurrences of chain by cochain, of ∂ by d, and reverse the direction of the operator between spaces (see Section 4.3.2): this will also define equivalence classes. Because cochains are dual of chains, and d is the adjoint of ∂, these equivalence classes define what are actually denoted as *cohomology groups*: the cohomology groups of the deRham complex for the coboundary operator are simply the quotient spaces Ker d/Im d. Finally, note that the homology and cohomology groups are not only dual notions, but they are also isomorphic; therefore, the cardinalities of their basis are equal.

4.4.4 Calculation of the Cohomology Basis

One usual way to calculate a cohomology basis is to calculate a Smith Normal Form to obtain the homology basis first (possibly using progressive mesh [Gu and Yau 2003]), with a worst case complexity is $O(n^3)$, and then find the corresponding cohomology basis derived from this homology basis. We provide an alternative method here with worst case complexity also equal to $O(n^3)$. The advantage of our method is that it directly calculates the cohomology basis.

Our algorithm is a modified version of an algorithm in [Edelsbrunner et al. 2000], although they did not the use it for the same purpose[1]. We will use $row\#(.)$ to refer to the row number of the *last non-zero coefficient* in a particular column.

The procedure is as follows:

1. Transform d^k (size $|\sigma^k| \times |\sigma^{k+1}|$) in the following manner:

```
// For each column of d^k
for(i = 0; i < |σ^k|; i++)
    // Reduce column i
    repeat
        p ← row#(d^k[i])
        find j < i such as p==row#(d^k[j])
        make d^k[i][p] zero by adding to d^k[i] a multiple of d^k[j]
    until j_not_found or column i is all zero
```

   In the end of this procedure, we get $d^{k\prime} = d^k\ N^k$, whose non-zero column vectors are linearly independent of each other and with different $row\#(.)$, and N^k is a non-singular upper triangular matrix.

2. Construct $K^k = \{N_i^k \mid {d^k}'_i = 0\}$
   (where N_i^k and D'_i are column vectors of matrices N and D' respectively).
   K^k is a basis for kernel of d^k.

3. Construct $I^k = \{N_i^k | i = row\#({d^{(k-1)}}'_j)\}$

4. Construct $P^k = K^k - I^k$
   P^k is a basis for cohomology basis.

Short proof of correctness: First thing to notice is that N_i^k's are all linearly independent because N^k is nonsingular. If $N_{i_1}^k, N_{i_2}^k \in P^k$ (with $i_1 < i_2$), $row\#(N_{i_1}^k - N_{i_2}^k) = i_2$. But i_2 is not $row\#(.)$ of any ${d^{(k-1)}}'_i$ (and thus any linear combination of them) by definition of P^k. Therefore, we know that $N_{i_1}^k - N_{i_2}^k$ is not in the image space of d^{k-1} (since the range of d^{k-1} is the same as $d^{k-1\prime}$, by construction).

One can also prove that I^k is a subset of K^k. Pick such an N_i^k, suppose $i = row\#({d^{(k-1)}}'_j)$. $d^k\ {d^{(k-1)}}'_j = 0$ (since $d^k \circ d^{k-1} = 0$). Now $row\#(\tau =\equiv (N^k)^{-1}\ {d^{(k-1)}}_j) = i$ (the inverse of an upper triangular matrix is also an upper triangular matrix). $0 = d^k\ {d^{(k-1)}}_j = d^{k\prime}(N^k)^{-1}\ {d^{(k-1)}}_j = d^{k\prime}\tau$ means ${d^k}'_i = 0$ because the columns of $d^{k\prime}$ are linearly independent or 0.

Therefore, $\text{Card}(P^k) = \text{Card}(K^k) - \text{Card}(I^k) = Dim(\text{Ker}(d^k)) - Dim(\text{Im}(d^{(k-1)}))$, and we conclude that, P^k spans $\text{Ker}(d^k)/\text{Im}(d^{k-1})$ as expected. ∎

4.4.5 Example

Consider the 2d simplicial complex in Figure 10(a) again. We will show an example of running the same procedure described above to compute homology basis. The only difference is that we use ∂ instead of d.

1. Compute the ∂''s and N's: $\partial^{2\prime}$ is trivial, which is the same as ∂^2.

$$\partial^{1\prime} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},\quad N^{1\prime} = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

[1] Thanks to David Cohen-Steiner for pointing us to the similarities

2. Contruct K's:

$$K^0 = \{\begin{pmatrix}1\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\0\\1\end{pmatrix}\}$$

$$= \{v_0, v_1, v_2, v_3, v_4\}$$

(N^0 is identity)

$$K^1 = \{\begin{pmatrix}-1\\-1\\1\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\-1\\0\\1\end{pmatrix}, \} = \{(-e_0 - e_1 + e_2 + e_3), (e_1 - e_2 + e_4)\}$$

3. Construct I's:

$$\begin{aligned} I^0 &= \{v_1\ (1 = row\#(\partial^{1'}{}_0)), \\ &\quad v_2\ (2 = row\#(\partial^{1'}{}_1), \\ &\quad v_3\ (3 = row\#(\partial^{1'}{}_2))\} \\ I^1 &= \{(e_1 - e_2 + e_4)\ (4 = row\#(\partial^{2'}{}_0)\} \end{aligned}$$

4. Construct homology basis

$$P^0 = \{v_0, v_1, v_2, v_3, v_4\} - \{v_1, v_2, v_3\} = \{v_0, v_4\}$$

$$P^1 = \{(-e_0 - e_1 + e_2 + e_3)\}$$

This result confirms the basis we gave in the example of Section 4.4.2 (Note that $-(-e_0 - e_1 + e_2 + e_3) - (e_1 - e_2 + e_4) = e_0 - e_4 - e_3 = \partial f_0$, thus $(-e_0 - e_1 + e_2 + e_3)$ spans the same homology space as $(e_1 - e_2 + e_4)$).

4.5 Dual Mesh

let us introduce the notion of *dual mesh*. The main idea is to associate to each *primal* k-simplex a *dual* $(n$-$k)$-cell. For example, consider the tetrahedral mesh in Figure 13, we associate a dual 3-cell to each primal vertex (0-simplex), a dual face (2-cell) to each primal edge (1-simplex), a dual edge (1-cell) to each primal face (2-simplex), and a dual vertex (0-cell) to the primal tet (3-simplex). By construction, the number of dual $(n$-$k)$-cells is equal to that of primal k-simplices. The collection of dual cells is called a *cell complex*, which need not be a simplicial complex in general. Yet, the dual complex inherits several properties and operations from the primal simplicial complex. Most important is the notion of *adjacency*. For instance, if two primal edges are adjacent (they share a vertex) then the corresponding dual faces are also adjacent, that is, they share a common dual edge (which is the dual of the primal common vertex). As a result of this adjacency property, one may easily derive a boundary operator on the dual cell complex and, consequently, a discrete exterior derivative. This means that the dual cell complex also carries the structure of a chain complex. The structure on the dual complex may be linked to that of the primal complex using the Hodge star (a metric-dependent operator), as we will discuss in Section 5.

4.5.1 Dualization $*$

For simplicity, we use the circumcentric (or Voronoi) duality to construct the dual cell complex. The circumcenter of a k-simplex is defined as the center of the k-circumsphere, which is the unique k-sphere that has all $k+1$ vertices of the k-simplex on its surface. In Figure 12, we show examples of circumcentric dual cells of a 2d mesh. The dual 0-cell associated with the triangular face is the circumcenter of the triangle. The dual 1-cell associated with one of the primal edges is the line segment that joins the circumcenter of

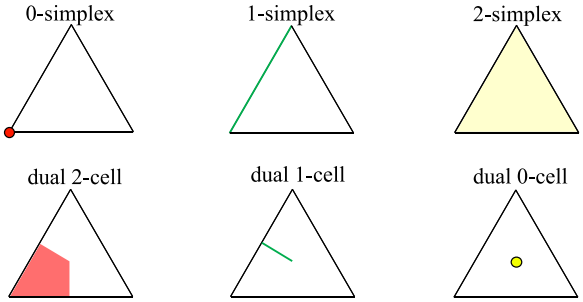


Figure 12: *A 2-dimensional example of primal and dual mesh elements. On the top row, we see the primal mesh with a representative of each simplicial complex being highlighted. The bottom row shows the corresponding circumcentric dual cells.*

the triangle to the circumcenter of that edge, while the dual 2-cell associated with a primal vertex is the union of the interior of the circumcenters of the triangle, the two adjacent edges and the vertex. Thereafter, we will denote as $*$ the operation of *duality*; that is, a primal simplex σ will have its dual called $*\sigma$ with the orientation induced by the primal orientation and the manifold's orientation. For a formal definition, we refer the reader to [Hirani 2003] for instance. It is also worth noting that other notions of duality such as the barycentric duality may be employed. For further details on dual cell decompositions, see [Munkres 1984].

4.5.2 Wedge Product

In the continuous setting, the wedge product $\wedge$ is an operation used to construct higher degree forms from lower degree ones; it is the antisymmetric part of the tensor product. For example, let α and β be 1-forms on a subset $\mathcal{R} \subset \mathbb{R}^3$, their wedge product $\alpha \wedge \beta$ is a 2-form on $\mathcal{R}$. In this case, one can relate the wedge product to the cross product of vector fields on $\mathcal{R}$. Indeed, if one considers the vector representations of α and β, the vector proxy to $\alpha \wedge \beta$ is the cross product of the two vectors. Similarly, the wedge product of a 1-form γ with the 2-form $\omega = \alpha \wedge \beta$ is a 3-form $\mu = \alpha \wedge \omega$ (also called volume-form) on $\mathcal{R}$ which is analogous to the scalar triple product of three vectors.

A discrete treatment of the wedge operator can be found in [Hirani 2003]. In this work, we only need to introduce the notion of a discrete *primal-dual wedge product*: given a primal k-cochain γ and a dual $(n$-$k)$-cochain ω, the discrete wedge product $\gamma \wedge \omega$ is an n-form (or a volume-form). For example, consider Figure 13, the wedge product of the primal 1-cochain with the dual 2-cochain is a 3-form associated with the diamond region defined by the primal edge and dual face.

5 Metric-Dependent Operators on Forms

Notice that up to now, we did *not* assume that a metric was available, *i.e.*, we never required anything to be *measured*. However, such a metric is necessary for many purposes. For instance, simulating the behavior of objects around us requires measurements of various parameters in order to be able to model laws of motion, and compare the numerical results of simulations. Consequently, a certain number of operations on forms can only be defined once a metric is know, as we shall see in this section.

5.1 Notion of Metric and Inner Product

A *metric* is, roughly speaking, a nonnegative function that describes the "distance" between neighboring points of a given space. For example, the Euclidean metric assigns to any two points in the Euclidean space $\mathbb{R}^3$, say $\mathbf{X} = (x_1, x_2, x_3)$ and $\mathbf{Y} = (y_1, y_2, y_3)$, the

number:

$$d(\mathbf{X},\mathbf{Y}) = \|\mathbf{X}-\mathbf{Y}\|^2 = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+(x_3-y_3)^2}$$

defining the "standard" distance between any two points in $\mathbb{R}^3$. This metric then allows one to measure length, area, and volume. The Euclidean metric can be expressed as the following quadratic form:

$$g^{\text{Euclid}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Indeed, the reader can readily verify that this matrix g satisfies: $d^2(\mathbf{X},\mathbf{Y}) = (\mathbf{X}-\mathbf{Y})^t g(\mathbf{X}-\mathbf{Y})$. Notice also that this metric *induces an inner product* of vectors. Indeed, for two vectors $\mathbf{u}$ and $\mathbf{v}$, we can use the matrix g to define:

$$\mathbf{u}\cdot\mathbf{v} = \mathbf{u}^t g\,\mathbf{v}.$$

Once again, the reader is invited to verify that this equality does correspond to the traditional dot product when g is the Euclidean metric. Notice that on a non-flat manifold, subtraction of two points is only possible for points infinitesimally close to each other, thus the metric is actually defined pointwise for the tangent space at each point: it does not have to be constant. Finally, notice that a volume form can be induced from a metric by defining $\mu^n = \sqrt{det(g)}dx^1 \wedge \ldots \wedge dx^n$.

5.2 Discrete Metric

In the discrete setting presented in this paper, we only need to measure length, area, volume of the simplices and dual cells. We therefore do not have a full-blown notion of a metric, only a *discrete metric*. Obviously, if one were to use a finer mesh, more information on the metric would be available: having more values of length, area, and volume known in each independent directions for all neighborhoods provides a better approximation of the real, continuous metric.

5.3 The Differential Hodge Star

Let us go back for a minute to the differential case to explain a new concept. Recall that the metric defines an inner product for vectors. This notion also extends to forms: given a metric, one can define the product of two k-forms $\in \Omega^k(\mathcal{M})$ which will measure, in a way, the projection of one onto the other. A formal definition can be found in [Abraham et al. 1988]. Given this inner product, we can introduce an operator denoted as $\star$, called the *Hodge star*, that maps a k-form to a complimentary $(n$-$k)$-form:

$$\star:\ \Omega^k(\mathcal{M}) \to \Omega^{n-k}(\mathcal{M}),$$

and is defined to satisfy the following equality:

$$\alpha \wedge \star\beta = \langle \alpha, \beta \rangle\ \mu^n$$

for any pair of k-forms α and β (recall that μ^n is the volume form induced by the metric g). However, notice that the wedge product is very special here: it is the product of k-form and a $(n$-$k)$-form, two complementary forms. This fact will drastically simplify the discrete counterpart of the Hodge star, as we now cover.

5.4 Discrete Hodge Star

In the discrete setting, the Hodge star becomes easier: we only need to define how to go from a *primal* k-cochain to a *dual* $(n$-$k)$-cochain, and vice-versa. By definition of the dual mesh, k-chains and dual $(n$-$k)$-chains are represented by vectors of the same dimension. Just like for the discrete exterior derivative (coboundary)

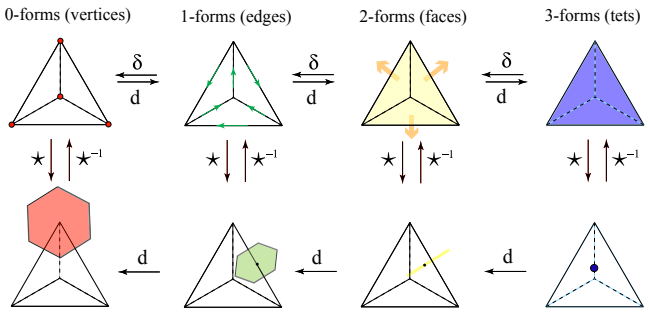


Figure 13: *On the first line, the 'primal' chain complex is depicted and on the second line we see the dual chain complex (i.e., cells, faces, edges and vertices of the Voronoi cells of each vertex of the primal mesh).*

operator, we may use a $|\mathcal{K}^k| \times |\mathcal{K}^k|$ square matrix to represent the Hodge star. Now the question is: what should the coefficients of this matrix be?

For numerical purposes we want it to be symmetric, positive definite, and sometimes, even diagonal for faster computations. One such diagonal Hodge star can be defined such as the diagonal elements are the ratio of sizes of a k-simplex and its dual $(n$-$k)$-simplex. In other words, we can define the discrete Hodge star through the following simple rule:

$$\frac{1}{|\sigma^k|}\int_{\sigma^k}\omega = \frac{1}{|*\sigma^k|}\int_{*\sigma^k}\star\omega \tag{8}$$

Therefore, any primal value of a form can be easily *transferred* to the dual mesh through proper scaling—and vice-versa; to be precise, we have:

$$\star\star = (-1)^{k(n-k)}\mathrm{Id}, \tag{9}$$

which means that $\star$ on the dual mesh is the inverse of the $\star$ on the primal *up to a sign* (the result of antisymmetry of wedge product, which happens to be positive for any k when $n = 3$).

So we use the inverse of this star to go from dual $(n$-$k)$-cochain to k-cochain. Since there is no ambiguity if we know the form is primal or dual, we also use $\star$ to denote the star for dual cochain, just as we use d as a context-dependent operator

Implementation Based on Eq. 8, the inner product of forms α^k and β^k at the diamond-shaped region formed by each k-simplex and its dual $(n-k)$-simplex is simply the product of the value of α at that k-simplex and value of $\star\beta$ at that dual $(n-k)$-simplex. Therefore, the sum over the whole space gives the following inner product (which involves only linear algebra matrix and vector multiplications)

$$\langle \alpha^k, \beta^k \rangle = \alpha^t \star \beta. \tag{10}$$

where the Hodge star matrix has, as its only non-zero coefficients, the following diagonal terms:

$$(\star_k)_{qq} = |(*\sigma)_q|/|(\sigma_q)|.$$

Notice that this definition of the inner product, when $\alpha = \beta$, induces the definition of the norm of k-forms.

Again, there are three different Hodge stars in $\mathbb{R}^3$, one for each simplex dimension. But as we discussed for all the other operators, the dimension of the form on which this operator is applied disambiguates which star is meant. So we will not encumber our notation with unnecessary indices, and will only use the symbol $\star$ for any of the three stars implied.

The development of an accurate, yet fast to compute, Hodge star is still an active research topic. However, this topic is beyond the

scope of the current paper and will be addressed in a future publication.

5.5 Discrete Codifferential Operator δ

We already have a linear operator d which maps a k-form to a $k+1$-form, but we do not have a linear operator which maps a $k+1$-form to a k-form. Having defined a discrete Hodge star, one can now create such an *adjoint* operator δ for the discrete exterior derivative d. Here, adjoint is meant with respect to the inner product of forms; that is, this operator δ satisfies:

$$\langle d\alpha, \beta \rangle \;=\; \langle \alpha, \delta\beta \rangle \qquad \forall \alpha \in \Omega^{k-1}(M), \beta \in \Omega^{k}(M)$$

For a smooth, compact manifold without boundary, one can prove that $(-1)^{n(k-1)+1} \star d \star$ satisfies the above condition [Abraham et al. 1988]. Let us try to use the same definition in the discrete setting; *i.e.*, we wish to define the discrete δ by the relation:

$$\delta \equiv (-1)^{n(k-1)+1} \star d \star = (-1)^{n(k-1)+1} \star (-1)^{k} d_p^t \star, \quad (11)$$

Beware that the exterior derivative here (of rank (n-k)) is the transpose of the corresponding primal d (of rank k, which we denote in the equation by d_p to emphasize we are using the actual matrix instead of the context-dependent operator) up to a sign, since the boundary operator on the dual is the transpose of the primal one up to a sign. It is now straightforward to verify that the following series of equalities:

$$\begin{aligned}
\langle d\alpha, \beta \rangle &\overset{Eq.\,(10)}{=} (d\alpha)^t \star \beta \\
&\overset{Eq.\,(9)}{=} \alpha^t (-1)^{(k-1)(n-(k-1))} \star \star d_p^t \star \beta \\
&= \alpha^t (-1)^{n(k-1)+1} \star \star (-1)^{k} d_p^t \star \beta \\
&\overset{Eq.\,(11)}{=} \langle \alpha, \delta\beta \rangle
\end{aligned}$$

holds on the discrete manifold. So indeed, the discrete d and δ are also adjoint in a similar fashion in the discrete setting. For this reason, δ is called the *codifferential operator*.

Implementation of the Codifferential Operator Thanks to this easily-proven adjointness, the implementation of the discrete codifferential operator is a trivial matter: it is simply the product of three matrices, mimicking exactly the differential definition mentioned in Eq. (11).

5.6 Exercise: Laplacian Operator

At this point, the reader is invited to perform a little exercise. Let us first state that the Laplacian Δ of a form is defined as: $\Delta = \delta d + d\delta$. Now, applied to a 0-form, notice that the latter term disappears. Question: in 2D, what *is* the Laplacian of a function f at a vertex i? The answer is actually known: it is the now famous *cotangent formula* [Pinkall and Polthier 1993], since the ratio of primal and dual edge sizes leads to such a trigonometric equality.

6 Interpolation of Discrete Forms

In Section 3.4, we argued that k-cochains are discretizations of k-forms. This representation of discrete forms on chains, although very convenient in many applications, is not sufficient to fulfill certain demands such as obtaining a point-wise value of the k-form. As a remedy, one can use an interpolation of these chains to the rest of space. For simplicity, these interpolation functions can be taken to be linear (by linear, we mean with respect to the coordinates of the vertices).

6.1 Interpolating 0-forms

It is quite obvious to linearly interpolate discrete 0-forms (as 0-cochains) to the whole space: we can use the usual vertex-based linear interpolation basis, often referred to as the *hat function* in the Finite Element literature. This basis function will be denoted as φ_i for each vertex v_i. By definition, φ_i satisfies:

$$\varphi_i = 1 \quad \text{at } v_i, \qquad \varphi_i = 0 \quad \text{at } v_j \neq v_i$$

while φ_i linearly goes to zero in the one-ring neighborhood of v_i. The reader may be aware that these functions are, within each simplex, *barycentric coordinates*, introduced by Möbius in 1827 as mass points to define a *coordinate-free geometry*.

With these basis functions, one can easily check that if we denote a vertex v_j by σ_j, we have:

$$\int_{v_j} \varphi_{v_i} = \int_{\sigma_j} \varphi_{\sigma_i} = \int_{\sigma_j} \varphi_i = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Therefore, these interpolating functions represent a basis of 0-cochains, that exactly corresponds to the dual of the natural basis of 0-chains.

6.2 Interpolating 1-forms

We would like to be able to extend the previous interpolation technique to 1-forms now. Fortunately, there is an existing method to do just that: the *Whitney* 1-form (used first in [Whitney 1957]) associated with an edge σ_{ij} between v_i and v_j is defined as:

$$\varphi_{\sigma_{ij}} = \varphi_i d\varphi_j - \varphi_j d\varphi_i.$$

A direct computation can verify that:

$$\int_{\sigma_{kl}} \varphi_{\sigma_{ij}} = \begin{cases} 1 & \text{if } i = k \text{ and } j = l, \\ -1 & \text{if } i = l \text{ and } j = k, \\ 0 & \text{otherwise.} \end{cases}$$

Indeed, it is easy to see that the integral is 0 when we are not integrating it on edge e_{ij}, because at least one of the vertex (say, i) is not on the edge, thus, $\varphi_i = 0$ and $d\varphi_i = 0$ on the edge. However, along the edge σ_{ij}, we have $\varphi_i + \varphi_j = 1$, therefore:

$$\int_{\sigma_{ij}} \varphi_{\sigma_{ij}} = \int_{\varphi_i=1}^{\varphi_i=0} (\varphi_i d(1-\varphi_i) - (1-\varphi_i) d\varphi_i) = \int_{\varphi_i=1}^{\varphi_i=0} (-d\varphi_i) = 1.$$

We thus have defined a correct basis for 1-cochains this time.

6.3 Interpolating with Whitney k-Forms

One can extend these 1-form basis functions to arbitrary k-simplices. In fact, Whitney k-forms are defined similarly:

$$\varphi_{\sigma_{i_0,i_1,\ldots,i_k}} = k! \sum_{j=0\ldots k} (-1)^j \varphi_{i_j} d\varphi_{i_0} \wedge \ldots \wedge \widehat{d\varphi_{i_j}} \ldots \wedge \varphi i_k$$

where $\widehat{d\varphi_{i_p}}$ means that $d\varphi_{i_p}$ is excluded from the product. Notice how this definition exactly matches the case of vertex and edge bases, and extends easily to higher dimensional simplices.

Remark If a metric is defined (for instance, the Euclidean metric), we can simply identify $d\varphi$ with $\nabla\varphi$ for the real calculation. This corresponds to the notion of *sharp* ($\sharp$), but we will not develop this point other than for pointing out the following remark: the traditional gradient of a linear function f in 2D, known to be constant per triangle, can indeed be re-written à la Whitney:

$$\nabla f = \sum_i f_i \nabla\varphi_i \overset{\varphi_i+\varphi_j+\varphi_k=1}{=} \sum_{i,j,i\neq j} (f_i - f_j)(\varphi_i \nabla\varphi_j - \varphi_j \nabla\varphi_i).$$

The values $(f_i - f_j)$ are the edge values associated with the gradient, *i.e.*, the values of the one-form df.

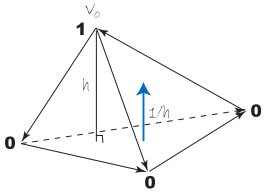


Figure 14: $\nabla\varphi$ *for the vertex on top*

Basis of Forms The integration of the Whitney form associated with a k-simplex will be 1 on that particular k-simplex, and 0 on all others. Indeed, it is a simple exercise to see that the integration of Whitney k-form basis associated to a k-simplex is 0 on a different k-simplex, because there is *at least one* hat function φ_j that is valued 0, and since φ_j or $d\varphi_j$ appears in every term, the integral is 0. To see that the integral is 1 on the simplex itself, we can use Stokes' theorem 9as our discrete forms satisfies it exactly on simplices): first, suppose $k < n$, and pick a $k+1$-simplex, such that the k-simplex is a face of it. Since it is 0 on other faces, the integral is equal to the integral of $d\varphi_\sigma = (k+1)! d\varphi_{i_0} \wedge ... \wedge \varphi i_k$ on the $k+1$-simplex, if we use φ_{i_j} as a local reference frame for the integration, $\int_{\sigma^{k+1}} d\varphi_{i_0} \wedge ... \wedge \varphi i_k$ is simply the volume of a standard simplex, which is $\frac{1}{(k+1)!}$, thus the integral is 1. The case when $k = n$ is essentially the same as $k = n-1$.

This means that these Whitney forms are forming a basis of their respective form spaces. In a way, these bases are an extension of the Finite Element bases defined on nodes, or of the Finite Volume elements that are constant per tet.

Note finally that the Whitney forms are not continuous; however, they *are* continuous *along the direction of the k-simplex* (*i.e.*, tangential continuity for 1-forms, and normal continuity for 2-forms); this is the only condition needed to make the integration well defined. In a way, this property is the *least* we can ask them to be. We would lose generality if we were to add any other condition! The interested reader is referred to [Bossavit 1998] for a more thorough discussion on these Whitney bases.

7 Application to Hodge Decomposition

We now go through a first application of the discrete exterior calculus we have set up up to now. As we will see, the discrete case is often much simpler that its continuous counterpart; yet it captures the same properties.

