Open access • Journal Article • DOI:10.1145/3197517.3201358

Curved optimal delaunay triangulation — Source link

Leman Feng, Pierre Alliez, Laurent Busé, Hervé Delingette ...+1 more authors

Institutions: École des ponts ParisTech, French Institute for Research in Computer Science and Automation, California Institute of Technology

Published on: 30 Jul 2018 - ACM Transactions on Graphics (ACMPUB27New York, NY, USA)

Topics: Delaunay triangulation and Polygon mesh

Curved Optimal Delaunay Triangulation

Leman Feng, Pierre Alliez, Laurent Busé, Hervé Delingette, Mathieu Desbrun

Curved Optimal Delaunay Triangulation

LEMAN FENG, Ecole des Ponts ParisTech
PIERRE ALLIEZ, Université Côte d'Azur, Inria
LAURENT BUSÉ, Université Côte d'Azur, Inria
HERVÉ DELINGETTE, Université Côte d'Azur, Inria
MATHIEU DESBRUN, Caltech

Meshes with curvilinear elements hold the appealing promise of enhanced geometric flexibility and higher-order numerical accuracy compared to their commonly-used straight-edge counterparts. However, the generation of curved meshes remains a computationally expensive endeavor with current meshing approaches: high-order parametric elements are notoriously difficult to conform to a given boundary geometry, and enforcing a smooth and non-degenerate Jacobian everywhere brings additional numerical difficulties to the meshing of complex domains. In this paper, we propose an extension of Optimal Delaunay Triangulations (ODT) to curved and graded isotropic meshes. By exploiting a continuum mechanics interpretation of ODT instead of the usual approximation theoretical foundations, we formulate a very robust geometry and topology optimization of Bézier meshes based on a new simple functional promoting isotropic and uniform Jacobians throughout the domain. We demonstrate that our resulting curved meshes can adapt to complex domains with high precision even for a small count of elements thanks to the added flexibility afforded by more control points and higher order basis functions.

CCS Concepts: • **Mathematics of computing → Mesh generation**;

Additional Key Words and Phrases: Higher-order meshing, Optimal Delaunay Triangulations, higher order finite elements, Bézier elements.

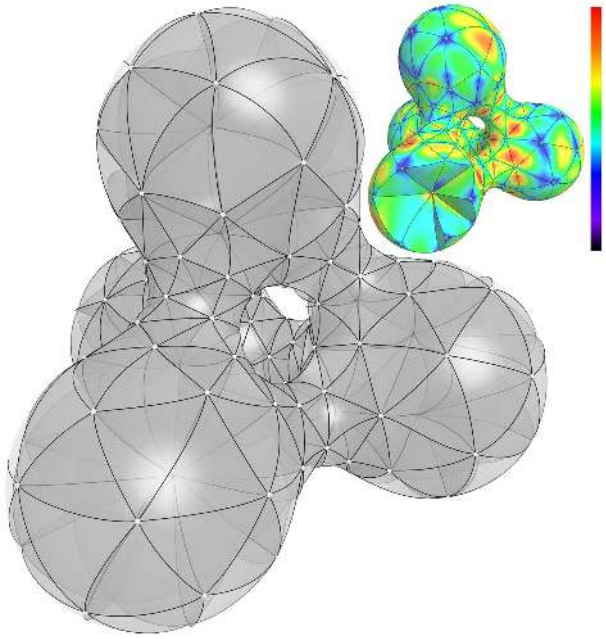


Fig. 1. **Curved Optimal Delaunay Triangulations:** A Bézier mesh (left, here with cubic patches) can capture a curved domain with orders of magnitude less elements for a given Hausdorff distance. Our extension of ODT to curved meshes results in a well-behaved Jacobian field (top: sizing-scaled determinant, with a few front elements removed to expose the interior).

1 INTRODUCTION

Simplicial meshing has found wide adoption in the graphics community over the years because of the prevalence of linear basis functions in computer animation. As the availability of computing power on commodity hardware continues to increase, so is the geometric complexity of the models simulated. However, properly capturing the boundaries of complex shapes with linear elements often requires an inordinate count of simplices for highly curved domains. Computational efficiency, instead, dictates the use of a low count of higher order piecewise polynomial elements: while using high-order elements—and thus, high-order basis functions—involves inevitable computational overhead, the drastic reduction of the number of elements needed to capture the domain boundary and the internal physical fields with a given accuracy lowers the total computational time appreciably compared to the linear case.

However, generating such a *curved* high order mesh for an arbitrary domain is significantly more difficult than its straight-edge analog: first, many staples of meshing (such as Delaunay triangulations or quality measures of mesh elements) only apply to straight-edge meshes; moreover, the use of higher-order polynomials to approximate the shape of a domain renders the numerical task more complex as it increases the occurrence of local minima in the energy landscape involved in the boundary fitting process and of local fold-overs of the polynomial map. As a result, most approaches for curved meshing limit themselves to deforming the boundary of a straight-edge mesh to improve the spatial matching of the boundary. However, this approach is limited and far from optimal as it largely relies on a good coarse linear mesh to start with, and leads to high deformation near the domain boundary, thus hampering subsequent numerical accuracy where it is sometimes most crucial (e.g., for boundary layers in fluid dynamics).

This paper extends the successful Optimal Delaunay Triangulation (ODT) approach for generating isotropic triangle and tetrahedron meshes [Alliez et al. 2005; Chen and Xu 2004] to now form Bézier meshes of arbitrary order. By exploiting the numerical properties of the original ODT technique and introducing a novel boundary

treatment, we show that our approach, which we coined *Curved Optimal Delaunay Triangulation*, generates curved meshes robustly and quite efficiently. The resulting meshes require far fewer elements to represent geometry or to guarantee accurate numerical solutions than their linear (straight-edge) counterparts.

1.1 Previous Work

We begin our exposition with a brief review of graphics and computational science efforts towards linear and high-order meshing.

From linear meshes... High quality meshing in graphics typically seeks the generation of non-degenerate meshes that offer good condition numbers for common discrete isotropic operators like the Laplacian [Cheng et al. 2012]. While Delaunay meshes with local refinements [Shewchuk 1998] have been shown most efficient at generating good isotropic simplicial meshes, *variational* approaches (that is, methods relying on energy minimization) can dramatically improve the quality of the resulting meshes. Most notably, the concept of Optimal Delaunay Triangulations (ODT) [Chen 2004; Chen and Xu 2004] anchored in functional approximation has garnered attention for providing what can be argued as the simplicial equivalent of Centroidal Voronoi Tessellations [Du et al. 1999; Liu et al. 2009]. Its implementation in 3D along with details on the sizing field computations and boundary handling was studied by Alliez et al. [2005], and a hybrid approach mixing Delaunay refinement and ODT optimization was later introduced [Tournois et al. 2009] to accelerate convergence and improve results. Recent improvements include an alternative boundary treatment [Gao et al. 2012] and various numerical accelerations [Chen and Holst 2011; Chen et al. 2014]. Today, the isotropic meshes obtained via ODT optimization consistently outperform other approaches (based on advancing front, bubble packing, etc) in terms of element quality (dihedral angles, aspect ratio, etc) for a given vertex budget.

To high-order basis functions... While simplicial meshes can be used for linear basis functions in the context of Finite Element Analysis, they can also accommodate the use of higher order basis functions which have been in high demand for simulation in computational fluid dynamics (CFD) and solid mechanics. Increasing the polynomial degree of basis functions within each element (classically referred to as p-refinement [Babuska et al. 1981]) allows for increased accuracy and faster convergence, often at a fraction of the computational cost of simply refining the simplicial mesh (h-refinement). Scalar fields are no longer just defined through values at nodes: more degrees of freedom per elements are now available, depending on the choice of basis functions and their polynomial orders. The accuracy gains brought by higher order basis in tetrahedral finite-element simulation were demonstrated in graphics by, e.g., [Roth et al. 1998] and [Weber et al. 2011]. However, the accuracy of finite element solutions is often strongly influenced by how well the geometry of the domain is approximated, which limits the applicability of straight-edge meshes in practice.

To high-order meshing. Coarse straight-edge meshes conform poorly to curved domain boundaries, impairing the correct numerical imposition of boundary conditions. Modern simulation techniques, in particular Isogeometric Analysis (IGA), define the geometry of the domain with the same high-order basis functions utilized for their Finite Element computations; that is, they use "high-order meshes" in the sense that the geometric description of the domains is a manifold assembly of *curved* isoparametric elements that are each defined through a geometric mapping with respect to a canonical simplex. However, degree elevation can be numerically difficult to handle near highly curved geometric boundaries: conforming to the shape of a boundary with piecewise higher order polynomials can easily create locally inverted mappings or severe distortion which precipitously decrease the accuracy of subsequent computations. Some graphics applications (notably, [Mezger et al. 2008] and [Suwelack et al. 2013]) use higher order meshes just to offer a coarse embedding of a fine geometry and accelerate computations; in this case, the presence of large Jacobians near the boundary is not necessarily an important issue. In practice, though, generating high-order meshes is typically accomplished by first constructing a straight-edge mesh that is subsequently curved and often regularized in case of invalidity (non-injectivity) of the geometric mapping. Regularization is usually achieved either through smoothing or morphing [Karman et al. 2016; Ruiz-Girones et al. 2017] (the deformation on the boundary is propagated onto the inside of the domain), or through optimization of control points and topological changes to increase the smallest value of the local determinant of the Jacobian of the map [Cardoze et al. 2004; Luo et al. 2002]. Global mesh optimization based on the extended notion of quality measures for curved elements has also been proposed: one can extend the element-based distortion measure by integrating a Jacobian-based distortion measure on the whole physical curved element [Gargallo-Peiró et al. 2013]. Distortion measures are often based on the Jacobian determinant, and include the ratio of minimum to maximum determinants in each element, or the ratio of the minimum determinant to the determinant of the corresponding straight-edge element [Johnen et al. 2013]. However, these functionals can be very high order even for moderately high order elements; Geuzaine et al. [2015] thus proposed a technique to compute provable bounds on the Jacobian determinant per element to check the mesh validity efficiently, and formulated a simpler functional to untangle invalid parts with a log barrier to penalize small determinants. Hierarchical methods proceed to curve edges first, then faces, then elements [Ziel et al. 2017] although behavior on complex 3D domains has not been demonstrated. Arguably, the most successful family of curved meshing methods rely on a mechanical analogy: these optimization-based methods use an initial straight-edge mesh as a reference domain and minimize a non-linear distortion using FEM between the reference domain and the parametric elements, thus effectively computing an *elastostatic* solution where the distortion they target defines a potential energy of deformation [Abgrall et al. 2012; Johnen et al. 2013]. However, the non-linearity of the potential is often an obstacle to finding a good minimum, and starting from a straight-edge mesh of the domain unnecessarily adds distortion if this mesh is not of high quality. Note that this approach was actually used in graphics, in the context of simulation by [Bargteil and Cohen 2014]: the authors suggested to use a rest pose of an elasticity simulation to be the end result of a previous simulation with a straight-edge tetrahedral mesh. However, they used a linear elastic model for simulation purposes, whereas high-quality meshing typically requires non-linear models [Johnen et al. 2013; Persson and Peraire 2009].

1.2 Contributions

In this paper, we revisit the ODT framework and extend it to provide both a better handling of the boundaries and new foundations for *curved meshing*. We show that the measure of element distortion underlying the ODT approach can be reexpressed as a particularly-simple potential energy whose minimization amounts to an equidistribution of the gradient of the deformation field, thus regularizing simultaneously the size and shape of all simplicial elements. After formulating a non-shrinking traction to favor uniform and isotropic elements at the boundary, we show that this interpretation of ODT applies nearly *as is* for curved meshes made of Bézier simplices. The resulting "Curved Optimal Delaunay Triangulations" provide coarse geometric descriptions of arbitrary 2D or 3D domains with a much improved fit to the domain boundary due to their piecewise polynomial nature, see Figs. 1 and 11. Moreover, our construction naturally promotes smoothness of the gradient of the induced geometric map inside and across elements, thus offering high-quality curved meshes ready to use in high-order finite element methods.

2 REVISITING OPTIMAL DELAUNAY TRIANGULATION

Before describing our extension of ODT to high-order meshes, we first review the foundations behind ODT for straight-edge simplicial meshes in $\mathbb{R}^d$ ($d = 2, 3$). We show that they can be neatly rewritten in the context of elastostatics, which provides us with a new approach to handle boundary conditions for improved isotropic simplicial meshing. This novel interpretation will, in later sections, render the extension to high-order meshes quite straightforward.

2.1 Primer on ODT

The quest for high quality simplicial meshes has sparked advances in mesh optimization through local vertex relocation to optimize a chosen notion of mesh quality [Amenta et al. 1999], local topological operations [Cheng et al. 2000], or both [Freitag and Ollivier-Gooch 1997]. Within the large body of work in mesh optimization for 2D and 3D unstructured triangulations, the Optimal Delaunay Triangulation approach (ODT for short) stands out by casting both geometric and topological mesh improvement as a single, unified optimization [Chen and Xu 2004].

Approximation-theoretical foundations. Exploiting functional approximation theory for piecewise linear interpolants, ODT finds a simplicial mesh in dimension d that minimizes in $\mathbb{R}^{d+1}$ the volume between the height field of the function $f : \mathbf{x} \to \|\mathbf{x}\|^2$ and the piecewise linear (PWL) interpolation of the mesh vertices lifted onto the height field; that is, the mesh connectivity and vertex positions are updated so as to minimize the energy E_{ODT}:

$$E_{\text{ODT}} = \|f - f_{\text{PWL}}\|_{L^1}^2 = \int_\Omega \left(f_{\text{PWL}}(\mathbf{x}) - f(\mathbf{x})\right)\, d\mathbf{x}. \tag{1}$$

The minimum of this energy is known to be asymptotically achieved when the mesh elements are isotropic [Nadler 1986]. Such theoretical grounding has far reaching practical implications: optimal connectivity, seemingly a high dimensional combinatorial problem, results from a simple numerical minimization akin to a convex hull algorithm, for which many robust, off-the-shelf tools exist. Moreover, various closed-form expressions of this geometric definition have been proposed that allow to evaluate efficiently the energy's gradient with respect to mesh vertices. In particular, if one denotes Ω_i as the one-ring neighborhood of vertex $\mathbf{x}_i$, we can rewrite the energy as a simple weighted sum plus a constant that solely depends on the shape of the domain Ω [Chen and Xu 2004]:

$$E_{\text{ODT}} = \frac{1}{d+1} \sum_{i=1..|V|} \|\mathbf{x}_i\|^2 |\Omega_i| - \int_\Omega \|\mathbf{x}\|^2 d\mathbf{x}. \tag{2}$$

A simple expression of the energy restricted to a single simplex τ was also formulated in [Chen and Holst 2011], containing no extra terms that depend on the domain:

$$E_{\text{ODT}}\Big|_\tau = \frac{|\tau|}{(d+1)(d+2)} \sum_{i,j=1,i<j}^{d+1} \|\mathbf{x}_i - \mathbf{x}_j\|^2, \tag{3}$$

meaning that the contribution of a simplex to E_{ODT} is proportional to its volume $|\tau|$ times the sum of all of its squared edge lengths.

