Optimizing Voronoi Diagrams for Polygonal Finite Element Computations

Daniel Sieger, Pierre Alliez, Mario Botsch

Optimizing Voronoi Diagrams for Polygonal Finite Element Computations

Daniel Sieger[1], Pierre Alliez[2], and Mario Botsch[1]

[1] Bielefeld University, Germany
{dsieger,botsch}@techfak.uni-bielefeld.de
[2] INRIA Sophia-Antipolis, France
pierre.alliez@sophia.inria.fr

Summary. We present a 2D mesh improvement technique that optimizes Voronoi diagrams for their use in polygonal finite element computations. Starting from a centroidal Voronoi tessellation of the simulation domain we optimize the mesh by minimizing a carefully designed energy functional that effectively removes the major reason for numerical instabilities—short edges in the Voronoi diagram. We evaluate our method on a 2D Poisson problem and demonstrate that our simple but effective optimization achieves a significant improvement of the stiffness matrix condition number.

1 Introduction

The finite element method (FEM) is an indispensable tool for solving partial differential equations in applied mathematics, computational physics, medical simulations and computer animation. The simulation domain is typically discretized into triangle and quadrangle elements in 2D, or tetrahedral and hexahedral elements in 3D, respectively. These conventional elements are well established and allow for simple and efficient implementations. However, they require complex remeshing when it comes to adaptive refinement, merging, cutting, or fracturing, i.e., general topological changes of the simulation domain [26, 27, 46, 45, 8, 44].

As a consequence, generalizations of FE methods to arbitrary polygonal or polyhedral meshes have gained increasing attention, both in computational physics [34, 36, 37] and in computer graphics [43, 25]. For instance, when cutting a tetrahedron into two pieces, polyhedral FEM can directly process the resulting two elements, while standard FEM has to remesh them into tetrahedra first. Polygonal FEM was also shown to yield good results in topology optimization, since the resulting meshes are less biased to certain principle directions than the typically employed regular tessellations [37].

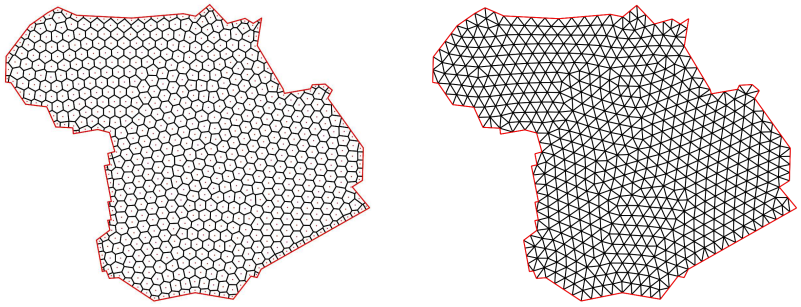

Fig. 1. A centroidal Voronoi tessellation (CVT) generated by interleaving refinement and Lloyd relaxation [38] (left). Most elements are well-shaped quasi-regular hexagons. The dual constrained Delaunay triangulation is shown on the right.

Independent of the employed element type, the accuracy and stability of numerical computations crucially depend on a high quality mesh consisting of well-shaped, non-degenerate elements only. A single bad element may be sufficient to destroy the numerical conditioning [30].

For triangular and tetrahedral meshes quality criteria for element shapes are well understood [30], and successful mesh generation and optimization techniques based on Delaunay refinement [28, 29, 7] or variational optimization [38, 39, 41] have been proposed.

In contrast, research on similar criteria and optimization techniques for general polygonal meshes has not yet reached a mature state. Most polygonal FE methods use Voronoi tessellations to discretize the simulation domain [35, 32, 37]. The corresponding mesh generation and optimization is then typically based on centroidal Voronoi tessellations (CVTs, see Figure 1). However, the resulting element quality and numerical conditioning is not yet comparable to high quality triangle or tetrahedral meshes.

In this paper we show that even high-quality CVT meshes can contain numerically ill-conditioned elements, which is due to very short edges in the Voronoi diagram. We present a simple variational mesh optimization that effectively removes short edges from a CVT input mesh by minimizing a suitable energy functional. We evaluate our method on a 2D Poisson problem by comparing the stiffness matrix condition number before and after optimization. Our method reliably improves the numerical condition—up to several orders of magnitude for complex examples.

2 Background

The numerical robustness of solving a partial differential equation (PDE) depends on the condition number of the resulting linear system, which itself is strongly influenced by the shape of the finite elements [30]. Our goal is to optimize the mesh in order to improve the numerical conditioning.

The relation between the element shape and the resulting matrix condition number cannot be answered in general as it depends on the type of PDE to be solved. In this paper we follow [30] and focus on the Poisson equation $-\Delta u = f$, since this problem has a large number of applications and is very similar in structure to linear elasticity computations.

In the following we briefly review the basic concepts of (polygonal) finite element methods, and refer the reader to the textbooks [4, 20] and the survey [34] for more details on classical FEM and polygonal FEM, respectively.