7.1 Introducing the Hodge Decomposition

It is convenient in some applications to use the Helmholtz-Hodge decomposition theorem to decompose a given continuous vector field or differential form (defined on a smooth manifold $\mathcal{M}$) into components that are mutually orthogonal (in $\mathcal{L}^2$ sense), and easier to compute (see [Abraham et al. 1988] for details). In fluid mechanics for example, the velocity field is generally decomposed into a part that is the gradient of a potential function and a part that is the curl of a stream vector potential (see Section 8.3 for further details), as the latter one is the incompressible part of the flow. When applied to k-forms, such decomposition is known as the *Hodge decomposition for forms* and can be stated as follows:

Given a manifold $\mathcal{M}$ and a k-form ω^k on $\mathcal{M}$ with appropriate boundary conditions, ω^k can be decomposed into the sum of the exterior derivative of a $(k\text{-}1)$-form α^{k-1}, the codifferential of a $(k\text{+}1)$-form β^{k+1}, and a harmonic k-form h^k:

$$\omega^k = d\alpha^{k-1} + \delta\beta^{k+1} + h^k.$$

Here, we use the term *harmonic* to mean that h^k satisfies the equation $\Delta h^k = 0$, where Δ is the Laplacian operator defined as $\Delta = d\delta + \delta d$. The proof of this theorem is mathematically involved and requires the use of elliptic operator theory and similar tools, as well as a careful study of the boundary conditions to ensure uniqueness. The discrete analog that we propose has a very simple and straightforward proof as shown below.

7.2 Discrete Hodge Decomposition

In the discrete setting, the discrete operators such as the exterior derivative and the codifferential can be expressed using matrix representation. This allows one to easily manipulate these operators using tools from linear algebra. In particular, the discrete version of the Hodge decomposition theorem becomes a simple exercise in linear algebra. Note that we will assume a boundaryless domain for simplicity (the generalization to domains with boundary is conceptually as simple).

Theorem 7.1 *Let $\mathcal{M}$ be a discrete manifold and let $\Omega^k(\mathcal{M})$ be the space of discrete Whitney k-forms on $\mathcal{M}$. Consider the linear operator $d^k : W^k \to W^{k+1}$, such that $d^{k+1} \circ d^k = 0$, and a discrete Hodge star which is represented as a symmetric, positive definite matrix. Furthermore, define the codifferential (the adjoint of the operator d) as done in Section 5.5; namely, let $\delta^{k+1} = (-1)^{n(k-1)+1} (\star^k)^{-1} (d^k)^t \star^{k+1}$. In this case, the following orthogonal decomposition holds for all k:*

$$\Omega^k(\mathcal{M}) = d\Omega^{k-1}(\mathcal{M}) \oplus \delta\Omega^{k+1}(\mathcal{M}) \oplus \mathcal{H}^k(\mathcal{M})$$

where $\oplus$ means direct sum, and $\mathcal{H}^k(\mathcal{M})$ is the space of harmonic k-forms on $\mathcal{M}$, that is, $\mathcal{H}^k(\mathcal{M}) = \{h \mid \Delta^k h = 0\}$.

Proof For notational convenience, we will omit the superscript of the operators when the rank is obvious. We first prove that the three component spaces are orthogonal. Clearly, using the facts that the Laplacian operator Δ is equal to $d\delta$+δd and that d and δ are adjoint operators, one has that $\forall h \in \mathcal{H}^k$:

$$\begin{aligned} \langle \Delta h, h\rangle = 0 \;&\Rightarrow\; \langle d\delta h, h\rangle + \langle \delta d h, h\rangle = \langle dh, dh\rangle + \langle \delta h, \delta h\rangle = 0 \\ &\Rightarrow\; dh = 0 \text{ and } \delta h = 0 \end{aligned}$$

Also, $\forall\alpha$ and $\beta \in \Omega^k(\mathcal{M})$, one has:

$$\langle d\alpha, \delta\beta\rangle = \langle dd\alpha, \beta\rangle = 0$$

and

$$\langle d\alpha, h\rangle = \langle \alpha, \delta h\rangle = 0 \qquad \langle h, \delta\beta\rangle = \langle dh, \beta\rangle = 0$$

Now, any k-form that is perpendicular to $d\Omega^{k-1}(\mathcal{M})$ and $\delta\Omega^{k+1}(\mathcal{M})$ must be in $\mathcal{H}^k(\mathcal{M})$, because this means $dh = 0$ and $\delta h = 0$, so $\Delta h = d\delta h + \delta dh = 0$.

Alternatively, we can prove that:

$$\Omega^k(\mathcal{M}) = \Delta\Omega^k(\mathcal{M}) \oplus \mathcal{H}^k(\mathcal{M}).$$

By analogy to the previous argument, it is easy to show that $\Delta\Omega^k$ is orthogonal to $\mathcal{H}^k$. Additionally, the dimension of these two spaces sum up to the dimension of Ω^k, which means the decomposition is complete. ∎

Note that the reader can find a similar proof given in Appendix B of [Frankel 2004], where it is used for Kirchhoff's Circuits Laws. There, Frankel does not mention that we can actually use cochains as the discretization of forms, and his operations using a "metric" of cochains can be interpreted as a Hodge star.

Implementation of the Discrete Hodge Decomposition Before we discuss how to numerically implement the discrete Hodge decomposition, we prove a useful result (that has a continuous analog).

Lemma 7.2 *In the discrete setting, one can find exactly one harmonic cochain from each cohomology equivalence class.*

Proof It is can be readily shown that the bases of harmonic cochains and the cohomology groups both have the dimension equal to $dim(\text{Ker } d^k) - dim(\text{Im } d^{k-1})$. To this end, recall that a cohomology basis is defined as is $\text{Ker}(d^k)/\text{Im}(d^{k-1})$ and has the $dim(\text{Ker } d^k) - dim(\text{Im } d^{k-1})$. Now, in order to see that the space of harmonic cochains has this same dimension, simply note that: $\text{Ker}(d^k) = d\Omega^{k-1} \oplus \mathcal{H}^k$.

Now, the equation $\delta(\omega + df) = 0$ has a solution for each ω in one cohomology equivalence class. We know that the cochains forming different cohomology groups are linearly independent, hence, we conclude that these harmonic cochains span $\mathcal{H}^k$. ∎

By virtue of the above lemma, the implementation of the Hodge decomposition is simply recursive in the rank of the form (i.e., cochain). The case of 0-forms is trivial: fix one vertex to a constant, and solve the Poisson equation for 0-forms. Now suppose that we have a decomposition working for $(k-1)$-forms, and we look for the decomposition of k-forms. Our approach is to get the harmonic component h^k first, so that we only need to solve a Poisson equation for the rest:

$$\Delta\omega^k = f^k - h^k \tag{12}$$

One is left with the problem of finding a basis of harmonic forms. Since we are given a Hodge star operator, we will use it to define the metric on the space of cochains. This metric allows us to define a basis for harmonic k-form (the dimension of this harmonic space is generally small, since it is the k-th Betti number β_k). First, one needs to calculate the cohomology basis $\{P_i\}$ based on the algorithm in Section 4.4.4. Once we have $\{P_i\}$, we solve one special decomposition of (k-1)-forms by first computing the forms f_i satisfying:

$$\Delta f_i = -\delta P_i \tag{13}$$

Now $H^k = P_i + df_i$ gives us the forms in basis of harmonic k-form space. After normalization, we have the basis to calculate the projection $h^k = HH^t f^k$, where we assemble all H^k into a matrix H. This completes the procedure of calculating the decomposition.

A non-singular matrix is often preferable when it comes to solve a linear system efficiently; we can change the Laplacian matrix slightly to make the Poisson equation satisfy this requirement. First, we can get a orthonormal basis for harmonic form space (the dimension is β^k). Now for basis e^j (column vector with j-th element equal to 1, and 0 everywhere else), take the distance of e^j to the harmonic space $|e^j - HH^t e^j|$; notice that this can be done in constant time. Now take out the j-th column and j-th row of Δ if e^j has the smallest distance from harmonic space, and repeat the step for β^k times. We are left with a non-singular matrix, and the solution to the new linear system is a solution to the original Poisson equation.

8 Others Applications

8.1 Form-based Proof of Tutte's Theorem

The notion of forms as convenient, intrinsic substitutes for vector fields has been used to provide a concise proof of the celebrated *Tutte's Embedding Theorem*. This important result in graph theory states that if one fixes the boundary of a 3-connected graph (i.e., a typical polygonal mesh) to a convex domain of the plane and ensures that every non-boundary vertex is a *strict convex combination* of its neighbors, then one obtains a planar straight-line embedding of the graph. In other words, this embedding procedure will not result in fold-overs. A significantly shorter alternative to the original proof of this theorem was proposed by Gortler, Gotsman, and Thurston [Gortler et al. 2004], using discrete 1-forms on edges. We now present a sketch of their approach, using a formulation more in line with the terms we used in this paper.

A Tutte embedding assigns to each vertex v_i of a graph G some 2D coordinates $\mathbf{X}(v_i) = (x(v_i), y(v_i))$. By definition, each interior vertex v_i satisfies a linear condition on its coordinates of the form: $\mathbf{X}(v_i) = \sum_{v_j \in \mathcal{N}(i)} w_{ij}\mathbf{X}(v_j)$, where $\mathcal{N}(i)$ is the set of 1-ring neighbors of vertex v_i. These coefficients w_{ij} are all positive due to the condition of strict convex combination mentioned above. Now, for a given Tutte embedding, one can construct a 0-form $z(v) = \alpha x(v) + \beta y(v)$ for any pair of positive coefficients α and β. Notice that this 0-form satisfies the same convex combination condition: $z(v_i) = \sum_{v_j \in \mathcal{N}(i)} w_{ij} z(v_j)$. Because of their strict positiveness, one can identify these coefficients w_{ij} to the diagonal Hodge star of primal 1-forms (see Section 7), defined by a particular metric. Therefore, the relationship $0 = \sum_{v_j \in \mathcal{N}(i)} w_{ij}(z(v_j) - z(v_i))$ is equivalent to: $d \star dz = 0$. There are two immediate conclusions:

- ◇ the 1-form $\omega = dz$ is closed (since it is the exterior derivative of a 0-form), and
- ◇ it is also co-closed since $\delta\omega = (\star d\star)dz = \star(d \star dz) = 0$.

To use the previously defined 1-form ω to prove Tutte's theorem, Gortler *et al.* then invoke the usual definition of index of vector fields. This concept is one of the oldest in Algebraic Topology, initially stated by Poincaré and then developed by Hopf and Morse in the continuous case. Its discrete counterpart was first proposed by Banchoff, and used for instance in [Lazarus and Verroust 1999]. A discrete Poincare-Hopf index theorem also holds, stating that the sum of all indices must be equal to 2 for a genus-0 patch. The final argument uses the link between (co)closed forms and their indices. Indeed, because we found a closed *and* coclosed form ω, it can be easily shown that these two properties induce that the index of each face must be less or equal to zero, as well as the index of each vertex. Because the boundary of the patch is convex, only two vertices on the boundary have index 1. Since all the indices must sum to 2 and each interior index must less than zero, we can conclude that *each interior index is zero*. Because this argument is valid for every positive pair (α, β), one can easily deduce that each interior face is convex and each vertex is a "wheel"; thus, injectivity can be guaranteed.

This rather elegant proof demonstrates how discrete forms and their obvious links to Algebraic Topology can be quite powerful in a variety of applications. We also point the interested reader to papers on conformal parameterizations, such as [Mercat 2001; Gu and Yau 2003], for which special discrete Hodge stars are defined to satisfy

a discrete definition of conformality: there are also very interesting research on this particular topic, once again using the calculus of exterior forms.

8.2 Electromagnetism with Forms

Electromagnetism can be formulated very elegantly using differential forms. For a detailed exposition of the geometric structure in E&M, we refer the reader to [Bossavit 1998] and [Warnick et al. 1997]. In this approach, the electric field E is represented by a 1-form, as the integral of E along a path traced by a test charge q is equal to the electromotive force experienced by that charge. The electric displacement L as well as the current density J are represented by 2-forms. The charge distribution ρ is a 3-form. The magnetic field B is represented by a 2-form since it is measured as a flux. whereas the magnetic field intensity H is a 1-form.

With these conventions, Maxwell's equations can be rewritten as follows:

$$\partial_t B + dE = 0, \qquad -\partial_t L + dH = J, \qquad dL = \rho, \qquad (14)$$

subject to the *constitutive* equations:

$$L = \epsilon E, \qquad H = \mu B, \qquad (15)$$

where ϵ is the permittivity, and μ is the permeability. The constitutive relations (15) are very similar to the Hodge star operator that transforms a k-form to an $(n\text{-}k)$-form. Here, ϵ operates on the electric field E (1-form) to yield the electric displacement L (2-form) while μ transforms the magnetic field B (2-form) into the magnetic field intensity H (1-form). To this end, one may think of both ϵ and μ as Hodge star operators induced from appropriately chosen metrics. Note that the balance laws in (14) are metric-independent.

As the reader can guess, one can readily discretize this representation of the physical quantities E, L, ... and the associated system of equations (14-15) using the tools presented in this chapter. The resulting numerical algorithm preserves *exactly* the geometric structure of the system, see [Bossavit 1998].

8.3 Fluids

The geometric structure of Fluid Mechanics, specifically Euler's equations for inviscid fluids, has been investigated (see [Marsden and Weinstein 1983] and references therein). In this geometric framework, vorticity are represented as a two-form (an area-form) and Euler's equations can be written as vorticity advection. Roughly speaking, vorticity measures the rotation of a fluid parcel; we say the fluid parcel has vorticity when it spins as it moves along its path. Vorticity advection means that the vorticity (as a two-form) moves dynamically as if it is pushed forward by the fluid flow. The integral of the vorticity on a given bounded domain is equal, by Stokes theorem, to the circulation around the loop enclosing the domain. This quantity as well as the total energy of the fluid are conserved in the absence of external forcing. Inspired by this geometric viewpoint and in light of the present development of Discrete Exterior Calculus, we propose a discrete differential approach to fluid mechanics and an integration scheme that satisfy the properties of conservation of circulation and energy, see Chapter 9 for further detail.

9 Conclusions

In this chapter, we have stated the importance of using discrete differential forms in computational science. We have also given a discrete version of the Hodge decomposition, useful for a number of computations in various fields. This geometric approach to computations is particularly novel, thus many details need to be explored and proven superior to the current approaches. In particular, we have the burden of proof when it comes to the generality of our approach. In order to work towards this goal, we are currently working on revisiting the foundations of elasticity, to demonstrate that this idea of forms as fundamental elements of differential equations can even be used in this context.

Acks

The authors wish to first thank Jerrold E. Marsden for his tremendous help and support along the way. We also wish to acknowledge the support of Anil Hirani, Melvin Leok, David Cohen-Steiner, Peter Schröder, Sharif Elcott, Pierre Alliez, and Eitan Grinspun.

References

ABRAHAM, R., AND SHAW, C., Eds. 1984. *Dynamics: The Geometry of Behavior*. Ariel Press (Santa Cruz, CA).

ABRAHAM, R., MARSDEN, J., AND RATIU, T., Eds. 1988. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences Vol. 75, Springer.

BJÖRNER, A., AND WELKER, V. 1995. The homology of "k-equal" manifolds and related partition lattices. *Advances in Math. 110*, 277–313.

BOBENKO, A., AND SEILER, R., Eds. 1999. *Discrete Integrable Geometry and Physics*. Clarendon Press.

BOSSAVIT, A. 1998. *Computational Electromagnetism*. Academic Press, Boston.

BOSSAVIT, A. 2003. *Personal Communications*.

BURKE, W. L. 1985. *Applied Differential Geometry*. Cambridge University Press.

CARROLL, S. 2003. *Spacetime and Geometry: An Introduction to General Relativity*. Pearson Education.

CARTAN, É. 1945. *Les Systèmes Differentiels Exterieurs et leurs Applications Géometriques*. Hermann, Paris.

DESBRUN, M., LEOK, M., AND MARSDEN, J. E. 2004. Discrete Poincaré Lemma . *Appl. Num. Math.*.

DIMAKIS, A., AND MÜLLER-HOISSEN, F. 1994. Discrete Differential Calculus, Graphs, Topologies, and Gauge Theory. *Journal of Mathematical Physics 35*, 6703–6735.

DORAN, C., AND LASENBY, A., Eds. 2003. *Geometric Algebra for Physicists*. Cambridge University Press.

EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN, A. 2000. Topological persistence and simplification. In *IEEE Symposium on Foundations of Computer Science*, 454–463.

ELCOTT, S., TONG, Y., KANSO, E., DESBRUN, M., AND SCHRÖDER, P. 2005. Discrete, Circulation-preserving and Stable Simplicial Fluid. *under review*.

FLANDERS, H. 1990. *Differential Forms and Applications to Physical Sciences*. Dover Publications.

FLANDERS, H., Ed. 2001. *Geometric Methods for Computational Electromagnetics*. EMW Publishing, Cambridge Mass.

FORMAN, R. 2005. Bochner's Method for Cell Complexes and Combinatorial Ricci Curvature. *to appear in the J. of Discrete and Computational Geom.*.

FRANKEL, T. 2004. *The Geometry of Physics*. Second Edition. Cambridge University Press, United Kingdom.

GORTLER, S., GOTSMAN, C., AND THURSTON, D. 2004. One-Forms on Meshes and Applications to 3D Mesh Parameteriza-

tion. Tech. Rep. TR-12-04, Harvard University.

GRAY, A., Ed. 1998. *Modern Differential Geometry of Curves and Surfaces*. Second edition. CRC Press.

GROSS, P. W., AND KOTIUGA, R. 2004. *Electromagnetic Theory and Computation: A Topological Approach*. Cambridge University Press.

GU, X., AND YAU, S.-T. 2003. Global conformal surface parameterization. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Eurographics Association, 127–137.

HARRISON, J. 2005. Ravello Lecture Notes on Geometric Calculus – Part I. Tech. rep., UC Berkeley.

HATCHER, A. 2004. *Algebraic Topology*. Cambridge University Press.

HILDEBRANDT, K., AND POLTHIER, K. 2004. Anisotropic filtering of non-linear surface features. In *Computer Graphics Forum*, M.-P. Cani and M. Slater, Eds., vol. 23. Proc. Eurographics 2004.

HIRANI, A. N. 2003. *Discrete Exterior Calculus*. PhD thesis, Caltech.

HYMAN, J. M., AND SHASHKOV, M. 1997. Natural Discretizations for the Divergence, Gradient, and Curl. *International Journal of Computers and Mathematics with Applications 33*, 277–313.

KANSO, E., ARROYO, M., DESBRUN, M., MARSDEN, J. E., AND TONG, Y. 2004. On the geometric character of continuum mechanics. *completed and to be submitted*.

LAZARUS, F., AND VERROUST, A. 1999. Level Set Diagrams of Polyhedral Objects. In *Proceedings of the* 5^{th} *ACM Symposium on Solid Modeling and Applications*, 130–140.

LOVELOCK, D., AND RUND, H. 1993. *Tensors, Differential Forms, and Variational Principles*. Dover Publications.

MADSEN, I. H., AND TORNEHAVE, J. 1997. *From Calculus to Cohomology : De Rham Cohomology and Characteristic Classes*. Cambridge University Press, United Kingdom.

MARSDEN, J. E., AND HUGHES, T. 1983. *Mathematical Foundations of Elasticity*. Dover, New York.

MARSDEN, J. E., AND WEINSTEIN, A. 1983. Coadjoint orbits, vortices and Clebsch variables for incompressible fluids. *Physica D 7*, 305–323.

MARSDEN, J. E., AND WEST, M. 2001. Discrete Mechanics and Variational Integrators . *Acta Numerica*.

MCCORMICK, S. F. 1989. *Multilevel Adaptive Methods for Partial Differential Equations — Chapter 2: The Finite Volume Method*, vol. 6. SIAM.

MERCAT, C. 2001. Discrete Riemann Surfaces and the Ising Model. *Commun. Math. Phys. 218*, 1, 177–216.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2002. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Proceedings of VisMath*.

MORITA, S. 2001. *Geometry of Differential Forms*. Translations of Mathematical Monographs, Vol. 201. Am. Math. Soc.

MUNKRES, J. R. 1984. *Elements of Algebraic Topology*. Addison-Wesley, Menlo Park, CA.

PINKALL, U., AND POLTHIER, K. 1993. Computing Discrete Minimal Surfaces. *Experimental Mathematics 2*, 1, 15–36.

POLTHIER, K., AND PREUSS, E. 2000. Variational Approach to Vector Field Decomposition. *Scientific Visualization, Springer Verlag (Proc. of Eurographics Workshop on Scientific Visualization)*.

POLTHIER, K., AND PREUSS, E. 2002. Identifying Vector Fields Singularities using a Discrete Hodge Decomposition. *Visualization and Mathematics III, Eds: H.C. Hege, K. Polthier, Springer Verlag*.

POLTHIER, K. 2002. Computational Aspects of Discrete Minimal Surfaces. *Proceedings of the Clay Summer School on Global Theory of Minimal Surfaces (Hass, Hoffman, Jaffe, Rosenberg, Schoen, and Wolf Editors)*.

SCHREIBER, U. 2003. On Superstrings in Schrödinger Representation. *Preprint*.

SHARPE, R. W. 1997. *Differential Geometry:Cartan's Generalization of Klein's Erlangen Programme*. Springer-Verlag, NY.

TONG, Y., LOMBEYDA, S., HIRANI, A. N., AND DESBRUN, M. 2003. Discrete Multiscale Vector Field Decomposition. *ACM Trans. Graph. 22*, 3, 445–452.

WARNICK, K. F., SELFRIDGE, R. H., AND ARNOLD, D. V. 1997. Teaching Electromagnetic Field Theory Using Differential Forms. *IEEE Trans. on Education 40*, 1, 53–68.

WHITNEY, H. 1957. *Geometric Integration Theory*. Princeton Press, Princeton.

ZAPATRIN, R. 1996. Polyhedral Representations of Discrete Differential Manifolds. *Preprint*.

Further Reading

Despite a large number of theoretical books, we are aware of only a few books with a truly "applied flavor", in line with this chapter. For applications based on this exterior calculus or other geometric algebras, see [Bossavit 1998; Flanders 2001; Bobenko and Seiler 1999; Doran and Lasenby 2003; Gross and Kotiuga 2004]. The reader interested in the application of differential forms to E&M is further referred to [Warnick et al. 1997], for applications in fluid mechanics see [Marsden and Weinstein 1983], and in elasticity see [Kanso et al. 2004] and [Frankel 2004]. The reader is also invited to check out current developments of variants of DEC, for instance, in [Dimakis and Müller-Hoissen 1994; Schreiber 2003; Zapatrin 1996; Harrison 2005].

Finally, the interested reader can find additional material on the following websites:

Graphics and Discrete Differential Calculus at Caltech:
http://www.multires.caltech.edu/pubs/
http://www.cs.caltech.edu/∼mathieu

Computational E&M (Alain Bossavit):
http://www.lgep.supelec.fr/mse/perso/ab/bossavit.html

Discrete Vector Fields and Combinatorial Topology (Robin Forman):
http://math.rice.edu/∼forman/

Discrete Mechanics at Caltech (Jerrold E. Marsden):
http://www.cds.caltech.edu/∼marsden/

Chapter 8:
Building Your Own DEC at Home

Sharif Elcott
Caltech

Peter Schröder
Caltech

1 Overview

The methods of Discrete Exterior Calculus (DEC) have given birth to many new algorithms applicable to areas such as fluid simulation, deformable body simulation, and others. Despite the (possibly intimidating) mathematical theory that went into deriving these algorithms, in the end they lead to simple, elegant, and straightforward implementations. However, readers interested in implementing them should note that the algorithms presume the existence of a suitable simplicial complex data structure. Such a data structure needs to support local traversal of elements, adjacency information for all dimensions of simplices, a notion of a *dual mesh*, and all simplices must be *oriented*. Unfortunately, most publicly available tetrahedral mesh libraries provide only *unoriented* representations with little more than vertex-tet adjacency information (while we need vertex-edge, edge-triangle, edge-tet, *etc.*). For those eager to implement and build on the algorithms presented in this course without having to worry about these details, we provide an implementation of a DEC-friendly tetrahedral mesh data structure in C++. This chapter documents the ideas behind the implementation.

1.1 Motivation

Extending a classic pointer-based mesh data structure to 3D is unwieldy, error-prone, and difficult to debug. We instead take a more abstract set-oriented view in the design of our data structure, by turning to the formal definition of an abstract simplicial complex. This gives our implementation the following desirable properties:

- We treat the mesh as a graph and perform all of our operations combinatorially.
- There is no cumbersome pointer-hopping typical of most mesh data structures.
- The design easily generalizes to arbitrary dimension.
- The final result is very compact and simple to implement.

In effect we are taking advantage of the fact that during assembly of all the necessary structures one can use high level, abstract data structures. That way formal definitions can be turned into code almost verbatim. While these data structures (*e.g.*, sets and maps) may not be the most efficient for *computation*, an approach which uses them during assembly is far less error prone. Once everything has been assembled it can be turned easily into more efficient packed representations (*e.g.*, compressed row storage format sparse matrices) with their more favorable performance during the actual computations which occur, *e.g.*, in physical simulation.

1.2 Outline

We will begin with a few definitions in Section 2, and see how these translate into our tuple-based representation in Section 3. The boundary operator, described in Section 4, facilitates mesh traversal and implements the discrete exterior derivative. We show how

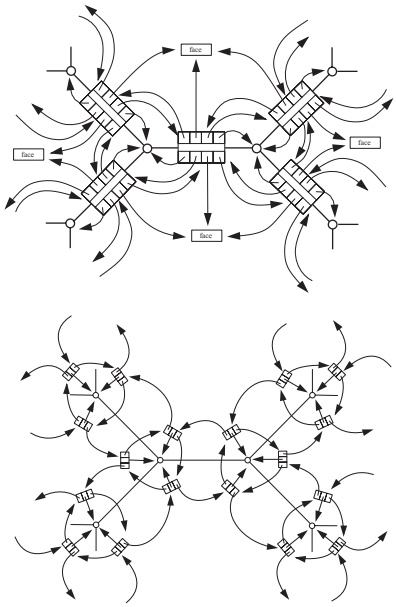


Figure 1: *Some typical examples of 2D mesh representations (from [Joy et al. 2002]; used with permission). Such pointer-based data structures become quite difficult to manage once they are extended to 3D.*

everything is put together in Section 5. Finally, we discuss our implementation of the DEC operators in Section 6.