Minimization of E_{ODT}. The minimization involved in finding an optimal ODT mesh in connectivity and vertex positions is particularly simple: starting from an arbitrary mesh of Ω, one alternatively updates mesh connectivity for fixed positions, then vertex positions for a fixed connectivity [Alliez et al. 2005]. Vertex optimization involves minimizing a quadratic energy for inner vertices with respect to each vertex position, while connectivity optimization corresponds to finding the Delaunay triangulation of the current vertex positions, leading to a steady decrease of the energy after each iteration. The effect on the mesh is very visible: all elements quickly tend to become equally sized and equally shaped, as predicted by approximation theory. Faster convergence can also be obtained using variants of Newton's method, where the energy Hessian is approximated through either previous values of the gradient [Chen et al. 2014] or a graph-Laplacian [Chen and Holst 2011].

Controlling mesh density. Another property of ODT making it particularly appropriate for isotropic meshing is that its formulation can be trivially altered to handle arbitrary mesh density: given a positive scalar field $\rho : \Omega \to \mathbb{R}^+$, we can modulate the L^1 norm in the definition of E_{ODT} in Eq. (1) by substituting a modulated local volume form $\rho(\mathbf{x})\ d\mathbf{x}$ for the original $d\mathbf{x}$ as introduced in [Chen and Xu 2004] and exploited in [Chen et al. 2014]. Modifying the volume form does not affect the isotropic properties of formulation, but the resulting minimizer of the ODT energy now has elements with edges proportional to $\rho^{-1/(d+2)}$. A given sizing field h over the domain Ω can thus be imposed by picking $\rho(\mathbf{x}) \propto h^{-(d+2)}(\mathbf{x})$.

Non-shrinking boundary conditions. As oftentimes for variational problems, setting proper boundary conditions is crucial to the success of ODT: like Eq. (3) clearly indicates, minimizing E_{ODT} without any constraints on the domain will make all the vertices of the mesh collapse to a point. Boundary conditions must thus be set to prevent shrinkage. Dirichlet conditions using a fixed sampling of the boundary of Ω produces nicely-shaped tetrahedra throughout the domain, but generates many degenerate elements near $\partial\Omega$ (in particular, "slivers" in 3D, i.e., almost flat tets with nearly equal edge lengths) as boundary vertices are not optimized, preventing the formation of equilateral elements. Reprojecting boundary vertices [Alliez et al. 2005] or letting them slide along the boundary by only taking the tangential part of the ODT gradient [Chen and Holst 2011; Gao

et al. 2012] improves the resulting meshes, but leaves degenerate elements as vertices that reach the boundary tend to stay on the boundary. The work of [Tournois et al. 2009] exploits a degree of freedom in the boundary terms of the update of a vertex to enforce a "circumsphere property" on $\partial\Omega$. This so-called *Natural ODT* boundary update guarantees that if ODT converges, it will form a mesh where boundary edges have nearly all the same lengths, further reducing the number of slivers. However, the precise boundary condition that this update corresponds to remains unclear, and as a consequence, how to modify this update to handle varying mesh density was not properly addressed either.

Discussion. While we will not explore anisotropy in this paper, we note that extensions of ODT to straight-edge anisotropic meshes have also been provided in [Chen and Xu 2004] and put into practice for flat and curved domains in [Boissonnat et al. 2006; Fu et al. 2014; Loseille and Alauzet 2009]. An interesting duality between graded polytopal tessellations and graded anisotropic simplicial complexes was also presented in [Budninskiy et al. 2016].

2.2 ODT as Elastostatics

We now reexpress the ODT energy in the context of continuum mechanics. This physical analogy will help us define a proper boundary condition (corresponding to a load on the boundary to render the elements isotropic there). This novel interpretation will facilitate the extension of ODT to high-order meshes later on as well.

Unit reference mesh. For any given d-dimensional simplicial complex $\mathcal{T}$, one can define a *unit reference simplicial complex* $\overline{\mathcal{T}}$ where both $\mathcal{T}$ and $\overline{\mathcal{T}}$ share the same connectivity, but each element of $\overline{\mathcal{T}}$ is regular with all of its edges of length 1. Note that this reference mesh is trivially manifold, but cannot be embedded in d dimensions: except in the rare case of 2D triangulation with only valence-6 vertices, such a "unit" mesh can only be embedded in a Euclidean space of dimension $D > d$ (and D is finite due to Nash's embedding theorem) since the generalized defect angles at vertices and edges are not zero. However, its exact embedding has no influence on our construction: this construction is only conceptual. Its purpose is to define a clear intrinsic reference d-manifold so that our target "physical" mesh in $\mathbb{R}^d$ can be seen as a *minimal deformation* from this ideal (but unrealizable in $\mathbb{R}^d$) reference domain by a map φ as in classical elastostatics, with:

$$\mathbb{R}^D \supset \overline{\mathcal{T}} \xrightarrow{\ \varphi\ } \mathcal{T} \subset \mathbb{R}^d$$

Our use of a reference domain with unit elements is quite different from any of the other methods anchored in continuum mechanics such as [Abgrall et al. 2012; Bargteil and Cohen 2014; Johnen et al. 2013; Persson and Peraire 2009] as they all use non-unit straight-edge triangulations as reference, which unnecessarily biases the notion of isotropy based on the quality of the tetrahedron mesh they start from. Instead, our interpretation phrases the optimization as a minimization of distortion with respect to a perfect mesh.

Jacobian-based expression of ODT. For the original straight-edge case of ODT, the mapping φ is piecewise linear, mapping each unit simplex $\bar{\tau}_i$ from $\overline{\mathcal{T}}$ to its associated physical simplex τ_i in $\mathcal{T}$. The Jacobian $\nabla\varphi$ of this map is thus piecewise constant: let's denote $\mathbf{J}_i$ this Jacobian from $\bar{\tau}_i$ to τ_i. We now present a simple, but important lemma that re-expresses the contribution of an element to ODT.

Lemma 2.1. *Using the linear map φ between $\bar{\tau}_i$ and τ_i, one has:*

$$E_{ODT}\Big|_{\tau_i} = \frac{|\bar{\tau}_i|}{2(d+2)} \det(\mathbf{J}_i)\ \|\mathbf{J}_i\|_{\mathrm{F}}^2,$$

where $\|\mathbf{J}_i\|_{\mathrm{F}}^2 = \mathrm{trace}\,(\mathbf{J}_i\mathbf{J}_i^t)$, and det *is the determinant.*

Proof. The squared Frobenius norm $\|\mathbf{J}_i\|_{\mathrm{F}}^2$ of $\mathbf{J}_i$ times the volume of $\bar{\tau}_i$ is nothing but the integral of the Dirichlet energy of φ over $\bar{\tau}_i$. From [Meyer et al. 2003; Pinkall and Polthier 1993], we know that the Dirichlet energy in 2D and 3D is a linear combination of squared edge lengths of τ_i:

$$\int_{\bar{\tau}_i} \|\nabla\varphi\|_{\mathrm{F}}^2 = \sum_{i,j=1..|V|} \bar{\alpha}_{ij}\ \|\mathbf{x}_i - \mathbf{x}_j\|^2,$$

where the coefficients $\bar{\alpha}_{ij}$ depend on geometric measures of $\bar{\tau}_i$ (half the cotangent of the tip angle in 2D [Pinkall and Polthier 1993], and one sixth of the cotangent of the dihedral angle times the associated edge length in 3D [Meyer et al. 2003]; more generally, they are the stiffness matrix coefficients for the Poisson problem). When $\bar{\tau}_i$ is a regular d-simplex, these coefficients are all equal to $2/(d+1)$. Moreover, we know that $\det(\mathbf{J}_i) = |\tau_i|/|\bar{\tau}_i|$ as the map is piecewise linear. We thus conclude that the expression of E_{ODT} restricted to τ_i as given in Eq. (3) is proportional to $\det(\mathbf{J}_i)\ \|\mathbf{J}_i\|_{\mathrm{F}}^2$, with a coefficient of proportionality equal to $\frac{1}{2}|\bar{\tau}_i|/(d+2)$ (for completeness, the volume of a d-dimensional unit and regular simplex is $|\bar{\tau}_i| = (d+1)^{1/2}/[2^{d/2}d!]$). Remark that this final expression is translation invariant (because of the gradient of φ) and rotation invariant ($\|\mathbf{J}\|_{\mathrm{F}}^2 = \|\mathbf{R}\mathbf{J}\|_{\mathrm{F}}^2$ for any rotation matrix $\mathbf{R}$), justifying a posteriori our claim that the precise embedding of the unit reference mesh does not matter. □

ODT energy as potential energy. Based on the previous lemma, we directly deduce that the ODT energy can be seen as a potential energy density W_{ODT} over the reference configuration $\overline{\mathcal{T}}$ in the context of hyperelasticity, expressed as:

$$W_{\mathrm{ODT}}(\mathbf{J}) = \frac{1}{2(d+2)} \det(\mathbf{J})\ \|\mathbf{J}\|_{\mathrm{F}}^2. \tag{4}$$

With this expression, the potential energy of a given physical mesh is exactly equal to E_{ODT}. A potential energy density W induces an internal (Cauchy) stress tensor over $\overline{\mathcal{T}}$ of the form [Ogden 1997]:

$$\boldsymbol{\sigma} = \frac{1}{\det(\mathbf{J})} \frac{\partial W}{\partial \mathbf{J}} \cdot \mathbf{J}^t.$$

Since $\partial\|\mathbf{J}\|_{\mathrm{F}}^2/\partial\mathbf{J} = 2\mathbf{J}$ and $\partial\det(\mathbf{J})/\partial\mathbf{J} = \det(\mathbf{J})\mathbf{J}^{-t}$, we get a very simple expression for the elementwise stress tensor in ODT:

Theorem 2.2. *The Cauchy stress tensor $\boldsymbol{\sigma}_{ODT}$ is expressed as:*

$$\boldsymbol{\sigma}_{ODT}\Big|_{\bar{\tau}_i} = \frac{1}{d+2}\left[\mathbf{J}_i\mathbf{J}_i^t + \frac{1}{2}\|\mathbf{J}_i\|_{\mathrm{F}}^2\ \mathbf{Id}\right],$$

where **Id** *denotes the $d \times d$ identity matrix.*

As a consequence, an optimal ODT mesh satisfies $\mathbf{div}(\boldsymbol{\sigma}_{\mathrm{ODT}}) = \mathbf{0}$ everywhere (in the weak sense of FEM), as it is the equilibrium equation for the equivalent elastostatics problem we just proved.

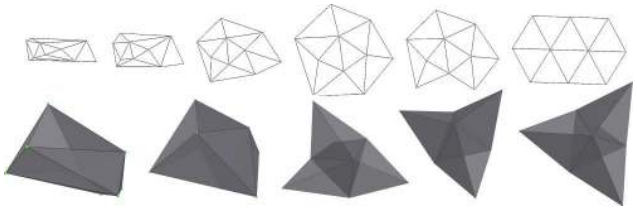


Fig. 2. **Non-shrinking ODT.** Starting from a random triangulation (top: 2D; bottom: 3D), minimizing the ODT energy with our non-shrinking boundary loads naturally converges (left to right) to a mesh with isotropic elements.

Connectivity of unit reference mesh. Since we showed that the ODT energy is in fact the integral of a potential energy density, we can also immediately conclude that the minimal potential energy *for a given set of vertex positions* is attained for the unit reference mesh that corresponds to the connectivity of the Delaunay triangulation of the vertices in physical space. As the reference mesh is purely conceptual, no computations besides finding the Delaunay triangulation is required: it simply means that our elastostatics problem consists in finding *both* the connectivity of the reference configuration *and* the deformed physical mesh that minimizes the total potential energy of the deformation, for given boundary constraints based on the domain Ω. We will formally write this variational principle after we discuss boundary conditions.

2.3 Non-shrinking boundary conditions

Dealing properly with boundaries is key to the success of ODT. Our elastostatics rewriting of the ODT energy suggests using boundary forces, or loads, to prevent the degeneracy of the minimization. The simplest constraint is to add a boundary force density along the boundary equal to $\boldsymbol{\sigma}_{\text{ODT}}\ \mathbf{n}$; in this case, boundary terms will exactly compensate for the forces induced by the internal stress, preventing the boundary vertices from moving: this choice corresponds to Dirichlet boundary conditions. As previously discussed, such a treatment is not quite useful in our case: we wish instead to let vertices move around to generate boundary elements that are as isotropic as possible. This leads us to propose another expression: we can compensate the *isotropic part* of the ODT stress tensor only, thus leaving the deviatoric part (containing shearing forces) active near the boundary: with the corresponding boundary forces, the elastostatics problem will find a solution where stress at the boundary is purely isotropic, i.e, elements are under equal compression/tension in all directions. Thus, at equilibrium, boundary elements will be *isotropic*. In order to impose such a traction, we directly apply the following boundary force field $\mathbf{f}_{\text{bdry}}$ on a boundary element τ_i in $\mathcal{T}$ (with the outwards unit normal denoted $\mathbf{n}$):

$$\mathbf{f}_{\text{bdry}} = \frac{1}{d}\,\text{trace}(\boldsymbol{\sigma}_{\text{ODT}})\ \mathbf{n} = \frac{1}{2\,d}\|\mathbf{J}_i\|_{\text{F}}^2\ \mathbf{n}, \tag{5}$$

since the isotropic part of $\boldsymbol{\sigma}$ is $\text{trace}(\boldsymbol{\sigma})\ \mathbf{Id}/d$ and $\text{trace}(\mathbf{Id}) = d$. As Fig. 2 demonstrates, this simple boundary condition added within the geometry updates unfolds a very uneven mesh into an assembly of perfect equilateral simplices, quickly and in a stable manner.