For a 2D Poisson problem we find a function $u\colon \mathbb{R}^2 \to \mathbb{R}$ such that

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \subset \mathbb{R}^2 \\ u &= \bar{u} \quad \text{on } \partial\Omega. \end{aligned} \tag{1}$$

We choose $f = \Delta\bar{u}$ such that $\bar{u}$ is the (known) solution of the PDE, to which we can compare our approximation. The function u is approximated by n basis functions N_i interpolating the nodal degrees of freedom u_i

$$u(\mathbf{x}) \approx \sum_{i=1}^{n} u_i N_i(\mathbf{x}). \tag{2}$$

This approximation is inserted into the weak form of the problem (1), which leads to a linear system to be solved for the nodal values u_i:

$$\mathbf{K}\begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \qquad \text{with} \qquad \begin{aligned} \mathbf{K}_{i,j} &= \textstyle\int_\Omega \nabla N_i \cdot \nabla N_j \,\mathrm{d}\Omega \\ \mathbf{f}_i &= \textstyle\int_\Omega f\, N_i \,\mathrm{d}\Omega. \end{aligned} \tag{3}$$

How difficult it is to numerically solve (3) depends on the condition number $\kappa(\mathbf{K}) = \lambda_{\max}/\lambda_{\min}$ of the stiffness matrix $\mathbf{K}$ [15]. A large condition number causes iterative solvers, such as conjugate gradients [3], to converge very slowly and direct solvers, such as sparse Cholesky factorizations [6], to suffer from round-off errors. For extreme condition numbers the linear system cannot be solved at all. While the smallest eigenvalue $\lambda_{\min}$ mainly depends on the size of the smallest element, the largest eigenvalue $\lambda_{\max}$ may become arbitrarily large already for a single badly-shaped element [30]. As a consequence, one bad element is sufficient to spoil the numerical robustness.

For triangle or tetrahedral meshes it is well known what makes up a numerically robust linear element [30]: Both small and large angles must be avoided. To this end, a number of very successful mesh improvement approaches have been proposed, being based on either Delaunay refinement [28, 29, 9] or some variational optimization [1, 38, 22, 39, 10, 41]. So-called *pliant* methods, which combine local topological changes (e.g., Delaunay refinement) and vertex relocation (e.g., Laplacian smoothing or Lloyd relaxation) have been found to be superior over methods involving one of the techniques only [5, 22, 38].

When general polygonal or polyhedral elements are to be used for FE computations, the N_i are no longer the classical linear barycentric coordinates, but are replaced by *generalized barycentric coordinates* [33]: These functions interpolate nodal values ($N_i(\mathbf{x}_j) = \delta_{ij}$), are linear on element edges, form a partition of unity ($\sum_i N_i \equiv 1$), are bounded non-negative ($0 \leq N_i \leq 1$), are C^0 across and C^∞ within elements, and have linear precision. Examples of generalized barycentric coordinates are Wachspress coordinates [42], mean value coordinates [12, 13, 18, 43], Laplace interpolants [17], harmonic coordinates [21, 25], and maximum entropy coordinates [31, 19].

For general polygonal or polyhedral meshes quality criteria for their numerical robustness are not well-established. One can observe, however, that short edges, small angles, and concave corners lead to numerical problems. As a consequence, Voronoi tessellation of the simulation domain are frequently used, since they guarantee convex elements. Well-shaped, isotropic Voronoi cells can be achieved by Lloyd clustering [38] or energy minimization [23], leading to centroidal Voronoi tessellations (CVTs, see Figure 1).

While CVTs guarantee convex elements and lead to well-behaved angles, they do not explicitly penalize short edges. Consequently, even "nice" CVT meshes may contain a number of very short Voronoi edges (see Figure 2). Since on a short edge $(\mathbf{x}_i, \mathbf{x}_j)$ the shape function N_i decreases from 1 (at $\mathbf{x}_i$) to 0 (at $\mathbf{x}_j$) over the very short distance $\|\mathbf{x}_i - \mathbf{x}_j\|$, the gradient ∇N_i becomes extremely large. These large gradients enter the stiffness matrix through the products $\int \nabla N_i \cdot \nabla N_j$ and lead to a large maximum eigenvalue $\lambda_{\max}$, and thus to a high condition number $\kappa = \lambda_{\max}/\lambda_{\min}$. Note that this is independent of the generalized barycentric coordinate employed, since they all satisfy the Lagrange property $N_i(\mathbf{x}_i) = \delta_{ij}$ and are linear on edges, i.e., they are actually identical on edges of the tessellation. Note also that a single short edge (a single bad element) may destroy the numerical robustness of the FE computation.

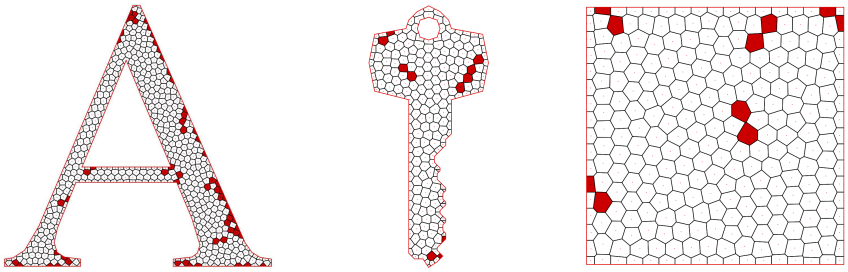

Fig. 2. While providing isotropic cells, centroidal Voronoi tessellations may still contain elements with short edges. Voronoi cells incident to an edge shorter than 5% of the specified edge length are depicted in red.