2 Definitions

We begin by recalling the basic definitions of the objects we are dealing with. The focus here is on the rigorous mathematical definitions in a form which then readily translates into high level algorithms. The underlying concepts are simply what we all know informally as *meshes* in either two (triangle) or three (tet) dimensions.

Simplices A *simplex* is a general term for an element of the mesh, identified by its dimension. 0-simplices are vertices, 1-simplices are edges, 2-simplices are triangles, and 3-simplices are tetrahedra.

Abstract Simplicial Complex This structure encodes all the relationships between vertices, edges, triangles, and tets. Since we are only dealing with combinatorics here the atomic element out of which everything is built are the integers $0 \leq i < n$ referencing the underlying vertices. For now they do not yet have point positions in space. Formally, an abstract simplicial complex is a *set of subsets* of the integers $0 \leq i < n$, such that if a subset is contained in the complex then so are all its subsets. For example, a 3D complex is a collection of tetrahedra (4-tuples), triangles (3-tuples), edges (2-tuples), and vertices (singletons), such that if a tetrahedron is present in the complex then so must be its triangles, edges, and

vertices. All our simplicial complexes will be proper three or two manifolds, possibly with boundary and may be of arbitrary topology (*e.g.*, containing voids and tunnels).

Manifold The DEC operators that we build on are defined only on meshes which represent manifolds. Practically speaking this means that in a 3D simplicial complex all triangles must have two incident tets only (for a boundary triangle there is only one incident tet). Every edge must have a set of tets incident on it which form a single "ring" which is either open (at the boundary) or closed (in the interior). Finally for vertices it must be true that all incident tets form a topological sphere (or hemisphere at the boundary). These properties should be asserted upon reading the input. For example, for triangles which bound tets one must assert that each such triangle occurs in at most two tets. For an edge the "ring" property of incident tets can be checked as follows. Start with one incident tet and jump across a shared triangle to the next tet incident on the edge. If this walk leads back to the original tet *and* all tets incident on the edge can thusly be visited, the edge passes the test. (For boundary edges such a walk starts at one boundary tet and ends at another.) The test for vertices is more complex. Consider all tets incident on the given vertex. Using the tet/tet adjacency across shared triangles one can build the adjacency graph of all such tets. This graph must be a topological sphere (or hemisphere if the vertex is on the boundary).

Since we need everything to be properly oriented we will only allow *orientable* manifolds (*i.e.*, no Möbius strips or Klein bottles).

Regularity To make life easier on ourselves we also require the simplicial complex to be *strongly regular*. This means that simplices must not have identifications on their boundaries. For example, edges are not allowed to begin and end in the same vertex. Similarly, the edges bounding a triangle must not be identified nor do we allow edges or triangles bounding a tet to be identified. In practice this is rarely an issue since the underlying geometry would need to be quite contorted for this to occur. Strictly speaking though such identifications are possible in more general, abstract settings without violating the manifold property.

Embedding It is often useful to distinguish between the *topology* (neighbor relationships) and the *geometry* (point positions) of the mesh. A great deal of the operations performed on our mesh can be carried out using only topological information, *i.e.*, without regard to the embedding. The embedding of the complex is given by a map $p : [0,n) \mapsto (x,y,z) \in \mathbb{R}^3$ on the vertices (which is extended piecewise linearly to the interior of all simplices). For example, when we visualize a mesh as being composed of piecewise linear triangles (for 2D meshes) or piecewise linear tets, we are dealing with the geometry. Most of the algorithms we describe below do not need to make reference to this embedding. When implementing these algorithms it is useful to only think in terms of combinatorics. There is only one stage where we care about the geometry: the computation of metric dependent quantities needed in the definition of the Hodge star.

3 Simplex Representation

Ignoring orientations for a moment, each k-simplex is represented as a $(k+1)$-tuple identifying the vertices that bound the simplex. In this view a tet is simply a 4-tuple of integers, a triangle is a 3-tuple of integers, an edge is a 2-tuple, and a vertex is a singleton. Note that all permutations of a given tuple refer to the same simplex. For example, (i,j,k) and (j,i,k) are different *aliases* for the same triangle. In order to remove ambiguities, we must designate one *representative* alias as the representation of the simplex in our data structures. We do this by using the *sorted* permutation of the tuple. Thus each simplex (tuple) is stored in our data structures as its canonical (sorted) representative. Then if we, for example, need to check whether two simplices are in fact the same we only need to compare their representatives element by element.

All this information is stored in lists we designate **V**, **E**, **F**, and **T**. They contain one representative for every vertex, edge, triangle, and tet, respectively, in the mesh.

3.1 Forms

The objects of computation in an algorithm using DEC are forms. Formally, a differential k-form is a quantity that can be integrated over a k dimensional domain. For example, consider the expression $\int f(x)dx$ (x being a scalar). The integrand $f(x)dx$ is called a 1-*form*, because it can be integrated over any 1-dimensional interval. Similarly, the dA in $\int\int dA$ would be a 2-form.

Discrete differential forms are dealt with by storing the results of the integrals themselves, instead of the integrands. That is, discrete k-forms associate one value with each k-simplex, representing the integral of the form over that simplex. With this representation we can recover the integral over any k-dimensional chain (the union of some number of k-simplices) by summing the value on each simplex (using the linearity of the integral).

Since all we have to do is to associate one value with each simplex, for our purposes forms are simply vectors of real numbers where the size of the vector is determined by the number of simplices of the appropriate dimension. 0-forms are vectors of size $|\mathbf{V}|$, 1-forms are vectors of size $|\mathbf{E}|$, 2-forms are vectors of size $|\mathbf{F}|$, and 3-forms are vectors of size $|\mathbf{T}|$. Such a vector representation requires that we assign an index to each simplex. We use the position of a simplex in its respective list (**V**, **E**, **F**, or **T**) as its index into the form vectors.

3.2 Orientation

Because the vectors of values we store represent integrals of the associated k-form over the underlying simplices, we must keep track of orientation. For example, reversing the bounds of integration on $\int_a^b f(x)dx$ flips the sign of the resulting value. To manage this we need an *intrinsic orientation* for each simplex. It is with respect to this orientation that the values stored in the form vectors receive the appropriate sign. For example, suppose we have a 1-form f with value f_{ij} assigned to edge $e = (i,j)$; that is, the real number f_{ij} is the integral of the 1-form f over the line segment (p_i, p_j). If we query the value of this form on the edge (j,i) we should get $-f_{ij}$.

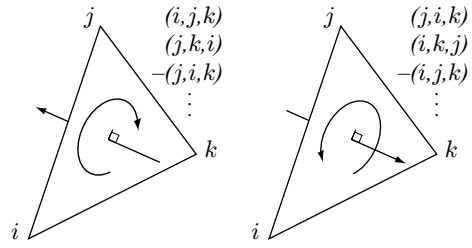


Figure 2: *All permutations of a triple (i,j,k) refer to the same triangle, and the sign of the permutation determines the orientation.*

Hence every tuple must be given a sign indicating whether it agrees $(+)$ or disagrees $(-)$ with the intrinsic orientation of the simplex. Given a set of integers representing a simplex, there are two equivalence classes of orderings of the given tuple: the even and odd permutations of the integers in question. These two equivalence classes correspond to the two possible orientations of the simplex (see Fig. 2).

Note that assigning a sign to any one alias (*i.e.*, the representative) implicitly assigns a sign to all other aliases. Let us assume for a moment that the sign of all representatives is known. Then the sign S of an arbitrary tuple t, with representative r, is

$$\mathbf{S}(t) = \begin{cases} \mathbf{S}(r) & \text{if } t \text{ is in the same equivalence class as } r \\ -\mathbf{S}(r) & \text{if } t \text{ is in the opposite equivalence class.} \end{cases}$$

More formally, let P be the permutation that permutes t into r (*i.e.*, $r = P(t)$). Then

$$\mathbf{S}(t) = \mathbf{S}(P)\mathbf{S}(P(t)).$$

(Here $\mathbf{S}(P)$ denotes the sign of the permutation P with $+1$ for even and -1 for odd permutations.)

All that remains, then, is to choose an intrinsic orientation for each simplex and set the sign of the representative alias accordingly. In general the assignment of orientations is arbitrary, as long as it is consistent. For all subsimplices we choose the representative to be positively oriented, so that the right-hand-side of the above expression reduces to $\mathbf{S}(P)$. For top-level simplices (tets in 3D, triangles in 2D), we use the convention that a positive volume corresponds to a positively oriented simplex. We therefore require a volume form which, together with an assignment of points to vertices, will allow us to orient all tets. Recall that a volume form accepts three (for 3D; two for 2D) vectors and returns either a positive or negative number (assuming the vectors are linearly independent). So the sign of a 4-tuple is:

$$\mathbf{S}(i_0,i_1,i_2,i_3) = \mathbf{S}(\mathrm{Vol}(p_{i_1} - p_{i_0}, p_{i_2} - p_{i_0}, p_{i_3} - p_{i_0})).$$

4 The Boundary Operator

The *faces* of a k-simplex are the $(k-1)$-simplices that are incident on it, *i.e.*, the subset of one lower dimension. Every k-simplex has $k+1$ faces. Each face corresponds to removing one integer from the tuple, and the relative orientation of the face is $(-1)^i$ where i is the index of the integer that was removed. To clarify:

- The faces of a tet $+(t_0,t_1,t_2,t_3)$ are $-(t_0,t_1,t_2)$, $+(t_0,t_1,t_3)$, $-(t_0,t_2,t_3)$, and $+(t_1,t_2,t_3)$.
- The faces of a triangle $+(f_0,f_1,f_2)$ are $+(f_0,f_1)$, $-(f_0,f_2)$, and $+(f_1,f_2)$.
- The faces of an edge $+(e_0,e_1)$ are $-(e_0)$ and $+(e_1)$.

We can now define the boundary operator ∂ which maps simplices to their their faces. Given the set of tets $\mathbf{T}$ we define $\partial^3 : \mathbf{T} \to \mathbf{F}^4$ as

$$\begin{aligned} \partial^3(+(i_0,i_1,i_2,i_3)) \quad = \quad & \{-(i_0,i_1,i_2), +(i_0,i_1,i_3), \\ & -(i_0,i_2,i_3), +(i_1,i_2,i_3)\}. \end{aligned}$$

Similarly for $\partial^2 : \mathbf{F} \to \mathbf{E}^3$ (which maps each triangle to its three edges) and $\partial^1 : \mathbf{E} \to \mathbf{V}^2$ (which maps each edge to its two vertices).

We represent these operators as sparse adjacency matrices (or, equivalently, signed adjacency lists), containing elements of type $+1$ and -1 only. So ∂^3 is implemented as a matrix of size $|\mathbf{F}| \times |\mathbf{T}|$ with 4 non-zero elements per column, ∂^2 an $|\mathbf{E}| \times |\mathbf{F}|$ matrix with 3 non-zero elements per column, and ∂^1 a $|\mathbf{V}| \times |\mathbf{E}|$ matrix with 2 non-zero elements per column (one $+1$ and one -1). The transposes of these matrices are known as the *coboundary* operators, and they map simplices to their *cofaces*—neighbor simplices of one higher dimension. For example, $(\partial^2)^T$ maps an edge to the "pinwheel" of triangles incident on that edge.

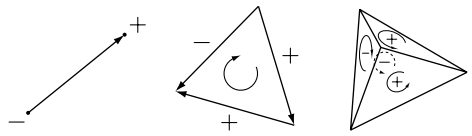


Figure 3: *The boundary operator identifies the faces of a simplex as well as their* relative *orientations. In this illustration, arrows indicate intrinsic orientations and signs indicate the relative orientation of a face to a parent.*

These matrices allow us to iterate over the faces or cofaces of any simplex, by walking down the columns or across the rows, respectively. In order to traverse neighbors that are more than one dimension removed (*i.e.*, the tets adjacent to an edge or the faces adjacent to a vertex) we simply concatenate the appropriate matrices, but without the signs. (If we kept the signs in the matrix multiplication any such consecutive product would simply return the zero matrix reflecting the fact that the boundary of a boundary is always empty.)

5 Construction

Although we still need a few auxiliary wrapper and iterator data structures to provide an interface to the mesh elements, the simplex lists and boundary matrices contain the entirety of the topological data of the mesh. All that remains, then, is to fill in this data.

We read in our mesh as a list of (x,y,z) vertex positions and a list of 4-tuples specifying the tets. Reading the mesh in this format eliminates the possibility of many non-manifold scenarios; for example, there cannot be an isolated edge that does not belong to a tet. We assume that all integers in the range $[0,n)$ appear at least once in the tet list (this eliminates isolated vertices), and no integer outside of this range is present.

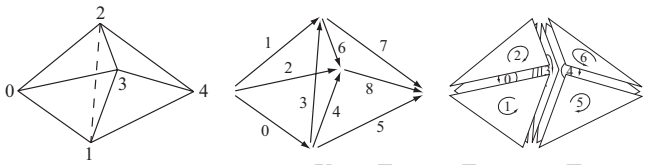
Once $\mathbf{T}$ is read in, building $\mathbf{E}$ and $\mathbf{F}$ is trivial; for each tuple in $\mathbf{T}$, append all subsets of size 2 and 3 to $\mathbf{E}$ and $\mathbf{F}$ respectively. We must be sure to avoid duplicates, either by using a unique associative container, or by sorting the list afterward and removing duplicates. Then the boundary operator matrices are constructed as follows:

```
for each simplex s
    construct a tuple for each face f of s
        as described in Section 4
    determine the index i of f by locating
        its representative
    set the entry of the appropriate matrix
        at row i, column s to S(f)
```

Figure 4 shows a complete example of a mesh and its associated data structure.

6 DEC Operators

Now we discuss the implementation of the two most commonly used DEC operators: the exterior derivative and the Hodge star. As we will see, in the end these also amount to nothing more than sparse matrices that can be applied to our form vectors.



$$\mathbf{V} = \begin{bmatrix}[0]\\[1]\\[2]\\[3]\\[4]\end{bmatrix} \quad \mathbf{E} = \begin{bmatrix}[0,1]\\[0,2]\\[0,3]\\[1,2]\\[1,3]\\[1,4]\\[2,3]\\[2,4]\\[3,4]\end{bmatrix} \quad \mathbf{F} = \begin{bmatrix}[0,1,2]\\[0,1,3]\\[0,2,3]\\[1,2,3]\\[1,2,4]\\[1,3,4]\\[2,3,4]\end{bmatrix} \quad \mathbf{T} = \begin{bmatrix}[0,1,2,3]\\[1,2,3,4]\end{bmatrix}$$

$$\partial^3 = \begin{pmatrix} -1 & \\ 1 & \\ -1 & \\ 1 & -1 \\ & 1 \\ & -1 \\ & 1 \end{pmatrix}$$

$$\partial^2 = \begin{pmatrix} 1 & 1 & & & & & & \\ -1 & & 1 & & & & & \\ & -1 & -1 & & & & & \\ 1 & & & 1 & 1 & & & \\ & 1 & & -1 & & 1 & & \\ & & & & -1 & -1 & & \\ & & & 1 & 1 & & 1 & \\ & & & & & 1 & -1 & \\ & & & & & & 1 & 1 \end{pmatrix}$$

$$\partial^1 = \begin{pmatrix} -1 & -1 & -1 & & & & & & \\ 1 & & & -1 & -1 & -1 & & & \\ & 1 & & 1 & & & -1 & -1 & \\ & & 1 & & 1 & & 1 & & -1 \\ & & & & & 1 & & 1 & 1 \end{pmatrix}$$

Figure 4: *A simple mesh and all associated data structures.*

6.1 Exterior Derivative

As we have seen earlier in the course, the discrete exterior derivative is defined using Stokes' theorem such that

$$\int_\sigma d\omega = \int_{\partial\sigma} \omega$$

where ω is a k-form, and σ is a $(k+1)$-simplex. In words, this equation states that the evaluation of $d\omega$ on a simplex is equal to the evaluation of ω on the boundary of that simplex.

Let us try to understand this theorem with a few examples. Consider a 0-form f, *i.e.*, a function giving values at vertices. With that, df is a 1-form which can be integrated along an edge (say with endpoints denoted a and b) and Stokes' theorem states the well known fact

$$\int_{[a,b]} df = f(b) - f(a).$$

The right hand side is simply the evaluation of the 0-form f on the boundary of the edge (*i.e.*, its endpoints), with appropriate signs indicating the orientation of the edge.

What about triangles? If f is a 1-form (one value per edge), then df is a 2-form that can be evaluated on a triangle abc as

$$\begin{aligned} \int_{\Delta abc} df &= \int_{\partial(\Delta abc)} f \\ &= \int_{[a,b]} f + \int_{[b,c]} f + \int_{[c,a]} f \\ &= f_{ab} + f_{bc} + f_{ca} \end{aligned}$$

using the subscript notation from Section 3.2. Again, the right hand side is simply the evaluation of the 1-form f on the boundary of the triangle—its three edges.

We can restate the general form of the theorem for our discrete forms as

$$d\omega_\sigma = \sum_{s\in\partial\sigma} \omega_s$$

Written this way, it is easy to see that this can be implemented as the multiplication of a form vector by the coboundary matrix ∂^T.

6.2 The Dual Mesh and the Hodge Star

Every complex has a dual. The dual of a simplicial complex is a *cell complex* where primal k-simplices correspond to dual $(n-k)$-cells. So in our case there are $|\mathbf{V}|$ dual polyhedra, $|\mathbf{E}|$ dual polygons, $|\mathbf{F}|$ dual edges, and $|\mathbf{T}|$ dual vertices, corresponding to primal vertices, edges, triangles, and tetrahedra, respectively (see Fig. 5). Note that, since every dual cell is co-located with a primal simplex and the cardinality is the same, in the code there is no explicit representation for the dual mesh. Where appropriate, dual cells are queried through the corresponding primal simplex index.

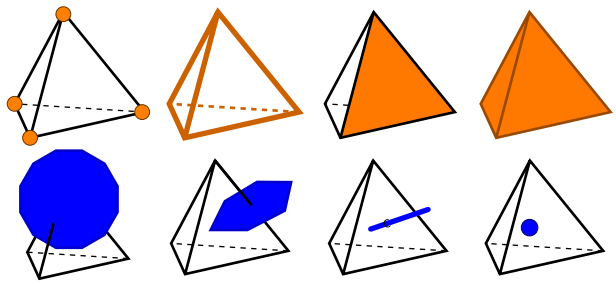


Figure 5: *There is one dual polyhedron for every primal vertex, one dual polygon for every primal edge, one dual edge for every primal triangle, and one dual vertex for every primal tetrahedron.*

The operator that transforms a primal k-form into a dual $(n-k)$-form is known as the *Hodge star*. There are many different kinds of Hodge stars, the simplest of which is the *diagonal Hodge star*.

We again attempt to motivate the definition with some intuition. When transferring a quantity from a primal simplex to a dual cell, the quantities must "agree" somehow. Since these are integral values, simply setting the value on the dual to be equal to the value on the primal does not make sense, as the domain of integration is unrelated. Instead, we require that the *integral density* be equal. So, if ω denotes the evaluation of a form on a primal k-simplex σ, then $\star\omega$ is the value on the dual $(n-k)$-cell $\tilde{\sigma}$ such that

$$\frac{\omega}{\mathrm{Vol}(\sigma)} = \frac{\star\omega}{\mathrm{Vol}(\tilde{\sigma})}$$

allowing us to define $\star$ as

$$\star = \frac{\mathrm{Vol}(dual)}{\mathrm{Vol}(primal)}.$$

In effect the diagonal Hodge star requires that the averages of the integrand over the respective domains agree.

This is represented as a diagonal matrix so that, again, application of the operator becomes a simple matrix-vector multiplication. Note that when transforming quantities from the dual to the primal, the inverse of this matrix is used. Since the matrix is diagonal we only store the diagonal entries. There are as many of these as there are simplices of the appropriate dimension. Consequently the diagonal Hodge star can be represented with vectors of length $|\mathbf{V}|$, $|\mathbf{E}|$, $|\mathbf{F}|$, and $|\mathbf{T}|$ respectively.

6.2.1 Calculating Dual Volumes

So far the entire implementation has been in terms of the combinatorics of the mesh, but when constructing the Hodge star we must finally introduce the geometry. After all, the purpose of the Hodge star is to capture the metric. The volumes of the primal simplices are straightforward: 1 for vertices, length for edges, area for triangles, and volume for tetrahedra. The dual volumes are similarly defined, but in order to avoid constructing the graph of the dual mesh explicitly, we calculate the dual volumes as follows.

If we use the circumcentric realization of the dual mesh (*i.e.*, dual vertices are at the circumcenters of the associated tets), we can exploit the following facts when calculating the dual volumes.[1]

- A dual edge (dual of a primal triangle t) is linear, is normal to t, and is collinear with the circumcenter of t (though the line segment need not necessarily pass through t).
- A dual polygon (dual of a primal edge e) is planar, is orthogonal to e, and is coplanar with the center of e (though it need not intersect e).
- A dual cell (dual of a primal vertex v) is the convex intersection of the half-spaces defined by the perpendicular bisectors of the edges incident on v.

Just as with primal vertices, the volume of a dual vertex is defined to be 1. For the others, we can conceptually decompose each cell into pieces bounded by lower dimensional cells, and sum the volumes of the pieces. For example, a dual polyhedron can be seen as the union of some number of pyramids, where the base of each pyramid is a dual polygon and the apex is the primal vertex. Similarly, a dual polygon can be seen as a union of triangles with dual edges at the bases, and dual edges can be seen as a union of (two) line segments with dual vertices at the bases. The following pseudocode illustrates how the volumes are calculated.

```
vec3 C( Simplex s );  // gives the circumcenter of s

// Initialize all dual volumes to 0.

// Dual edges
for each primal triangle f
    for each primal tet t_f incident on f
        b ← t_f.dualVolume  // 1
        h ← ||C(f) − C(t_f)||
        f.dualVolume ← f.dualVolume + (1/1) bh

// Dual polygons
for each primal edge e
    for each primal triangle f_e incident on e
        b ← f_e.dualVolume
        h ← ||C(e) − C(f_e)||
        e.dualVolume ← e.dualVolume + (1/2) bh

// Dual polyhedra
for each primal vertex v
    for each primal edge e_v incident on v
        b ← e_v.dualVolume
        h ← ||C(v) − C(e_v)||
        v.dualVolume ← v.dualVolume + (1/3) bh
```

Note that, even when dealing with the geometry of the mesh, this part of the implementation still generalizes trivially to arbitrary dimension.

[1] Circumcentric duals may only be used if the mesh satisfies the Delaunay criterion. If it does not, a barycentric dual mesh may be used. However, care must be taken if a barycentric dual mesh is used, as dual edges are no longer straight lines (they are piecewise linear), dual faces are no longer planar, and dual cells are no longer necessarily convex.

7 Summary

All the machinery discussed above can be summarized as follows:

- k-forms as well as the Hodge star are represented as vectors of length $|\mathbf{V}|$, $|\mathbf{E}|$, $|\mathbf{F}|$, and $|\mathbf{T}|$;
- the discrete exterior derivative is represented as (transposes of) sparse adjacency matrices containing only entries of the form $+1$ and -1 (and many zeros); the adjacency matrices are of dimension $|\mathbf{V}| \times |\mathbf{E}|$ (boundary of edges), $|\mathbf{E}| \times |\mathbf{F}|$ (boundary of triangles), and $|\mathbf{F}| \times |\mathbf{T}|$ (boundary of tets).

In computations these matrices then play the role of operators such as grad, curl, and div and can be composed to construct operators such as the Laplacian (and many others).

While the initial setup of these matrices is best accomplished with associative containers, their final form can be realized with standard sparse matrix representations. Examples include a compressed row storage format, a vector of linked lists (one linked list for each row), or a two dimensional linked list (in effect, storing the matrix and its transpose simultaneously) allowing fast traversal of either rows or columns. The associative containers store integer tuples together with orientation signs. For these we suggest the use of sorted integer tuples (the canonical representatives of each simplex). Appropriate comparison operators needed by the container data structures simply perform lexicographic comparisons.

And that's all there is to it!

Acknowledgments This work was supported in part through a James Irvine Fellowship to the first author, NSF (DMS-0220905, DMS-0138458, ACI-0219979), DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Alias, and Pixar.

References

JOY, K. I., LEGAKIS, J., AND MACCRACKEN, R. 2002. *Data Structures for Multiresolution Representation of Unstructured Meshes*. Springer-Verlag, Heidelberg, Germany.

Chapter 9:
Discrete, Vorticity-Preserving, and Stable Simplicial Fluids

Sharif Elcott Yiying Tong Eva Kanso Peter Schröder Mathieu Desbrun
Caltech

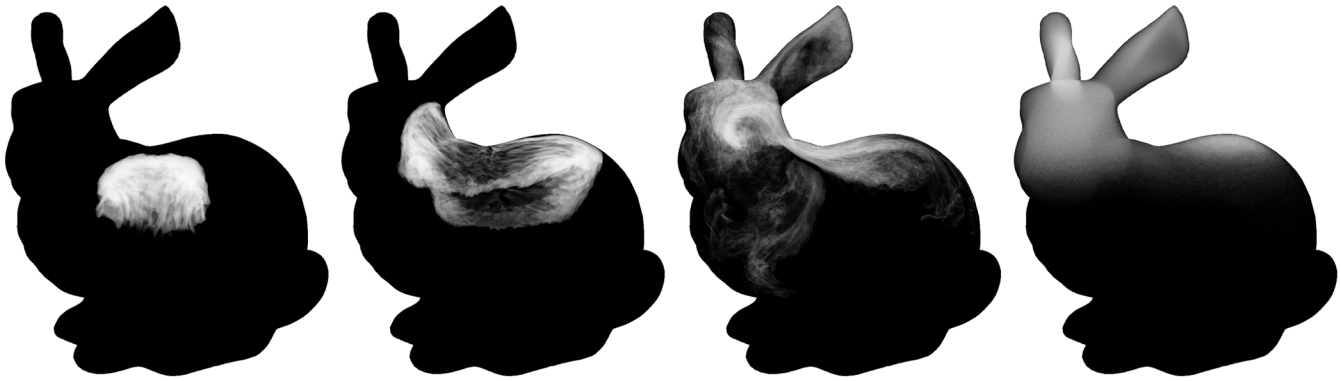


Figure 1: *Discrete Fluids: we present a novel integration scheme for fluid simulation applicable to tetrahedral meshes of arbitrary domains. Aside from resolving the exact boundaries, our approach also provides an accurate treatment of the vorticity through a discrete preservation of Kelvin's circulation theorem. Here, a hot smoke cloud rises inside a bunny shaped domain of* $32K$ *tets, significantly reducing the computational complexity of the simulation for such an intricate boundary compared to regular grid-based techniques (less than* $2s$ *per frame on Pentium IV 3GHz).*

Abstract

Visual accuracy, low computational cost, and numerical stability are foremost goals in computer animation. An important ingredient in achieving these goals is the conservation of fundamental motion invariants. For example, rigid or deformable body simulation have benefited greatly from conservation of linear and angular momenta. In the case of fluids, however, none of the current techniques focuses on conserving invariants, and consequently, they often introduce a visually disturbing numerical diffusion of *vorticity*. Visually just as important is the resolution of complex simulation domains. Doing so with regular (even if adaptive) grid techniques can be computationally delicate.

