Quality Encoding for Tetrahedral Mesh Optimization

Kai Xu[a,b], Zhi-Quan Cheng[a], Yanzhen Wang[a], Yueshan Xiong[a], Hao Zhang[*,b]

[a] *School of Computer Science, National University of Defense Technology, China*
[b] *School of Computing Science, Simon Fraser University, Canada*

Abstract

We define *quality differential coordinates* (QDC) for per-vertex encoding of the quality of a tetrahedral mesh. QDC measures the deviation of a mesh vertex from a position which maximizes the combined quality of the set of tetrahedra incident at that vertex. Our formulation allows the incorporation of different choices of element quality metrics into QDC construction to penalize badly shaped and inverted tetrahedra. We develop an algorithm for tetrahedral mesh optimization through energy minimization driven by QDC. The variational problem is solved efficiently and robustly using gradient flow based on a stable semi-implicit integration scheme. To ensure quality boundary of the resulting tetrahedral mesh, we propose a harmonic-guided optimization scheme which leads to consistent handling of both the interior and boundary tetrahedra.

Key words: Tetrahedral mesh optimization, Tetrahedral element quality, Algebraic metrics, Quality differential coordinates

1. Introduction

Tetrahedral meshes are widely used in computer graphics for physically-based modeling, in particular realistic simulation of deformable objects [27] and fluid [11] via the finite element or finite volume method. High-quality tetrahedral meshes can remarkably improve the numerical accuracy and convergence of the simulation, as well as the visual appearance of the object surface. Tetrahedral meshes reconstructed from volume data or generated by tiling a scanned 3D surface often do not possess the desired quality. Their surfaces are typically rough due to physical noises in the data. Badly shaped, degenerate or tangled (i.e., inverted) tetrahedra are often present and they can significantly hinder the performance of numerical simulation.

To produce tetrahedral meshes with smooth boundaries and well-shaped interior tetrahedra, one natural solution is to employ a two-stage method. In the first stage, feature-preserving surface fairing is performed to denoise the surface mesh and improve its quality. The second stage then aims to improve the quality of the mesh interior while keeping the smoothed surface boundary approximately unchanged. Typically, the quality of a mesh element, a tetrahedron in our context, is measured based on its effect on interpolation error, discretization error, and the conditioning of the stiffness matrix [32].

Surface fairing has been extensively researched during the last decade and many excellent feature-preserving faring algorithms [9, 14, 21, 34, 37] are now available. Many of them can produce smooth, high-quality surface (triangular) meshes while preserving geometric features. However, surface fairing is generally oblivious to the quality of the interior tetrahedral mesh. By repositioning boundary vertices during faring, one may even damage the quality of certain boundary tetrahedra.

The handling of boundary tetrahedra is a difficult problem for both tetrahedral mesh generation and optimization. Previous methods often trade off between two competing goals: quality improvement and boundary conformation. Some methods [13, 17, 16, 23, 28] choose to fix all boundary vertices during optimization of tetrahedral element quality, which constrains the processing of boundary tetrahedra. Other methods [19, 22] optimize the boundary tetrahedra by repositioning boundary vertices or by changing boundary mesh connectivity under necessary constraints. However, these methods can still produce degenerate boundary tetrahedra. Additional processing to remove the degeneracies [7] is often needed. However this may in turn introduce noticeable boundary error and hence new surface noises as well as distortion of surface features.

As a result, the main difficulties with the two-stage approach as we describe above calls for an algorithm which can *consistently* improve the quality of the interior and boundary tetrahedra without introducing noticeable errors at the boundary. Other issues of concern include numerical stability and speed. Previous mesh optimization methods [15, 18, 19] mostly employ nonlinear optimization whose objective function is built directly from certain mesh quality metrics [32]. These methods can turn an initial mesh with sufficient quality into an even better one, however they may fail to converge on an input whose quality is too low. In addition, due to the slow convergence rate and high computational cost of each iteration, these approaches often cannot deal with large data sets.