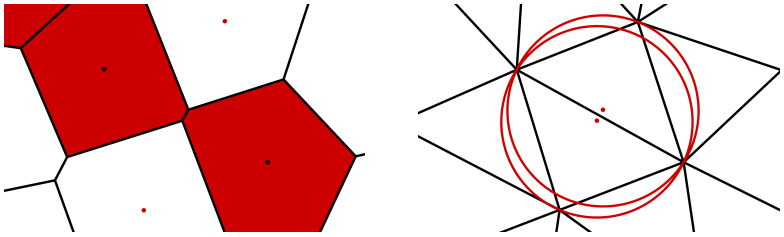

Fig. 3. Short edges in the Voronoi diagram (left) correspond to triangle circumcenters of the dual Delaunay mesh (right) being spatially close to another.

3 Mesh Optimization

We now describe a mesh improvement scheme that effectively removes short edges from a CVT, while keeping all the beneficial properties of a Voronoi tessellation. Since the Voronoi diagram is fully defined by the vertices $\mathbf{v}_i$ of its dual (constrained) Delaunay triangulation, the m Delaunay vertices $(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ are the degrees of freedom of our optimization.

Let us first investigate the relationship between the Delaunay vertices $\mathbf{v}_i$ and short edges $(\mathbf{x}_i, \mathbf{x}_j)$ in the Voronoi diagram. The Voronoi vertices $\mathbf{x}_i$ are circumcenters of Delaunay triangles. A short Voronoi edge $(\mathbf{x}_i, \mathbf{x}_j)$ corresponds to two spatially close circumcenters $\mathbf{x}_i$ and $\mathbf{x}_j$ (see Figure 3). While a CVT maximizes the compactness of Voronoi cells [23] and yields well-shaped dual Delaunay triangles, it does *not* prevent short Voronoi edges. As depicted in Figure 3, four almost co-circular Delaunay vertices lead to two almost coincident circumcenters, i.e., to a short Voronoi edge.

We propose to prevent circumcenters from coming close to each other by shifting them as much as possible into the interior of their corresponding triangles. The point that is "as interior as possible" is the triangle's incenter, which maximizes the distance from the triangle's edges. The question is therefore how to relocate the Delaunay vertices $\mathbf{v}_i$ such that the triangles' circumcenters are shifted towards their incenters. To this end we setup an energy that measures the (squared) distances of circumcenters to incenters and minimize this energy with respect to the Delaunay vertices $\mathbf{v}_i$.

The squared distance d^2 from a triangle's circumcenter to its incenter can elegantly be computed from its circumradius R and inradius r through Euler's triangle formula as $d^2 = R\,(R - 2r)$. Our energy simply accumulates the squared distances of all triangles $t \in \mathcal{T}$:

$$E(\mathbf{v}_1, \ldots, \mathbf{v}_m) \;=\; \frac{1}{2} \sum_{t \in \mathcal{T}} R_t \,(R_t - 2r_t) \;.$$

For a triangle with edge lengths a, b, c and area A (see Figure 4) the circumradius R and inradius r can be computed as

$$R \;=\; \frac{abc}{4A} \qquad \text{and} \qquad r \;=\; \frac{2A}{a+b+c} \,.$$

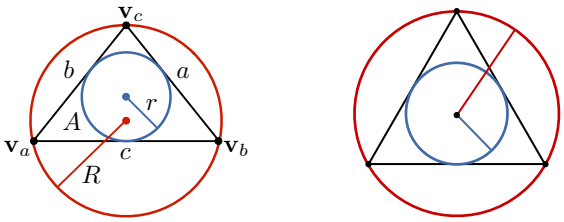

Fig. 4. Left: A triangle $(\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c)$ with edge lengths a, b, c, area A, circumradius R, and inradius r. Right: For equilateral triangles circumcenter and incenter coincide.

The simple form of E allows us to evaluate and minimize the energy easily and efficiently. The energy landscape is very steep, such that it turned out to be sufficient to minimize E by a simple iterative gradient descent [14]. The resulting mesh optimization algorithm is sketched in the pseudo-code below and described in more detail in the following.