2.4 Discussion

Revisiting the original ODT energy through elastostatics provides some key advantages, as we review briefly to conclude this section.

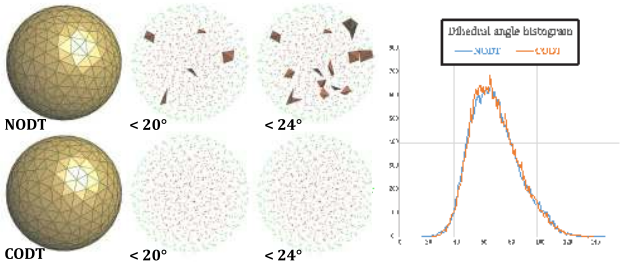


Fig. 3. **Slivers.** Given a spherical domain and 1,000 vertices, Natural ODT [Tournois et al. 2009] (top) achieves a minimum dihedral angle of 17.25° and a maximum of 149°, with 5 tets with angles below 20° (top); our new isotropic-stress boundary treatment (bottom) brings the minimum angle to 24.75° and the maximum to 141° instead.

Variational formulation. First, we can now formulate the exact elastostatics problem that our modified ODT solves:

$$(\overline{\mathcal{T}}^*, \varphi^*) = \underset{\overline{\mathcal{T}},\varphi}{\arg\min} \int_{\overline{\mathcal{T}}} W_{\text{ODT}}(\nabla\varphi(\bar{\mathbf{x}}))\ d\bar{\mathbf{x}}\ \ \text{s.t.}\ \begin{cases} \boldsymbol{\sigma}\Big|_{\partial\mathcal{T}} = \dfrac{1}{2\,d}\|\nabla\varphi\|_{\text{F}}^2\ \mathbf{Id} \\ \text{Dist}\big[\varphi(\overline{\mathcal{T}}), \Omega\big] \le \epsilon, \end{cases}$$

where $\text{Dist}[\cdot,\cdot]$ measures how far two subsets of $\mathbb{R}^d$ are from each other. In practice, this last condition can be enforced by adding a boundary fitting force field that attracts boundary elements of $\mathcal{T}$ towards $\partial\Omega$. Such a force can be derived from, e.g., the symmetric Hausdorff distance between mesh boundary and domain boundary, or simply from the volume in between the two [Alliez et al. 1999]. This attractive force field then acts as a penalty to enforce a close match between $\partial\mathcal{T}$ and $\partial\Omega$, thus enforcing the last constraint. We will provide a simpler shortest-distance based force field in Sec. 3.5 when we treat the more general case of Bézier meshes. Our novel treatment of boundary elements brings much added robustness to the ODT procedure : the constant reprojection on $\partial\Omega$ used in Natural ODT slows down convergence, introducing constant jittering of boundary vertices; in sharp contrast, the CODT treatment of boundaries does converge reliably in practice, always reaching a high quality mesh. Even if we compare to the (non-converged) best result of natural ODT after many iterations, our new approach converges to a lower number of slivers as illustrated in Fig. 3.

Interpretation and other potential energies. A posteriori, expressing ODT as a minimization of the deformation of a map between an ideally isotropic mesh and the current embedding of a mesh in $\mathbb{R}^d$ raises the question of why this particular choice of potential energy is special. We first note that the potential energy that ODT corresponds to is a local rescaling of the usual Dirichlet energy density: without the volume $|\tau|$ in Eq. (3), we would be left with only the Dirichlet energy, enforcing conformality of the map but *not* size uniformity. Moreover, one could also argue that using a geometric energy density of the form $\min_{\mathbf{R}\in\text{SO}(d)} \|\mathbf{J} - \lambda\mathbf{R}\|_{\text{F}}^2$ in the spirit of what is advocated in [Chao et al. 2010] for elastic simulation would seem like a perfect substitute: it would force each element to be isometric (up to a global scale λ) to its reference element. In fact, the potential energies of [Johnen et al. 2013; Persson and Peraire 2009] (or more generally, any energy that derives from shape functionals through density integration [Gargallo-Peiró et al. 2013])

target the same ideal: having a Jacobian as close to constant and isotropic as possible. However, two important differences make these other choices of energy less attractive. First, finding the optimal connectivity of these energies for given positions of the physical mesh nodes is not known, which will thus require local and costly exploration of the huge combinatorial space of all connectivities; instead, the ODT energy is known to be minimized for a Delaunay connectivity [Chen and Xu 2004], which is efficiently computed by a number of existing libraries. Second, solving for the optimal position of vertices given a fixed connectivity requires a non-linear solve, and the minimization of high-order energies is evidently slower and more prone to local extrema than ODT, for which the optimal positions are found through a sparse linear system. One can thus consider ODT as the *d-D continuum equivalent of 1-D zero-rest-length springs that regularize simultaneously the size and shape of all simplices*: the ODT energy goes to zero when the volume of simplex goes to zero, whereas previous elastic energies had a non-zero volume at rest state. In fact, the ODT energy is nothing but the (integrated) L_2 norm of the Jacobian over the physical domain, so its minimization will tend to equidistribute its singular values. This simple energy is thus a more effective solution to target a uniform, isotropic Jacobian field than springs with unit rest lengths, as they would require non-linear solves. Note also that other quality measures [Knupp 2001; Shewchuk 2002] could be used as substitute potential energies to enforce good mesh elements, but none can be simpler (i.e., lower order in vertex positions) than ODT.

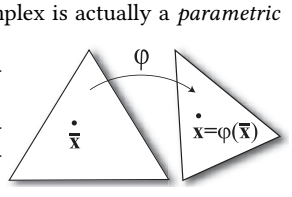
3 CURVED OPTIMAL DELAUNAY TRIANGULATION

We are ready to delve into our extension of ODT for high-order meshes. We focus on Bézier meshes in dimension d (for $d=2$ or $d=3$), made out of Bézier simplices of order n. This choice of parametric elements has commonly been used in graphics, as they require fewer points (thus less memory) to represent curved domains and have much better continuity properties than linear elements.

3.1 Primer on Bézier Meshes

While meshes made out of Bézier simplices (sometimes referred to as Bernstein-Bézier meshes) have often been used in the graphics literature (see, e.g., [Bargteil and Cohen 2014; DeRose 1988; Roth et al. 1998; Weber et al. 2011]), we briefly review their construction here for completeness.

Preamble. In the previous section, we used a linear map φ between a regular simplex and an arbitrary simplex embedded in $\mathbb{R}^d$. With this convention, a straight-edge simplex is actually a *parametric* patch, where the parameter domain is the regular simplex, and the interpolation between the vertices of the physical simplex is through linear basis functions. The Jacobian of this geometric map φ (which is constant per simplex since the basis functions are linear) indicates the local deviation between the canonical and the physical element, and the ODT energy tries to render this Jacobian field as uniform and isotropic as possible. Bézier simplices extend this geometric map notion to now *higher-order* polynomial deformation functions, where vertices



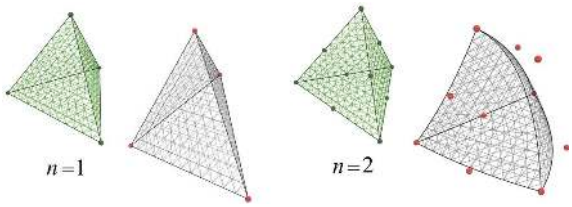


Fig. 4. **Bézier simplices:** Examples of Bézier simplices for $n=1$ (left, simplicial meshes) and $n=2$ (right, quadratic patches), with control points highlighted in red; top: their reference (straight-edge) regular simplex.

are enriched with other control points per simplex to increase the space of possible deformations.

Barycentric Coordinates. The use of barycentric coordinates is particularly convenient when dealing with simplices: any point $\bar{\mathbf{x}}$ in a simplex $\bar{\tau}$ in $\mathbb{R}^d$ is encoded by a column vector of $(d+1)$ non-negative coefficients $\mathbf{u}(\bar{\mathbf{x}}) \coloneqq (u_1,\dots,u_{d+1})^t$, which sum to one. We denote by $\overline{\mathcal{U}}$ the matrix of the differential of the linear map from $\bar{\mathbf{x}}$ to $\mathbf{u}$ (i.e., the Jacobian matrix); it is constant over the simplex since barycentric coordinates are linear. Given that we only consider *regular* simplices $\bar{\tau}$, its expression is the same for each simplex, and is given in App. B for 2D and 3D.

Bernstein polynomials. A barycentric index $\mathbf{i}$ of order n is defined by a vector of $(d+1)$ non-negative integers $\mathbf{i}=(i_1,\dots,i_{d+1})\in\mathbb{N}^{d+1}$ that sum to n, i.e., $|\mathbf{i}| \coloneqq \sum_k i_k = n$. For each such index corresponds a Bernstein polynomial $B^n_{\mathbf{i}}$ over the simplex $\bar{\tau}$ (the superscript n refers to the order of the subscript $\mathbf{i}$), defined through:

$$B^n_{\mathbf{i}}(\mathbf{u}) = \frac{n!}{\prod_{k=1}^{d+1} i_k!} \prod_{k=1}^{d+1} u_k^{i_k}, \tag{6}$$

where $\mathbf{u}$ is a shorthand for $\mathbf{u}(\bar{\mathbf{x}})$. These polynomials have a number of interesting mathematical properties (e.g., partition of unity, positiveness over the simplex), not the least being that their partial derivatives are, themselves, scaled Bernstein polynomials:

$$\frac{\partial}{\partial u_k} B^n_{\mathbf{i}}(\mathbf{u}) = n B^{n-1}_{\mathbf{i}-\mathbf{e}_k}(\mathbf{u}) \tag{7}$$

where $\mathbf{e}_k$ denotes the vector of size $(d+1)$ with 1 at the kth position and all zeros otherwise (to be valid for any n, Bernstein polynomials are assumed to be identically zero if one of the barycentric indices is negative). Note that we will denote by $\mathcal{B}^n_{\mathbf{i}}$ the Jacobian (row vector) of $B^n_{\mathbf{i}}$ with respect to the barycentric coordinates, i.e.,

$$\begin{aligned}\mathcal{B}^n_{\mathbf{i}}(\mathbf{u}) &= \left[\frac{\partial}{\partial u_1} B^n_{\mathbf{i}}(\mathbf{u}) \quad \cdots \quad \frac{\partial}{\partial u_{d+1}} B^n_{\mathbf{i}}(\mathbf{u})\right] \\ &\overset{\text{Eq. (7)}}{=} n\left[B^{n-1}_{\mathbf{i}-\mathbf{e}_1}(\mathbf{u}) \quad \cdots \quad B^{n-1}_{\mathbf{i}-\mathbf{e}_{d+1}}(\mathbf{u})\right].\end{aligned} \tag{8}$$

Bézier simplex. We now can properly define a d-dimensional Bézier simplex of order n: given a set $\mathbf{C}$ of control points with

$$\mathbf{C} = \{\mathbf{c}_{\mathbf{i}} \in \mathbb{R}^d \,|\, \mathbf{i}\in\mathbb{N}^{d+1}, |\mathbf{i}| = n\},$$

the associated Bézier simplex is defined via its geometric map φ:

$$\mathbf{x} = \varphi(\bar{\mathbf{x}}) \coloneqq \sum_{|\mathbf{i}|=n} \mathbf{c}_{\mathbf{i}}\, B^n_{\mathbf{i}}(\mathbf{u}(\bar{\mathbf{x}})). \tag{9}$$

The control points $\mathbf{c}_{\mathbf{i}} \in \mathbb{R}^d$ are degrees of freedom for the Bézier simplex, allowing it to curve. Note, as mentioned earlier, that the case $n=1$ is nothing but the piecewise-linear map described in Sec. 2

since the Bernstein polynomials are linear, and the control points are restricted to be the vertices of the linear mesh. More generally, a Bézier simplex only interpolates its "corner" control points (i.e., those of the form $\mathbf{c}_{ne_k}$), whereas the other control points smoothly pull and stretch the simplex towards them, see Fig. 4. Note also that this construction has the property that the boundary of a Bézier k-simplex is, itself, a Bézier $(k-1)$-simplex.

Map Jacobian. Knowing the Jacobian $\overline{\mathcal{U}}$ of $\mathbf{u}$ w.r.t $\bar{\mathbf{x}}$ and the Jacobian $\mathcal{B}^n_{\mathbf{i}}$ of the Bézier simplex w.r.t. $\mathbf{u}$, we can directly express the Jacobian of the geometric map φ of the Bézier simplex w.r.t. $\bar{\mathbf{x}}$ as:

$$\mathbf{J}(\bar{\mathbf{x}}) \coloneqq \nabla\varphi(\bar{\mathbf{x}}) = \left(\sum_{|\mathbf{i}|=n} \mathbf{c}_{\mathbf{i}} \cdot \mathcal{B}^n_{\mathbf{i}}\left(\mathbf{u}(\bar{\mathbf{x}})\right)\right) \overline{\mathcal{U}}(\bar{\mathbf{x}}) \tag{10}$$

We deduce directly that the change $\partial\mathbf{J}(\bar{\mathbf{x}})/\partial\mathbf{c}_{\mathbf{i}}$ of the Jacobian with respect to a control point $\mathbf{c}_{\mathbf{i}}$ is given by:

$$\frac{\partial\mathbf{J}(\bar{\mathbf{x}})}{\partial\mathbf{c}_{\mathbf{i}}} = \mathcal{B}^n_{\mathbf{i}}\left(\mathbf{u}(\bar{\mathbf{x}})\right) \overline{\mathcal{U}}(\bar{\mathbf{x}}). \tag{11}$$

Bézier mesh. A Bézier mesh is a manifold assembly of Bézier simplices, that is, any two adjacent Bézier simplices have their associated parametric regular k-simplices sharing a common $(k-1)$-simplex for $k \leq d$. Moreover, two adjacent k-simplices share the *same* (locations of) control points on their shared $(k-1)$-simplex as well, thus enforcing continuity across simplices. Note that the expression of the Jacobian with respect to a control point $\mathbf{c}_{\mathbf{i}}$ derived for a single element in Eq. (11) remains valid as is, but one has to sum the contributions of each simplex sharing this control point if needed to get the final Jacobian.