In this paper, we propose a variational approach to tetrahedral mesh optimization based on quality encoding. Our approach is inspired by existing works on Laplacian mesh editing [1, 5, 33, 38], where the main idea is to use a representation that captures the local differential properties of the surface and

[*]Corresponding author. Tel.: +1 778-782-6843.
Email addresses: kaixu@nudt.edu.cn (Kai Xu), cheng.zhiquan@gmail.com (Zhi-Quan Cheng), yzwang@nudt.edu.cn (Yanzhen Wang), ysxiong@nudt.edu.cn (Yueshan Xiong), haoz@cs.sfu.ca (Hao Zhang)

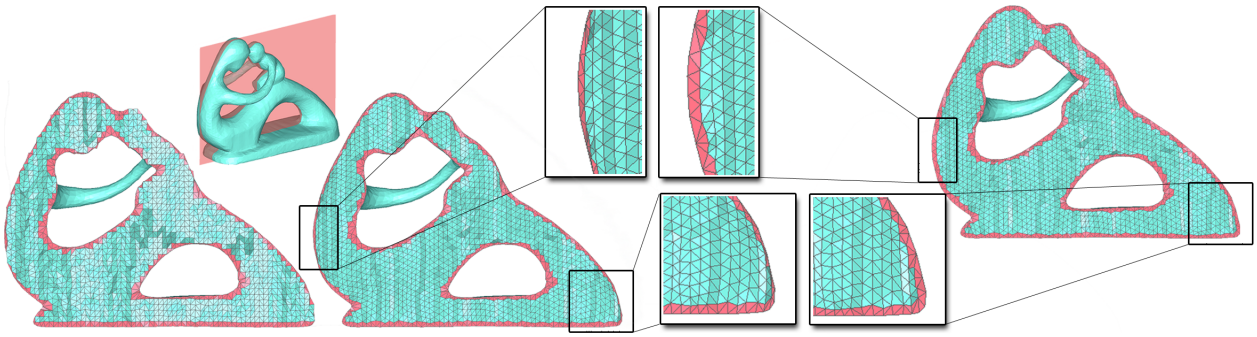


Figure 1: Given an input tetrahedral mesh (left), our method can well improve the mesh quality (middle). With harmonic field guidance, the quality of boundary tetrahedra (highlighted in red) can be better improved (right). Note the zoomed-in comparison of the boundary tetrahedra with and w/o harmonic field guidance.

to preserve these properties during deformation [5]. Specifically, differential coordinates are used to encode surface details. A deformed mesh is obtained by reconstructing the mesh geometry under positional constraints of edited vertices while preserving the surface details as much as possible.

Analogously, we introduce a volumetric representation, which we call *quality differential coordinates* (QDC), to encode the "quality details" at the vertices of a tetrahedral mesh. Specifically, QDC encodes mesh quality as the deviation of a vertex from a position which maximizes the combined quality of the tetrahedra incident at that vertex, where the element quality is measured by quality metric [32]. Based on QDC, a quadric energy is built to measure the distance between the input mesh and its counterpart with the total element quality maximized. Minimizing the energy under the positional constraints of boundary vertices leads to optimization of mesh quality.

Different from preserving surface details in the context of mesh editing, the variational process in our method improves the "quality details". The resulting nonlinear variational problem is solved by a semi-implicit gradient flow solver derived from [3]. Through analysis and numerical experiments, we show that our solver obtains both robustness and efficiency by improving the conditioning of the system matrix. To better optimize boundary tetrahedra, we propose harmonic-guided optimization. Under the guidance of harmonic weight fields, the boundary tetrahedra can benefit from weighted least-squares optimization, resulting in high-quality boundary tetrahedra (see Figure 1). Figure 2 gives an overview of our algorithm. The main contributions of our work are:

- Flexibility: Different quality metrics and their combinations can be used for quality encoding to obtain different optimization behavior and suit different applications.
- Consistent optimization: With harmonic-guided optimization, interior and boundary tetraheda can be improved more consistently.
- Simultaneous smoothing and untangling: By integrating a quality metric which is continuous over $\mathfrak{R}^3$, our algorithm can simultaneously untangle (removal of inverted elements) and smooth a mesh.
- Robustness and efficiency: Our algorithm can robustly optimize meshes with low quality. The semi-implicit gradient flow solver provides fast convergence and efficiency.