Algorithm 1: Mesh Optimization

Input: Constrained Delaunay Triangulation with vertices $\mathbf{v}_i \in \mathcal{V}$ and faces $\mathcal{T}$

```
// Flip edges to improve vertex valences
T = improve_valences(T)
// Initialize vector of vertices
V^(0) = (v_1, ..., v_m)
// Iteratively minimize energy
for k = 0, 1, 2, ... do
    // Compute gradient
    G = ∇E(V^(k))
    // Determine step size
    for h = 1, 1/2, 1/4, ..., ε do
        if E(V^(k) − h G) < E(V^(k)) then
            // Update vertex positions
            V^(k+1) = V^(k) − h G
        end
    end
    // Convergence check
    if h == ε then
        // Re-establish Delaunay property
        return Delaunay(V^(k), T)
    end
end
```

For the iterative optimization we stack all degrees of freedom, i.e., the positions of the Delaunay vertices, into a $2m$-dimensional vector $\mathbf{V}^{(0)} = (\mathbf{v}_1^T, \ldots, \mathbf{v}_m^T)^T$. In each iteration k all vertices are shifted in the direction of the negative gradient $-\nabla E(\mathbf{V}^{(k)})$ by a step size h:

$$\mathbf{V}^{(k+1)} \leftarrow \mathbf{V}^{(k)} - h\,\nabla E(\mathbf{V}^{(k)}). \tag{4}$$

Gradient Computation (Line 4): In order to construct the gradient, we have to compute the partial derivatives of E with respect to all vertices $\mathbf{v}_a$, which thanks to the simple energy can be done analytically:

$$\frac{\partial E}{\partial \mathbf{v}_a} = \sum_{t \in \mathcal{T}} \left[(R_t - r_t)\, \frac{\partial R_t}{\partial \mathbf{v}_a} - R_t\, \frac{\partial r_t}{\partial \mathbf{v}_a} \right].$$

The ingredients are the partial derivatives of circumradius and inradius, which also have a simple analytical form. For a triangle $(\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c)$ the derivatives of circumradius and inradius are

$$\frac{\partial R}{\partial \mathbf{v}_a} = R \left[\frac{1}{c^2} (\mathbf{v}_a - \mathbf{v}_b) + \frac{1}{b^2} (\mathbf{v}_a - \mathbf{v}_c) - \frac{1}{2A} (\mathbf{v}_c - \mathbf{v}_b)^\perp \right],$$

$$\frac{\partial r}{\partial \mathbf{v}_a} = \frac{-2}{(a+b+c)^2} \left[\frac{A}{c} (\mathbf{v}_a - \mathbf{v}_b) + \frac{A}{b} (\mathbf{v}_a - \mathbf{v}_c) - \frac{a+b+c}{2} (\mathbf{v}_c - \mathbf{v}_b)^\perp \right],$$

where $(x, y)^\perp = (-y, x)$ denotes rotation by 90°.

Step Size (Lines 5, 6): After computing the gradient, we determine the step size h by a simple bisection line search. Starting with $h = 1$ we successively halve h until the step decreases the energy. Once we found a step size h to reduce the energy we perform the step. In our experiments, we chose $\varepsilon =$ `FLT_MIN`. In order to prevent triangles from inverting during the optimization we return an infinite energy value as soon as a triangle has negative area. We stop the iterative optimization if even the minimum step size $h = \varepsilon$ does not manage to further decrease the energy.

Vertex Update (Line 7): After computing the gradient and determining a step size all vertex positions are updated at once. Unconstrained vertices $\mathbf{v}_i$ in the *interior* of the domain Ω are simply moved in the direction of their negative gradient as described by (4). Constrained vertices on the *boundary* $\delta\Omega$ need special treatment, where we consider two different cases: Boundary vertices representing constrained corners of the boundary polygon are not moved. Vertices on straight boundary edges (that have been refined by the initial CVT computation) are optimized along the constrained edge by projecting their gradients onto that edge.

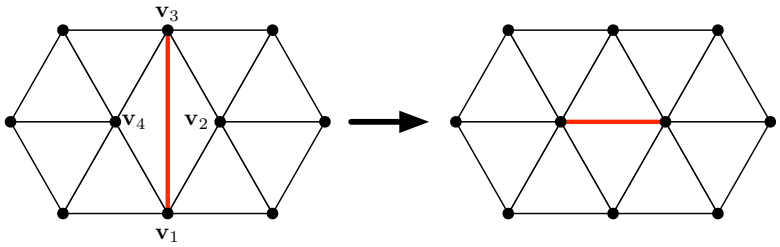

Fig. 5. In a pre-process we flip edges to reduce the deviation from valence 6.

Boundary Weighting: The Voronoi cells on the domain boundary are clamped against the input boundary polygon using a bounded Voronoi diagram [38] (see Figures 1, 2). As a consequence, Voronoi edges that are dual to the domain boundary are clipped, thereby roughly halving the lengths of these Voronoi edges. The optimization therefore has to try about twice as hard to keep their interior edge endpoints (which are circumcenters of boundary triangles) away from the domain boundary in order to avoid short Voronoi edges. We therefore give boundary triangles twice the weight by multiplying their contributions to both the energy and the gradient by 2. This simple heuristic effectively avoids situations where interior triangles push the circumcenters of boundary triangles towards (or even across) boundary edges.