Cubature points and cubature samples. For a function $g(\mathbf{u})$ defined in barycentric coordinates over a d-simplex, its integral over the Bézier parametric simplex can be evaluated numerically using a cubature scheme. For this purpose, we define a set of weighted points $\{(\mathbf{u}_i, w_i)\}_i$ over the reference simplex $\bar{\tau}$ such that $\sum_i w_i = 1$: they define a *cubature scheme* with which a function g can be integrated through a linear combination of sample values as

$$\int_{\bar{\tau}} g(\mathbf{u}) \;\approx\; |\bar{\tau}| \sum_j w_j \, g(\mathbf{u}_j)\,.$$

Additionally, we map these cubature points to the physical domain to create a series of *cubature samples* $\{\mathbf{x}_j \coloneqq \mathbf{x}(\mathbf{u}_j)\}_j$ for each Bézier simplex. This sampling of the Bézier simplices will be useful when dealing with boundary fitting: they provide an efficient way to evaluate surface integrals by only using values at these spatial samples.

3.2 Extending ODT to High-Order Elements

Straight-edge triangulations correspond to Bézier meshes of order one, so one may be tempted to extend the approximation theory foundations behind ODT to the general case of order n. However, the best discrete approximant of the paraboloid is no longer relevant here: any Bézier simplex of order more than one can exactly capture this low-order polynomial height field. Using a higher-order polynomial to replace the paraboloid is also bound to fail, as one loses the convenient property that the function and its discrete approximant are below one another, rendering the L_1 norm difficult to evaluate.

However, our analogy with continuum mechanics still holds in the high-order case. We proved in Lemma 2.1 that the ODT energy

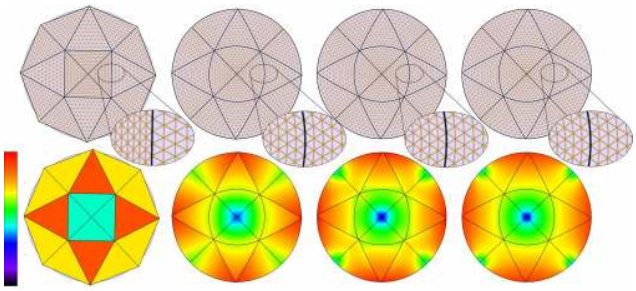


Fig. 5. **Determinant of Jacobian.** From left to right: ODT mesh ($n = 1$), then curved ODT meshes with order 2, 3 and 5 respectively, showing increasing continuity across elements. Bottom: determinant of the Jacobian in the domain visualized with a rainbow colormap.

of a mesh corresponds to a specific potential energy of the piecewise linear map between an idealized unit reference mesh of same connectivity and this mesh. Since each element of a Bézier mesh has a simplicial parametric domain and the map is now piecewise polynomial, the exact same expression is well defined in this Bézier context. We rename it E_{CODT} to underline its validity on any Bézier mesh of order $n \geq 1$:

Definition 3.1. Given a Bézier mesh $\mathcal{T}$ made out of d-simplices of order n and its unit reference mesh $\overline{\mathcal{T}}$ made out of its associated equilateral simplicial charts with the same connectivity, we define its *Curved Optimal Delaunay Triangulation* energy as:

$$E_{\text{CODT}} = \frac{1}{2(d+2)} \int_{\overline{\mathcal{T}}} \det(\mathbf{J}(\bar{\mathbf{x}})) \, \|\mathbf{J}(\bar{\mathbf{x}})\|_{\text{F}}^2 \, d\bar{\mathbf{x}}, \tag{12}$$

where $\mathbf{J}(\bar{\mathbf{x}})$ is the Jacobian of the Bézier simplex at point $\bar{\mathbf{x}}$ on the parametric domain.

Polynomial degree of E_{CODT}. For Bézier simplices, this energy is in fact polynomial in the barycentric coordinates. In a Bézier d-simplex of order n, the Jacobian has degree $n-1$, and its determinant has degree $d\,(n-1)$. Thus, the energy density is of polynomial degree $(d+2)(n-1)$. In practice, order-2 (resp., order-3) Bézier elements in 2D and 3D are often used, leading to polynomial degrees 4 and 5 (resp., 8 and 10). Note that more complex energy densities (as discussed in Sec. 2.4) would raise the degree of this polynomial, or even render the energy rational, adding significant difficulty to its evaluation and, arguably more troublesome, its minimization.

Interpretation for higher-order meshing. While the ODT energy is clearly promoting equilateral simplices, a closer look is needed to understand the effect of a low CODT energy. In the case of a piecewise linear map φ, the energy in a simplex was proportional to its volume and to the squared Frobenius norm of Jacobian describing its linear transformation from a regular simplex, see Lemma 2.1. In a Bézier simplex, the reference (parametric) element is still a straight-edge simplex, so the notion of unit reference mesh remains valid and unchanged. But now, the actual elements are curved, so the traditional notion of "well-shaped elements" in simplicial meshing is no longer appropriate. However, what this energy does is, in essence, to consider the Bézier mesh subdivided to an infinitely dense mesh, where now the edges are nearly straight, and it sums up the classical ODT energy for these fine sub-simplices. So minimizing

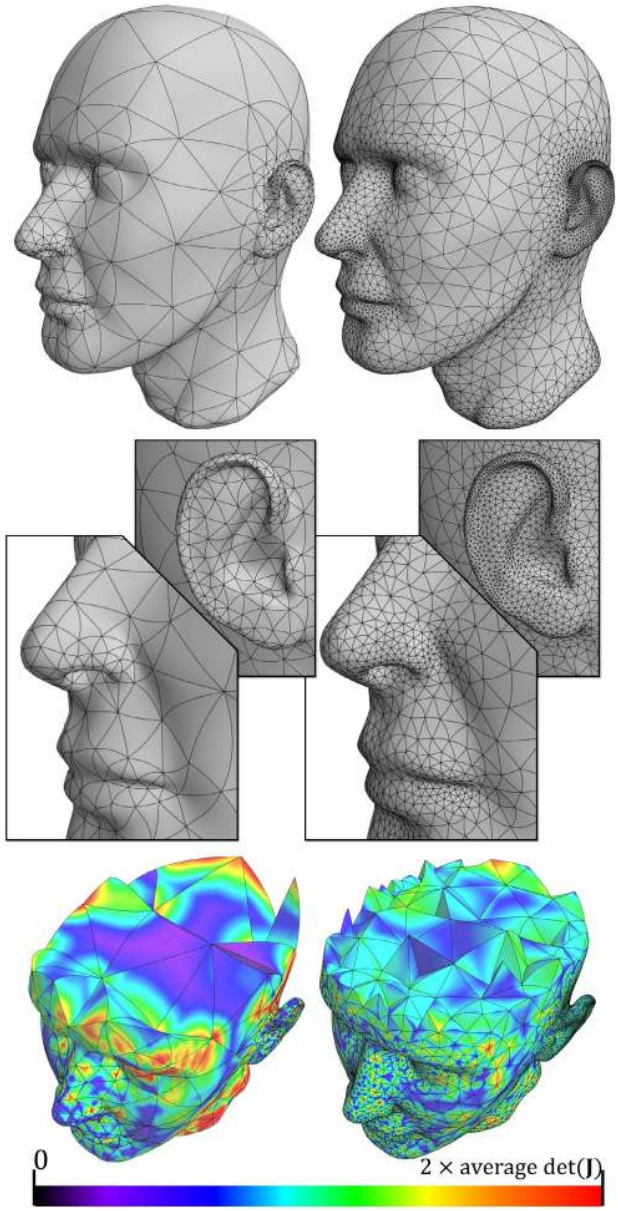


Fig. 6. **Man head.** Order-3 Bézier meshes based on a sizing field derived from the local feature size with 1K vertices (left, no sliver peeling) vs. 10K vertices (right); the sizing-scaled Jacobian determinant is visualized (bottom) via a rainbow colormap; respective closeups on the ear and nose (middle).

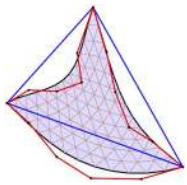
this energy will lead, instead, to a Jacobian *field* of the deformation map between the Bézier parametric domain and its curved elements as close to being uniform and isotropic throughout the domain. This effect can be clearly noticed in Fig. 5: starting from an ODT mesh, optimizing the control points of a Bézier mesh to minimize the energy E_{CODT} makes the local Jacobian field more continuous across edges—not to mention that the same number of elements now approximates the smooth domain much better (see also in Fig. 11 how the Hausdorff distance to the boundary quickly decreases with order). This continuity of the Jacobian field will improve within, and across, elements as we go higher in the order of the Bézier simplices, but not necessarily at vertices that form a cone singularity in the reference domain: Fig 5 shows this expected behavior by displaying the determinant of the Jacobian for a high-order Bézier mesh, which is smooth everywhere except at vertices with irregular valences.

Derivative of CODT energy. The derivative of CODT energy defined in Def. 3.1 with respect to a Bézier control point $\mathbf{c_i}$ is found by the product rule (see App. A for the full derivation), leading to:

$$\frac{\partial E_{\text{CODT}}}{\partial \mathbf{c_i}} = \frac{1}{2(d+2)} \int_{\overline{\mathcal{T}}} \mathcal{B}_{\mathbf{i}}^n(\mathbf{u}(\bar{\mathbf{x}})) \overline{\mathcal{U}}(\bar{\mathbf{x}}) \left[\|\mathbf{J}\|_{\text{F}}^2 \mathbf{J}^{-1} + 2\mathbf{J}^t \right] \det(\mathbf{J})\, d\bar{\mathbf{x}}. \quad (13)$$

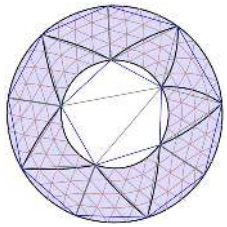
3.3 Bézier mesh topology

We also need to consider the topology of the Bézier mesh, in particular to find how to optimize the connectivity based on the CODT energy, and which elements are inside the domain.



Proxy straight-edge simplices. As often used for Bézier meshes, we first define the notion of "proxy simplex" for a Bézier simplex to be the straight-edge simplex formed by the vertices (i.e., corner control points) of the Bézier patch (inset: order-5 simplex, proxy in blue). For $n = 1$, this is exactly the Bézier simplex; for higher orders, it is just a straight-edge approximation of the curved element.

Delaunay proxy connectivity. Remember that for the straight-edge case, the minimum of the E_{ODT} for fixed vertex positions is achieved for the Delaunay connectivity of these positions. With this new, extended CODT energy, the situation becomes more complex: now, the position of the control points do depend on this connectivity, so finding which connectivity minimizes the CODT energy is less obvious. We are not aware of how to define the optimal connectivity of the Bézier mesh; however, since we are trying to form a Bézier mesh with a distortion as small and uniform as possible, we proceed with the following assumption: we simply enforce *a Delaunay connectivity for the proxy elements*. That is, from the vertices of the Bézier mesh only (disregarding the other control points), we compute the Delaunay triangulation of these positions, and assume that the resulting Bézier mesh with these Delaunay simplices as proxy elements is the one with minimal energy.



Restriction to Ω. Given a set of vertices and a Delaunay connectivity for the proxy elements, we have formed a Bézier mesh. However, not all of the elements are relevant to the meshing of the domain. In particular, concave regions may contain Bézier simplices that should really be considered as outside, and thus disregarded when it comes to optimizing the meshing of Ω. For a Delaunay (straight-edge) first-order Bézier mesh, there is a clear way to identify elements inside Ω through the notion of *restricted Delaunay elements* [Rineau and Yvinec 2008]: these are simplices for which their Voronoi dual is inside Ω. *Only* those restricted mesh elements are active during ODT optimization, as they correspond to the set of inside elements. We thus proceed exactly the same way

for our case, but by testing the proxy elements instead: every Bézier simplex whose proxy element is restricted is tagged as active. A finer approximation could be done by considering the subdivided control net; but only testing the proxy elements has shown sufficient in all our examples.

Finally, we also deactivate restricted boundary elements that have two faces or more on the domain boundary and are not on a sharp feature: this allows to quickly "peel off" sliver-like curved elements. Such curved slivers can be useful if only a concise representation of the domain is sought after, but they are not acceptable for finite element computations as they lead to singularities of $\det \mathbf{J}$. Fig. 7 depicts such an example in 2D and Fig. 16 (bottom) in 3D, each involving a simplex angle close to 180 degrees.

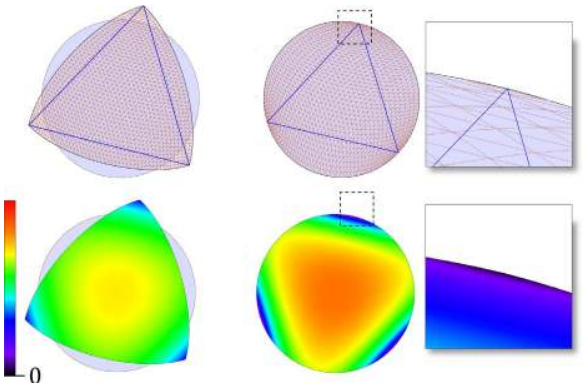


Fig. 7. **Singularities at obtuse angles.** If a curved triangle loosely approximates a disk in 2D (left; proxy triangle in blue), the Jacobian is well behaved everywhere. For strong fitting forces (middle), near-zero Jacobians may occur at vertices when the angles become close to 180 degrees, see the dark magenta color and the squeezed subtriangles in the two insets (right).

Boundary elements. Our identification of restricted elements also provides a direct way to find which Bézier faces are on the external boundary of our Bézier mesh: during the testing of restricted elements, any d-dimensional simplex for which one of its $(d-1)$-faces' dual edges crosses $\partial\Omega$ is tagged as "boundary simplex", and the corresponding boundary face is tagged as "boundary": they will play an important role in fitting the domain.