In this chapter, we propose a novel technique for the simulation of fluid flows. It is designed to respect the defining differential properties, i.e., the *conservation of circulation* along arbitrary loops as they are transported by the flow. Consequently, our method offers several new and desirable properties: (1) arbitrary simplicial meshes (triangles in 2D, tetrahedra in 3D) can be used to define the fluid domain; (2) the computations are efficient due to discrete operators with small support; (3) the method is stable for arbitrarily large time steps; and (4) it preserves a *discrete circulation* avoiding numerical diffusion of vorticity. The underlying ideas are easy to incorporate in current approaches to fluid simulation and should thus prove valuable in many applications.
Keywords: Fluid Dynamics, Discrete Exterior Calculus, Computational Algorithms, Circulation Preservation

1 Introduction

It is now taken for granted that properties such as conservation of linear and angular momentum in solid mechanics simulations are a key ingredient in both numerical stability and the realism of the resulting animations. Much of the progress in this direction has been enabled by a deeper understanding of the underlying geometric structures and how they can be preserved as we go from continuous models to discrete computational realizations. So far, advances of this type have not yet deeply impacted fluid flow simulations. Current methods in fluid simulation are rarely designed to conserve *defining physical properties*. Consider, for example, the need in many methods to continually project the numerically updated velocity field onto the set of divergence free velocity fields.

1.1 Previous Work

Fluid Mechanics has been studied extensively in the scientific community both mathematically and computationally. The physical behavior of incompressible fluids is usually modeled by Navier Stokes (NS) equations for viscous fluids and by Euler equations for inviscid (non-viscous) fluids. Numerical approaches in computational fluid dynamics typically discretize the governing equations through Finite Volumes (FV), Finite Elements (FE) or Finite Differences (FD) methods. We will not attempt to review the many methods proposed (an excellent survey can be found in [Langtangen et al. 2002]) and instead focus on approaches used for fluids in computer graphics. Some of the first fluid simulation techniques used in the movie industry were based on Vortex Blobs [Yaeger et al. 1986] and Finite Differences [Foster and Metaxas 1997]. To circumvent the ill-conditioning of these iterative approaches for large time steps and achieve unconditional stability, Jos Stam [1999; 2001] pioneered in graphics the use of the *method of characteristics* for fluid advection, and of the Helmholtz-Hodge decomposition to preserve the divergence-free nature of the fluid motion [Chorin and Marsden 1979]. This extremely successful semi-Lagragian approach based on an Eulerian discretization through a regular space partitioning has led to a series of refinement over the past years. We mention the use of Galilean invariance [Shah et al. 2004], staggered grids, and monotonic cubic interpolation [Fedkiw et al. 2001], which have all significantly contributed to visual impact of fluid animations. In the wake of this success, improvements were made on the handling of the interfaces with air [Foster and Fedkiw 2001], extensions to curved surfaces [Stam 2003; Tong et al. 2003; Shi and Yu 2004] and visco-elastic objects [Goktekin et al. 2004], and goal oriented control of the fluid motion [Treuille et al. 2003; McNamara et al. 2004; Pighin et al. 2004].

However, the Stable Fluids technique is not without drawbacks. First, complex domain boundaries are difficult to handle with regular grids due to scaling issues. This can be addressed through local adaptivity of the domain [Losasso et al. 2004], but the associated

octree structures require significant overhead and lead to possible loss of accuracy. Second, regular as well as octree partitionings of space suffer from preferred direction sampling, leading to artifacts similar to aliasing in rendering. Lastly, due to numerical dissipation, the current methods do not preserve fundamental invariants aside from the divergence-free nature of the flow. While exaggerated loss of total energy is often difficult to notice, excessive diffusion of vorticity affects the motion significantly. The presence of vortices in liquids and volutes in smoke is one of the most important visual clues to our perception of fluidity. Vorticity confinement [Steinhoff and Underhill 1994; Fedkiw et al. 2001] counteracts this diffusion by locally reinjecting vorticity. Unfortunately, it is hard to control how much can safely be added back into the flow without affecting stability.

Our main argument in this chapter is that a careful setup of *discrete differential quantities* together with their structural relationship (*e.g.*, well known vector calculus identities) is a necessary first step in building discrete simulation methods for fluids which can respect some of the underlying physics and in this way overcome limitations of earlier approaches. As it turns out this setup also provides guidance in designing time integration methods with attractive features such as stability and efficiency. The key ingredient to this approach is a return to the *geometric* foundations of physics.

1.2 Towards a Geometric Approach to Simulation

In recent years, there has been a renewed emphasis on the geometric structure of physical systems as a key feature for developing reliable and efficient numerical methods that better respect the underlying physics. Computational Electro-Magnetism (E&M) and Discrete Variational Mechanics, for instance, have independently demonstrated that geometric understanding of the continuous model and proper geometric discretization are crucial for obtaining stable numerical results that conserve charge, momentum, and even energy (see, for example, [Bossavit 1998; Marsden and West 2001; Kane et al. 2000; Lew et al. 2003; Fetecau et al. 2003]).

The geometric structure of Fluid Mechanics, specifically Euler's equations for inviscid fluids, has been investigated from a theoretical point of view (see [Marsden and Wenstein 1983] and references therein). In this geometric framework, vorticity plays a central role since Euler's equations can be written directly as a simple vorticity advection (see Section 2 for details). Inspired by this geometric viewpoint and the recent advances in Discrete Exterior Calculus (DEC—see [Bossavit 1998; Hirani 2003] , and Chapter 7), we propose to mimic these geometric properties on the discrete level through a *discrete differential approach to fluid mechanics*.

1.3 Contributions and Outline

In this chapter, we present a radically different approach to fluid simulation based on a tailored discretization of the geometric structure of the fluid equations. We depart from the concepts of most previous computational approaches by locating physical quantities on vertices, edges, faces, or cells, depending on their geometric nature. Through a proper discrete calculus on simplicial complexes, our novel integration scheme directly manipulates intrinsically divergence-free variables, alleviating the need for a numerically-detrimental Hodge projection. Our technique offers control over the structural invariants in fluid flows thanks to our structure-preserving space and physical discretization. Finally, our novel technique fits the specific requirements of the CG community that are simplicity and unconditional stability, with high visual quality even for very large time steps.

The organization of this chapter is as follows. In Section 2, we motivate our approach through a brief overview of the theory and computational algorithms for Fluid Mechanics. We propose a novel discrete fluid theory in Section 3 and we discuss the associated circulation-preserving integration algorithm in Section 4. Several numerical examples are shown and discussed in Section 5.

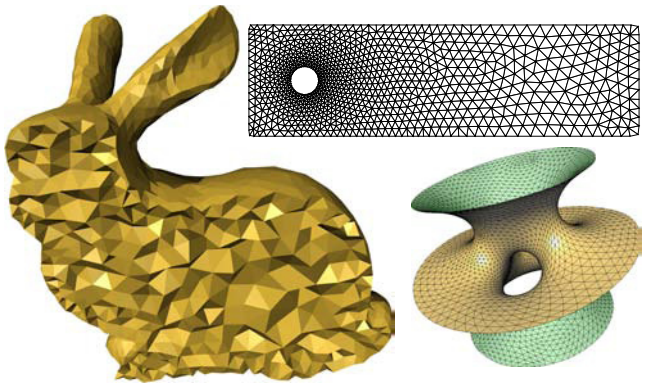


Figure 2: *Domain Mesh: our fluid simulator uses a simplicial mesh to discretize the equations of motion; (left) the domain mesh (shown as a cutaway view) used in Fig. 1; (up) a coarser version of the flat 2D mesh used in Fig. 8; (right) the curved triangle mesh used in Fig. 10.*

2 Background on Fluid Mechanics

2.1 Theory and Geometry of Euler Equations

Consider an ideal (inviscid, incompressible and homogeneous) fluid flow on a domain $\mathscr{D}$ in two- or three-dimensional physical space. The Euler equations, governing the motion of this fluid (with no external forces for now), can be written as:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p\,, \\ div(\mathbf{u}) &= 0\,, \quad \mathbf{u} \parallel \partial\mathscr{D}\,. \end{aligned} \tag{1}$$

Here, we have set the density of the fluid $\rho = 1$ and used $\mathbf{u}$ to denote the fluid velocity, p the pressure, and $\partial\mathscr{D}$ the boundary of the fluid region $\mathscr{D}$. The pressure term in Eq. (1) can be easily dropped by rewriting the Euler equations in terms of *vorticity*. Recall first that, in traditional vector calculus notation, the vorticity $\boldsymbol{\omega}$ is defined as the curl of the velocity field; then, by taking the curl ($\nabla\times$) of Eq.(1), we obtain:

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} + \mathscr{L}_{\mathbf{u}}\boldsymbol{\omega} &= \mathbf{0}\,, \\ \boldsymbol{\omega} = \nabla\times\mathbf{u}\,, \quad div(\mathbf{u}) &= 0\,, \quad \mathbf{u} \parallel \partial\mathscr{D}\,. \end{aligned} \tag{2}$$

where $\mathscr{L}$ represents the Lie derivative. To put it simply, this last expression states that *vorticity is advected along the fluid flow*. Roughly speaking, vorticity measures the local rotation of a fluid parcel. We say the fluid parcel has vorticity when it spins as it moves along its path. Therefore, vorticity advection means that the local spin moves dynamically as if pushed forward by the flow.

Now, since the integral of the vorticity on a given bounded domain is equal, by Stokes' theorem, to the *circulation* around the loop enclosing the domain, one can explain the geometric nature of an ideal fluid flow in particularly simple terms: **the circulation around any closed loop $\mathscr{C}$ is conserved throughout the motion of this loop in the fluid**. This key result is known as Kelvin's circulation theorem, and is usually written as:

$$\Gamma(t) = \oint_{\mathscr{C}(t)} \mathbf{u}\cdot\mathbf{dl} = \text{constant}\,, \tag{3}$$

where $\Gamma(t)$ is the circulation of the velocity on the loop $\mathscr{C}$ at time t as it gets advected in the fluid.

Additionally, one can readily verify that Euler equations (1), equivalently (2), also preserve the total energy of the fluid which can be

written as:

$$E = \frac{1}{2}\int_{\mathscr{D}} \| \mathbf{u} \|^2 , \quad \text{or, equivalently,} \quad \mathrm{E} = \frac{1}{2}\int_{\mathscr{D}} \boldsymbol{\omega}\cdot\Delta^{-1}\boldsymbol{\omega} . \quad (4)$$

2.2 Navier-Stokes Equations

In contrast to ideal fluids, incompressible *viscous* fluids generate very different fluid behaviors. However, they can be modelled by the Navier-Stokes equations which look very similar to Euler equations:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p + \nu\Delta\mathbf{u} , \\ div(\mathbf{u}) &= 0 , \quad \mathbf{u}|_{\partial\mathscr{D}} = \mathbf{0} . \end{aligned} \quad (5)$$

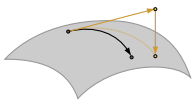
where Δ represents the Laplacian operator, and ν is a parameter called the *kinematic viscosity*. Note that various types of boundary conditions are sometimes added, depending on the chosen model. Despite the apparent similarity between these two models for fluid flows, it is important to notice that the added diffusion term dampens the motion, resulting in a slow decay of both circulation *and* total energy. This diffusion also implies that the velocity of a viscous fluid at the boundary of a domain must be null, whereas an inviscid fluid could have a non-zero tangential component on the boundary. Here again, one can avoid the pressure term by taking the curl of the equations, finally yielding :

$$\begin{aligned} \frac{\partial \boldsymbol{\omega}}{\partial t} + \mathscr{L}_{\mathbf{u}}\boldsymbol{\omega} &= \nu\Delta\boldsymbol{\omega} , \\ \boldsymbol{\omega} = \nabla\times\mathbf{u} , \quad div(\mathbf{u}) &= 0 , \quad \mathbf{u}|_{\partial\mathscr{D}} = \mathbf{0} . \end{aligned} \quad (6)$$

2.3 Stable Fluids Discretization

The different variants of the original Stable Fluids algorithm [Stam 1999] are all based on a class of discretization approaches known in Computational Fluid Dynamics as fractional step methods. In order to numerically solve the Euler equations over a time step h, they proceed in two stages. They first update the velocity field assuming the fluid is inviscid and disregard the divergence-free constraint of Eq. (1). Then, the resulting velocity is projected onto the closest divergence-free flow (in the $\mathscr{L}^2$ sense) through a Helmholtz-Hodge decomposition.

Although each step of this approach is unconditionally stable, one of the consequences of this fractional integration is the exaggerated energy loss it creates: advecting velocity before reprojecting onto a divergence-free field creates major energy loss and, more importantly in a CG context, diffusion of vorticity as reported in [Fedkiw et al. 2001] for instance. One can understand this numerical flaw through the following geometric argument: physically speaking, the solution of Euler equations are *geodesic* (i.e., shortest) paths *on* the manifold of all possible divergence-free flows; advecting the fluid *out* of the manifold is not a proper substitute to this intrinsic constrained minimization, even the post re-projection is, in itself, exact.



2.4 Our Geometric Approach

Given the difficulties discussed above a natural question is whether these problems can be overcome by designing more careful discretizations which are better suited to maintain the underlying geometric structures—for example, flows that are always divergence free without the need to continually project onto the space of divergence free fields and incurring the associated losses. Or, perhaps even more importantly for visual simulation, one may wonder if it is possible to find discretizations that conserve circulation.

It is known that it is not possible to exactly preserve momenta *and* total energy simultaneously in the discrete setting [Zhong and Marsden 1988]. However, stable numerical techniques have been reported to exactly preserve momenta while keeping the total energy remarkably close to constant [Marsden and West 2001]. Such properties are obtained through the use of *variational integrators*, *i.e.*, by guaranteeing a discrete version of the least-action principle. Their design proceeds by keeping underlying geometric structures intact as one goes from the continuous to the discrete formulation. More precisely, appropriate geometric discretization of the physics allows one to construct discrete analogs of momenta and energy. Equipped with these discrete structure-preserving quantities, integration schemes can then be designed to enforce their invariant nature. We will loosely follow this path by using vorticity as our primary simulation variables (see Eq. (2)) and designing a time integration scheme which will conserve circulation (Eq. (3)) through vorticity advection. As a by-product our velocity fields will be divergence free *without* any need to continually reproject to keep this property. For comparison, and to the best of our knowledge, none of the integration schemes proposed in CFD have been designed to satisfy the conservation properties of the underlying equations (aside from the limited case of linearized NS equations [Morton and Roe 2001]).

We will limit ourselves to the investigation of such a scheme without focusing on the separate issue of order of accuracy. Coming up with an integration scheme that is of higher-order accuracy will be the object of further research.

3 Spatial and Physical Discretization

In this section, we define proper discrete analogs for the velocity and vorticity fields $\mathbf{u}$ and $\boldsymbol{\omega}$ on simplicial grids. We emphasize that the construction of these discrete fields is quite general as it does *not* depend on the assumption of an ideal fluid.

3.1 Space Discretization

We discretize the spatial domain (in which the flow takes place) using a locally oriented simplicial complex, *i.e.*, either a tet mesh for 3D domains or a triangle mesh for 2D domains, and refer to this discrete domain as $\mathscr{M}$ (see Figure 2). The domain may have non-trivial topology, *e.g.*, it can contain tunnels and voids (3D) or holes (2D), but is assumed to be compact. To ensure good numerical properties in the subsequent simulation we require the simplices of $\mathscr{M}$ to be well shaped, *i.e.*, the aspect ratios of tets (resp., triangles) are not near zero. This assumption is quite common since many numerical error estimates depend heavily on the element quality. Collectively we refer to the sets of vertices, edges, triangles, and tets as V, E, F, and T.

We will also need the concept of a *dual mesh*. It associates with each original simplex (vertex, edge, triangle, tet, respectively) its dual (dual cell, dual edge, dual face, and dual vertex, respectively) (see Fig. 3). The geometric realization of this dual mesh is defined as follows: we take circumcenters of tets as the dual vertices and the Voronoi cells as the dual cells; dual edges are then line segments connecting dual vertices of neighboring tets and dual faces are the faces of the Voronoi cells.

3.2 Physical Quantity Discretization

In order to faithfully capture the geometric structure of fluid mechanics on the discrete mesh, we need to define the usual physical quantities such as velocity and vorticity, for example, through *integral values over the simplices of the mesh* $\mathscr{M}$. This is the sharpest departure from traditional numerical techniques in CFD: we not only use values at nodes and tets (as in FEM and FVM), but also allow association (and storage) of field values at *any* appropriate

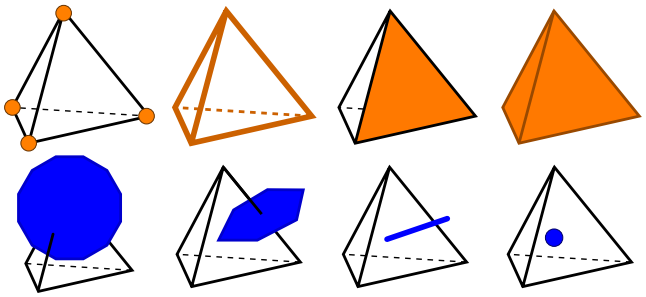


Figure 3: *Primal and Dual Cells: the simplices of our mesh are vertices, edges, triangles and tets (up); their circumcentric duals are dual cells, dual faces, dual edges and dual vertices (bottom).*

simplex. In particular, some quantities will live on edges (primal or dual), others on faces. Before our formal exposition of how these quantities are defined, we motivate our discretization choice with some physical intuition first.

Velocity as a Discrete Flux We wish to define a discrete quantity that encodes the fluid velocity field while being intrinsic to the mesh, i.e., with a coordinate-free representation. To do this, we consider the *flux* of the fluid, *i.e.*, the mass of fluid transported across a given surface per unit time. Note that the flux across a surface incorporates the area of the surface, indicating that it is an integrated quantity rather than a pointwise quantity. On the discrete mesh, a natural place to store the flux is on the triangles of a tet mesh (or edges in a 2D triangle mesh). This discrete flux is coordinate free, *i.e.*, it does not depend on whatever local or global coordinate frame we choose (vectors on the other hand have different representations depending on the coordinate system).

One can equivalently think of the flux as living on dual edges; the proper term should, however, be *circulation* in that case: recall that a dual edge connects the dual vertices associated with the two incident tets and is thus (in a sense) "transverse" to a shared primal face. This point of view is reminiscent of the staggered grid method used in [Fedkiw et al. 2001] and other non-collocated grid techniques (see [Goktekin et al. 2004]). In the staggered grid approach one does not store the x,y,z components of a vector at nodes but rather associates them with the corresponding grid faces. We may therefore think of the idea of storing fluxes on the triangles of our tet mesh as a way of *extending* the idea of staggered grids to the more general simplical mesh setting. This was previously exploited in [Bossavit and Kettunen 1999] in the context of E&M computations. It also makes the usual no-transfer boundary conditions easy to encode: boundary faces experience no flux across them. Encoding this boundary condition when storing velocity vectors at vertices is far more cumbersome.

Divergence as Net Flux on Tets Given the incompressibility of the fluid, the velocity field must be divergence-free ($\nabla \cdot \mathbf{u} = 0$), hence the integral of $\nabla \cdot \mathbf{u}$ is constant. We would like to write this condition in the discrete setting. Given the flux across all the faces of a tet, the integral of the divergence over the tet, or, said differently, the net flux of the tet, becomes particularly simple. According to the generalized Stokes' theorem this integral equals the sum of the integral of the flux on all four faces. Divergence is thus naturally thought of as a value at each tet (see Fig. 4). Physically speaking, the notion of a divergence-free velocity field is equivalent to saying that, at each tet, everything that gets in must get out.

Vorticity as Flux Spin Finally we need to define vorticity on the mesh. To see the physical intuition behind our definition, consider an edge in the mesh. It has a number of faces incident on it, akin to a paddle wheel (see Figure 4). The flux on each face contributes a net torque to the edge. The sum of all these, when going around an edge, is the net torque that would "spin" the edge. We

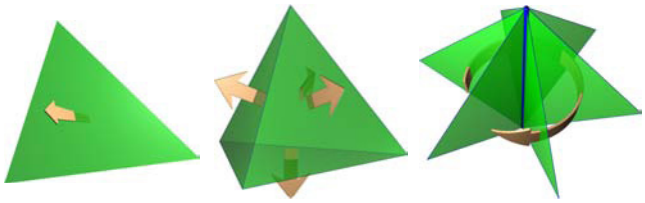


Figure 4: *Discrete Physical Quantities: in our discrete geometric discretization of fluid mechanics, fluid flux lives on faces (left), divergence lives on tets (middle), and vorticity lives on edges (right).*

can thus give a physical definition of vorticity as the sum of fluxes on all faces incident to a given edge: this quantity is now associated with primal edges—or, equivalently, dual faces.

3.3 Discrete Differential Structure

The definition of these intuitive physical quantities living at different simplices on the mesh can be made precise through the definition of a *discrete differential structure*. In this framework, a mesh is seen as the only given structure to work with, with no reference to the continuous space that it approximates, and Discrete Exterior Calculus (DEC) defines a coherent calculus on the mesh using only discrete combinatorial and geometric operations [Munkres 1984; Hirani 2003; Tong 2004][1]. Although we can not discuss at length such a vast mathematical machinery, we briefly cover the fundamental aspects and the discrete differential operators we need to link flux and vorticity (just as in the differential case). For a comprehensive exposition, we refer the interested readers to Chapter 7 on discrete differential forms.

Discrete Forms As Integrals Before we discuss the discrete exterior structure inherent to the mesh, we briefly review exterior forms in the continuous setting (for a more comprehensive discussion, see, for example, [Abraham et al. 1988]). To this end, recall that, given a three-dimensional space, a 0-form is simply a function on that space; a 1-form ω is a proxy to a vector field; a 2-form, or area-form, is to be *integrated over a surface*, that is, it can be viewed as a proxy to the vector perpendicular to that surface; and, a 3-form, or volume-form, is *to be integrated over a volume* and is viewed as a function.

A *discrete* differential k-form, $k = 0,1,2$, or 3, is then the evaluation (i.e., the integral) of the differential k-form on all k-cells, or k-simplices. In practice, discrete k-forms can simply be considered as *vectors of numbers* according to the simplices they live on: 0-forms, live on vertices, and are expressed as a vector of length $|V|$; and correspondingly, 1-forms live on edges (length $|E|$), 2-forms live on faces (length $|F|$), and 3-forms live on tets (length $|T|$). Dual forms are treated similarly. In the examples above, flux is thus a primal 2-form (integrated over faces), vorticity a dual 2-form (integrated over dual faces), and divergence a primal 3-form (integrated over tets).

Discrete Differential Calculus on Simplicial Meshes These *discrete* differential forms can now be used to build the tools of calculus. The mesh has a natural structure, called the DeRham complex, which offers discrete operators on discrete forms, mimicking the continuous setting. At the core of its construction is the definition of the discrete d_k operators (analog to the continuous exterior derivative).

Discrete Exterior Derivatives A key ingredient to define this discrete derivative is Stokes' theorem on a k-form:

$$\int_{\sigma_{k+1}} d_k \omega_k = \int_{\partial_{k+1}\sigma_{k+1}} \omega_k,$$

[1] Although we use many notions from DEC in this chapter, the theory presented here is self-contained and does *not* assume previous knowledge of this machinery.

where σ_k denotes a k-cell while ω_k is a k-form. Stokes' theorem states that the integral of $d_k\omega_k$ (a $(k+1)$-form) over a $(k+1)$-cell equals the integral of the k-form ω_k over the *boundary* of the $(k+1)$-cell. The boundary of a $(k+1)$-cell of course consists of k-cells, making everything well defined. Stokes' theorem can thus be used as a way to *define* the d operator in terms of the boundary operator ∂. Or, said differently, once we have the boundary operator, the operator d follows immediately if we wish Stokes' theorem to hold on the simplicial complex.

To use a very simple example, consider a 0-form ω_0, *i.e.*, a function giving values at vertices. With that $d_0 f_0$ is a 1-form which can be integrated along an edge (say with end points denoted a and b) and Stokes' theorem states the well known fact

$$\int_{[a,b]} d_0 f_0 = f_0(b) - f_0(a).$$

The right hand side is simply the evaluation of the 0-form f_0 on the boundary of the edge, *i.e.*, its endpoints (with appropriate signs indicating the orientation of the edge). Actually, one can define a hierarchy of these operators that mimic the operators given in the continuous setting by the gradient (∇), curl ($\nabla\times$), and divergence ($\nabla\cdot$), namely,

- d_0: maps 0-forms to 1-forms and corresponds to the **Gradient**;
- d_1: maps 1-forms (values on edges) to 2-forms (values on faces). The value on a given face is simply the sum (by linearity of the integral) of the 1-form values on the boundary (edges) of the face with the signs chosen according to the local orientation. d_1 corresponds to the **Curl**;
- d_2: maps 2-forms to 3-forms and corresponds to the **Divergence**.