Preprocessing (Line 1): As shown in Figure 4, minimizing the distance between circumcenter and incenter also optimizes for equilateral triangles, since those are minima of the energy. Perfectly equilateral triangles, however, can only be achieved around vertices of valence 6. In a pre-process we therefore modify the input triangulation in order to bring the valences closer to the optimal valence 6 (respectively 4 on the boundary). In particular, we repeatedly iterate over all edges and flip an edge if this decreases the squared deviation from the optimal valences of the four affected vertices $\mathbf{v}_1, \dots, \mathbf{v}_4$:

$$\sum_{i=1}^{4} \left(\text{valence}(\mathbf{v}_i) - \text{optimal\_valence}(\mathbf{v}_i)\right)^2$$

This simple topological preprocessing allows for significantly better results of the geometric optimization of vertex positions.

Delaunay Update (Line 11): After flipping edges in a pre-process and changing all vertex positions during the optimization, the resulting triangulation might no longer be a Delaunay triangulation. Since in the end we rely on the Delaunay property in order to obtain a valid Voronoi diagram with convex elements, we finally reconstruct a (constrained) Delaunay triangulation $(\mathbf{V}^{(k)}, \mathcal{T})$. To this end, we simply check each edge of the triangulation and flip it if the Delaunay property is not satisfied. A more efficient implementation would be possible by filtering Delaunay relocations, as described in [24].

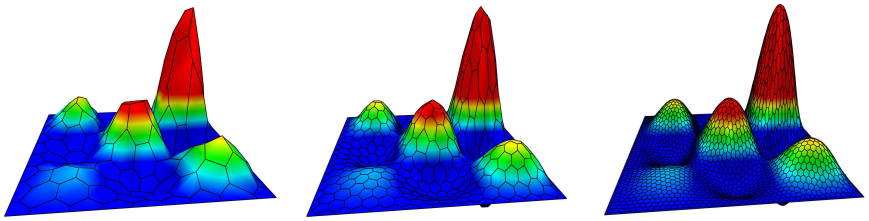

Fig. 6. Polygonal FEM approximation with Voronoi meshes of increasing degrees of freedom: 187 (left), 712 (middle), 4760 (right).

Note that even before this step the mesh is already almost a Delaunay triangulation, since the geometric optimization leads to very regular triangulations with well-shaped faces. Consequently, only very few edge flips are necessary, which therefore hardly affects the energy.

4 Results

In this section we evaluate our mesh improvement technique based on the 2D Poisson problem (1) with the constraint function

$$\bar{u}(x, y) \;=\; xy \,\sin(3\pi x)\,\sin(3\pi y).$$

For the numerical experiments we employ mean value coordinates [18] as the shape functions N_i in (2). We also compared to harmonic coordinates [21] and maximum entropy coordinates [19], but found that the choice of the shape function does not make a significant difference. We solve the linear system (3) using the sparse Cholesky factorization of CHOLMOD [6]. Figure 6 shows the resulting approximations for several resolutions.

Our mesh optimization technique is implemented in `C++` using the CGAL library [11]. It requires less than one second for the smaller examples and about one minute for the complex mesh of Figure 10 (Mac Pro, 2.66 GHz Intel Xeon, single-threaded). In order to evaluate the numerical improvement we compare the condition numbers before and after the mesh optimization. Our experiments indicate that edges in the range of 1%–5% of the mean edge length start to severely hurt the numerical robustness. In the following figures we therefore highlight elements with edges shorter than 5% of the target edge length specified by the sizing field.

As depicted by Figure 7 a CVT of a trivial shape may already contain arbitrarily short edges. Our mesh improvement technique removes all edges shorter than 5% of the specified edge length and decreases the condition number from 2943 to 407. On the slightly more complex A-shape shown in Figure 8 the optimization also removes all the short edges below 5% and thereby reduces the condition number from 421 to 75.

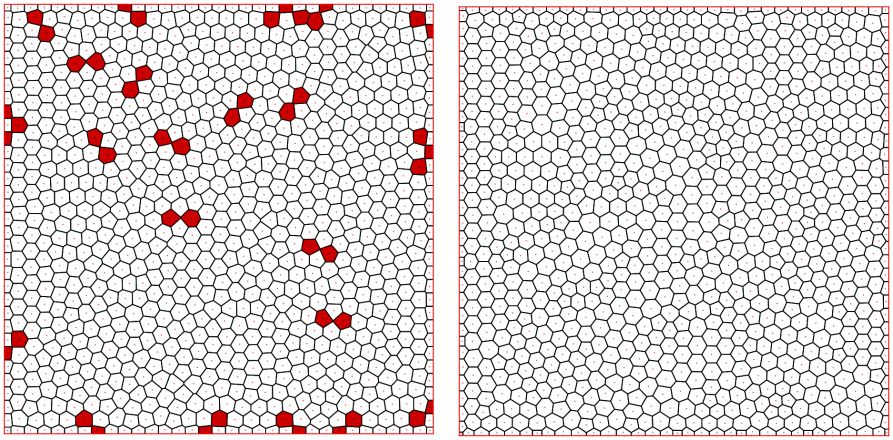

Fig. 7. Comparison between a CVT (left) and an optimized mesh (right) for the unit square. The underlying Delaunay triangulation contains 1722 triangles. The condition number decreases from 2943 to 407.