3.4 Non-shrinking boundary conditions

As we discussed in Sec. 2.1, defining proper boundary handling is important to obtain meshes with low shape distortion. While previous boundary conditions do not always carry over straightforwardly to our curved meshing context, our new isotropic-stress boundary condition discussed in Sec. 2.3 can be adapted rather easily.

At first, one may be tempted to simply apply a force $\mathbf{f}_{\text{bdry}}$ (given in Eq. (5)) *pointwise* along boundary elements, since the gradient $\mathbf{J}$ of the map is now spatially varying. This is, however, forgetting part of the goal of this boundary force. Remember that the internal CODT energy tries to get the Jacobian as uniform and isotropic as possible. But the boundary condition should also promote *both* isotropy *and* uniformity of the Jacobian.

We thus propose to compute an *average tensor per boundary simplex*, and apply this tensor to form the boundary traction. More specifically, we compute the average $\langle \|\mathbf{J}\|_{\text{F}}^2 \rangle_{\tau_i}$ (over area in 2D, over volume in 3D) of the isotropic part of the CODT stress tensor of a boundary simplex. Then we apply the following boundary force field for the face(s) of τ_i that are tagged as "boundary":

$$\mathbf{f}_{\text{bdry}} = \frac{1}{2d} \langle \|\mathbf{J}\|_{\text{F}}^2 \rangle_{\tau_i} \, \mathbf{n}.$$

This averaging of the isotropic part of the stress at the boundary used as an element-wise traction now promotes both isotropy and uniformity: equilibrium will be reached when the Jacobian is, on average, isotropic and uniform. It also exactly fits the straight-edge ODT case, as the average and the pointwise Jacobian are equal in the piecewise linear map case.

3.5 Boundary fitting

Minimizing our CODT energy with added traction at the boundary makes any Bézier mesh as "regular" as possible in the sense that it has a Jacobian with respect to the parametric domain quite uniform and isotropic: in fact, like in Fig. 2, running the usual control-points / proxy-connectivity procedure to minimize the CODT energy leads to nearly linear elements, as these correspond to the best possible element in terms of their Jacobian fields. Yet, the domain Ω may not be particularly well approximated by restricted elements, unless the mesh is very dense. One of the key advantages of Bézier meshes is that they can adapt to curved domains with high precision even for a small count of elements thanks to the added flexibility afforded by the control points and high order basis functions.

In order to achieve this desirable property, we add a boundary fitting force field on the boundary faces of $\mathcal{T}$ to push them towards the boundary $\partial\Omega$ of the domain. The magnitude of this extra force offers a compromise between boundary fitting and low distortion of the Bézier map. Depending on the application for which a mesh is constructed, vertices may have to be precisely on the boundary, or the mesh boundary just needs to approximate $\partial\Omega$. While the former can be achieved using methods as in [Alliez et al. 2005; Chen et al. 2014; Gao et al. 2012], we derive a simple approach to the latter by exploiting the stability of our non-shrinking minimization.

We simply add a boundary force field $\mathbf{f}_{\text{fit}}$ that attracts the boundary face elements of $\mathcal{T}$ to the domain boundary $\partial\Omega$ using a two-sided notion of distance minimization. For any cubature sample $\mathbf{x}_i$ of a d-simplex τ that is on $\partial\mathcal{T}$, we apply a local fitting force that sums a force field $\mathbf{f}_{\mathcal{T}\to\Omega}$ based on the shortest distances from $\mathcal{T}$ to Ω and another force field $\mathbf{f}_{\Omega\to\mathcal{T}}$ based on the shortest distances from Ω to $\mathcal{T}$ to ensure tight geometric closeness:

$$\mathbf{f}_{\text{fit}}(\mathbf{x}_i) = \lambda \, |\tau|^{1/d} \, \big(\mathbf{f}_{\mathcal{T}\to\Omega}(\mathbf{x}_i) + \mathbf{f}_{\Omega\to\mathcal{T}}(\mathbf{x}_i)\big), \tag{14}$$

where $|\tau|$ is the volume of the d-simplex on which point $\mathbf{x}_i$ lies, and λ is the global strength of the force. The first force is defined as:

$$\mathbf{f}_{\mathcal{T}\to\Omega}(\mathbf{x}_i) = \Pi_{\partial\Omega}(\mathbf{x}_i) - \mathbf{x}_i,$$

where $\Pi_\Omega(\mathbf{x}_i)$ is the projection of $\mathbf{x}_i$ to the nearest point on the domain boundary $\partial\Omega$. The vector $\Pi_{\partial\Omega}(\mathbf{x}) - \mathbf{x}$ is known to be the local gradient of the one-sided squared distance function between $\partial\mathcal{T}$ and $\partial\Omega$ [Pottmann and Hofer 2003]. By applying this force to all cubature samples (weighted by their associated areas), we obtain a simple one-sided boundary fitting term whose integral is zero when the boundary of the Bézier mesh is straddling the domain

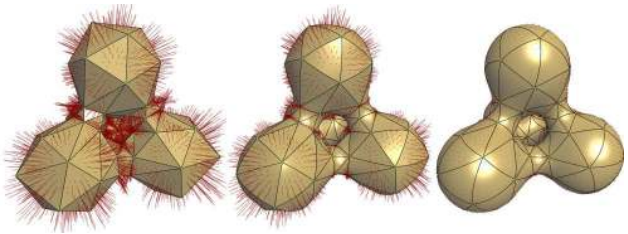


Fig. 8. **Fitting force field.** The fitting force field decreases in magnitude as the mesh of Fig. 1 ($n = 3$) is optimized from a very coarse linear mesh. Force vectors are scaled by a factor 20 for better visual depiction.

boundary. The use of this force alone is usually enough to attract the Bézier mesh to the boundary of the domain, but its one-sidedness may lead to regions of the domain boundary being not covered. So we compute the force $\mathbf{f}_{\Omega\to\mathcal{T}}$ derived from a set of sample points $\{\mathbf{s}_k\}_k$ sampling the boundary $\partial\Omega$: for each boundary sample $\mathbf{s}_k$, we compute their closest cubature sample $\mathbf{x}_i$ on the boundary $\partial\mathcal{T}$ of the Bézier mesh; we then add a force $\mathbf{f}_{\Omega\to\mathcal{T}}$ equal to the vector between the cubature sample and the centroid of all closest boundary sample points, restricted to be normal to the boundary (to prevent spurious sliding due to the use of the centroid):

$$\mathbf{f}_{\Omega\to\mathcal{T}} = \left(\frac{1}{|S(\mathbf{x}_i)|} \sum_{k \in S(\mathbf{x}_i)} \mathbf{s}_k \;-\; \mathbf{x}_i \right) \cdot \mathbf{n}_i \; \mathbf{n}_i, \tag{15}$$

where $S(\mathbf{x}_i)$ is the set of boundary sample indices such that $\mathbf{x}_i$ is their closest cubature sample, and $\mathbf{n}_i$ is the unit normal to the Bézier surface at $\mathbf{x}_i$. This extra forcing term helps ensure that every region of $\partial\Omega$ is close to the Bézier mesh. The resulting force is further multiplied in Eq. (14) by the local edge length h (obtained by taking the d-th root of the volume of τ) and a dimensionless coefficient λ for two reasons: first, it makes the fitting force be of the same dimensionality as the gradient of CODT, thus making these forces comparable in strength locally; second, the value λ can be interpreted as controlling h/ϵ where ϵ is the average distance of the boundary face of τ_i to the boundary of Ω. That is, λ is a scale-invariant parameter defining how close to the boundary we want the mesh to be compared to its local element size. This coefficient is thus easy to adjust based on the balance between mesh distortion and boundary fit, independent of mesh density or domain shape. Refer to Fig. 9 to see the effect of increasing values of λ.

3.6 Mesh gradation

For a sizing field $h(\mathbf{x})$ over Ω, all the expressions we provided need to be altered to lead to the requested sizing constraints. Most changes are very simple, and are best understood as thinking of sizing as a modulation of the local metric by $1/h$. This means that any gradient needs to account for h (for instance, the evaluation of $\mathbf{J}$ should become $\mathbf{J}/h$), and any local evaluation of length, area, or volume in should be multiplied by h^{-1}, h^{-2} or h^{-3} respectively. As a result, the ODT energy is effectively multiplied by $h^{-(d+2)}$ pointwise as mentioned in the straight-edge case, since it involves $\|\mathbf{J}\|_{\mathrm{F}}^2$ (bringing a factor h^{-2}) and $\det(\mathbf{J})$ (responsible for a factor h^{-d}). The only complication is the computation of the derivative of the sizing-weighted CODT energy, as it now involves a derivative in the sizing

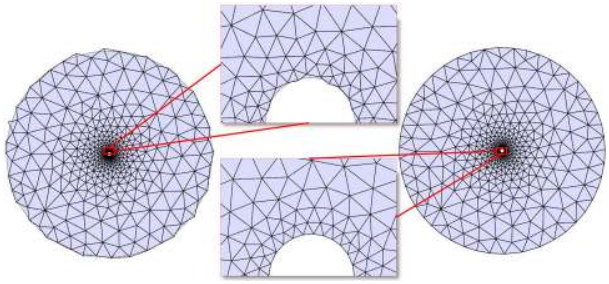


Fig. 9. **Boundary fitting.** The dimensionless fitting strength λ in Eq. (15) controls the tightness of the fit to the domain boundary: a small value (left, $\lambda = 1$) will only weakly capture the boundary shape, while a large value (right, $\lambda = 100$) provides a close fit. The effect is adapted to the size of local elements, making its value scale invariant, thus intuitive and easy to set.

field itself; for completeness, we provide its derivation in App. A and its complete expression to ease implementation in Eq. (23).

As reported in prior work, we found that having a slowly-varying sizing field that is adapted to the local feature size (lfs) of the domain (see, for instance, the lfs-based K-Lipschitz field in [Alliez et al. 2005]) is best for the gradation of the resulting mesh, as well as for the robustness of the meshing process as the size of elements fits the local geometry well. See Fig. 10 to see the effect of varying mesh densities for a given domain, using degree-3 elements.

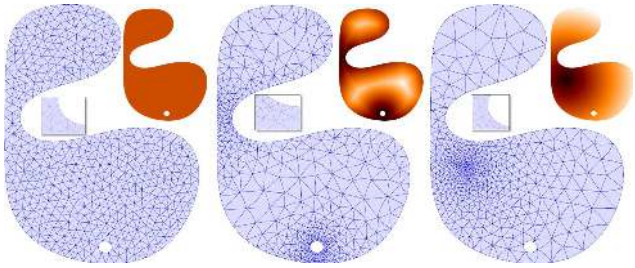


Fig. 10. **Sizing.** Left: uniform sizing; middle: lfs-based sizing; right: radially linear sizing function. Respective sizing fields h are depicted on top.

3.7 Putting it all together

While minimizing the CODT energy can be incorporated into any meshing technique with minimal effort, we describe next how one can construct an isotropic curved Bézier mesh from an input domain Ω and a sizing field h (possibly constant for uniform meshes). In particular, we mention a few important implementation details to improve efficiency and robustness.

Mesh initialization. Starting from a decent mesh is preferable if computational time is an important issue—but any mesh can do. In practice, we initialize the vertices by drawing them randomly from a density distribution adapted to the sizing field. Our implementation follows a usual Delaunay refinement/deletion process [Cheng et al. 2012; Rineau and Yvinec 2008] to construct a restricted tetrahedron mesh that roughly follows the sizing field h and captures the topology of the domain Ω. This coarse mesh defines proxy elements from which a first Bézier mesh can be initiated.

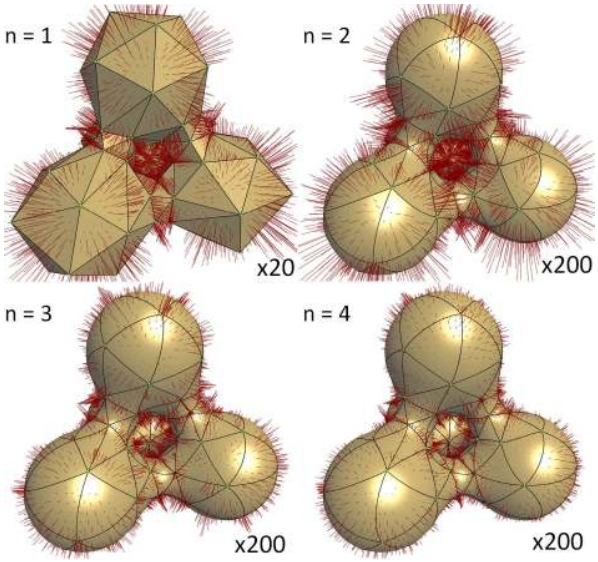


Fig. 11. **Boundary approximation.** Input domain approximated for a given number of vertices, from Bézier order 1 to 4. The respective 2-sided Hausdorff errors (as a percentage of bounding box diagonal) are 3.42, 0.315, 0.133 and 0.073. The boundary forces are rescaled for better depiction.

Numerical integration using cubature. Since the Bézier mapping we employ is polynomial and the CODT energy is a function of its Jacobian, the integrals involved in our CODT energy and its derivative can be evaluated efficiently and exactly using a cubature scheme with the proper order (and with a pre-evaluation of the Bernstein polynomials of the cubature points for efficiency) as their pointwise values are known in analytical form. Unless a constant sizing field is chosen, we use Witherden-Vincent cubature points [Witherden and Vincent 2015]: the presence of an *arbitrary* (i.e., non-polynomial) sizing field h is robusly handled through this cubature scheme—with, again, pre-evaluation of the field at the cubature points—because its weights are all positive. In our code, we used the points and weights of this cubature scheme provided in [Schlömer 2018].