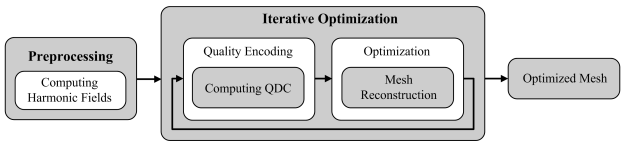


Figure 2: Algorithm overview. Given a tetrahedral mesh, we first compute a harmonic weight field over the entire mesh. Then our algorithm alternates between quality encoding with QDC and mesh reconstruction guided by the harmonic field until convergence. The reconstruction, from QDC to Cartesian coordinates, minimizes the energy function that measures the distance between the input mesh and its counterpart with the total element quality maximized.

2. Related works

Tetrahedral mesh optimization, designed to improve the quality of a mesh, is an important research topic in both computer graphics and several industrial applications. Many algorithms have been proposed during the last decade. We only review those works most related to ours.

The basic methods for mesh quality improvement can be classified into two categories. The first category of methods apply topological transformations to improve a mesh, typically by changing its connectivity. The operations include local face swapping, element or vertex insertion/deletion, etc. The other type of methods, referred to as smoothing or vertex repositioning, improve a mesh by moving its vertices while keeping the mesh connectivity unchanged. For this class of techniques [13, 15, 16, 17, 23, 28], quality improvement is reduced to a numerical optimization problem where the objective function measures one or more mesh properties. Minimizing the objective function through vertex repositioning leads to improvement in those mesh properties. Combining the two strategies can often result in higher quality meshes [19, 22].

Vertex repositioning employs quality metrics defined on mesh elements to measure their quality. Shewchuk [32] provided a clear exposition of the relations between the metrics and (1) the conditioning of stiffness matrices in finite element methods and (2) the accuracy of linear interpolation of functions and their gradients. Based on the quality metrics, two classes of approaches, local and global optimizations, have been proposed.

For local methods [15, 16, 23], an objective function is built on a submesh (see Figure 3) to locally measure the mesh quality. Quality improvement is achieved by repeating a local optimization of the objective function defined on each submesh.

Objective functions for global methods [12, 17, 28] are constructed by accumulating contributions from each local measure into one scalar function of vertex positions. The overall mesh is optimized through minimizing the global objective function. In [17], two methods, inexact Newton and block coordinate descent, for numerically optimizing global objective functions were compared. While inexact Newton method leads to *all-vertex* optimization [17] where all of the free vertices are moved simultaneously within a single iteration, the block coordinate descent results in a *single-vertex* optimization [17] in which only one vertex at a time is modified through a sub-optimization for that vertex. Local methods can also be seen as single-vertex methods. Generally, all-vertex approaches have faster convergence rate and often provide more accurate optimization results than single-vertex ones, although they need more computation for a single iteration [17]. Our method fits into the all-vertex category since our energy function is built globally and all the vertices are repositioned simultaneously.

Methods reviewed above can only optimize a valid mesh, i.e., one that does not have tangled elements, since their objective functions present singularities when any element is tangled. To overcome this problem, Freitag et al. [15] proposed a two-stage method to untangle and smooth tetrahedral meshes separately. By modifying two quality metrics and their corresponding objective functions to ensure their continuity over $\Re^3$, Escobar et al. [13] obtained an algorithm which can simultaneously smooth and untangle a tetrahedral mesh.

Minimization of the above objective functions is often a nonlinear problem. For meshes with very low quality, most of the existing methods will run into slow convergence or even divergence. Furthermore, previous methods [12, 13, 17] often result in bad boundary tetrahedra since boundary vertices are fixed for boundary conformation and hence boundary tetrahedra benefit less from the optimization than interior ones. Some methods [19, 22] relax the problem to approximately preserve the boundary shape. They employ constrained smoothing of boundary vertices, in which a boundary vertex can be moved within a common plane or edge of its neighbor vertices to avoid boundary shape distortion [19]. However, curved boundaries can not benefit from such an approach as few neighboring vertices share a common plane or edge [22].

Alliez et al. [2] proposed a variational meshing algorithm, in which both vertex positions and connectivity are updated to minimize the same quadric energy. This energy is defined based on optimal Voronoi partition [6] where no element quality metric is considered. In our approach, quality metrics are encapsulated explicitly into the objective function. Moreover, the variational meshing algorithm also shares the problem of boundary degeneracy due to the requirement of boundary conformation.