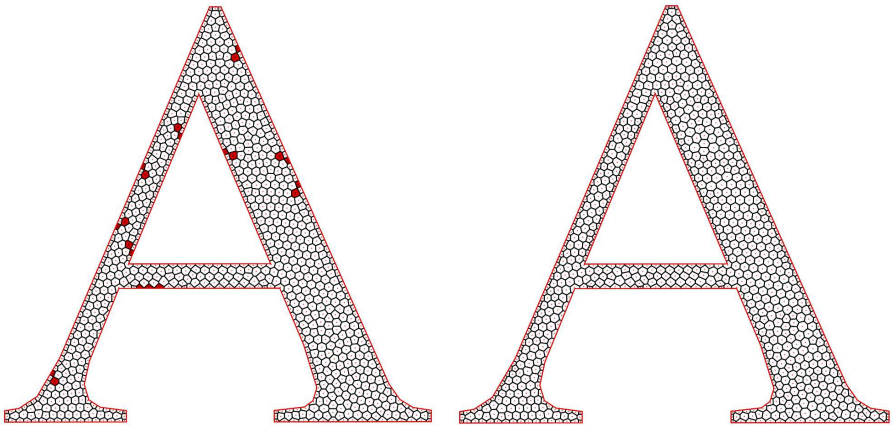

Fig. 8. CVT (left) and optimized mesh (right), corresponding to a dual Delaunay triangulation of 1799 triangles. The condition number decreases from 421 to 75.

Mesh sizing is an important aspect of high quality mesh generation, for instance for complex input shapes or if higher accuracy is required in certain regions of the simulation domain. Although our simple energy does not explicitly take the sizing field into account, the grading is nicely preserved while short edges are removed. We experimented with an explicit incorporation of the sizing function as well as with a normalization of the squared distance of circum- and incenter by the triangle's area, but did not notice any significant difference, such that we kept our simple energy formulation.

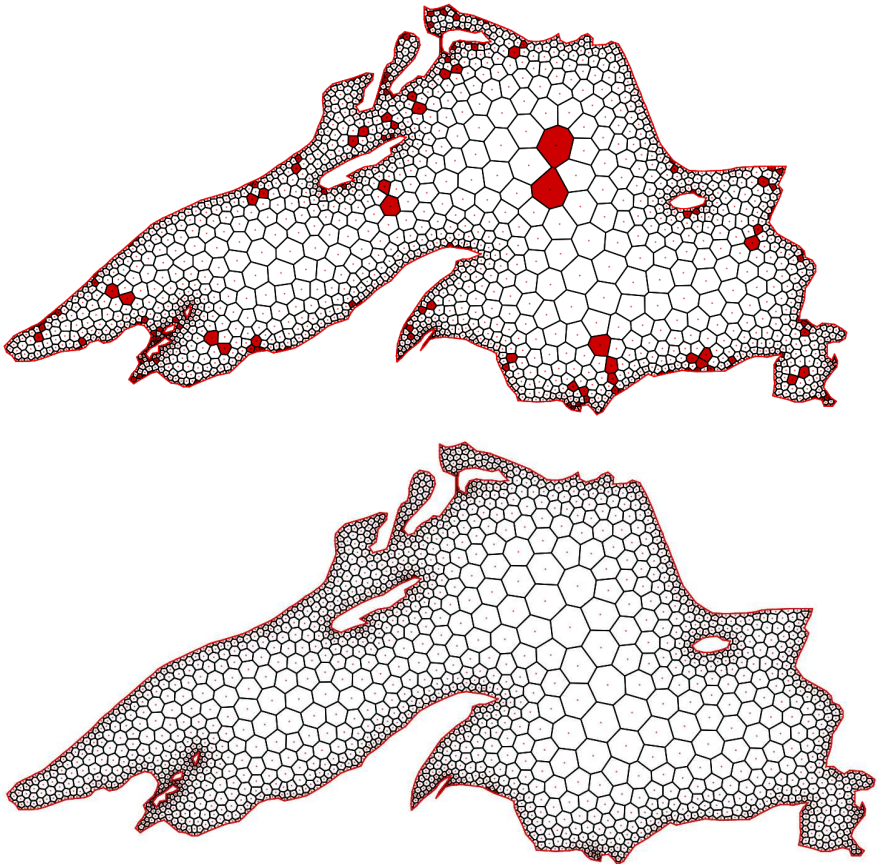

Fig. 9. CVT (top) and optimized mesh (bottom) for Lake Superior using a K-Lipschitz sizing function [2] with $K = 0.7$. The underlying Delaunay triangulation contains 4036 triangles. The condition number reduces from 371877 to 190.

Optimization results for graded meshes are shown in Figure 9 for the Lake Superior and in Figure 10 for a strongly graded mesh generated from a gray-level photograph. Note that the extreme condition number for the CVT of the Lake Superior is not necessarily a typical example, but it demonstrates that the chance for degenerate edges in complex CVT meshes increases significantly. The high condition number in Figure 10 is mainly due to the high number of elements and the extreme grading field.

We compared the results of our mesh optimization to two other methods capable of removing short edges from Voronoi diagrams, namely (1) collapsing the 10% shortest edges and (2) Laplacian smoothing of the Voronoi vertices (see [10] for an overview). Table 1 compares the condition numbers resulting from the different optimization techniques.