Optimization of E_{CODT}. From a current Bézier mesh, we perform multiple "rounds" of optimization of the CODT energy, i.e., alternate steps of geometry and topology updates to improve the mesh:

- *Geometry.* Control points (including vertices) are moved through quasi-Newton (more precisely, LBFGS) iterations, where we add to the gradient of E_{CODT} (Eq. 11) the contribution of the boundary forces $\mathbf{f}_{\mathrm{fit}}$ and $\mathbf{f}_{\mathrm{bdry}}$ to the control points. More precisely, for each control point $\mathbf{c_i}$, we add to $\partial\mathbf{J}/\partial\mathbf{c_i}$ the boundary force for each cubature sample $\mathbf{x}_k$ of neighboring simplices weighted by the value $B^n_{\mathbf{i}}(\mathbf{u}_k)$ of the Bernstein polynomial $B^n_{\mathbf{i}}$ for that cubature point $\mathbf{u}_k$. Since Bernstein polynomials form a partition of unity over each simplex, forces are properly distributed to the control points; the resulting vector forms the gradient (righthand side) of the quasi-Newton step, while the Hessian is approximated from previous gradients (we use a history size of 5 in our code).
- *Topology.* Every 10 geometry updates, we verify that every proxy element is still Delaunay, performing a topology update if needed. Once a connectivity update is triggered, we make sure not to alter the control points of the unaffected elements; newly-created elements are assigned either existing control points for shared faces, or canonical internal control points as if they were forming a straight-edge simplex: these new control points will move at the next round of energy minimization anyway. To accelerate computations, this topology check is only done every 30 geometry updates once the energy change per iteration goes below a certain threshold: this indicates near convergence, and thus a low probability of frequent connectivity changes.

Early on, Jacobians with negative determinants may appear temporarily, corresponding to local foldovers in a Bézier patch. While minimizing the CODT energy eventually removes these cases, we found beneficial to replace the determinant `det` of $\mathbf{J}$ by:

$$\det < 0 \,?\, -100\det : \det.$$

This simple trick heavily penalizes any locally negative Jacobian by artificially inflating its effect. Even with large fitting forces, our resulting meshes never have Jacobians with negative determinants.

Boundary fitting. The boundary forces are directly used as a change of the gradient for the quasi-Newton step in each geometry update. Faster convergence is observed if we start with a small fitting force magnitude λ: we start with $\lambda = 1$ and increase λ gradually during the geometry updates of the Bézier mesh to reach $\lambda = 1000$. With this scheduling (see Fig. 14 for an example), mesh elements have time to quickly position themselves near the boundary and equidistribute their Jacobian before the fitting force is too strong and starts counteracting the effect of the CODT minimization.

Sharp features. With the algorithm described above, sharp features and corners may or may not be properly handled, as shown in the inset (where depending on the initialization, the features get exactly captured (left), or at times, an element is bent to only approximate the feature (right)). Sharp features can, however, be tackled properly if needed. Recall that the mesh boundary corresponds to facets of "restricted" proxies. Instead of having only fitting forces between the mesh boundary and the domain boundary, we can put fitting forces between all domain features and restricted Delaunay elements (facets, edges, vertices). More generally, restricted Delaunay elements are elements whose dual intersects the corresponding feature in the domain. For 3D meshing, this includes all features of lower dimension (2D, 1D, 0D): the dual of a restricted facet (a Voronoi edge) pokes the domain 2D boundary as we used before; but now, an edge is restricted if its dual (a Voronoi face bisecting this edge) contains a sharp crease of the domain; and a vertex is restricted if its dual (Voronoi cell) contains a sharp corner. In practice, we thus sample all existing features. During optimization, we detect all these restricted elements within the proxy mesh after each connectivity update, and we call a Bézier element restricted if its proxy element is restricted. Before each quasi-Newton iteration, we calculate cubature points on all these restricted Bézier elements, then for each kind of features, we find the two-sided projection vectors between Bézier cubatures and domain

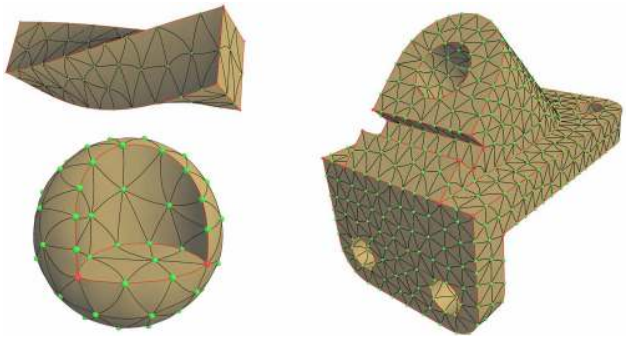


Fig. 12. **Sharp features.** With simple changes, our approach can accommodate sharp features, ensuring that the curved edges of the Bézier mesh approximate the feature curves, while corners are at vertices. Restricted Bézier edges and vertices (i.e., creases and corners) are shown in red.

samples, just like $\mathbf{f}_{\Omega\to\mathcal{T}}$ and $\mathbf{f}_{\mathcal{T}\to\Omega}$ in Sec. 3.5. The only difference with Eq. 14 is the rescaling factor: to keep λ dimensionless, we use e^2 and s^3 for edges and vertices respectively, where e is the local edge length and s is the average length of edges incident to the vertex. Three meshes with sharp features are shown in Fig. 12.

Interleaving refinement/deletion/teleportations. Note that most of the previous accelerations proposed in the context of straight-edge ODT meshes can also be incorporated in the curved mesh extension. In particular, we found that the strategy of Tournois et al. [2009] consisting in triggering occasional refinements (resp., deletions) of elements in regions where the determinant of the Jacobian is smallest (resp., largest) helps converge faster. The use of vertex teleportation recommended in [Alliez et al. 2005] achieves a similar goal, where now insertion and deletion are done in pairs.

Degree elevation. As recommended in previous works, proceeding through incremental degree elevation helps both accelerate convergence and improve the final mesh quality: even if CODT is significantly simpler than previous mesh-optimization functionals, starting from lower order polynomials avoids stepping early into local minima, and minimize the amount of transient wiggling that high order polynomials are known to create. The use of lower-order numerical integration with fewer cubature points also improves efficiency in early stages of the meshing.

3.8 Discussion

Our extension to ODT allows the creation of curved meshes with excellent properties, akin to the isotropy of ODT meshes. We review next a few notable properties that end users may benefit from, as well as further possible extensions.

Isogeometric Analysis. In computational science, recent years have seen a growing interest in Isogeometric Analysis (IGA): the geometry of the computational domain is defined through parametric functions that are *also* used to approximate the unknown physical fields in a numerical simulation. Such approaches avoid the typical mismatch between the basis functions describing the geometry and the ones defining the solution space in the domain, thus improving the analysis and convergence of numerical methods. Our curved extension of ODT may be particularly relevant to this community: because we optimize the parameterization to ensure a high quality map gradient within and across elements, IGA-based numerical simulations should have both improved condition numbers of stiffness matrices and global accuracy [Pilgerstorfer and Jüttler 2014].

Hybrid meshes. While we discuss the optimization of the CODT energy with respect to *all* control points in the mesh, one may want to keep most internal d-simplices with straight edges: computing stiffness matrices in FEM simulation is computationally simpler when elements are not curved. This is easily achieved in our framework by optimizing all the control points within a certain distance of the boundary, but only corner control points (i.e., vertices) in the rest of the domain. As a result, we get a hybrid mesh with curved elements near the boundary and straight elements inside. Unlike previous approaches that propagate boundary deformation to inside elements through smoothing or blending, this approach targets low distortion of parametric elements directly and reliably.

Anisotropy. One interesting extension of our isotropic, curved meshing approach is to introduce anisotropy. The ODT approach is quite naturally extended to anisotropic meshes, through the insertion of a metric field (locally indicating both anisotropy and sizing) and the use of weighted Delaunay meshes (see [Budninskiy et al. 2016] for a recent summary). However, extending our approach to handle any metric field raises a number of issues (such as the design of anisotropic metric fields that are consistent with the geometry of the boundary) that deserve more attention than this document can dedicate to it. We thus leave anisotropy as future work.

4 RESULTS

We implemented our approach to curved meshing through ODT minimization in C++ using mostly the CGAL [The CGAL Project 2017], Eigen [Guennebaud et al. 2018] and OpenCL libraries. Meshing is achieved in a very systematic way starting from a domain Ω given as a triangle mesh or an implicit function (we use an oracle to make our approach independent of the input format). We begin with a given number of points generated with a spatial density proportional to a user-given sizing field, from which the restricted Delaunay elements within the domain Ω are determined. Then we begin rounds of geometric and connectivity updates for $n=1$ (keeping it as a linear mesh for now) as described in Sec. 3.7. When a round has converged (i.e., the last round had no connectivity updates and no vertex motions larger than a small threshold), we raise the order of the Bézier mesh (never by more than two orders at a time to avoid wiggling of the elements near the boundary) and start another set of rounds until convergence. All our results were generated with this simple approach. While this is unlikely to be the fastest approach to generate our final meshes, the robustness and efficiency of the approach made it quite straightforward to use as is. Note finally that we always display the *sizing-scaled Jacobian determinant* in the figure, i.e., $\det(\mathbf{J})/h^d$ (since its pointwise value is more meaningful in the case of steep sizing fields), and colorbars range from 0 to twice the average sizing-scaled Jacobian determinant.

Fig. 11 demonstrates the approximation power of curved meshes on a blobby-shaped domain. Starting from a desired number of vertices (110), we increase the order n from 1 to 4 to show how the

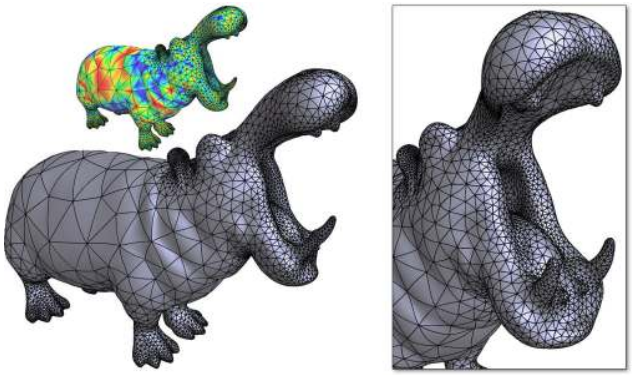


Fig. 13. **Hippo.** This CODT mesh used an lfs-based sizing field and Bézier elements of order 3, with 10K vertices and $\lambda = 10000$.

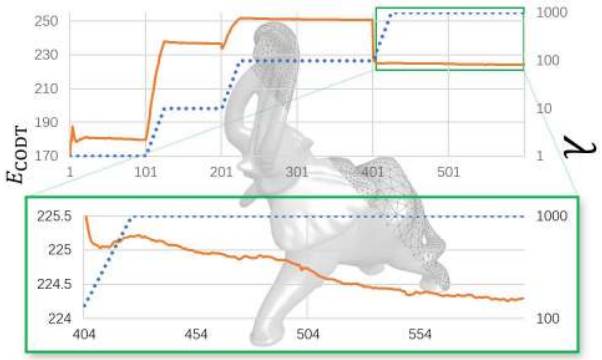


Fig. 14. **Energy minimization.** Energy plot (orange curve) and fitting strength λ scheduling (blue dots) during the meshing of the elephant model with 10K vertices, throughout the 600 rounds of optimization (horizontal axis). A closeup of the final stage is shown below. An energy drop is seen at the 401st round, where the mesh starts to curve.

boundary is closely captured. In order to estimate how well, we measured the 2-sided Hausdorff error between the boundary of the curved mesh and the input boundary as a percentage of the longest bounding box diagonal of the input. Using the error of the linear mesh as reference, the Hausdorff distance decreases by a factor 11, 26 and 47, respectively. For reference, matching similar errors with linear elements (generated by Delaunay refinement) requires 2K, 4.3K and 15K vertices, respectively.

Fig. 6 shows our approach on a 1-Lipschitz sizing field based on an estimate of the local feature size of the input domain [Alliez et al. 2005]. We generated two order-3 curved meshes of the same domain: one with 1K vertices (877 vertices ended up on the boundary, 123 inside, for a total of 2,971 elements) and one with 10K vertices (6,051 vertices on the boundary, 3,949 inside, 40,323 elements). For 1K (resp., 10K) vertices, the 2-sided Hausdorff distance is 0.35% (resp., 0.05%) of the longest bounding box diagonal, with an average Euclidean distance of 0.01% (resp., 0.00125%). A cross-section illustrates the gradation of the mesh inside. The Jacobian field is smooth, with distortion concentrated at mesh vertices. Similarly, Fig. 15 presents two order-3 curved meshes of the Bimba model, with respectively 2K and 10K vertices for a smoothly-varying sizing field; Fig. 17 shows the kitten model for three different resolutions. Finally, Fig. 13 depicts an order-3 curved mesh of the Hippo model with 10K vertices, for an LFS-based sizing field.

Other Jacobian-based methods. As discussed early on, current high-order meshing methods often use a Lagrangian mechanics analogy to find a curved mesh using a high-quality straight mesh as a reference. To allow for high curving, a neo-Hookean constitutive model is desirable [Persson and Peraire 2009], containing a logarithm of the determinant of the Jacobian. Instead, we use a low-order polynomial CODT energy, and do not have to rely on a PWL reference mesh, thus allowing both more freedom for the connectivity of the curved mesh and much simpler numerics.

Energy behavior. For fixed boundary, the CODT energy strictly decreases at each step of control point optimization and connectivity update. Because we use boundary forces to let vertices slide over the domain, this is no longer true. However, the energy minimization still behaves as expected: if the fitting strength λ increases, the energy increases; if λ stays constant for a while, then the energy decreases as the mesh deforms to improve the shape of its elements; finally, increasing the order of the Bézier elements allows for a lowering of the energy as more DOFs become available. Fig. 14 shows the energy throughout the construction of the elephant mesh as λ increases. When we switch from PWL to quadratic elements (after 400 iterations), the energy drops precipitously.