From this basic setup, we see that all that is required now is to define the boundary operator. This is done using *incidence matrices*, which then act on the vectors of our discrete k-forms. For example d_0 follows as the incidence matrix of vertices and edges. The incidence matrix has $|E|$ rows and $|V|$ columns. Each row contains a $+1$ and -1 for the two end points of the given edge (and zero otherwise). The sign is determined from the orientation of the edge. Similarly for the incidence relations of edges and faces: this is a sparse matrix with $|F|$ rows and $|E|$ columns, with appropriate $+1$ and -1 entries according to the relation of the orientation of edges as one moves around a face (according to its orientation). More generally d_k is the incidence matrix of k-cells on $k+1$-cells.

Implementation Given the oriented mesh $\mathscr{M}$ all that is required to implement the necessary operators is to assemble the incidence matrices. Note that these are sparse and contain only entries of type 0, $+1$, and -1. As we pointed out earlier, care is required in assembling these incidence matrices: the orientation must be taken into account in a *consistent* manner. A simple debugging sanity check (necessary but not sufficient) is to compute consecutive products: d_0 followed by d_1 must be a matrix of zeros, as must be d_1 multiplied by d_2. This reflects the fact that the boundary of any boundary is the empty set. It also corresponds to the calculus fact that curl of grad is zero as is divergence of curl.

Hodge Stars We need one last operation to complete our machinery. We noted earlier that fluxes can be seen as 2-forms on primal faces, or as dual 1-forms on dual edges. This desirable projection of a primal k-form to a conceptually-equivalent dual $(3-k)$-form is called the k^{th} *Hodge star*. We will denote $\star_0$ (resp., $\star_1, \star_2, \star_3$) the Hodge star taking a 0-form (resp., 1-form, 2-form, and 3-form) to a dual 3-form (resp., dual 2-form, dual 1-form, dual 0-form). These linear operators, describing the local metric, can also be stored in sparse matrices. In this chapter, we will use what is known as the *diagonal Hodge stars* [Bossavit 1998] as they are particularly simple to compute: only the diagonal terms are nonzero, and they are equal to the ratio of sizes of corresponding dual and primal cells: let vol(.) denote the volume of a cell (*i.e.*, 1 for vertices, length for edges, area for 2D cells, and volume for 3D cells), then the diagonal matrix entries are

$$(\star_k)_{qq} = \text{vol}(\tilde{\sigma}_q)/\,\text{vol}(\sigma_q)$$

where σ is a primal k-simplex, and $\tilde{\sigma}$ its dual. The subscript q indicates the index in the list of all cells. The Hodge star matrices are therefore symmetric positive definite. More accurate metric representations can be used, leading to less sparse Hodge stars; however, for our purposes, this one is sufficient. It also has the nice property that its inverse is trivial to compute.

The Hodge stars allow us to go from primal forms to dual forms of complementary dimension. In order to complete the deRham complex, we now need to define the *dual* version of the operators d_k; i.e., their equivalents on the dual side. These operators turn out to be quite simple: one can prove that the *transpose* of the d_k operators (and therefore, the transpose of their matrices) serve this purpose [Bossavit 1998]. Figure 5 summarizes the various operations between forms that we just defined. Equipped with these

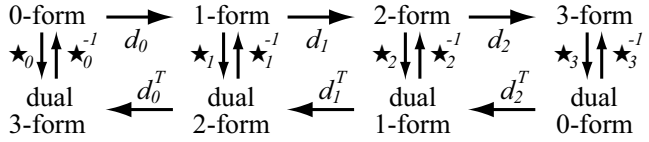


Figure 5: *Discrete Differential Calculus: the operators d_k and $\star_k$ allow proper manipulation of arbitrary discrete forms on the domain mesh.*

matrices, we can now formalize the definition of the various quantities we need for fluid mechanics—and see how they parallel their continuous analogs.

3.4 Revisiting Fluid Discretization

The fluid velocity $\mathbf{u}$ is treated as a flux, *i.e.*, as a 2-form. It is therefore represented by a vector U of values on faces (size $|F|$). Since we store fluxes on faces, the *circulation* can be derived as values on dual edges through $\star_2 U$ (Hodge star of the 2-form U). *Vorticity*, typically a 2-form in fluid mechanics [Marsden and Wenstein 1983], is easily computed by summing this circulation along the dual edges that form the boundary of a dual face. So we store the vorticity (seen as a dual 2-form) on dual faces. In other words, our choice of flux representation imposes that ${d_1}^T \star_2 U$ be used to represent $\boldsymbol{\omega} = \nabla \times \mathbf{u}$ (see Fig. 5). This is a vector Ω of size $|E|$, representing the vorticity on each dual face. These formal discretizations match our physically-motivated definitions previously given in Section 3.2.

With the appropriate Hodge star, we can go from a 2-form to a dual 1-form and vice-versa, so the reader may notice that it is in a sense equivalent to use a 2-form (resp., a dual 2-form) to represent a flux or to use the associated dual 1-form (resp., primal 1-form) to represent circulation. The reason we chose a 2-form instead of dual 2-form for vorticity is that it is easier to represent the boundary of our 3D domain by faces rather than dual faces.

3.5 From Vorticity Back To Flux

We have just seen how the vorticity can be directly derived from the set of all face fluxes. However, during the simulation, we will also need to recover flux *from* vorticity. For this we employ the Helmholtz-Hodge decomposition theorem, stating that any vector field $\mathbf{u}$ can be decomposed into three components (given appropriate boundary conditions)

$$\mathbf{u} = \nabla\phi + \nabla \times \boldsymbol{\psi} + \mathbf{h}.$$

A generalization to nD-domains and for k-forms reads as follows:

$$f_k = d_{k-1}\phi_{k-1} + \star_k{}^{-1} {d_k}^T \star_{k+1} \psi_{k+1} + h_k \qquad (7)$$

Because of our use of 2D fluxes, we only need the latter for $k = 2$. For the case of incompressible fluids (i.e., with zero divergence), two of the three components are sufficient to describe the velocity field: the curl of a vector potential and a harmonic field. This implies that when decomposing the 2-form U, we may set ψ_3 to 0. If the topology of the domain is trivial, we can furthermore ignore the harmonic part h_2 (we will discuss a full treatment of arbitrary topology in Section 4.8), leaving us with $U = d_1\phi_1$.

Thus, we can recover the velocity field solely from the vorticity by solving a Poisson equation to get the potential ϕ_1 and then applying the curl operator to the potential. The Poisson equation to solve for the 1-form ϕ (values on primal edges) is as follows:

$$(\star_1 d_0 \star_0{}^{-1} d_0^T \star_1 + d_1^T \star_2 d_1)\phi_1 = d_1^T \star_2 U = \omega \qquad (8)$$

To arrive at this equation, we applied $d_1^T \star_2$ to both sides of Eq. (7), and set the gauge of this Poisson problem as ${d_0}^T \star_1 \phi_1 = 0$. As the Laplacian Δ in differential calculus is $d \star d \star + \star d \star d$, one can readily verify that the previous equation is, indeed, a discrete version of the Poisson equation: it literally corresponds to $\Delta\phi_1 = (\nabla\nabla\cdot - \nabla\times\nabla\times)\phi_1 = \nabla\times\mathbf{u}$. Notice that the left-side matrix is *symmetric and sparse*, thus ideally suitable for fast numerical solvers.

Our linear operators (and, in particular, the discrete Laplacian) differ sharply from another discrete Poisson setup on simplicial complexes proposed in [Tong et al. 2003]: the ones we use have smaller support, which results in sparser and better conditioned linear systems [Bossavit 1998]—an attractive feature in the context of numerical simulation.

3.6 Interpolating Velocity and Circulation

So far we have only defined physical quantities as values on simplices. Although most computations can be carried out in this format, evaluation of such quantities *anywhere in space* is also necessary in practice.

Figure 6: *Bunny Snow Globe: the snow in the globe is advected by the inner fluid, initially stirred by a vortex to simulate a spin of the globe.*

Piecewise-Constant Velocity Field When considering the fluxes on the primal mesh, we can interpolate within each tet using the usual 1-form linear basis functions for discrete forms, known as Whitney forms, and described in detail in [Bossavit 1998] and in these course notes in Chapter 7. These interpolating basis functions are piecewise linear within each tet, and easy to compute. However, in our context of incompressible fluids, it turns out that we do not even need to use them. Since our velocity field is divergence free (the sum of the four fluxes on a tet equals 0), it can be shown that this piecewise-linear interpolation of the velocity field is *piecewise constant within each tet*. That is, there is a unique vector $\mathbf{u}_i$ per tet T_i that simultaneously *agrees with all four fluxes*. It is thus a trivial matter to deduce these vectors inside the tets given the vector U of all fluxes, rendering the Whitney forms unnecessary. Notice finally that the normal component to a face F_i of such a tet-based vector agrees (by definition) with the normal component of its neighboring tet through F_i, as they must both be equal to the flux of the velocity field on that face; therefore, advection along this velocity field is properly defined (i.e., you can always step over a face and will never get stuck somewhere) and easy to compute.

Piecewise Rational Circulation We will also need to interpolate the circulation from dual edges to the whole space. Unfortunately, Whitney forms are defined *only* for primal forms. In order to bypass this limitation, we propose a novel dual 1-form interpolant based on generalized barycentric coordinates. Taking a dual cell in isolation, note that each vertex of this dual cell has 3 dual edges incident on it. Given values of circulation on these adjacent dual edges, there exists a unique vector (or covector, to be mathematically correct) at the vertex that will fit these circulations. That is, we find the vector whose projection onto each dual edge is equal to the magnitude of the circulation along that edge. This amounts to reconstructing a vector based on its projection onto 3 independent vectors. Coincidentally, in our application (and once again, because of the divergence-free property), this vector turns out to be the vector value $\mathbf{u}_i$ of the velocity field defined in the tet associated with this dual vertex—we already have stored this value in the tet, and have no need to recompute it on the fly. With these vectors at each dual vertex, we can now *interpolate* them using generalized barycentric coordinates on 3D polytopes as recently proposed in [Warren et al. 2004]. This technique offers a fast evaluation of weights at each corner of an arbitrary polytope that provide a smooth interpolation of the corner values. It is also proved that this interpolation is *linear acurate*, i.e., it reconstructs exactly constant and linear fields. Finally, because these 3D barycentric weights have the property to degenerate into their 2D equivalents on the polytope's faces, the reader can verify that such an interpolation of circulation does fit the initial values of the circulation on dual edges, providing a good and fast computational method to handle circulation. Notice finally that this interpolation, just like their (simpler) primal equivalent (Whitney forms), are only tangentially-continuous across dual edges, reflecting that a dual 1-form has only meaning as a circulation as already explained in Chapter 7.

4 A Circulation-Preserving Integration

Once the proper discretization of space has been defined, we can now turn our attention to the actual integration of the Euler and Navier-Stokes equations. We propose a numerical integration scheme that solves Eq. (2) and preserves the circulation as stated in Eq. (3). We give details on how to efficiently implement this novel integration scheme.

4.1 Rationale: Vorticity Advection

Equipped with the spatial discretization defined above, we want to integrate the fluid equations. As noticed earlier, we wish to avoid having to resort to a projection to divergence-free (i.e., incompressible) flows. Therefore, what is truly needed is an *integration of the vorticity of the fluid*: this particular physical variable is by nature

divergence-free, and we have shown in Section 3.5 that the flow itself can be entirely derived from its vorticity (modulo a proper treatment of the genus of the domain and boundary condition, which we will detail in Section 4.8). In other words, and as we cover next, our approach can be summarized as effectively performing a *vorticity advection*.

Discrete Loops Guided by Kelvin's theorem, we wish to drive the integration by preserving the circulation along loops as they are advected in the flow. Since we are now working in a discrete space, satisfying this property for *any* loop $\mathscr{C}$ does not make sense. Remember that we are limited by the resolution of the spatial discretization that the mesh $\mathscr{M}$ provides and that we decided to describe the vorticity as a dual 2-form. Thus, we propose to *satisfy Kelvin's property on each boundary of dual 2-cells*. These Voronoi loops can indeed generate *any* discrete, dual loop C : the sum of adjacent loops is a larger, outer loop since all the interior edges cancel out due to opposite orientation as sketched in Fig. 7(right). Consequently, preserving Kelvin's theorem on each of these Voronoi loops will provide a discrete analog of the continuous case.

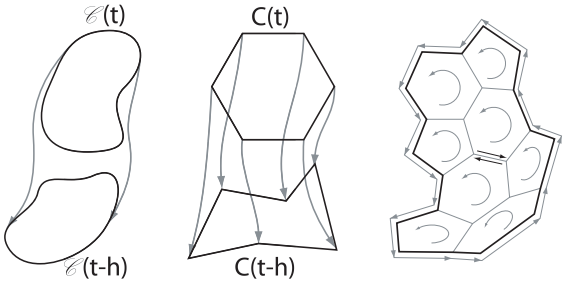


Figure 7: *Kelvin's Theorem: (left) in the continuous setting, the circulation on any loop being advected by the flow is constant. (middle) our discrete integration scheme enforces this property on each Voronoi loop, (right) thus on any discrete loop.*

Backtracking Discrete Loops For a given discrete Voronoi loop $C_i(t)$ considered at time t, we can conceptually *backtrack* it in time to find the loop $C_i(t-h)$ it originated from if the fluid velocity is assumed constant during the time step. Notice that this amounts, in the discrete case, to backtracking each circumcenter, as the piecewise-linear loops are exclusively defined by their positions. Ensuring circulation preservation is now a trivial matter: we simply have to compute the *current* circulation of this backtracked loop $C_i(t-h)$ and assign this value to the loop $C_i(t)$. By construction, we have *advected* the circulation along the flow. Using Stokes' theorem, this circulation is also the integral of the vorticity over the Voronoi face: thus we have formally found the new, advected vorticity. Notice that this backtracking is similar in spirit to the original Stable Fluids approach, but with a fundamental difference: the *vorticity* is advected instead of velocity. Again, its divergence-free nature makes it the natural variable to advect to avoid spurious numerical diffusion.

4.2 Setup and Pseudocode

An implementation of this vorticity advection algorithm requires rather usual data structures. We input a tet mesh $\mathscr{M}$ of the domain first, and start the preprocessing stage by storing the (signed) incidences between all simplices. The incidence matrices d_k described in Section 3.3 are subsequently stored. We also precompute the position of the circumcenter of each tet as we will repeatedly need them throughout the algorithm. Finally, we assemble the dual (Voronoi) cell of each vertex as they are used in the generalized barycentric coordinates interpolation described in Section 3.6. The integration from time t to time $t+h$ is then performed by successive updates of the vorticity according to the following pseudocode, starting from an initial set of fluxes U_t and corresponding vorticities Ω_t:

- **Advect Vorticity** (see Section 4.3 and Fig. 7)
  1. Backtrack each circumcenter through the flow, along the current piecewise-linear primal velocity field defined by U_t.
  2. Integrate circulation along each backward-advected Voronoi loop using numerical quadrature.
  3. Deduce the advected vorticity on each edge accordingly, and store the result in Ω^A_{t+h}
- **Add Body Forces**: From a set of external body forces (gravity, buoyancy) on each face, we further update the previous vorticity and store it in Ω^B_{t+h} as explained in Section 4.4.
- **Apply Diffusion** If we wish to simulate NS equations, we add an (optional) diffusion step on the resulting vorticity field to get Ω_{t+h} (see Section 4.5). Otherwise, we set $\Omega_{t+h} = \Omega^B_{t+h}$.
- **Convert Vorticity to Fluxes** We finally need to update the velocity field U_{t+h} for the next step, by converting the final value of Ω_{t+h} as detailed in Section 4.6.

For each item of this pseudocode, we now give details on what is involved and how it relates to the machinery previously developed.

4.3 Vorticity Advection

As sketched earlier, we can compute the new, advected vorticity on each edge through the following procedure: given the current velocity field U_t, we calculate the new vorticity on each dual face at time $t+h$ by backtracking its boundary (Voronoi) loop $C_i(t)$ through U_t and integrating the current circulation around $C_i(t-h)$ using numerical quadrature. The backtracking of all the loops is done by simply backtracking each tet's circumcenter through the flow using, for instance, a simple Euler integration. Note that this is particularly easy as the primal, divergence-free velocity field is constant per tet (see Section 3.6). We then use a simple quadrature to compute the circulation of the piecewise-linear loop defined by the backtracked circumcenters. The loop is considered as the union of each segment going from one backtracked circumcenter to the next. Using the generalized barycentric interpolation described in Section 3.6, we evaluate the interpolated velocity field at each segment end point. The circulation over each segment is then taken as the average of the velocity field at its two endpoints dotted with the segment. The new circulation around the loop is just the sum of the circulations on all these sample segments, giving us the advected vorticity Ω^A_{t+h}.

The quadrature accuracy could be easily improved by increasing the number of quadrature points on each segment, or by formally computing the circulation on the resulting piecewise-linear loops (a numerically expensive procedure as the circulation is locally expressed as a rational polynomial due to the necessary use of generalized barycentric coordinates). However, note that our spatial discretization is entirely based on linear basis functions; a linear-accurate quadrature suffices, as our numerical tests confirmed.

Note that one would be tempted to shortcut this quadrature by simply using the circulation computed from the primal velocity field. Unfortunately, such a procedure introduces significant inaccuracy since even in the case of stationary flows, the circulation would not be exactly preserved: the use of generalized barycentric coordinates and linear-accurate quadrature is a computationally-efficient *must*.

4.4 External Body Forces

The use of external body forces, like buoyancy, gravity, or stirring, is common practice to create interesting motions. Incorporating external forces into Eq. (5) is, fortunately, straightforward, resulting in:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} \;=\; -\nabla p + \nu\Delta\mathbf{u} + \mathbf{f}\,.$$

Again, taking the curl of this equation allows us to recast this equation in terms of vorticity:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \mathscr{L}_{\mathbf{u}}\boldsymbol{\omega} \;=\; \nu\Delta\boldsymbol{\omega} + \nabla\times\mathbf{f}\,. \tag{9}$$

Thus, we note that an external force influences the vorticity only through the force's curl (the $\nabla\cdot\mathbf{f}$ term is compensated for by the pressure term keeping the fluid divergence-free). Thus, if we express our forces through the vector F of their resulting fluxes in each face, we can directly add the forces to the domain by incrementing Ω^A_{t+h} by the circulation of F, i.e.:

$$\Omega^B_{t+h} = \Omega^A_{t+h} + h\, d_1^T \star_2 F.$$

4.5 Adding Diffusion

If we desire to simulate a viscous fluid, we must add the diffusion term present in Eq. (6). Note that previous methods were sometimes omitting this term because their numerical dissipation was already creating (uncontrolled) diffusion. In our case, however, this diffusion needs to be properly handled if viscosity is desired. This is easily done through an unconditionally-stable implicit integration as done in Stable Fluids (i.e., we also use a fractional step approach). Using the discrete Laplacian in Eq. (8), we simply solve for the diffused vorticity Ω_{t+h} using the following linear system:

$$(I - \nu h\Delta)\Omega_{t+h} = \Omega^B_{t+h}.$$

4.6 Converting Vorticity Back To Velocity

Finally, given the updated value of the vorticity Ω_{t+h}, we need to update the corresponding velocity field. This step is straightforward by solving the linear system given in Eq. (8), and taking the circulation of the resulting potential field ϕ_1 around each face to derive the new set of fluxes U_{t+h}.

One may argue that solving this Poisson equation is strictly equivalent to the projection step in Stable Fluids. While it is true that this step has the same computational cost, their respective roles are *very* different: while the Poisson equation is used as a projection in Stable Fluids, it is used as a mere conversion in our case, and does *not* incur numerical dissipation.

4.7 Boundary Conditions

Special treatment of boundaries is needed to ensure proper behavior of the resulting simulations.

Enforcing Boundary Conditions No-transfer boundary conditions are easily imposed by setting the fluxes through the boundary triangles to zero. Non-zero flux boundary conditions (i.e, forced fluxes through the boundary as in the case of Fig. 8) are, however, more subtle to handle. First, remark that all these boundary fluxes *must* sum to zero; otherwise, we would have little chance of getting a divergence-free fluid in the domain! As the total divergence is zero, there *must exist* a harmonic velocity field satisfying exactly these conditions, as stated by the Helmholtz-Hodge decomposition theorem with normal boundary conditions [Chorin and Marsden 1979]. Thus, this harmonic part $\mathbf{h}^{\partial,\mathscr{M}}$ can be computed *once and for all* through a Poisson equation using the same setup as described in Section 3.5. This precomputed velocity field allows us to deal very elegantly with these boundary conditions: we simply perform the same algorithm as we described by setting all boundary conditions to zero (with the exception of backtracking which takes the precomputed velocity into account), and *reinject* the harmonic part at the end of each time step (i.e., add $\mathbf{h}^{\partial,\mathscr{M}}$ to the current velocity field).

Viscous Fluids near Boundaries The Voronoi cells at the boundaries are slightly different from the usual, interior ones, since boundary vertices do not have a full 1-ring of tets around them. In the case of NS equations, this has no significant consequence: we set the velocity on the boundary to zero, resulting also in a zero circulation on the dual edges on the boundary. The rest of the algorithm can be used as is.

Inviscid Fluids near Boundaries For Euler equations, however, the tangent velocity at the boundary is not explicitly stored anywhere. Consequently, the boundary Voronoi faces need an additional variable to remedy this lack of information. We store in these dual faces the current integral *vorticity*, bootstrapping it with an initial vorticity imposed by the initial velocity field. From this additional information, we can *deduce* at each time step the missing circulation on the boundary (since the circulation over the inside dual edges is known, and the total integral must sum to the vorticity through Stokes' theorem).

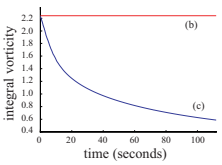
4.8 Handling Arbitrary Topology

Although the problem of arbitrary domain topology (e.g., when its first Betti number is not zero) is rarely discussed in CFD or in our field, it is important nonetheless. We first note that for Euler equations, Kelvin's theorem is also valid for *loops that are not shrinkable to a point* (i.e., loops around a tunnel, or obstacle). Therefore, in the absence of external forces, the circulation along each loop around a tunnel (note: for one period around the obstacle!) is constant in time. So once again, we precalculate a constant harmonic field based on the initial circulation around each tunnel, and simply *add it* to the current velocity field for advection purposes. This procedure serves two purposes: first, notice that we now automatically enforce the discrete equivalent of Kelvin's theorem on *any* (shrinkable or non-shrinkable) loop; second, arbitrary topologies are handled very efficiently.

5 Results and Discussion

We have tested our method on some of the usual "obstacle courses" in CFD. We start with the widely studied example of a flow past a disk (see Fig. 9). Starting with zero vorticity, it is well known that in the case of an inviscid fluid, the flow remains irrotational at all times. By construction, our method does respect this physical behavior since circulation is preserved for Euler equations. We then increase the viscosity of the fluid incrementally, and observe the formation of a vortex wake behind the obstacle, in agreement with physical experiments. As evidenced by the vorticity plots, vortices are shed from the boundary layer formed as a result of the adherence of the fluid to the obstacle, thanks to our proper treatment of the boundary conditions.

The behavior of vortex interactions observed in existing experimental results is now compared to numerical results based on our novel model and those obtained from the semi-Lagrangian advection method. It is known from theory that two like-signed vortices with a finite vorticity core will merge when their distance of separation is smaller than some critical value. This behavior is captured by the experimental data and shown in the first series of snapshots of Fig. 9. As the next row of snapshots indicates, the numerical results that our model generates present striking similarities to the experimental data. In the last row, we see that a traditional semi-Lagrangian advection followed by re-projection misses most of the fine structures of this phenomenon. This can be attributed to the loss of total integral vorticity as evidenced in this inset; in comparison our technique preserves this integral exactly.