Fig. 10. CVT (left) and optimized mesh (right) for a complex highly-graded mesh generated from a photograph. The underlying Delaunay triangulation contains 113196 triangles. The condition number reduces from 354630 to 44208.

	CVT	Ours	EC	LS
Unit Square	2943	407	401	274
A-Shape	421	75	66	43
Lake Superior	371877	190	200797	1.2288e06
Photo	354630	44208	352170	39976

Table 1. Comparison of the stiffness matrix condition number for different mesh improvement techniques. From left to right: initial Centroidal Voronoi Tessellation (CVT), our energy minimization, collapsing short edges (EC), and Laplacian smoothing (LS).

While edge collapses and Laplacian smoothing improve the conditioning on the smaller examples, they do not work reliably on the more complex meshes. In contrast, our optimization manages to improve the condition number for all examples. It is important to note that both alternative methods no longer maintain the dual relationship to the underlying Delaunay triangulation and might lead to non-convex elements.

Furthermore, they suffer from another disadvantage: The resulting elements in the polygonal mesh are not as well-shaped as in the original CVT or as in our method, which leads to a decrease in approximation accuracy. The L^2 errors for the Poisson problem (1) on the different Unit Square meshes (Figures 6 and 7) are 3.2e-6 for our method, 2.4e-5 for edge collapses, and 4.4e-6 for Laplacian smoothing.

5 Discussion

Our mesh optimization technique avoids short Voronoi edges by moving each triangle's circumcenter towards its incenter. Despite its simplicity this approach reliably improves the numerical conditioning in all our experiments.

Although motivated by a different downstream application, we realize that our energy is closely related to the concept of *well-centered triangulations* [41]. A triangulation is called well-centered if each of its triangles contains its circumcenter, a property important for simulations based on discrete exterior calculus [16].

VanderZee and colleagues compute well-centered triangulations by iteratively maximizing the (signed) distance of mesh vertices to the opposite edges of their incident triangles, normalized by circumradii [41]. This per-vertex measure is minimized with respect to an L^p-norm, with p typically being 4, 6, 8, or 10, or a combination thereof. The minimization is performed by a nonlinear conjugate gradients solver with carefully tuned line-search and using numerically approximated gradients.

In a recent comparison to other mesh improvement schemes [10], the well-centered optimization (its earlier but equivalent 2D version [40]) was rated as complicated to implement and computationally rather slow. In contrast, our simple L^2 energy can be successfully minimized with an easy-to-implement gradient descent based on analytical gradients. For the smaller examples (Unit Square, A-Shape, Lake Superior) the optimization took less than a second, for the complex Photo-mesh it took about one minute. In all examples our method led to a significant improvement in numerical conditioning.

Although not our primary goal, we tested our optimized meshes for well-centeredness. Besides situations where the mesh connectivity simply does not allow for well-centered triangles (e.g., around vertices with valence 3 or 4), almost all triangles are well-centered.

An interesting direction for future work would therefore be to combine our simple energy minimization with the more sophisticated topological preprocessing proposed in [40], which could lead to a simpler optimization for well-centered triangulations. Furthermore, our energy can be generalized to 3D in a straightforward manner, which is another promising direction for further investigations.

Acknowledgments

The authors are grateful for N. Sukumar for inspiring discussion about polygonal FEM and for providing source code for Maximum Entropy Coordinates. Daniel Sieger and Mario Botsch are supported by the Deutsche Forschungsgemeinschaft (Center of Excellence in "Cognitive Interaction Technology", CITEC). The authors thank Jonathan R. Shewchuk for providing the various input shapes used throughout this paper. We thank Jan for growing a mustache and providing the awesome photo used for Figure 10.

References

1. P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24(3):617–625, 2005.
2. L. Antani, C. Delage, and P. Alliez. Mesh sizing with additively weighted Voronoi diagrams. In *Proceedings of the 16th International Meshing Roundtable*, pages 335–346, 2007.
3. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 2nd edition, 1994.
4. K. J. Bathe. *Finite Element Procedures*. Prentice Hall, second edition, 1995.
5. F. J. Bossen and P. S. Heckbert. A pliant method for anisotropic mesh generation. In *Proceedings of 5th International Meshing Roundtable*, pages 63–74, 1996.
6. Y. Chen, T. Davis, W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, 35(3):1–14, 2008.
7. S. W. Cheng, T. K. Dey, and J. Levine. A Practical Delaunay Meshing Algorithm for a Large Class of Domains. In *Proc. of the 16th Int. Meshing Roundtable*, pages 477–494, 2007.
8. N. Chentanez, R. Alterovitz, D. Ritchie, L. Cho, K. K. Hauser, K. Goldberg, J. R. Shewchuk, and J. F. O'Brien. Interactive simulation of surgical needle insertion and steering. *ACM Transactions on Graphics*, 28(3):88:1–88:10, 2009.
9. H. Erten and A. Üngör. Triangulations with locally optimal steiner points. In *SGP07: Eurographics Symposium on Geometry Processing*, pages 143–152, 2007.
10. H. Erten, A. Üngör, and C. Zhao. Mesh smoothing algorithms for complex geometric domains. In *Proceedings of 18th International Meshing Roundtable*, pages 175–193, 2009.
11. A. Fabri, G. Giezeman, L. Kettner, and S. Schirra. On the design of CGAL a computational geometry algorithms library. *Software Practice and Experience*, 30(11):1167–1202, 2000.
12. M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.
13. M. S. Floater, G. Kos, and M. Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22:623–631, 2005.
14. P. R. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, 1981.
15. G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989.
16. A. N. Hirani. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology, 2003.
17. H. Hiyoshi and K. Sugihara. Two generalizations of an interpolant based on Voronoi diagrams. *International Journal of Shape Modeling*, 5(2):219–231, 1999.
18. K. Hormann and M. S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics*, 25(4):1424–1441, 2006.
19. K. Hormann and N. Sukumar. Maximum entropy coordinates for arbitrary polytopes. *Computer Graphics Forum*, 27(5):1513–1520, 2008.
20. T. J. R. Hughes. *The Finite Element Method*. Dover Publications, 2000.