Timings. We measured our timings on HP Z420 workstation, with a quadcore E5-1620-0 clocked at 3.6GHz and 16 GBytes of memory. We parallelized most steps of our CODT minimization using OpenMP in shared memory mode. Optionally, the compute-intensive evaluation of the energy and its gradient at internal and boundary cubature points are performed on GPU, bringing an acceleration of around a factor 5 on a NVIDIA GTX 1060. Compared to Optimal Delaunay (linear) Tetrahedralizations, curved meshing is more compute-intensive, even more so when the order of the elements increases. The complete process for meshing the skin surface shown in Fig. 1 with 200 vertices and order 3, takes nearly 50s. Each quasi-Newton iteration takes 0.03s in the linear stage, and 0.14s in the order-3 stage (we go directly from degree 1 to degree 3). Meshing the ManHead model with 1K vertices and order-3 takes less than 40 mins (but only 15 mins on GPU). Going up to 10K vertices increases the computational time to 6 hours (only 80 mins on GPU); each order-1 iteration then takes 0.41s (0.27s with GPU), while an order-3 iteration takes 5s (1s with GPU), and we observe a near-linear time complexity in the order to the Bézier simplices for each iteration. In our experiments the total number of rounds required for convergence depends heavily on the number of vertices, the domain complexity and the grading of the sizing field. Using a uniform sizing function is usually 20% faster or more, as the CODT energy and its derivatives are simpler to evaluate. In practice, we found that almost two thirds of the meshing time is used to reach convergence for $n = 1$: once this stage is obtained, the rest (further degree elevations) goes relatively fast. Within each stage, most of the time is spent in computing cubatures and boundary fitting forces (hence the large time improvements when GPU acceleration is used). While we have not tried to fully optimize computational time, using

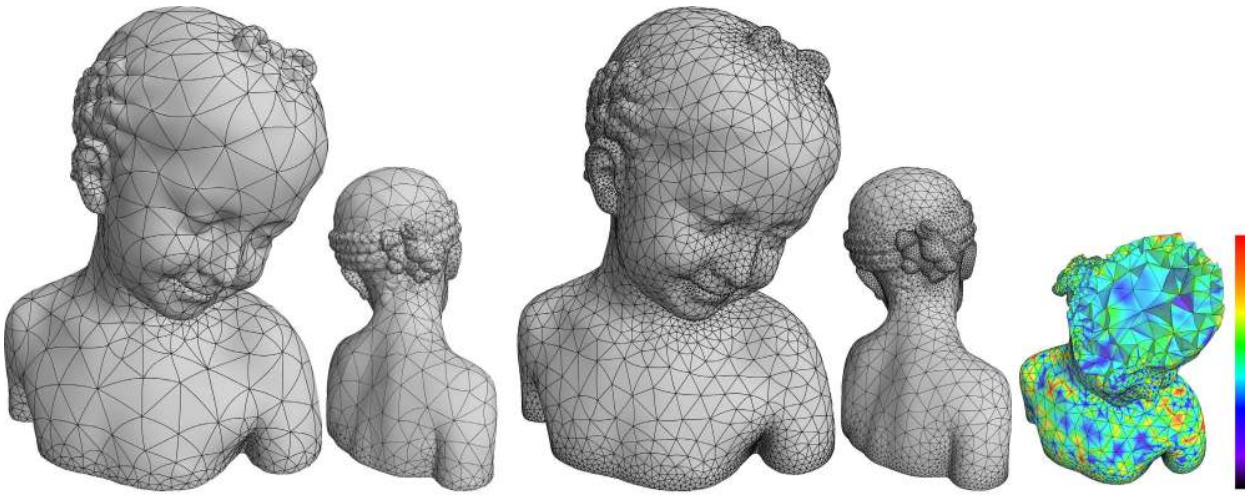


Fig. 15. **Bimba con nastrino.** This model is remeshed with an lfs-based sizing field using Bézier elements of order 3, with 2K (left) and 10K (right) vertices, for $\lambda = 1000$. The Jacobian determinant of the 10K-vertex result (rightmost) is shown, with a cross section to highlight the quality of internal elements.

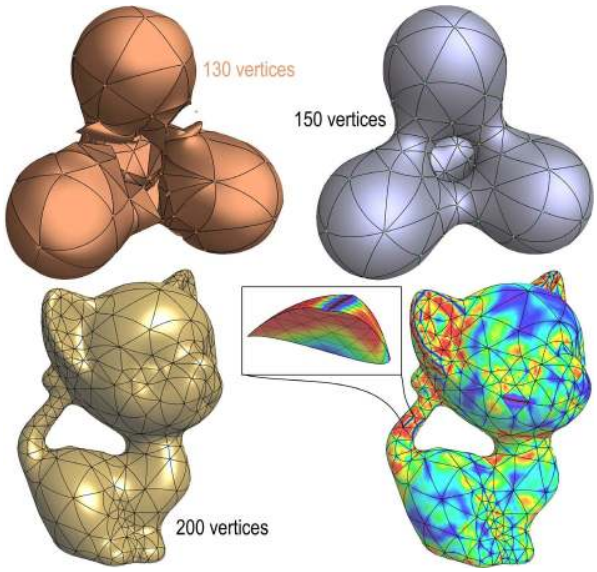


Fig. 16. **Failure cases.** Top: An insufficient number of vertices (130 in this case, left) may fail to capture the proper topology of a domain and generate self-intersections. Increasing the vertex count to 150 produces the correct topology and no self-intersection. Bottom: The kitten from Fig. 17 with only 200 vertices shows that degeneracies of $\det(\mathbf{J})$ appear because of a dihedral angle nearing 180 degrees (note that sliver peeling was turned off).

a cubature scheme of order 10 for the inside and order 20 for the boundary (to offer more accurate boundary fitting estimates) has proven adequate for all our examples; but adapting cubatures may improve timings significantly.

Limitations and Possible Extensions. There are mostly two main types of limitations to our current approach. First, geometry and topology failures may appear: using too few vertices can fail to capture the proper topology of the domain as shown in Fig. 16; similarly, using a badly-adapted sizing field may lead to a loss of geometric detail. Both cases are easily addressed by interleaving refinements as in [Tournois et al. 2009]. In the same vein, and while our proxy mesh is guaranteed to be free of self-intersection by construction, we do not currently test for self-intersection of the curved elements in our code since using a lfs-adapted sizing field basically prevents such a case; but localized refinements can remove these issues if detected. The second limitation concerns continuity of the patches. By construction, the output curved meshes are only strictly C^0 across faces, even if the CODT energy tends to render them nearly C^1 for higher-order Bézier meshes (see Fig. 5). Enforcing strict G^1 continuity is feasible, but adds many constraints on control points; finding how to efficiently minimize the CODT energy under these continuity constraints would be a valuable extension.

Supplemental material. We include short video clips demonstrating our approach. In 2D, we show a simple realtime sequence illustrating how CODT optimization with free boundary behaves, then the same optimization with boundary forces activated, and finally a visualization of the Jacobian field after degree elevation. In 3D, we show an accelerated timelapse of the meshing of the 10K-vertex ManHead mesh from Fig. 6, starting from $n=1$ until convergence, then going straight to $n=3$ until convergence.

5 CONCLUSIONS

We showed in this paper that the common ODT approach to generating isotropic triangle and tetrahedron meshes can be directly extended to curved meshing for higher-order basis functions. Our construction requires only usual meshing tools, such as Delaunay and restricted Delaunay triangulations, and a low-order polynomial CODT energy to minimize using quasi-Newton iterations. We showed that because of CODT energy benefits from the simplicity of the original ODT energy, minimization is efficiently achieved via simple control point updates and Delaunay connectivity in sharp contrast to previous methods. Finally, while we focused our exposition on Bézier simplices, other simplex-based high-order functions may be accommodated as well.

Just like ODT for tetrahedron meshing, the concept of CODT for curved meshing can be used not only for meshing a domain from scratch, but also as a generic tool for local mesh improvements in a variety of contexts. Our approach may thus benefit from better heuristics and cubatures in specific applications, such as local remeshing in physical simulation techniques like Arbitrary Lagrangian-Eulerian (ALE) methods. Other obvious directions for future work include anisotropic curved meshes—but the topic deserves its own focus. Due to the duality between Centroidal Voronoi Tessellations and Optimal Delaunay Triangulations [Budninskiy et al. 2016], our approach may also extend to Optimal Voronoi Tessellations, where now curved cell elements would be used.

ACKNOWLEDGMENTS

All meshes in this paper are courtesy of AIM@SHAPE and Luxology/Foundry. This work was supported by the European Union under grant 675789 (ITN ARCADES), and by the French government through the UCAJEDI Investments managed by the National Research Agency (ANR-15-IDEX-01). MD gratefully acknowledges the INRIA International Chair program, and Zhejiang University for hosting him superbly well during the final editing of this work.

REFERENCES

Remi Abgrall, Cécile Dobrzynski, and Algiane Froehly. 2012. *A method for computing curved 2D and 3D meshes via the linear elasticity analogy: preliminary results.* Technical Report RR-8061. INRIA. https://hal.inria.fr/hal-00728850

Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. 2005. Variational tetrahedral meshing. In *ACM Trans. Graph.*, Vol. 24. ACM, 617–625.

Pierre Alliez, Nathalie Laurent, Henry Sanson, and Francis Schmitt. 1999. Mesh approximation using a volume-based metric. In *Pacific Conf. on Computer Graphics and Applications*. 292–301.

Nina Amenta, Marshall Bern, and David Eppstein. 1999. Optimal Point Placement for Mesh Smoothing. *J. Algorithms* 30, 2 (1999), 302–322.

Ivo Babuska, Barna Szabo, and Norman Katz. 1981. The p-version of the Finite Element Method. *SIAM J. Num. Anal.* 18, 3 (1981), 515–545.

Adam W. Bargteil and Elaine Cohen. 2014. Animation of Deformable Bodies with Quadratic Bézier Finite Elements. *ACM Trans. Graph.* 33, 3 (June 2014), Art. 27.

Jean-Daniel Boissonnat, David Cohen-Steiner, and Mariette Yvinec. 2006. Comparison of algorithms for anisotropic meshing and adaptive refinement. *Tech. Rep. ACS-TR-362603,* INRIA (2006).

Max Budninskiy, Beibei Liu, Fernando de Goes, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. 2016. Optimal Voronoi Tessellations with Hessian-based Anisotropy. *ACM Trans. Graph.* 35, 6, Article 242 (2016).

David Cardoze, Alexandre Cunha, Gary L. Miller, Todd Phillips, and Noel Walkington. 2004. A Bézier-based Approach to Unstructured Moving Meshes. In *Symp. Computat. Geom.* 310–319.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A Simple Geometric Model for Elastic Deformations. *ACM Trans. Graph.* 29, 4, Article 38 (2010).

Long Chen. 2004. Mesh Smoothing Schemes based on Optimal Delaunay Triangulations. In *Int. Meshing Roundtable.* 109–120.

Long Chen and Michael Holst. 2011. Efficient mesh optimization schemes based on optimal Delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering* 200, 9 (2011), 967–984.

Long Chen and Jin-chao Xu. 2004. Optimal delaunay triangulations. *J. Computational Mathematics* (2004), 299–308.

Zhonggui Chen, Wenping Wang, Bruno Lévy, Ligang Liu, and Feng Sun. 2014. Revisiting Optimal Delaunay Triangulation for 3D Graded Mesh Generation. *SIAM J. Sci. Comput.* 36, 3 (2014), 930–954.

Siu-Wing Cheng, Tamal K. Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. 2000. Sliver Exudation. *J. ACM* 47, 5 (2000), 883–904.

Siu-Wing Cheng, Tamal K. Dey, and Jonathan R. Shewchuk. 2012. *Delaunay Mesh Generation.* CRC Press.

Tony D. DeRose. 1988. Composing Bézier Simplexes. *ACM Trans. Graph.* 7, 3 (July 1988), 198–221.

Qiang Du, Vance Faber, and Max Gunzburger. 1999. Centroidal Voronoi tesselations. *SIAM Rev.* 41, 4 (1999), 637–676.

Lori A. Freitag and Carl Ollivier-Gooch. 1997. Tetrahedral Mesh Improvement using Swapping and Smoothing. *Int. J. Numer. Meth. Engng.* 40, 21 (1997), 3979–4002.

Xiao-Ming Fu, Yang Liu, John Snyder, and Baining Guo. 2014. Anisotropic Simplicial Meshing Using Local Convex Functions. *ACM Trans. Graph.* 33, 6 (2014), Art. 182.

Zhanheng Gao, Zeyun Yu, and Michael Holst. 2012. Quality Tetrahedral Mesh Smoothing via Boundary-optimized Delaunay Triangulation. *Comput. Aided Geom. Design* 29, 9 (2012), 707–721.

Abel Gargallo-Peiró, Xevi Roca, Jaimie Peraire, and Josep Sarrate. 2013. High-order mesh generation on CAD geometries. In *Int. Conf. on Adaptive Modeling & Simulation.* 301–312.

Christophe Geuzaine, Amaury Johnen, Jonathan Lambrechts, Jean-Fraçois Remacle, and Thomas Toulorge. 2015. The Generation of Valid Curvilinear Meshes. In *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, Vol. 128. 15–39.

Gaël Guennebaud, Benoît Jacob, and others. 2018. Eigen v3. http://eigen.tuxfamily.org. (2018).

Amaury Johnen, Jean-François Remacle, and Christophe Geuzaine. 2013. Geometrical validity of curvilinear finite elements. *J. Comp. Phys.* 233, Supplement C (2013), 359–372.

Steve L. Karman, J. T. Erwin, Ryan S. Glasby, and Douglas Stefanski. 2016. High-Order Mesh Curving Using WCN Mesh Optimization. In *46th AIAA Fluid Dynamics Conference.*

Patrick M. Knupp. 2001. Algebraic Mesh Quality Metrics. *SIAM J. Sci. Comput.* 23, 1 (2001), 193–218.

Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. 2009. On Centroidal Voronoi Tessellation – Energy Smoothness and Fast Computation. *ACM Trans. Graph.* 28, 4 (2009), Art. 101.

Adrien Loseille and Frédéric Alauzet. 2009. Optimal 3D Highly Anisotropic Mesh Adaptation Based on the Continuous Mesh Framework. In *Int. Meshing Roundtable.* 575–594.

Xiao-Juan Luo, Mark S. Shephard, Jean-Francois Remacle, Robert M. OBara, Mark W. Beall, Barna Szabo, and Ricardo Actis. 2002. p-Version Mesh Generation Issues. In *Int. Mesh Roundtable.* 343–354.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. 2003. *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds.* 35–57.

Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. 2008. Interactive Physically-based Shape Editing. In *Symp. Solid Phys. Modeling.* 79–89.