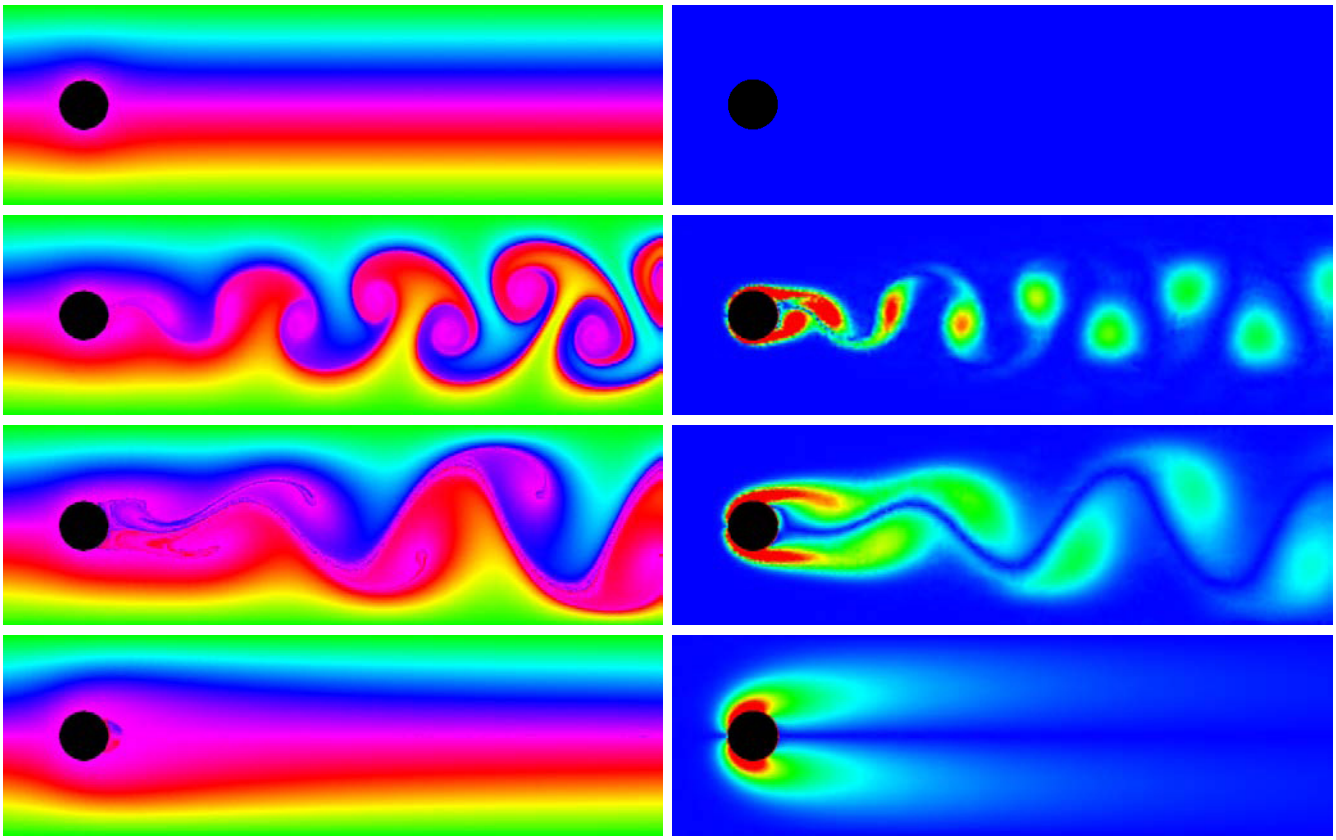


Figure 8: *Obstacle Course: in the usual experiment of a flow passing around a disk, the viscosity as well as the velocity can significantly affect the flow appearance; (left) our simulation results for increasing viscosity and same left boundary flux; (right) the vorticity magnitude (shown in false colors) of the same frame. Notice how the usual irrotational flow is obtained (top) for zero viscosity, while the von Karman vortex street appears as viscosity is introduced.*

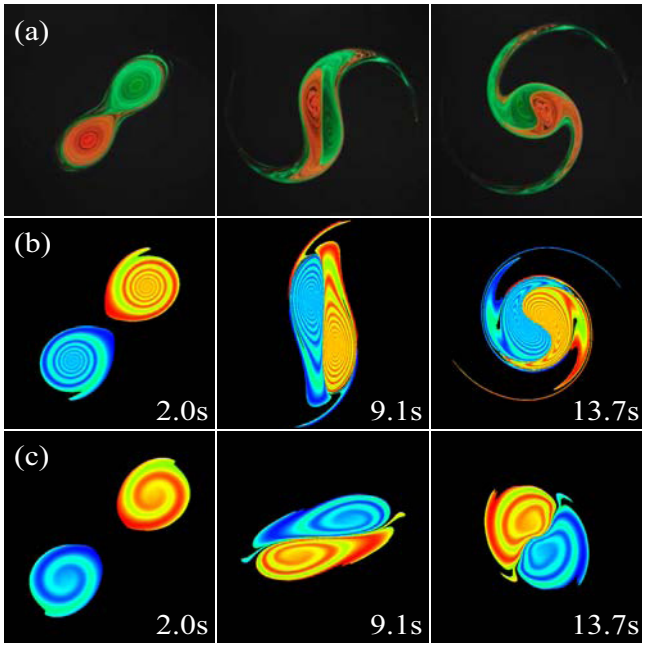


Figure 9: *Two Merging Vortices: discrete fluid simulations are compared with a real life experiment (courtesy of Dr. Trieling, Eindhoven University; see* `http://www.fluid.tue.nl/WDY/vort/index.html`*) where two vortices (colored in red and green) merge slowly due to their interaction (a); while our method faithfully simulates the merging phenomenon (b), a traditional semi-lagrangian scheme does not capture the correct motion because of vorticity damping (c).*

We have also considered the flow on curved surfaces in 3D with complex topology, as depicted in Fig. 10. We were able to easily extend our implementation of two-dimensional flows to this curved case thanks to the intrinsic nature of our approach.

The integration method we proposed can be directly applied to solve three-dimensional fluid flows. We consider a smoke cloud surrounded by air filling the body of a bunny as an example of flow in a domain with complex boundary. The buoyancy drives the air flow which, in turn, advects the smoke cloud in the three-dimensional domain bounded by the bunny mesh as shown in Fig. 1.

In the last simulation, we show a snow globe with a bunny inside in Fig. 6. We emulate the flow due to an initial spin of the globe using a swirl described as a vorticity field. The snow particles are transported by the flow as they fall down under the effect of gravity.

6 Conclusion

In this chapter, we have introduced a novel theoretical approach to fluid dynamics, along with its practical implementation and various simulation results. We have carefully discretized the physics of flows to respect the most fundamental geometric structures that characterize their behavior. Amongst the several specific benefits that we demonstrated, the most important is the circulation preservation property of the integration scheme, as evidenced by our numerical examples. The discrete quantities we used are intrinsic, allowing us to go to curved manifolds with no additional complication. Finally, the machinery employed in our approach can be used on any simplicial complex. We wish to emphasize, however, that the same methodology also applies directly to more general spatial partitionings, and in particular, to regular grids or hybrid

meshes [Feldman et al. 2005]—rendering our approach widely applicable to existing fluid simulators.

For future work, a rigorous analysis (beyond the scope of this chapter) of the advantages of the current method over some of the standard approaches should be properly investigated.

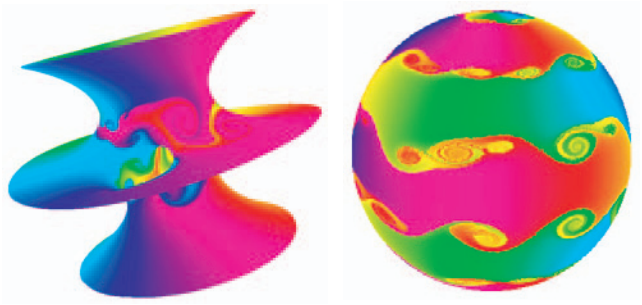


Figure 10: *Weather System on Planet Funky: the intrinsic nature of the variables used in our algorithm makes it amenable to the simulation of flows on arbitrary curved surfaces.*

References

ABRAHAM, R., MARSDEN, J., AND RATIU, T., Eds. 1988. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences Vol. 75, Springer.

BOSSAVIT, A., AND KETTUNEN, L. 1999. Yee-like schemes on a tetrahedral mesh. *Int. J. Num. Modelling: Electr. Networks, Dev. and Fields 12* (July), 129–142.

BOSSAVIT, A. 1998. *Computational Electromagnetism*. Academic Press, Boston.

CHORIN, A., AND MARSDEN, J. 1979. *A Mathematical Introduction to Fluid Mechanics*, 3rd edition ed. Springer-Verlag.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual Simulation of Smoke. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND KLINGNER, B. M. 2005. A method for animating viscoelastic fluids. *ACM Transactions on Graphics (SIGGRAPH)* (Aug.).

FETECAU, R. C., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2003. Nonsmooth Lagrangian Mechanics and Variational Collision Integrators. *SIAM J. Applied Dynamical Systems 2*, 381–416.

FOSTER, N., AND FEDKIW, R. 2001. Practical Animation of Liquids. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 23–30.

FOSTER, N., AND METAXAS, D. 1997. Modeling the Motion of a Hot, Turbulent Gas. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 181–188.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics 23*, 3 (Aug.), 463–468.

HIRANI, A. 2003. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology.

KANE, C., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2000. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *Internat. J. Numer. Methods Engrg. 49*, 1295–1325.

LANGTANGEN, H.-P., MARDAL, K.-A., AND WINTER, R. 2002. Numerical Methods for Incompressible Viscous Flow. *Advances in Water Resources 25*, 8-12 (Aug-Dec), 1125–1146.

LEW, A., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2003. Asynchronous Variational Integrators. *Arch. Rational Mech. Anal. 167*, 85–146.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics 23*, 3 (Aug.), 457–462.

MARSDEN, J. E., AND WENSTEIN, A. 1983. Coadjoint orbits, vortices and Clebsch variables for incompressible fluids. *Physica D 7*, 305–323.

MARSDEN, J. E., AND WEST, M. 2001. Discrete Mechanics and Variational Integrators. *Acta Numerica*, 357–515.

MCNAMARA, A., TREUILLE, A., POPOVIC, Z., AND STAM, J. 2004. Fluid Control Using the Adjoint Method. *ACM Transactions on Graphics 23*, 3 (Aug.), 449–456.

MORTON, K. W., AND ROE, P. 2001. Vorticity-Preserving Lax-Wendroff-Type Schemes for the System Wave Equation. *SIAM Journal on Scientific Computing 23*, 1 (July), 170–192.

MUNKRES, J. R. 1984. *Elements of Algebraic Topology*. Addison-Wesley.

PIGHIN, F., COHEN, J. M., AND SHAH, M. 2004. Modeling and Editing Flows Using Advected Radial Basis Functions. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 223–232.

SHAH, M., COHEN, J. M., PATEL, S., LEE, P., AND PIGHIN, F. 2004. Extended Galilean Invariance for Adaptive Fluid Simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 213–221.

SHI, L., AND YU, Y. 2004. Inviscid and Incompressible Fluid Simulation on Triangle Meshes. *Journal of Computer Animation and Virtual Worlds 15*, 3-4 (June), 173–181.

STAM, J. 1999. Stable Fluids. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 121–128.

STAM, J. 2001. A Simple Fluid Solver Based on the FFT. *Journal of Graphics Tools 6*, 2, 43–52.

STAM, J. 2003. Flows on Surfaces of Arbitrary Topology. *ACM Transactions on Graphics 22*, 3 (July), 724–731.

STEINHOFF, J., AND UNDERHILL, D. 1994. Modification of the Euler Equations for *Vorticity Confinement*: Applications to the Computation of Interacting Vortex Rings. *Physics of Fluids 6*, 8 (Aug.), 2738–2744.

TONG, Y., LOMBEYDA, S., HIRANI, A. N., AND DESBRUN, M. 2003. Discrete Multiscale Vector Field Decomposition. *ACM Trans. Graph. 22*, 3, 445–452.

TONG, Y. 2004. *Towards Applied Geometry in Graphics*. PhD thesis, University of Southern California.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe Control of Smoke Simulations. *ACM Transactions on Graphics 22*, 3 (July), 716–723.

WARREN, J., SCHAEFER, S., HIRANI, A., AND DESBRUN, M., 2004. Barycentric Coordinates for Convex Sets. Preprint.

YAEGER, L., UPSON, C., AND MYERS, R. 1986. Combining Physical and Visual Simulation - Creation of the Planet Jupiter for the Film 2010. *Computer Graphics (Proceedings of SIGGRAPH 86) 20*, 4, 85–93.

ZHONG, G., AND MARSDEN, J. E. 1988. Lie-Poisson Hamilton-Jacobi Theory and Lie-Poisson Integrators. *Physics Letters A 133*, 3 (Nov.).

Chapter 10:
Variational Tetrahedral Meshing

Pierre Alliez
INRIA

David Cohen-Steiner
INRIA

Mariette Yvinec
INRIA

Mathieu Desbrun
Caltech

Abstract

In this chapter, a novel Delaunay-based variational approach to isotropic tetrahedral meshing is presented. To achieve both robustness and efficiency, we minimize a simple mesh-dependent energy through global updates of both vertex positions *and* connectivity. As this energy is known to be the $\mathcal{L}^1$ distance between an isotropic quadratic function and its linear interpolation on the mesh, our minimization procedure generates well-shaped tetrahedra. Mesh design is controlled through a gradation smoothness parameter and selection of the desired number of vertices. We provide the foundations of our approach by explaining both the underlying variational principle and its geometric interpretation. We demonstrate the quality of the resulting meshes through a series of examples.
Work published in ACM SIGGRAPH'05 proceedings

Keywords: Isotropic meshing, Delaunay mesh, sizing field, slivers.

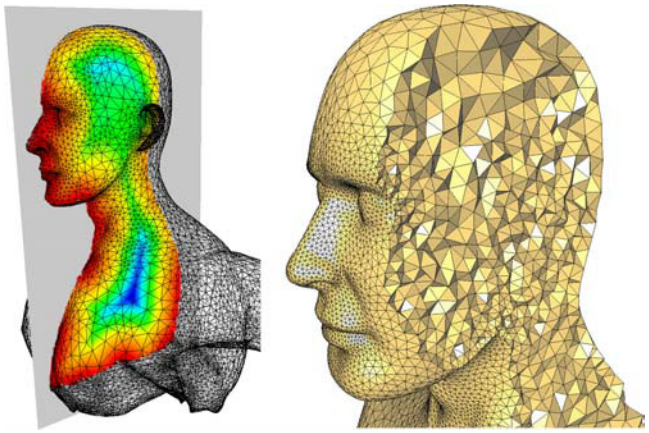


Figure 1: *Variational Tetrahedral Meshing: Given the boundary of a domain (here, a human torso), we automatically compute the local feature size of this boundary as well as an interior sizing field (left, cross-section), before constructing a mesh with a prescribed number of vertices (here $65K$) and a smooth gradation conforming to the sizing field (right, cutaway view). The resulting tetrahedra are all well-shaped (i.e., nearly regular).*

1 Introduction

Three-dimensional simplicial mesh generation aims at tiling a bounded 3D domain with tetrahedra so that any two of them are either disjoint or sharing a lower dimensional face. Such a discretization of space is required for most physically-based simulation techniques: realistic simulation of deformable objects in computer graphics, as well as more general numerical solvers for partial differential equations in computational science, need a discrete domain to apply finite-element or finite-volume methods. Most applications have specific requirements on the size and shape of simplices in the mesh. *Isotropic meshing* is desirable in the common case where nearly-regular tetrahedra (nearly-equal edge lengths) are preferred.

Creating high quality tetrahedral meshes is a difficult task for a variety of reasons. First, the mere size of the resulting meshes requires robust, disciplined data structures and algorithms. There are also basic mathematical difficulties which make tetrahedral meshing significantly harder than its 2D counterpart: the most isotropic 3D simplex, the regular tetrahedron, does *not* tile 3D space (let alone specific domains), while the equilateral triangle does tile the plane; unlike the 2D case, even well-spaced vertices can create degenerate 3D elements such as slivers (see Fig. 2). Dealing with boundaries is also fundamentally more difficult in 3D: while there exists 2D triangulations conforming to any set of non intersecting constraints, this is no longer true in 3D [Shewchuk 1998a]. All these facts conspire to make both the development of algorithms and suitable error analysis for the optimal 3D meshing problem very challenging. Given that one can often observe in applications that the worst element in the domain dictates accuracy and/or efficiency [Shewchuk 2002a], it is clear that great care is required to design the underlying meshes and ensure that they meet the desired quality standards.

1.1 Previous Work & Nomenclature

The meshing community has extensively studied a number of techniques over the last 20 years. We do not aim at covering all previous work since comprehensive surveys are available [Carey 1997; Owen 1998; Frey and George 2000; Teng et al. 2000; Eppstein 2001]. To motivate our work we briefly review both the usual nomenclature and the main difficulties involved in isotropic tetrahedral mesh generation. Throughout *tet* will be the abbreviation for tetrahedron.

Proper mesh generation requires a number of successive stages, which are governed by a number of key factors:

◇ **Shape Quality Measures**: Element shape/size requirements are typically application-dependent. Consequently, an extraordinarily large number of quality measures has been proposed, ranging from minimum or maximum bounds on dihedral or solid angles, to more complex geometric ratios. We warmly recommend [Shewchuk 2002a] for a clear exposition of both the history behind these measures and their relation to (1) the conditioning of finite element stiffness matrices and (2) the accuracy of linear interpolation of functions and their gradients. Among the most popular quality measures of a tet are the radius and radius-edge ratios. The latter measures the ratio between the circumsphere radius and the shortest edge length. It is not a *fair* measure since it does not approach zero for a class of degenerate tets called *slivers* (slivers result when four tet vertices are close to a great circle of a sphere and spaced roughly equally along this circle, see Fig. 2). The radius ratio, which takes the quotient of inscribed and circumscribed sphere radii (times three for normalization purposes), *is* a good measure for any kind of degeneracy.

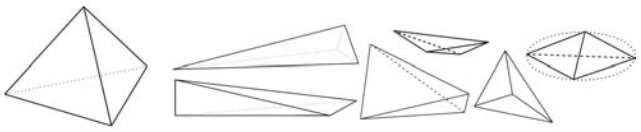


Figure 2: *Tet shapes: the regular tet (leftmost) is well shaped, unlike the other tets displayed: each represents a type of degeneracy. The rightmost one with 4 near-cocircular vertices is usually referred to as a* sliver.

◇ **Sizing requirement**: Accuracy and efficiency of numerical solvers depend on the local size of tets. Consequently, a *sizing field*, prescribing the ideal local edge length as a function of space, must be added. Obvious choices include the constant field for a

uniform mesh, and a priori or a posteriori error estimators for simulations. To avoid bad dihedral angles in the simplices one typically requires the sizing field to vary smoothly [Ruppert 1993].

- ◇ **Boundary Requirements**: Some approaches aim at *conforming* to (i.e., matching exactly) the domain boundary by adding Steiner points if necessary [Cohen-Steiner et al. 2002; Krysl and Ortiz 2001; Cheng and Poon 2003]. Others require of the mesh boundary to only *approximate* the domain boundary. The latter allows for higher tet quality since the boundary is not required to match the input surface. In particular the latter is important when the initial input is a low quality surface triangulation.
- ◇ **Strategy**: Existing meshing techniques can be roughly classified by the general strategy they employ:
  - ◇ *Advancing front*: Starting from the boundary of the domain, new vertices are added by a local heuristic to ensure that the generated tets have acceptable shapes and sizes and conform to the desired sizing field. Global optimization steps can also be performed sporadically to improve the mesh quality further. A number of variants exist, such as sphere or bubble packing [Li et al. 2000], which provide better tet shape and size control albeit adding a significant computational overhead.
  - ◇ *Octree-based methods*: An octree is first refined until each of its leaves is either strictly inside or strictly outside of a finely voxelized version of the domain. Proper connections of the interior leaves through, for instance, a red-green strategy [Molino et al. 2003] then ensure a good initial mesh of the domain, usually improved through optimization or physically-based relaxation in particular to better approximate the domain boundary. Other similar methods offer bounds of worst dihedral angles even without a relaxation stage [Mitchell and Vavasis 2000]. Unfortunately, octree-based meshes have preferred edge directions, which may be detrimental to subsequent use in simulation.
  - ◇ *Delaunay approaches*: For a given set of sample points in $3D$, its *Delaunay triangulation* has the canonical property of minimizing the maximum radius of the minimum containment sphere. This property is very useful in approximation theory: this radius provides an upper bound on the $\mathcal{L}^\infty$ difference between any function f and its *piecewise linear approximant*, assuming f has bounded second derivatives. Thus a Delaunay triangulation provides good control over the worst interpolation error inside a domain. Consequently a large body of work in numerical analysis provides error estimates for a variety of applications using these meshes. Because of these as well as many other optimality properties, mesh generation relying on Delaunay triangulation such as Delaunay refinement [Ruppert 1993; Shewchuk 1998b; Shewchuk 2002b; Cheng et al. 2004], unit mesh [Borouchaki et al. 1997a; Borouchaki et al. 1997b], or centroidal Voronoi tessellations [Du and Wang 2003] have flourished in the meshing and Computational Geometry communities. Delaunay refinement methods offer some theoretical guarantees on the resulting meshes: they provide bounds on the radius-edge ratio, and are shown to be asymptotically optimal with respect to the number of elements in the mesh. Delaunay refinement, however, can generate slivers; some attempts have been made to handle the sliver problem within Delaunay refinement [Cheng et al. 1999; Cheng and Dey 2002; Li and Teng 2001]. Unfortunately the theoretical guarantees are quite poor, and the mesh either is *no longer Delaunay* but a regular (weighted Delaunay) triangulation, or comes with degraded bounds on the radius-edge ratio.
  - ◇ *Mesh Optimization Techniques*: Even if fast and robust Delaunay triangulators are available, the previous strategies can require substantial implementation effort to make them robust to arbitrary input domains. A large number of practical meshing techniques instead employ local optimization methods which move vertices adjacent to poorly-shaped tets to improve mesh quality. Coupled with local face swapping between adjacent tets as well as tet insertions and deletions, these strategies can result in nice final meshes [Freitag and Ollivier-Gooch 1996; Cutler et al. 2004]. Unfortunately, these optimizations often use highly non-convex functionals and get easily stuck in local minima.

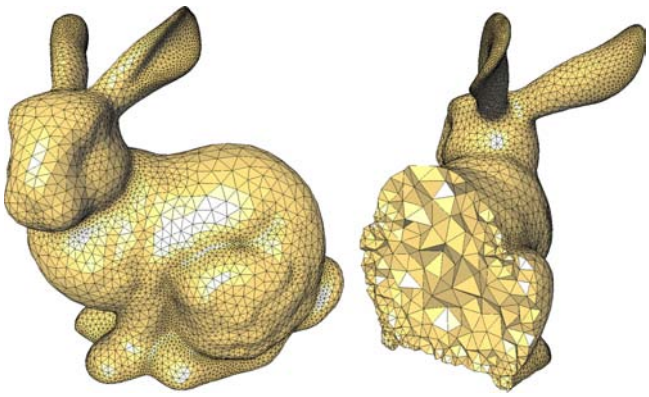


Figure 3: *Stanford bunny: meshing the interior of the bunny with adapted tets (smaller near the boundary, larger inside, and smooth gradation ($K = 1$) in between). The cutaway views show the well-shapedness of the mesh elements inside the domain; notice also the quality of the boundary mesh.*

From this brief overview we see that meshing has been approached with two very different emphases: theory and practice. Theoretical methods, most commonly using iterative Delaunay refinement approaches, come with quality guarantees that are often not suited to further use in practical applications: the presence of fairly degenerated tets are a serious problem for many numerical methods. Alternatively, optimization methods provide viable solutions with relatively little implementation effort, and the quality obtained is satisfactory for a class of applications. Alas, their ad-hoc nature does not warrant high-quality meshes. When seeking high quality meshes, a method combining optimization with solid theoretical foundations would provide the best of both worlds, promising meshes of a quality that none of the existing approaches could obtain by themselves.

1.2 Approach and Contributions

In this chapter, we present a Delaunay-based optimization technique, that we call *Variational Tetrahedral Meshing*, to efficiently mesh a bounded 3D domain Ω of arbitrary topology or number of connected components. The domain boundary $\partial\Omega$ is assumed to be a manifold, watertight and intersection-free triangular mesh. Drawing on recent work on surface approximation [Cohen-Steiner et al. 2004] and Optimal Delaunay Triangulations [Chen and Xu 2004], we propose a simple minimization procedure that alternates global 3D Delaunay triangulation and local vertex relocation to *consistently and efficiently minimize a global energy* over the domain. It results in a robust meshing technique that generates high quality isotropic meshes in terms of radius ratios, as well as angles. A notable feature of the method is that it removes slivers inside the domain. To provide a flexible meshing tool, we also introduce an automatic sizing field construction that guarantees an arbitrary smooth gradation of the mesh together with faithful approximation of the domain boundary. Equipped with these tools, the user has full control over the mesh design, and can require a specific number of vertices for the final mesh. We demonstrate the versatility and robustness of our method through a series of results and comparisons; we also give details on the current limitations.

2 Variational Approach to Meshing

Variational approaches (that is, methods relying on energy minimization) have been advocated as a powerful and robust tool in meshing both in graphics for triangle [Hoppe et al. 1993; Cohen-Steiner et al. 2004] and tet [Molino et al. 2003; Cutler et al. 2004]

meshes and in mechanical engineering for volumetric meshes [Freitag and Ollivier-Gooch 1996; Du and Wang 2003]. These methods basically define (often highly) non-convex energies that they minimize through vertex displacements and/or connectivity changes in the current mesh. Our method also falls into this broad category. However, in contrast to earlier work, we use a simple quadratic energy (which we analyze) and allow for global changes in mesh connectivity during energy minimization. We will point out both the theoretical and practical consequences of such a strategy. We begin by motivating our choice of energy.

2.1 Consistent Energy Minimization

Among the earlier variational approaches, a few [Cohen-Steiner et al. 2004; Du and Wang 2003] have an attractive theoretical property resulting in remarkable results: vertex positions *and* connectivity updates are performed alternately in order to *minimize the same quadratic energy*. This specificity has rich consequences. First, each update can be done *optimally* due to the simplicity of the energy used. Second, assuming convexity of the boundary, the energy decreases monotonically, implying eventual convergence. Lastly, since both optimization steps minimize the same energy, their final meshes have a concrete, variational nature with all the qualities that it entails: these meshes are (quasi) minimizers of a "quality" functional.

Centroidal Voronoi Tessellations Du and Wang [2003] propose to generate meshes that are *dual* to optimal Voronoi diagrams. These diagrams are achieved by minimizing the quadratic energy:

$$E_{\text{CVT}} = \sum_{i=1..N} \int_{V_i} ||\mathbf{x} - \mathbf{x}_i||^2 d\mathbf{x} \qquad (1)$$

where the $\mathbf{x}_i$ are vertex positions and V_i a local cell associated with each $\mathbf{x}_i$; the union of these cells forms a partition of the domain Ω. Du and Wang used Lloyd relaxation [Lloyd 1957] to robustly minimize this energy: for a given set of vertices, compute their Voronoi diagram (restricted to the domain Ω) since it *is* the energetically optimal partition for the current vertex positions. In a second phase, the partition is held fixed and vertex positions $\mathbf{x}_i$ are optimized. Even though these steps of *partitioning* and *vertex position* optimization are quite different in character, each of them decreases the same energy. Du and Wang explain how a mesh that minimizes this energy has each vertex *at the centroid* of its own Voronoi cell: hence the name *Centroidal Voronoi Tessellation* (CVT). Aside from the theoretical properties of CVTs, Du and Wang also note the superior results they get in comparison to conventional Laplacian smoothing (a widespread technique in graphics due to its simplicity, but for which the associated energy only relies on edge length, not on spatial distribution).