21. P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics*, 26(3):71, 2007.
22. B. W. Klingner and J. R. Shewchuk. Agressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, 2007.
23. Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellation—energy smoothness and fast computation. *ACM Trans. on Graphics*, 28(4):1–17, 2009.
24. P. M. Manhães de Castro, J. Tournois, P. Alliez, and O. Devillers. Filtering relocations on a delaunay triangulation. *Computer Graphics Forum (Symp. on Geometry Processing)*, 28(5):1465–1474, 2009.
25. S. Martin, P. Kaufmann, M. Botsch, M. Wicke, and M. Gross. Polyhedral finite elements using harmonic basis functions. *Computer Graphics Forum*, 27(5):1521–1529, 2008.
26. J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Transactions on Graphics*, pages 291–294, 2002.
27. J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of ACM SIGGRAPH 1999*, pages 137–146, 1999.
28. J. R. Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, Pittsburg, 1997.
29. J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 22:21–74, 2002.
30. J. R. Shewchuk. What is a good linear finite element? Interpolation, conditioning, anisotropy, and quality measures. Unpublished Preprint, 2002.
31. N. Sukumar. Construction of polygonal interpolants: A maximum entropy approach. *International Journal for Numerical Methods in Engineering*, 61(12):2159–2181, 2004.
32. N. Sukumar and J. E. Bolander. Voronoi-based interpolants for fracture modelling. *Tessellations in the Sciences; Virtues, Techniques and Applications of Geometric Tilings*, 2009.
33. N. Sukumar and E. A. Malsch. Recent advances in the construction of polygonal finite element interpolants. *Archives of Computational Methods in Engineering*, 13(1):129–163, 2006.
34. N. Sukumar and A. Tabarraei. Conforming polygonal finite elements. *International Journal for Numerical Methods in Engineering*, 61(12):2045–2066, 2004.
35. N. Sukumar and A. Tabarraei. Numerical formulation and application of polygonal finite elements. In *Proceedings of the Seventh International ESAFORM Conference on Material Forming*, pages 73–76, 2004.
36. A. Tabarraei and N. Sukumar. Application of polygonal finite elements in linear elasticity. *International Journal of Computational Methods*, 3(4):503—520, 2006.
37. C. Talischi, G. Paulino, and A. Pereira. Polygonal finite elements for topology optimization: A unifying paradigm. *International Journal for Numerical Methods in Engineering*, 82(6):671–698, 2010.
38. J. Tournois, P. Alliez, and O. Devillers. Interleaving Delaunay refinement and optimization for 2D triangle mesh generation. In *Proceedings of the 16th International Meshing Roundtable*, pages 83–101, 2007.
39. J. Tournois, C. Wormser, P. Alliez, and M. Desbrun. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Transactions on Graphics*, 28(3):75:1–75:9, 2009.

40. E. VanderZee, A. N. Hirani, D. Guoy, and E. Ramos. Well-centered planar triangulation – an iterative approach. In *Proceedings of the 16th International Meshing Roundtable*, pages 121–138, 2007.
41. E. VanderZee, A. N. Hirani, D. Guoy, and E. Ramos. Well-centered triangulation. *SIAM Journal on Scientific Computing*, 31(6):4497–4523, 2010.
42. E. L. Wachspress. *A Rational Finite Element Basis*. Academic Press, 1975.
43. M. Wicke, M. Botsch, and M. Gross. A finite element method on convex polyhedra. *Computer Graphics Forum*, 26:355–364, 2007.
44. M. Wicke, D. Ritchie, B. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on Graphics*, 29(3):(to appear), 2010.
45. C. Wojtan, N. Thürey, M. Gross, and G. Turk. Deforming meshes that split and merge. *ACM Transactions on Graphics*, 28(3):76:1–76:10, 2009.
46. C. Wojtan and G. Turk. Fast viscoelastic behavior with thin features. *ACM Transactions on Graphics*, 27(3):47:1–47:8, 2008.