Edmond Nadler. 1986. Piecewise linear best L_2 approximation on triangulations. In *Approx. Theory V*, C. K. Chui et al. (Ed.). Academic Press, 499–502.

Ray W. Ogden. 1997. *Non-linear Elastic Deformations.* Dover Publications.

Per-Olof Persson and Jaime Peraire. 2009. Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics.

Elisabeth Pilgerstorfer and Bert Jüttler. 2014. Bounding the influence of domain parameterization and knot spacing on numerical stability in Isogeometric Analysis. *Comput. Methods in Appl. Mech. Eng.* 268, Supplement C (2014), 589–613.

Ulrich Pinkall and Konrad Polthier. 1993. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics* 2, 1 (1993), 15–36.

Helmut Pottmann and Michael Hofer. 2003. Geometry of the squared distance function to curves and surfaces. In *Visualization and mathematics III.* Springer, 221–242.

Laurent Rineau and Mariette Yvinec. 2008. *Meshing 3D Domains Bounded by Piecewise Smooth Surfaces.* 443–460.

Samuel Roth, Markus Gross, Silvio Turello, and Friedrich Carls. 1998. A Bernstein-Bézier Based Approach to Soft Tissue Simulation. *Computer Graphics Forum* 17, 3 (1998), 285–294.

Eloi Ruiz-Girones, Abel Gargallo-Peiró, Josep Sarrate, and Xevi Roca. 2017. An augmented Lagrangian formulation to impose boundary conditions for distortion based mesh moving and curving. In *Int. Meshing Roundtable.*

Nico Schlömer. 2018. Numerical integration (quadrature, cubature) in Python. (2018). https://github.com/nschloe/quadpy

Jonathan R. Shewchuk. 1998. Tetrahedral Mesh Generation by Delaunay Refinement. In *Symp. on Comp. Geometry.* 86–95.

Jonathan R. Shewchuk. 2002. What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Int. Meshing Roundtable.* 115–126.

Stefan Suwelack, Dimitar Lukarski, Vincent Heuveline, Rüdiger Dillmann, and Stefanie Speidel. 2013. Accurate Surface Embedding for Higher Order Finite Elements. In *Symp. Comp. Anim.* 187–192.

The CGAL Project. 2017. *CGAL User and Reference Manual* (4.11 ed.). CGAL Editorial Board. http://doc.cgal.org/4.11/Manual/packages.html

Jane Tournois, Camille Wormser, Pierre Alliez, and Mathieu Desbrun. 2009. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph.* 28, 3 (2009), Art–No.

Daniel Weber, Thomas Kalbe, André Stork, Dieter Fellner, and Michael Goesele. 2011. Interactive deformable models with quadratic bases in Bernstein–Bézier-form. *The Visual Computer* 27, 6 (2011), 473–483.

Freddie D. Witherden and Peter E. Vincent. 2015. On the identification of symmetric quadrature rules for finite element methods. *Comp. & Math. Appl.* 69, 10 (2015), 1232–1241.

Verena Ziel, Hadrien Bériot, Onur Atak, and Génaël Gabard. 2017. Comparison of 2D curving methods with modal shape functions and a piecewise linear target mesh. In

A DERIVATIVES OF E_{CODT}

To derive the gradients of the CODT energy and of the determinant of J, we first recall Jacobi's formula for an invertible matrix $\mathbf{A}$:

$$\frac{\partial \det \mathbf{A}(t)}{\partial t} = \operatorname{tr}\left[\operatorname{adj}(\mathbf{A})\frac{\partial \mathbf{A}(t)}{\partial t}\right] \tag{16}$$

where $\operatorname{adj}(\mathbf{A}) = \det(\mathbf{A})\mathbf{A}^{-1}$ is the adjugate matrix of $\mathbf{A}$. We can now derive the derivative of the determinant w.r.t. control points:

$$\begin{aligned}\frac{\partial \det \mathbf{J}}{\partial \mathbf{c_i}} &= \operatorname{tr}\left[\operatorname{adj}(\mathbf{J})\frac{\partial \mathbf{J}}{\partial \mathbf{c_i}}\right] = \operatorname{tr}\left[\operatorname{adj}(\mathbf{J})_{bi}\delta_{ia}\mathcal{B}^n_{\mathbf{i}k}\overline{\mathcal{U}}_{kj}\right] \\ &= \operatorname{adj}(\mathbf{J})_{ji}\delta_{ia}\mathcal{B}^n_{\mathbf{i}k}\overline{\mathcal{U}}_{kj} = \mathcal{B}^n_{\mathbf{i}}\overline{\mathcal{U}}\operatorname{adj}(\mathbf{J}).\end{aligned} \tag{17}$$

Similarly, the gradient of $\|\mathbf{J}\|^2$ is computed as:

$$\frac{\partial \|\mathbf{J}\|^2}{\partial \mathbf{c_i}} = 2\,\mathbf{J}_{ij}\delta_{ia}\mathcal{B}^n_{\mathbf{i}k}\overline{\mathcal{U}}_{kj} = 2\,\mathcal{B}^n_{\mathbf{i}k}\,\overline{\mathcal{U}}_{kj}\,\mathbf{J}^t_{ji}\,\delta_{ia} = 2\,\mathcal{B}^n_{\mathbf{i}}\,\overline{\mathcal{U}}\,\mathbf{J}^t. \tag{18}$$

Gradient for constant sizing field. From the expressions above, we can derive the gradient by the product rule as:

$$\begin{aligned}\frac{\partial E_{\text{CODT}}}{\partial \mathbf{c_i}} = \frac{1}{2(d+2)}\int_{\overline{\mathcal{T}}}\Bigg[&\frac{\partial \det(\mathbf{J}(\bar{\mathbf{x}}))}{\partial \mathbf{c_i}}\,\|\mathbf{J}(\bar{\mathbf{x}})\|_{\mathrm{F}}^2 \\ &+ \det(\mathbf{J}(\bar{\mathbf{x}}))\,\frac{\partial \|\mathbf{J}(\bar{\mathbf{x}})\|_{\mathrm{F}}^2}{\partial \mathbf{c_i}}\Bigg]\,d\bar{\mathbf{x}},\end{aligned} \tag{19}$$

Then based on Eqs. (17) and (18), we find directly Eq. (13).

Derivative with sizing field. Notice that when CODT energy is modulated by sizing field h, the energy becomes

$$E_{\text{CODT}} = \frac{1}{2(d+2)}\int_{\overline{\mathcal{T}}} \det(\mathbf{J})\|\mathbf{J}\|_{\mathrm{F}}^2\, h^{-(d+2)}(\mathbf{x})\,d\bar{\mathbf{x}}. \tag{20}$$

Its gradient with respect to a control point $\mathbf{c_i}$ thus involves a gradient of sizing field as well:

$$\begin{aligned}\frac{\partial E_{\text{CODT}}}{\partial \mathbf{c_i}} = &\frac{1}{2(d+2)}\int_{\overline{\mathcal{T}}}\frac{\partial \det(\mathbf{J})\|\mathbf{J}\|_{\mathrm{F}}^2}{\partial \mathbf{c_i}}\,h^{-(d+2)}(\mathbf{x})\,d\bar{\mathbf{x}} \\ &-\frac{1}{2}\int_{\tau_0} h^{-(d+3)}(\mathbf{x})\frac{\partial h(\mathbf{x})}{\partial \mathbf{c_i}}\det(\mathbf{J})\|\mathbf{J}\|^2 d\bar{\mathbf{x}}\end{aligned} \tag{21}$$

Since we know that

$$\mathbf{x}(\mathbf{u}(\bar{\mathbf{x}})) = \sum_{|\mathbf{i}|=n}\mathbf{c_i}\,B^n_{\mathbf{i}}(\mathbf{u}(\bar{\mathbf{x}})) \text{ and } \frac{\partial \mathbf{x}}{\partial \mathbf{c_i}} = B^n_{\mathbf{i}}(\mathbf{u}(\bar{\mathbf{x}}))\mathbf{Id}, \tag{22}$$

we can compute the gradient of h w.r.t. to $\mathbf{c_i}$ as:

$$\frac{\partial h(\mathbf{x})}{\partial \mathbf{c_i}} = \nabla_{\mathbf{x}}h(\mathbf{x})\frac{\partial \mathbf{x}}{\partial \mathbf{c_i}} = \nabla_{\mathbf{x}}h(\mathbf{x})B^n_{\mathbf{i}}(\mathbf{u}(\bar{\mathbf{x}}))\,\mathbf{Id} = B^n_{\mathbf{i}}(\mathbf{u}(\bar{\mathbf{x}}))\nabla_{\mathbf{x}}h(\mathbf{x}). \tag{23}$$

Finally, we deduce that:

$$\begin{aligned}\frac{\partial E_{\text{CODT}}}{\partial \mathbf{c_i}} = &\frac{n}{2(d+2)}\int_{\tau_0} h^{-(d+2)}(\mathbf{x})\mathcal{B}^n_{\mathbf{i}}\overline{\mathcal{U}}(\|\mathbf{J}\|_{\mathrm{F}}^2\mathbf{J}^{-1} + 2\mathbf{J}^t)\det(\mathbf{J})d\bar{\mathbf{x}} \\ &-\frac{1}{2}\int_{\tau_0} h^{-(d+3)}(\mathbf{x})\,B^n_{\mathbf{i}}(\mathbf{u}(\bar{\mathbf{x}}))\nabla_{\mathbf{x}}h(\mathbf{x})\det(\mathbf{J})\|\mathbf{J}\|_{\mathrm{F}}^2 d\bar{\mathbf{x}}.\end{aligned} \tag{24}$$

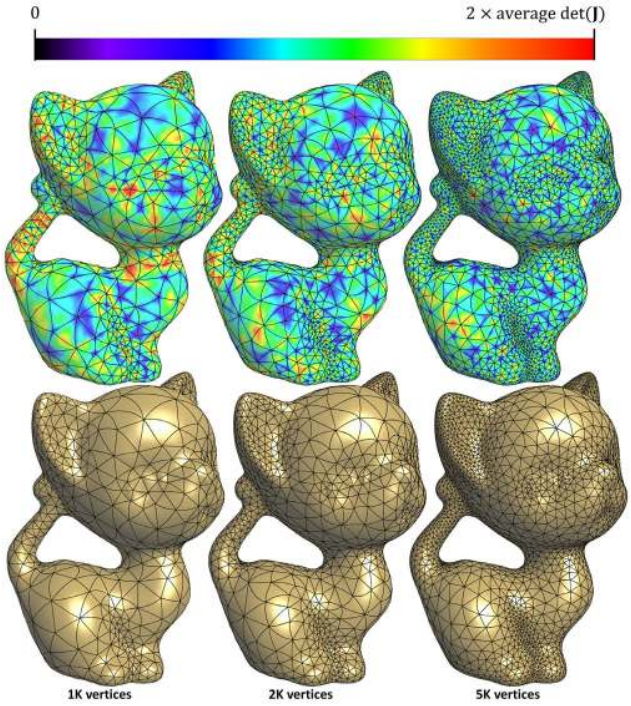


Fig. 17. **Kitten.** This kitten model is approximated with order-3 curved meshes for an increasing number of vertices (bottom), with their corresponding sizing-adapted Jacobian determinant (top).

B FORMULAS FOR $\overline{\mathcal{U}}$

The barycentric coordinates of a point $\bar{\mathbf{x}}$ in $\mathbb{R}^d$ inside a unit, regular simplex $\bar{\tau}$ defined by vertices $\bar{\mathbf{v}}_1,\ldots,\bar{\mathbf{v}}_{d+1}$ (their exact embedding has no significance, and can be chosen arbitrarily as long as every edge has unit length) are: $\mathbf{u}(\bar{\mathbf{x}}) = [u_1(\bar{\mathbf{x}}),\ldots,u_{d+1}(\bar{\mathbf{x}})]$ with

$$u_k = \frac{|\bar{\mathbf{v}}_1,\ldots,\bar{\mathbf{v}}_{k-1},\bar{\mathbf{v}},\bar{\mathbf{v}}_{k+1},\ldots,\bar{\mathbf{v}}_{d+1}|}{|\bar{\mathbf{v}}_1,\ldots,\bar{\mathbf{v}}_{d+1}|} \tag{25}$$

where $|\cdot|$ represents the signed volume in $\mathbb{R}^d$ (area in 2D, volume in 3D) of the simplex formed by the vertices in the argument.

The Jacobian matrix $\overline{\mathcal{U}}$ corresponding to the differential of $\mathbf{u}$ with respect to $\bar{\mathbf{x}}$ is simple to assemble in both 2D and 3D, respectively:

$$\overline{\mathcal{U}} = \frac{1}{2|\bar{\tau}|}\begin{bmatrix}(\bar{\mathbf{v}}_2-\bar{\mathbf{v}}_3)^T\\(\bar{\mathbf{v}}_3-\bar{\mathbf{v}}_1)^T\end{bmatrix}\begin{bmatrix}0 & -1\\1 & 0\end{bmatrix} \tag{26}$$

$$\overline{\mathcal{U}} = \frac{1}{3|\bar{\tau}|}\begin{bmatrix}\mathbf{N}(\bar{\mathbf{v}}_2,\bar{\mathbf{v}}_4,\bar{\mathbf{v}}_3)^T\\\mathbf{N}(\bar{\mathbf{v}}_1,\bar{\mathbf{v}}_3,\bar{\mathbf{v}}_4)^T\\\mathbf{N}(\bar{\mathbf{v}}_1,\bar{\mathbf{v}}_4,\bar{\mathbf{v}}_2)^T\\\mathbf{N}(\bar{\mathbf{v}}_1,\bar{\mathbf{v}}_2,\bar{\mathbf{v}}_3)^T\end{bmatrix} \tag{27}$$

where $\mathbf{N}(\mathbf{a},\mathbf{b},\mathbf{c})$ is the area-weighted face normal of triangle $(\mathbf{a},\mathbf{b},\mathbf{c})$:

$$\mathbf{N}(\mathbf{a},\mathbf{b},\mathbf{c}) = \frac{1}{2}(\mathbf{b}\times\mathbf{c} + \mathbf{c}\times\mathbf{a} + \mathbf{a}\times\mathbf{b}). \tag{28}$$