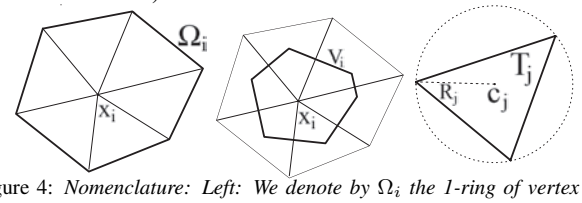


Figure 4: *Nomenclature: Left: We denote by Ω_i the 1-ring of vertex $\mathbf{x}_i$. Middle: V_i is the Voronoi cell of vertex $\mathbf{x}_i$. Right: The center of the circumcircle of triangle T_j, is denoted $\mathbf{c}_j$, while its radius is denoted R_j.*

From the analysis of E_{CVT} it is well known that its minimization corresponds to minimizing the volume between a paraboloid $f(\mathbf{x}) = ||\mathbf{x}||^2$ and an *underlaid, circumscribing piecewise linear approximant* $f^{\text{dual}}_{\text{PWL}}$, which is formed by planar patches tangent to the paraboloid (see Fig. 5(a)):

$$E_{\text{CVT}} = ||f - f^{\text{dual}}_{\text{PWL}}||_{\mathcal{L}^1}.$$

In 2D, without invoking much approximation theory, the reader can convince herself that this approach will lead to isotropically sampled meshes since it has been shown that any $\mathcal{L}^p$ optimal approximation of a smooth function asymptotically tends to align and shape its elements according to the eigenvectors and eigenvalues of its Hessian [Shewchuk 2002b]: given the isotropic nature of the Hessian of the quadratic function $f(\mathbf{x}) = ||\mathbf{x}||^2$ the resulting meshes must have nearly hexagonal Voronoi cells, i.e., nearly equilateral triangles in the dual Delaunay mesh.

Unfortunately, and despite Du's proposal [2003] to use CVTs for tet meshing, there exists no proof of such a dual property in 3D. Our own tests show that using Du's suggestion for tet meshing gives rise to numerous degenerate *sliver* tets (see Fig. 2). We can attribute the slivers to the fact that E_{CVT} tends to optimizes the compactness of the dual Voronoi cells, but not the compactness of simplices in the primal Delaunay triangulation: therefore, the presence of a sliver is *not* penalized by this energy. In other words, this variational approach ensures that the vertices in the domain are well spaced (i.e., isotropic point sampling—see [Hardin and Saff 2004] for an overview of this interesting problem); sadly, well-spacedness does not guarantee anything in terms of the quality of the resulting 3D mesh [Eppstein 2001].

Optimal Delaunay Triangulations Recently, Chen [2004] proposed an approach "dual" to the above in the context of mesh optimization. He used the following energy:

$$E_{\text{ODT}} = ||f - f^{\text{primal}}_{\text{PWL}}||_{\mathcal{L}^1},$$

i.e., the volume between a paraboloid and an *overlaid, circumscribing piecewise linear approximant* $f^{\text{primal}}_{\text{PWL}}$ formed by a linear interpolation of points on the paraboloid (see Fig. 5(b)). Chen made the observation that changing the energy from E_{CVT} to E_{ODT} amounts to only a slight change in Eq. (1), turning it into:

$$E_{\text{ODT}} = \frac{1}{n+1} \sum_{i=1..N} \int_{\Omega_i} ||\mathbf{x} - \mathbf{x}_i||^2 d\mathbf{x}. \qquad (2)$$

The integral is now taken over each *1-ring* region Ω_i (also called the star of the vertex $\mathbf{x}_i$, see Fig. 4). Notice that these regions overlap. These quadratic energies differ quite significantly: Chen's E_{ODT} energy measures a quality of the *simplicial mesh*, not of its dual. It is thus more prone to generate well-shaped primal elements, while E_{CVT} was maximizing the compactness of the dual Voronoi cells.

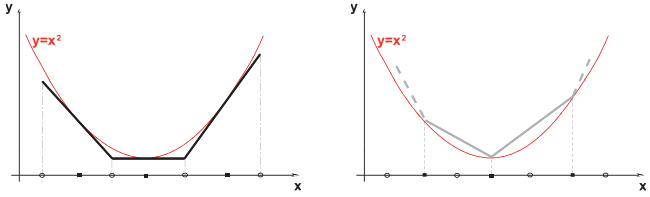


Figure 5: *PWL approximations: A paraboloid can be approximated by an underlaid circumscribed PWL function (left), or by an overlaid one (right).*

Although no formal guarantee on the resulting meshes is given in [Chen and Xu 2004], the 2D results presented are of high quality. The smoothing technique presented in [Chen 2004] updates the mesh connectivity through only-local edge flips when an inverted triangle is detected. Unfortunately, this local connectivity optimization in the 2D triangle case does not carry over to 3D: there is no theorem proving that an arbitrary mesh is only a few flips away from the optimal connectivity.

2.2 Our Variational Approach

We propose an algorithm to consistently minimize the primal energy E_{ODT}. This is achieved not just through a *smoothing* procedure (as suggested in [Chen 2004]), but through a full-blown minimization procedure for both vertex positions *and* connectivity.

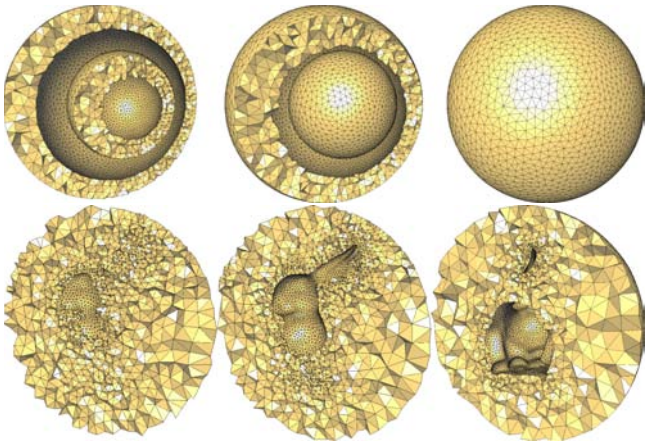


Figure 6: *Complex Topology: As stressed in this chapter, our approach can as well mesh complex domains with arbitrary genus. Four nested spheres define a multi-layer object (top); a bunny immersed in a sphere (bottom).*

Optimizing Connectivity Connectivity optimization is easily achieved: for a given set of vertex positions $\mathbf{x}_i$, its Delaunay triangulation is again (remarkably!) *the* optimal connectivity which minimizes E_{ODT} (as shown in [Chen and Xu 2004]) just as it is optimal for E_{CVT}. Therefore, we compute the (global) Delaunay connectivity systematically, guaranteeing optimality of the connectivity at each iteration.

Optimizing Vertex Positions One can show that for a given mesh $\mathcal{M}$ with vertices $\mathbf{x}_i$'s, E_{ODT} can be written as:

$$E_{\text{ODT}} = \frac{1}{4}\sum_i {\mathbf{x}_i}^2 |\Omega_i| - \int_{\mathcal{M}} \mathbf{x}^2 d\mathbf{x}, \tag{3}$$

where $|\Omega_i|$ is the measure (volume in 3D) of the 1-ring neighborhood of vertex $\mathbf{x}_i$ (Appendix B gives a short proof). From this new expression and after noting that the last term is constant given a fixed boundary $\partial\mathcal{M}$, a simple derivation of this quadratic energy in $\mathbf{x}_i$ leads to the following *optimal position* $\mathbf{x}_i^\star$ of the interior vertex $\mathbf{x}_i$ in its 1-ring:

$$\mathbf{x}_i^\star = -\frac{1}{2\,|\Omega_i|}\sum_{T_j\in\Omega_i}\left(\nabla_{\mathbf{x}_i}|T_j|\left[\sum_{\substack{\mathbf{x}_k\,\in\,T_j\\ \mathbf{x}_k\,\neq\,\mathbf{x}_i}} ||\mathbf{x}_k||^2\right]\right). \tag{4}$$

The term $\nabla_{\mathbf{x}_i}|T_j|$ is the gradient of the volume of the tet T_j with respect to $\mathbf{x}_i$. Replacing function $f(\mathbf{x}) = ||\mathbf{x}||^2$ by the translated function $f(\mathbf{x}) = ||\mathbf{x}-\mathbf{x}_i||^2$, which has the same interpolation error and thus leads to the same optimal position, we get the following equivalent expression used to update a vertex position :

$$\mathbf{x}_i^\star = \mathbf{x}_i - \frac{1}{2\,|\Omega_i|}\sum_{T_j\in\Omega_i}\left(\nabla_{\mathbf{x}_i}|T_j|\left[\sum_{\mathbf{x}_k\in T_j} ||\mathbf{x}_i-\mathbf{x}_k||^2\right]\right). \tag{5}$$

Geometric and Physical Interpretations As shown in the Appendix C, we can express the latter optimality condition in more obvious geometric terms, to further our understanding of the benefits of this variational approach:

$$\mathbf{x}_i^\star = \frac{1}{|\Omega_i|}\sum_{T_j\in\Omega_i}|T_j|\mathbf{c}_j. \tag{6}$$

where $\mathbf{c}_j$ is the circumcenter of tet T_j (see Fig. 4). This last expression shows that, although we move each vertex to a local average, the optimal placement heavily depends on the local distribution. For instance, if all the 1-ring neighbors are on a common sphere, the optimal position will be the sphere center. In fact, as evidenced by Eq. 4, this optimal location depends *only* on the 1-ring neighbors, not on the current location. Note also the similarities with the generalized barycentric coordinates in the Voronoi polytope proposed in [Warren et al. 2004].

If we further transform the energy (see Appendix B), we get:

$$E_{\text{ODT}} = \sum_j \left(|T_j|R_j^2 - \int_{T_j}||\mathbf{x}-\mathbf{c}_j||^2 d\mathbf{x}\right) = 2\sum_j |M_{S_j} - M_{T_j}| \tag{7}$$

where M_{T_j} is the sum of the principal moments (i.e., the trace of the inertia tensor) of the tet T_j w.r.t. the circumcenter $\mathbf{c}_j$, while M_{S_j} is the same quantity for the *circumshell* S_j of equivalent mass (i.e., a shell in the shape of the circumsphere, with the same mass as tet T_j). Minimizing this energy amounts to *make the average moment of inertia of each tet match the one of its circumshell of equivalent mass*. Even Eq. (6) can be re-expressed in terms of these circumshells: a vertex is moved at the *barycenter* of its neighboring circumshells, which is reminiscent of the CVT property, this time on the primal mesh. We believe that these series of observations provide further insights on this simple quadratic energy, and how it relates to the well-shapedness of the resulting tets. It also provides a straightforward generalization to graded meshing as explained next.

2.3 Extension to Graded Meshes

Since the previous expressions only apply to uniform meshing, we extend the optimality condition next to allow for more flexible meshing capacities. For this purpose, we will make use of a *sizing field* μ as a roadmap to the desired tet sizes within the domain.

Generalized Optimality Conditions Eq. (6) gives us a straightforward means to extend the previous approach to create *graded* meshes. One can simply define a *mass density* in space, and use this mass density in the computation of the inertia tensors. This density should be in agreement with the sizing field, i.e., the desired size of a tet locally. In order to simplify the computations, we use a one-point approximation of the sizing field μ in a tet and define the mass density as being $1/\mu^3$ since the local volume of a tet should be roughly equal to the third power of the ideal edge size. Thus, we modify the optimality condition of a vertex as follows:

$$\mathbf{x}_i^\star = \frac{1}{\sum_{T_k\in\Omega_i}\frac{|T_k|}{\mu^3(\mathbf{g}_k)}}\sum_{T_j\in\Omega_i}\frac{|T_j|}{\mu^3(\mathbf{g}_j)}\mathbf{c}_j. \tag{8}$$

where $\mathbf{g}_k$ is the centroid of T_k. A formal integration of the sizing field within each tet would provide more precision. However it would also significantly affect the computational cost without a drastic change in the results thanks to our choice of sizing field (described next). Finally, we keep the Delaunay triangulation (of the new point positions) as the optimal connectivity.

Automatic Design of Sizing Field The sizing field can be virtually any function tuned to the specific application the mesh is being designed for. However we wish to provide a default sizing field construction that can robustly achieve a large spectrum of mesh types. Notice that the sizing field defined here is *relative*; it describes the inhomogeneity of the desired edge length. The actual edge length will be proportional to this relative value to fit whatever prescribed vertex budget. (Alternatively the user may want to use as many vertices as needed to produce a specific edge size.)

Because we aim at an isotropic approximation of the input domain boundary, the sizing field on the boundary should be a function of the local absolute maximum curvature. Since we also aim at approximating the domain topology, we need to make sure that the boundary approximation error will never exceed the local "thickness" of the domain: for instance, a dumbbell shape should have small tets in its bottleneck. Therefore we propose to build our sizing field on the notion of *local feature size* (*lfs*) introduced by Amenta

and Bern [1998] and widely used in the field of shape reconstruction: it corresponds to the combination of curvature *and* thickness as we require. To define the local feature size, one first introduces the medial axis $Sk(\Omega)$, of the domain (its intuitive skeleton) which is the locus of all the centers of maximal balls included in either Ω or its complement. Note that this skeleton has already been identified in the meshing community as playing a central role in sizing [Quadros et al. 2004]. Then the local feature size $lfs(\mathbf{x})$ at a point $\mathbf{x}$ of $\delta\Omega$ is defined as the distance $d(\mathbf{x}, Sk(\Omega))$ from $\mathbf{x}$ to the medial axis (where $d(.,.)$ is the Euclidean distance function).

Given the local feature size on the boundary, we need a canonical, as well as controllable way to extrapolate this function to the interior. Two conflicting constraints are desirable: we could try to minimize the number of total tets by forcing the inside of the object to have the largest tets possible; however, in order to maintain good shape quality, the mesh gradation (i.e., how fast tet sizes vary within a neighborhood) must remain bounded [Ruppert 1993]. We propose to recast the problem of finding an ideal sizing field to finding the *maximal K-Lipschitz function* that does not exceed $lfs(\mathbf{x})$ on $\delta\Omega$. The parameter K will control the gradation (0 being the uniform case) of the resulting field. As we prove in the Appendix A, the function:

$$\mu(\mathbf{x}) = \inf_{\mathbf{s}\in\partial\Omega} [K\ d(\mathbf{s},\mathbf{x}) + lfs(\mathbf{s})] \tag{9}$$

satisfies these requirements. Consequently, we used it in all the examples shown in the figures. Now that the theoretical aspects of our approach have been addressed, we describe in the next section the details of a concrete implementation of these ideas.

3 Algorithm

In this section, we go through the details of each step of the following pseudo-code which summarizes our approach:

```
Read the input boundary mesh ∂Ω
Setup Data Structure & Preprocessing
Compute sizing field μ
Generate initial sites x_i inside Ω
Do
    Construct Delaunay triangulation({x_i})
    Move sites x_i to their optimal positions x_i*
Until (convergence or stopping criterion)
Extract interior mesh
```

For efficiency as well as robustness, we opted to use the Computational Geometry Algorithms Library (CGAL [Fabri et al. 2000]) for the input mesh data structure, as well as for the 3D Delaunay triangulation using robust arithmetics.

3.1 Input Domain Boundary

Our algorithm takes as input an intersection free closed surface triangle mesh defining the domain boundary $\partial\Omega$. We have no restriction on the topology of the domain Ω: it may contain multiple connected components, or have multiple voids, or both (see Fig. 6).

3.2 Setup & Preprocessing

The vertices of the input surface mesh $\partial\Omega$ are inserted in a 3D Delaunay triangulation, to create what we call the *control mesh*. This control mesh is used by our algorithm to estimate the local feature size of $\partial\Omega$ as well as to answer inside/outside queries. For efficient inside/outside queries, we require that the control mesh contains all triangle facets of the input boundary $\partial\Omega$, guaranteeing that it is the *restricted Delaunay triangulation* of the input vertices [Cohen-Steiner et al. 2002]. This allows us to tag the corresponding faces of the control mesh and in turn its tets with inside/outside tags. To achieve these goals, $\partial\Omega$ is originally either enriched or remeshed using an isotropic surface meshing algorithm [Boissonnat and Oudot 2005].

Discrete Skeleton Once the control mesh has been generated, we extract its *poles* by selecting a subset of Voronoi vertices (i.e., circumcenters of tets) in the following manner. For each Delaunay vertex v we first select as a pole the farthest circumcenter from all incident tets. The vector formed by v and this pole is considered as the local normal estimate. We deduce a local tangent plane estimate in v, and two half-spaces bounded by this plane. Next we search in the half-space that does not contain the pole the farthest Voronoi vertex incident to the site. If it exists it is added as a pole too. We refer the reader to [Amenta and Bern 1998] for a more detailed description of this simple procedure. By definition, and assuming a sufficiently dense set of points sampled on the boundary, the resulting set of all these poles is a discrete approximation of the *skeleton* (or medial axis) of the domain boundary $\partial\Omega$ (see Fig. 7).

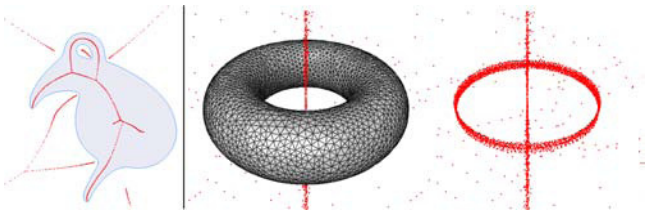


Figure 7: *Poles: The set of all poles (depicted as red dots) represents a discrete approximation of the* skeleton *of a 2D (left) or 3D (right) shape.*

Local Feature Size At each vertex of the boundary $\partial\Omega$, we approximate its local feature size *lfs* by measuring the distance to its closest pole (Fig. 8). To improve the efficiency of these queries (the set of poles is a dense point set for a complex boundary), we create a static kD-tree search data structure from the poles.

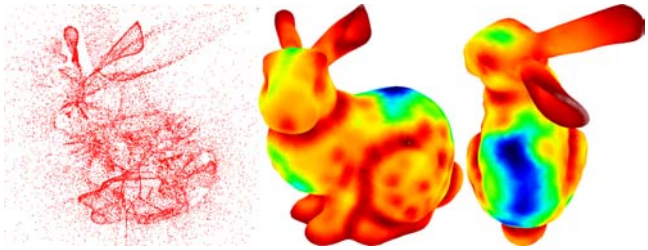


Figure 8: *Left: poles extracted from the Bunny model. Middle, right: the distance from each input vertex to the set of poles is the approximated* local feature size*, capturing both local thickness and curvature of the shape.*

Boundary Supersampling The input boundary is initially sampled with a large number of *quadrature samples* (used later on to find a good approximation of the surface). More precisely, three sets of quadrature samples are generated: on the boundary itself, on its sharp creases, and on its corners (for piecewise smooth domains—see, e.g., Fig. 16); this will allow us to both approximate the boundary and fit its features. Each quadrature sample stores its position $\mathbf{x}$ as well as a quadrature value, incorporating the area ds (for surface quadrature samples) or length dl (for feature quadrature samples) it covers. These values are set to $ds/\mu(x)^4$ and $dl/\mu(x)^3$ respectively, in order to conform to our mass density field [Du and Wang 2003]. Each corner sample is given an infinite density to guarantee that a vertex will be assigned there.

3.3 Fast Marching Construction of Sizing Field

Recall that a parameter K is used to adjust the sizing field according to the desired mesh gradation (see Section 2.3, and Figure 9 for illustration). We store the sizing field on a uniform grid bounding the domain. Each node of the grid must store the local sizing field value μ and an additional bit to specify whether this grid node lies inside or outside the domain as these grid nodes will be used to efficiently generate initial positions for the vertices of the mesh. However, computing μ in the interior of Ω would require the evaluation of a minimum over *each* vertex of $\partial\Omega$. To provide a faster grid

initialization we use a fast marching method on the uniform grid using the 6-neighborhood incidence relationship beginning with the grid cells that intersect $\partial\Omega$. We define a *candidate cell* $\mathbf{x}$ as a cell for which we have stored a temporary *buddy cell*, denoted hereafter $y(\mathbf{x})$. The latter is astride of the boundary, and has the property that $K\ d(\mathbf{x}, y(\mathbf{x})) + lfs(y(\mathbf{x}))$ is the current known minimum value of the sizing field $\mu(\mathbf{x})$. The candidate cells are maintained in a priority queue ordered by their current estimated value of μ. This queue is initialized with all grid cells that are neighbors of a grid cell intersecting $\partial\Omega$. At each step of the marching process we pop the candidate cell with minimum μ value out of the queue, set its final sizing field value to μ, and push other possible adjacent candidates in the queue with the same buddy cell. This fast marching method will thus propagate values of μ from the initial boundary to the inside of the domain. Note that this could introduce an approximation in the evaluation of the sizing field, since the boundary cell $c(\mathbf{x})$ such that $K\ d(\mathbf{x}, c(\mathbf{x})) + lfs(c(\mathbf{x}))$ is globally minimal might not be among the buddy cells of $\mathbf{x}$'s neighbors. We argue that the error is negligible. The reason is that the set of points $\mathbf{p}$ that have a same buddy cell $\mathbf{y}$ is star-shaped around $\mathbf{y}$. Indeed, on the line segment from $\mathbf{p}$ to $y(\mathbf{p})$, the function $\lambda(\mathbf{s}) = K\ d(\mathbf{s}, y(\mathbf{p})) + lfs(y(\mathbf{p}))$ decreases with speed K; as μ is K-Lipschitz, we have that $\lambda \leq \mu$, therefore $\lambda = \mu$ since μ is the minimum over all $\mathbf{y} \in \partial\Omega$, and finally $y(\mathbf{s}) = y(\mathbf{p})$. Hence, the first grid cell $\mathbf{q}$ met by the ray $\mathbf{p} - y(\mathbf{p})$ is most likely such that $y(\mathbf{q}) = y(\mathbf{p})$. One then has $\mu(\mathbf{q}) \leq \mu(\mathbf{p})$; thus, $\mu(\mathbf{q})$ must have been already computed by the time $\mathbf{p}$ is taken care of. This simple procedure enables an efficient and robust initialization of our sizing field grid.

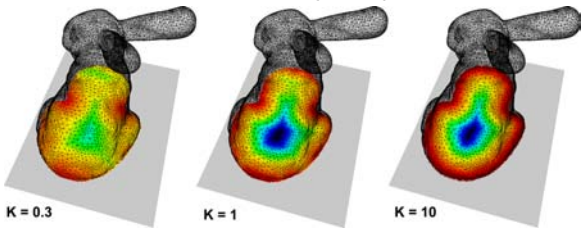


Figure 9: *Sizing fields computed for three increasing values for K. The smaller K, the smoother the grading. For large values, the ideal edge size is rapidly increasing as one moves away from the boundary.*

3.4 Initial Point Sampling

Given the potential complexity of the input boundary and the optimized 3D Delaunay triangulation in terms of geometry and topology (with possibly multiple connected components and holes), a good initialization of the tet mesh vertices is desirable. We "spread" the requested number of vertices throughout the domain while roughly matching the desired local density through error diffusion over the sizing field. This initial sampling proceeds in two passes. In order to calibrate the sampling so as to fit the vertex budget specified by the user, the first pass sums up the values $dv/\mu(\mathbf{x})^3$ for each interior node of the sizing field grid, where dv denotes the volume of the node and $\mathbf{x}$ its position. The second pass iterates over the same grid nodes in serpentine order, computing for each node its corresponding (floating point) number of initial vertices to lay down locally, quantizes this number to the nearest integer and diffuses the corresponding residual to its neighbors: this process is a straightforward extension of [Ostromoukhov 2001] to volumetric images. Although these placements do not guarantee any quality on the resulting Delaunay mesh, we achieve a local density of vertices consistent with the sizing field for a very low computational effort.

3.5 Energy Minimization

The energy minimization phase, alternating connectivity and geometry optimization, is the core of our algorithm. From the current vertex positions, the energy is minimized by computing the 3D Delaunay triangulating of these sites. For a given connectivity, the energy is further minimized by moving each *interior* vertex $\mathbf{x}_i$ to its optimal placement within its 1-ring (Eq. 8).

Boundary Vertices require a different treatment to provide adequate boundary conditions to our minimization, as well as a good, isotropically-sampled approximation of the domain boundary. A simple and practical solution is to use a variant of the constrained centroidal Voronoi tessellation approach (CCVT [Du et al. 2003]). First, we go over all the boundary quadrature samples $\mathbf{s}_i$ mentioned earlier. For each $\mathbf{s}_i$ we locate the nearest vertex in the current mesh (through a fast kD tree query), and accumulate at that vertex the quadrature value at $\mathbf{s}_i$ times the coordinates of $\mathbf{s}_i$. Subsequently, we focus on the vertices with a non-zero quadrature sum, since those are the boundary vertices that require a specific treatment. Their position update is straightforward: we move these boundary vertices to the average value they each have accumulated during the pass over all quadrature samples. This position provides, at low cost, a good approximation of the *centroid* of the intersection between the 3D Voronoi cell of the boundary vertex and the input boundary $\partial\Omega$. Note that we proceed similarly for the feature quadrature samples involved in the piecewise smooth case, where we have to fit sharp creases as well. As demonstrated in our results (see Fig. 10 for an illustration of several steps of optimization on a simple boundary), this simple procedure results in well-shaped triangles that fit the domain boundary, and whose size is in agreement with the sizing field.

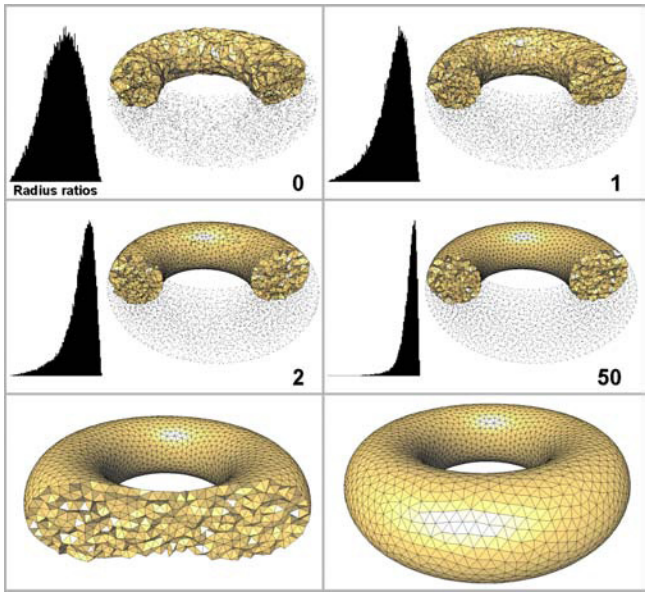


Figure 10: *Optimization steps for* 1000 *vertices in a torus.* $100K$ *quadrature samples were spread on the boundary, and the sizing field is constant to get a uniform mesh. Observe how the radius ratio distribution shifts towards* 1.

3.6 Accelerating Convergence

Although our quadratic energy minimization provides a powerful tool to design high-quality meshes, the convergence rate can be slow for large number of vertices. We have successfully experimented with the following practical shortcuts to get faster results.

Delaunay Refinement Direct minimization of more than $100,000$ vertices can be computationally expensive. Instead, we prefer starting the energy minimization with a much smaller number of vertices. Before it even reaches a minimum, we increase the number of vertices by adding a specified fraction (typically, 50%) of the vertices at a subset of the Voronoi centers of the current mesh. To select the latter subset, we sort all tets by decreasing size of circumradius divided by the local desired edge length (to know where refinement is most needed). After another round of minimization steps, we repeat this procedure, until the requested number of vertices is reached. The speed-up is considerable, while achieving the same final quality.

Selective Optimizations A straightforward improvement of vertex position update optimizes *only* the vertices adjacent to bad quality tets (say, with a radius ratio less than 0.3) and their immediate neighbors. Although no theoretical guarantees back up this trick, it works remarkably well in practice. We recommend switching to such a selective optimization once the full-blown optimization steps are relatively small in amplitude.

Boundary Vertex Jittering As expected from our energy, the inside tets are well shaped after minimization. However, because of the boundary constraints that we must satisfy, a few slivers can remain *adjacent* to boundary vertices. We have implemented a fast "jittering" of these points; in order to snap a sliver, we slightly move one of its adjacent boundary vertices in the local tangent plane. Similar in spirit to the more general procedure of sliver exudation [Cheng et al. 1999], but for the easier case of tets on boundaries, this jittering suffices to remove the remaining slivers.

Vertex Teleportation or Insertion We also recommend to sporadically remove the vertex with the smallest Voronoi cell w.r.t. the desired edge length (*i.e.*, in the densest region), and insert it at the centroid of the interior tet with the worst radius ratio. Such tunnelling of vertices is particularly useful when tight control over the worst element is required. If the vertex budget does not have to be maintained, one can directly add vertices inside the worst tets.

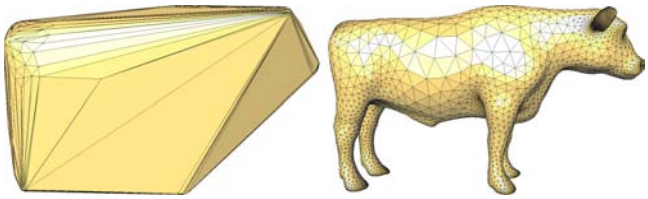


Figure 11: *Final Extraction: As the Delaunay triangulation covers the convex hull, the last stage of our algorithm must extract the inside tets.*

3.7 Final Mesh extraction

To produce the final mesh, we need to peel off the Delaunay tets of our mesh that are outside the domain (remember that a Delaunay mesh triangulates the convex hull of the vertices). A first approach is to consider the Delaunay triangulation restricted to the input domain, by tagging a tet as *outside* when its circumsphere center is located outside Ω. Similar in spirit to the Cocone algorithm [Amenta et al. 200], we consider instead the Delaunay triangulation restricted to a slightly *thicker* version of the input domain Ω. So for each tet initially classified outside, we compute the ratio between the distance d from the circumsphere center to the boundary $\partial\Omega$ and the circumradius; if this ratio is smaller than a predefined threshold (we used 0.4 in our experiments) we tag the tet inside.

4 Results and Discussions

The figures in this chapter illustrate the robustness and versatility of our technique: our implementation can handle large and/or complex domains of arbitrary topology in a matter of minutes. Although a visual inspection cannot provide a thorough assessment of our results, all the cutaway views as well as the radius ratio distributions that we obtained exhibit high quality tet shapes throughout the domain. In contrast to many other methods we *do not* a priori assign vertices to be either on the boundary or in the interior. It is the minimization procedure that will make them stick to boundary or not, driven by the sizing field and number of vertices required. This feature partially explains the quality of the results, since the mesh is not constrained to a given budget of boundary vertices. Also, our experience shows that global optimization of the connectivity through a Delaunay triangulation renders the results significantly better: this handling of the connectivity is possibly the sharpest departure from common approaches that perform local updates only.

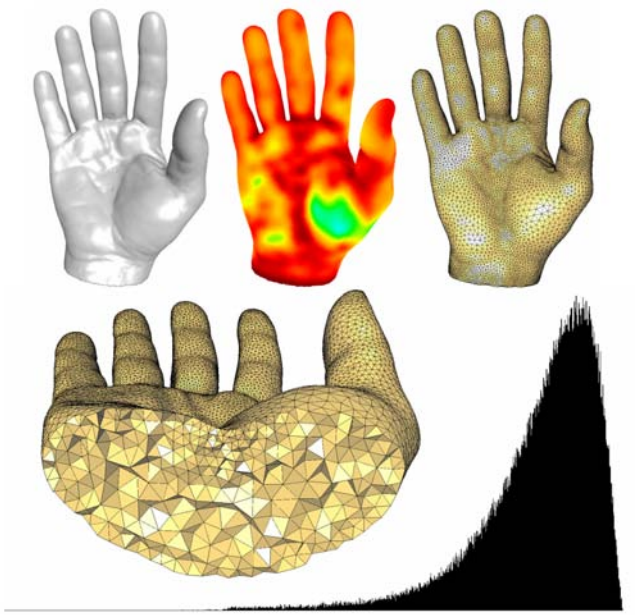


Figure 12: *Scanned hand: on this highly detailed mesh ($36K$ vertices, $174K$ tets, with color-coded* lfs*), mesh gradation is a must: a uniform mesh fine enough to capture the surface details would have millions of tets; Instead, our algorithm can reproduce all the fine surface features while using large tets inside the domain. Sizing field parameter: $K = 1$. As the radius ratio distribution shows, there are no degenerate tets. Worst radius ratio $= 0.29$, average radius ratio $= 0.86$, average dihedral angle = 70^o. Mean symmetric distance from input boundary: 0.024% of the bounding box.*

Results can be obtained in a matter of seconds or minutes. For instance, Fig. 10 was obtained in 16 seconds (for the 50 iterations, which include a Delaunay triangulation and the vertex position optimizations at *each* iteration) on a Pentium IV 3GHz. A more complex model, such as the hand in Fig. 12 requires on average 2.1 seconds per iteration, including Delaunay triangulation, boundary quadrature, and vertex updates for the $36K$ vertices. For good meshes, 10 to 20 iterations are sufficient, but we often increase this number to 50, and use the speed-ups described in Section 3.6 for a final high quality result. Although very few timings are available for previous optimization methods, we consider the time involved in our technique for mesh design practical. Notice that we can also deal with sharp features, as Figure 16 demonstrates—a full treatment treatment of such mechanical would however require a good Constrained Delaunay mesher.

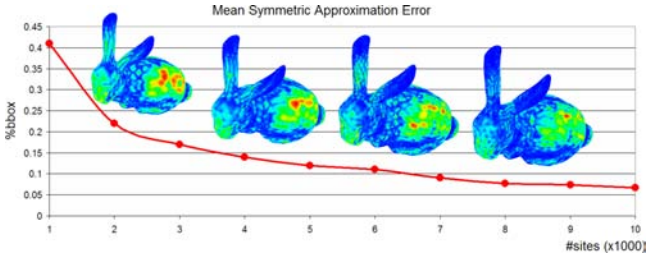


Figure 13: *Mean symmetric approximation error against the number of sites measured with Metro. The approximation error is expressed in percentage of the bounding box. The four bunnies shown correspond respectively to 2, 4, 6 and 8K sites, with their approximation error in false colors.*

Figure 13 demonstrates the quality of approximation of the boundary for the Bunny model, for increasing vertex budgets ranging from 1K to 10K vertices. We plot the mean symmetric ($\mathcal{L}^2$) distance in percentage of the bounding box against the number of sites, as well as a color-coded illustration of the approximation error.

Judging the quality of the results is, however, a difficult task. First,

Figure 14: *Gargoyle: a comparison with [Cutler et al. 2004] further demonstrates the excellent results of a variational approach. Our distribution in terms of radius ratios (one of the fairest measures [Shewchuk 2002a]) is far superior to a standard optimization technique (mesh courtesy B. Cutler).*

many (often contradictory) quality measures have been proposed over the years. Second, averages of radius ratios are often given, but they do not tell the whole story: many slivers can be present even when the average is high. Finally, we could not find or get tet meshes of usual CG models (such as the bunny) or of canonical shapes, aside from the results presented in [Cutler et al. 2004] (see Fig. 14 for comparison). As a consequence, we simply provide some relevant numbers of our typical results. For a torus mesh with $8.4K$ tets obtained in less than 1 min., the radius ratios are between 0.42 and 0.99 (average=0.88), and the dihedral angles between 20.86^o and 145.8^o (average=69.95^o). The hand in Fig. 12 has radius ratios from 0.29 to 0.99, aspect ratios from 0.22 to 1.13, and dihedral angles from 15^o to 157^o (see figure for more statistics).

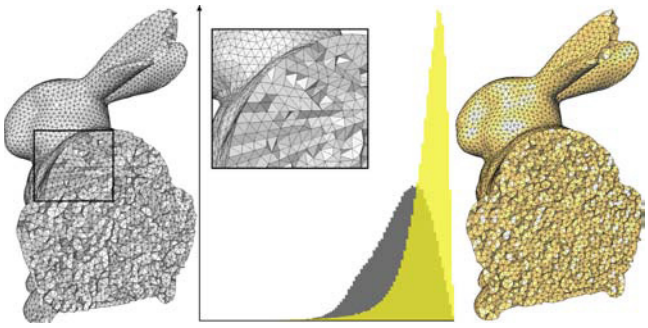


Figure 15: *Comparison of our method (right, obtained in 4 minutes) with the unit mesh approach [Frey and George 2000] (left, obtained in 12 seconds). To make a fair comparison we provide as input a uniform mesh and specify a uniform density for both methods. The distribution of radius ratios (middle) demonstrates the quality of our resulting tetrahedra.*

Finally, in Fig. 15 we compare our optimization technique with the unit-mesh approach [Frey and George 2000] used in commercial meshers. This technique has been applied on a high-quality uniform input boundary mesh of the bunny generated using the technique presented in [Surazhsky et al. 2003], resulting in 275K tets and 49K vertices in 12.5s (the number of vertices cannot be specified and results from the conforming of the boundary). As this unit-mesh approach splits long edges into smaller equal-length edges during the meshing process, the final mesh exhibits directional aliasing in the form of lines of tets, potentially detrimental to the simulation of isotropic phenomena. To provide a fair comparison, we used our technique with a uniform sizing field and the same number of final vertices. Our mesh was obtained in 4 minutes. Both distributions of radius ratios show no slivers; however, our approach results in better shaped tets overall, albeit at the price of a higher computational cost.

Limitations Our design choice to approximate the input boundary instead of conforming to it can also be seen as a limitation for certain applications. Additionally, we do not have theoretical bounds on the quality measures of the resulting tets. However, our results indicate that in practice, our minimization procedure generates well-shaped tets inside the domain, with better radius ratio distribution curves than any of the tet meshes we came across. Note that this high quality, most desirable for simulation purposes, naturally comes with higher computational cost than typical greedy (*e.g.* Delaunay refinement) methods.

5 Conclusions

We have introduced a novel approach to the construction of high-quality, isotropic tetrahedral meshes. Based on a sound variational principle, our technique provides a robust mesh design tool that can accommodate requirements on the final number of vertices and on the mesh gradation, for arbitrary domain complexity and topology. We demonstrated the scalability of our approach by meshing large, complex domains, even with sharp features. In future work, we wish to explore how to extend our approach to anisotropic meshing using not just a mass density, but a tensor field. Other boundary conditions for our optimization could also be studied.

Acknowledgments The authors wish to thank Peter Schröder as one of the instigators of this project. Many thanks to Alexandre Olivier-Mangon and George Drettakis for providing us with the torso model. Our gratitude also goes to Joe Warren, Sean Mauch, Peter Krysl, Fehmi Cirak and Tamer, Barbara Cutler, Steve Oudot, Sylvain Pion, and Andreas Fabri for precious help along the way. Sponsors include NSF (CARGO DMS-0221669 and DMS-0221666, CAREER CCR-0133983, and ITR DMS-0453145), DOE (DE-FG02-04ER25657), the EU Network of Excellence AIM@SHAPE (IST NoE No 506766), and Pixar.

References

AMENTA, N., AND BERN, M. 1998. Surface Reconstruction by Voronoi Filtering. In *Proc. of* 14^{th} *Symp. on Computational Geometry (SCG'98)*, 39–48.

AMENTA, N., CHOI, S., DEY, T., AND LEEKHAU, N. 200. A Simple Algorithm for Homeomorphic Surface Reconstruction. In *Proceedings of the Symposium on Computational geometry*, 213–222.

BOISSONNAT, J.-D., AND OUDOT, S. 2005. Provably Good Sampling and Meshing of Surfaces. *Graphical Models (special issue on Solid Modeling)*. To appear.

BOROUCHAKI, H., GEORGE, P., HECHT, F., LAUG, P., AND SALTEL, E. 1997. Delaunay mesh generation governed by metric specifications. Part 1 : Algorithms. *Finite Elements in Analysis and Design 25*, 61–83.

BOROUCHAKI, H., GEORGE, P., AND MOHAMMADI, B. 1997. Delaunay mesh generation governed by metric specifications. Part 2 : Application examples. *Finite Elements in Analysis and Design 25*, 85–109.

CAREY, G. F. 1997. *Computational Grids: Generation, Adaptation, and Solution Strategies*. Taylor & Francis eds.

CHEN, L., AND XU, J. 2004. "Optimal Delaunay triangulations". *Journal of Computational Mathematics 22*, 2, 299–308.

CHEN, L. 2004. Mesh smoothing schemes based on optimal Delaunay triangulations. In *Proceedings of 13th International Meshing Roundtable*, 109–120.

CHENG, S. W., AND DEY, T. K. 2002. Quality meshing with weighted Delaunay refinement. In *Proc.* 13^{th} *ACM-SIAM Sympos. Discrete Algorithms*, 137–146.

CHENG, S.-W., AND POON, S.-H. 2003. Graded conforming Delaunay tetrahedralization with bounded radius-edge ratio. In *Proc, of the* 14^{th} *ACM-SIAM Symposium on Discrete algorithms (SODA)*, 295–304.

CHENG, S.-W., DEY, T. K., EDELSBRUNNER, H., FACELLO, M. A., AND TENG, S.-H. 1999. Sliver Exudation. In *Proc. 15th ACM Symp. Comput. Geom.*, 1–13.

CHENG, S.-W., DEY, T. K., RAMOS, E., AND RAY, T. 2004. Quality Meshing for Polyhedra with Small Angles. In *Proc. of ACM Symp. on Comp. Geom.*, 290–299.

COHEN-STEINER, D., DE VERDIERE, E. C., AND YVINEC, M. 2002. Conforming Delaunay triangulations in 3D. In *Proc. of Symp. on Comp. Geom.*, 237–246.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational Shape Approximation. *ACM Trans. on Graphics (SIGGRAPH)*, 905–914.

CUTLER, B., DORSEY, J., AND MCMILLAN, L. 2004. Simplification and Improvement of Tetrahedral Models for Simulation. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 95–104.

DU, Q., AND WANG, D. 2003. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal on Numerical Methods in Engineering 56(9)*, 1355–1373.

DU, Q., GUNZBURGER, M., AND JU, L. 2003. Constrained Centroidal Voronoi Tessellations for Surfaces. *SIAM J. Sci. Comput. 24*, 5, 1488–1506.

EPPSTEIN, D., 2001. Global optimization of mesh quality. Tutorial at the 10^{th} Int. Meshing Roundtable, Newport Beach.

FABRI, A., GIEZEMAN, G.-J., KETTNER, L., SCHIRRA, S., AND SCHÖNHERR, S. 2000. On the Design of CGAL, a Computational Geometry Algorithms Library. *Softw. – Pract. Exp. 30*, 11, 1167–1202. www.cgal.org.

FREITAG, L., AND OLLIVIER-GOOCH, C. 1996. A comparison of Tetrahedral Mesh Improvement Techniques. In *Proc. of 6th Int. Meshing Roundtable*, 87—1000.

FREY, J. L., AND GEORGE, P. L. 2000. *Mesh Generation: Applications to Finite Elements*. Hermès, Paris.

HARDIN, D. P., AND SAFF, E. B. 2004. Discretizing Manifolds via Minimum Energy Points. *Notices of the AMS 51(10)*, 1186–1194.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. "Mesh Optimization". *ACM Trans. on Graphics (SIGGRAPH)*, 19–26.

KRYSL, P., AND ORTIZ, M. 2001. Variational Delaunay Approach to the Generation of Finite Element Meshes. *Int. J. for Num. Meth. in Eng. 50(7)*, 1681–1700.

LI, X.-Y., AND TENG, S.-H. 2001. Generate Sliver Free Three Dimensional Mesh. In *Proc.* 12^{th} *ACM-SIAM Sympos. Discrete Algorithms*.

LI, X.-Y., TENG, S.-H., AND UNGOR, A. 2000. "Biting: Advancing Front Meets Sphere Packing". *Int. J. on Num. Methods in Eng. 49*, 1, 61–81.

LLOYD, S. P. 1957. Least Squares Quantization in PCM's. Tech. rep., Bell Telephone Laboratories, Murray Hill, NJ.

MITCHELL, S., AND VAVASIS, S. 2000. Quality Mesh Generation in Higher Dimensions. *SIAM J. Sci. Comput. 29*, 1334–1370.

MOLINO, N., BRIDSON, R., TERAN, J., AND FEDKIW, R. 2003. A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra. In *Proceedings of the 12th International Meshing Roundtable*, 103–114.

OSTROMOUKHOV, V. 2001. A simple and efficient error-diffusion algorithm. In *Proceedings of ACM SIGGRAPH*, 567–572.

OWEN, S. J. 1998. A Survey of Unstructured Mesh Generation Technology. In *Proceedings of the 7th International Meshing Roundtable*, 239–267.

QUADROS, W. R., SHIMADA, K., AND OWEN, S. J., 2004. 3D Discrete Skeleton Generation by Wave Propagation on PR-Octree for Finite Element Mesh Sizing. *Poster*, Solid Modeling Conference.

RUPPERT, J. 1993. A New and Simple Algorithm for Quality 2-Dimensional Mesh Generation. In *Proc. of the* 4^{th} *ACM/SIAM Symp. on Disc. Algo. (SODA)*, 83–92.

SHEWCHUK, J. R. 1998. A Condition Guaranteeing the Existence of Higher-Dimensional Constrained Delaunay Triangulations. In *Proc.* 14^{th} *Annu. ACM Sympos. Comput. Geom.*, 76–85.

SHEWCHUK, J. R. 1998. Tetrahedral mesh generation by Delaunay refinement. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, 86–95.

SHEWCHUK, J. 2002. What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measure. In *Proc. of* 11^{th} *Int. Meshing Roundtable*, 115–126.

SHEWCHUK, J. R. 2002. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Computational Geometry: Theory and Applications 22*, 21–74.

SURAZHSKY, V., ALLIEZ, P., AND GOTSMAN, C. 2003. Isotropic Remeshing of Surfaces: a Local Parameterization Approach. In *Proc. of* 12^{th} *Int. Meshing Roundtable*.

TENG, S.-H., WONG, C. W., AND LEE, D. T. 2000. Unstructured Mesh Generation: Theory, Practice, and Perspectives. *International Journal Computational Geometry and Applications 10*, 3 (June), 227–266.

WARREN, J., SCHAEFER, S., HIRANI, A., AND DESBRUN, M., 2004. Barycentric Coordinates for Convex Sets. Preprint.

A Lipschitz Sizing Field

We wish to prove that the sizing field defined in Eq. (9) is both K-Lipschtiz and maximal. Because we want the function to be K-Lipschitz and agree with *lfs* on the boundary, one can easily show the following property:

$$\mu(x) \leq \inf_{s \in \partial\Omega} \left[K\ d(x,s) + \mathit{lfs}(s) \right].$$

We now need to show that the rhs is K-lipschitz and coincides with the *lfs* on the boundary: if so, the rhs will be the maximal sizing field we seek.

For $\mathbf{x} \in \Omega$, let

$$y(\mathbf{x}) = \text{argmin}_{\mathbf{s} \in \partial\Omega} \left[K\ d(\mathbf{x},\mathbf{s}) + \mathit{lfs}(\mathbf{s}) \right].$$

If $\mathbf{x}'$ is in Ω, we have by definition:

$$\begin{aligned} \mu(\mathbf{x}') &\leq K\ d(\mathbf{x}', y(\mathbf{x})) + \mathit{lfs}(y(\mathbf{x})) \\ &\leq K\ d(\mathbf{x}', \mathbf{x}) + K\ d(\mathbf{x}, y(\mathbf{x})) + \mathit{lfs}(y(\mathbf{x})) \\ &\leq K\ d(\mathbf{x}', \mathbf{x}) + \mu(\mathbf{x}) \end{aligned}$$

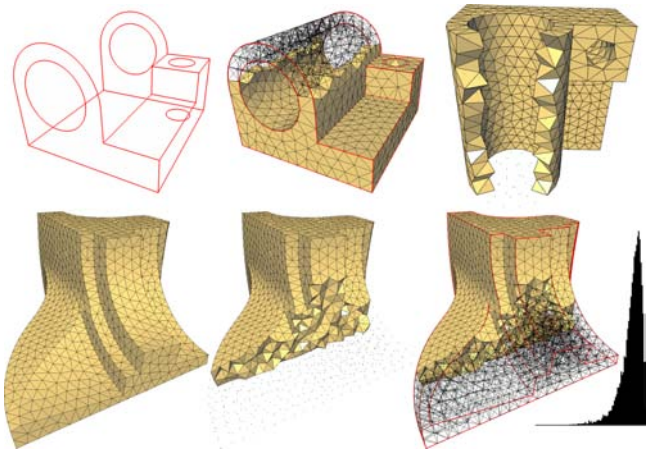
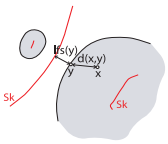


Figure 16: *Mechanical parts: Even in the presence of sharp features, our tet meshing algorithm exhibits excellent behavior for low or high vertex count while capturing features and corners remarkably well. We tagged edges as sharp features if their dihedral angles are more than* 20^o*—more sophisticated segmentation techniques could of course be used. (Top, middle) Joint model (*$1.2K$ *vertices); (Bottom) Fan disk model (*$3K$ *vertices, mean/max symmetric distance from input boundary: 0.021%/0.5% of the bounding box), along with its radius ratio distribution. Notice the good aspect ratio of both tets (see cutaway views) and surface triangles. A constant sizing field has been used for both models to obtain uniform tet meshes.*

which shows that μ is K-Lipschitz. Since *lfs* is 1-Lipschitz, we cannot hope that μ coincides with *lfs* on $\partial\Omega$ unless K is at least 1. If so, then for $\mathbf{x} \in \partial\Omega$ we have

$$K\ d(\mathbf{x},\mathbf{y}) + \mathit{lfs}(\mathbf{y}) \geq \mathit{lfs}(\mathbf{x})$$



for all $\mathbf{y} \in \partial\Omega$, with equality when $\mathbf{y} = \mathbf{x}$. Thus, μ does coincide with *lfs* on $\partial\Omega$. Note that when $K = 1$, $\mu(\mathbf{x})$ boils down to the length of the shortest path from x to the medial axis of $\partial\Omega$ while passing by a point on $\partial\Omega$. When K is less than 1, we get that $\mu(x)$ can be less than *lfs*(x) on the boundary due to the Lipschitz constraint; however, the gradation is respected and the boundary sampling will be better than what is necessary: it is therefore still a good choice of sizing field.

B Transforming the Energy E_{ODT}

Let us start with the definition:

$$E_{\text{ODT}} = ||f - f^{\text{primal}}_{\text{PWL}}||_{\mathcal{L}^1} = \sum_j \int_{T_j} |f - f^{\text{primal}}_{\text{PWL}}|. \qquad (10)$$

In the tet T_j with vertices $\mathbf{x}_i$ $i = 1 \ldots 4$, the error function can be expressed as a function of the barycentric coordinates $\lambda_i(\mathbf{x})$:

$$|f(\mathbf{x}) - f^{\text{primal}}_{\text{PWL}}(\mathbf{x})| = \sum_i \lambda_i(\mathbf{x})\mathbf{x}_i^2 - \mathbf{x}^2 = \sum_i \lambda_i(\mathbf{x})\,(\mathbf{x}_i - \mathbf{x})^2. \qquad (11)$$

Notice that Eq. (3) is easily derived from this last expression by plugging it into Eq. (10). Rewriting $\mathbf{x}_i - \mathbf{x}$ as $((\mathbf{x}_i - \mathbf{c}_j) + (\mathbf{c}_j - \mathbf{x}))$, where $\mathbf{c}_j$ is the circumcenter of T_j, and plugging it into Eq. (10), we get the following confirmation of Eq. (7):

$$E_{\text{ODT}} = \sum_j \int_{T_j} \left(R_j^2 - ||\mathbf{x} - \mathbf{c}_j||^2 \right) d\mathbf{x} = \sum_j \left(|T_j| R_j^2 - \int_{T_j} ||\mathbf{x} - \mathbf{c}_j||^2 d\mathbf{x} \right).$$

C Updates as Weighted Circumcenters

Notice that the energy E_{ODT} inside a tet T is always extremal at the circumcenter $\mathbf{c}_T$. As a consequence, the optimal position of a vertex that has only four neighbors is exactly *at* $\mathbf{c}_T$. Using Eq. (5) in this special case of a 1-ring in the shape of a tet $T = (\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r, \mathbf{x}_s)$, and taking the point $\mathbf{x}_i$ to be located on $\mathbf{x}_p$, we get:

$$\begin{aligned} \mathbf{c}_T = \mathbf{x}_p - \frac{1}{2\,|\Omega_i|} (\quad & \nabla_{\mathbf{x}_p} |T|) \left[\sum_{\mathbf{x}_k \in T} ||\mathbf{x}_p - \mathbf{x}_k||^2 \right] + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_r) \\ & + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_s) + \mathrm{F}(\mathbf{x}_p, \mathbf{x}_r, \mathbf{x}_s)) \end{aligned}$$

where the extra terms on the rhs only depend on each face of the tet. Applying this formula to an arbitrary 1-ring centered on $\mathbf{x}_p$, the face terms cancel each other if we sum the contributions from all the tets, simplifying the expression drastically, and resulting in Eq. (6).