Laplacian mesh processing has been extended to optimize triangular meshes recently [26, 29]. Their methods can be successfully used for quality improvement of triangular mesh. However, for optimization of tetrahedral mesh, simple Laplacian smoothing can produce tangled tetrahedra when the mesh boundary is non-convex.

3. Mesh Quality Encoding

In this section, we give a brief review of mesh quality measurement. Then, we introduce QDC-based quality encoding.

Let a tetrahedral mesh $\mathscr{M}$ be described by a triple $(\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ the set of edges and $\mathcal{T}$ the set of tetrahedra. $\hat{\mathcal{V}} \subset \mathcal{V}$ denotes the set of boundary vertices, and $\hat{\mathcal{E}} \subset \mathcal{E}$ the set of boundary edges. Let $|\bullet|$ denote the size of a set. The position of vertex $v_i \in \mathcal{V}$ is represented by Cartesian coordinates $\mathbf{v}_i = (x_i, y_i, z_i)^{\mathrm{T}} \in \Re^3$. $\mathbf{X} = (\mathbf{x}^{\mathrm{T}}, \mathbf{y}^{\mathrm{T}}, \mathbf{z}^{\mathrm{T}})^{\mathrm{T}} \in \Re^{3\times|\mathcal{V}|}$ refers to the collection of all vertex coordinates, where $\mathbf{x} = (x_i)^{\mathrm{T}}_{v_i\in\mathcal{V}} \in \Re^{|\mathcal{V}|}$, $\mathbf{y} = (y_i)^{\mathrm{T}}_{v_i\in\mathcal{V}}$ and $\mathbf{z} = (z_i)^{\mathrm{T}}_{v_i\in\mathcal{V}}$. Throughout the paper "tet" will be the abbreviation for tetrahedron. Each tet t consists of a small subset of the vertices, denoted by $\mathcal{V}_t$. $\mathbf{X}_t \in \Re^{3\times|\mathcal{V}_t|}$ is the coordinate matrix of all vertices in t. Tet t is a boundary tet if and only if at least one of its vertices is a boundary vertex.

3.1. Mesh Quality Measurement

Mesh generation and optimization algorithms often evaluate element quality by a continuous function $q : \Re^{3\times|\mathcal{V}_t|} \to \Re$. In particular, $q(\mathbf{X}_t)$ or q_t for short, measures the quality of the tet t, where we assume a larger value indicates higher quality throughout this paper. Many such quality metrics are available [24, 32]. We, however, prefer those algebraic ones which can measure the orientation of a tet, allowing to optimize against tangled tets. Specifically, $q_t = 1$ if t is a regular tet, $q_t = 0$ if it is flat, and $q_t < 0$ if it is tangled (inverted). Objective functions are often built with the inverse of q_t (denoted by $\tilde{q}_t$, where $\tilde{q}_t \in [1, \infty)$ when t is not tangled). so that quality improvement can be achieved by a minimization process. In our experiments, we use the original versions of the quality metrics to evaluate and compare the quality of resulting meshes.

We employ three algebraic quality metrics which measure a tet's deviation from a regular tet and its volume-(edge)length ratio respectively:

Modified inverse mean-ratio (MIMR). Mean-ratio [24] is a well-known algebraic metric in the literature. It measures the quality of a tet using the norm of the affine mapping matrix that maps the tet to a regular one. Let $\mathbf{S}_t$ denote the weighted Jacobian matrix of the tet t, which is the affine map that takes t to a regular tet. The mean-ratio of $\mathbf{S}_t$ is the scalar:

$$\eta_t = \frac{3\sigma_t^{2/3}}{\|\mathbf{S}_t\|_F^2},$$

where $\|\mathbf{S}_t\|_F = \sqrt{\mathrm{tr}(\mathbf{S}_t^{\mathrm{T}}\mathbf{S}_t)}$ is the Frobenius norm of $\mathbf{S}_t$ and $\sigma_t = \det(\mathbf{S}_t)$. Escobar et al. [13] replace σ by $h(\sigma) = (\sigma + \sqrt{\sigma^2 + 4\delta^2})/2$ (see [13] for details on choosing δ) to remove the singularities appearing in the inverse mean-ratio metric when t is tangled, leading to a metric that is continuous over $\Re^3$:

$$\tilde{\eta}_t = \frac{\|\mathbf{S}_t\|_F^2}{3h(\sigma_t)^{(2/3)}}.$$

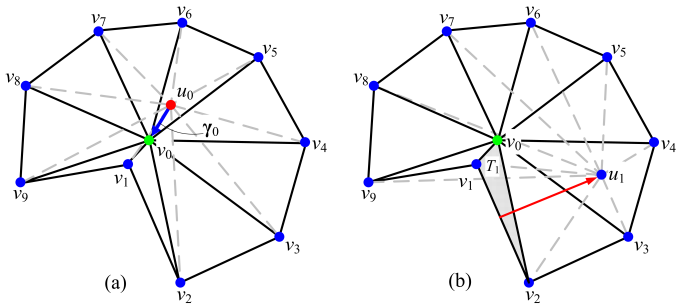


Figure 3: Illustration of QDC vector (a) and QDC weighting scheme (b) in a 2D planar submesh $\mathscr{M}(v_0)$, where v_0 is the free vertex. u_0 is the optimal position of v_0, maximizing the overall quality of $\mathcal{T}(v_0)$ while holding the positions of the other vertices in $\mathscr{M}(v_0)$ fixed. u_0 is the weighted average of the 1-ring vertices, where the weights are computed based on the optimal position for each triangle in the submesh. For instance, in (b), u_1 is the optimal position of v_0 with respect to the quality of triangle T_1 while holding the positions of v_1 and v_2 fixed.

Modified inverse condition number (MICN). This metric is also derived by [13] through modifying the condition number metric [24] with the same consideration as for MIMR:

$$\tilde{\kappa}_t = \frac{\|\mathbf{S}_t\|_F \cdot \left\|\mathbf{S}_t^*\right\|_F}{3h(\sigma_t)},$$

where $\mathbf{S}_t^* = \sigma_t \mathbf{S}_t^{-1}$ is the adjoint matrix of $\mathbf{S}_t$.

Modified inverse volume-length (MIVL). The volume-length metric, suggested by Parthasarathy et al. [30] and denoted by $\rho_t = 6\sqrt{2}V\big/l_{rms}^3$, is the signed volume of a tet divided by the cubic of the root-mean-square of its edge lengths. We modify this metric similar to [13] and use:

$$\tilde{\rho}_t = \frac{\sqrt{2}l_{rms}^3}{12h(V)}.$$

Note that the three modified metrics are all continuous over $\mathfrak{R}^3$ with respect to the coordinates of tet vertices and reach their minimum (equal to 1) when t reaches its highest quality, the measure of a regular tet. They are also referred to as the smooth quality metrics in the literature [32].

3.2. Quality Encoding

In detail-preserving shape editing, differential representations have gained significant popularity over the past few years [1, 5, 33, 38]. Given a surface mesh, the differential coordinates of a vertex v_i are defined by a displacement vector between v_i and the weighted average of its 1-ring neighborhood:

$$\boldsymbol{\delta}_i = [\delta_i^{(x)}, \delta_i^{(y)}, \delta_i^{(z)}]^{\mathrm{T}} = \mathbf{v}_i - \sum_{j \in \hat{N}_1(i)} w_j \mathbf{v}_j, \tag{1}$$

where $\hat{N}_1(i) = \{j | (i, j) \in \hat{\mathcal{E}}\}$ is the set of 1-ring neighborhood of v_i. In [38], the volume differential coordinates which encode the so-called volumetric details were introduced for volume-preserving mesh deformation. For a tetrahedral mesh, volume differential coordinates are defined through extending the 1-ring neighbor to the interior, where the 1-ring set is given by $N_1(i) = \{j | (i, j) \in \mathcal{E}\}$.

In a sense, differential coordinates locally encode the geometric details through measuring the deviation of a surface mesh from its smoothed version. Enlightened by this observation, we define analogously the "quality details" as the deviation of a tetrahedral mesh from its counterpart with the total element quality maximized.

Similarly, we use a differential representation to represent the (volumetric) "quality details". We consider the local *submesh* $\mathscr{M}(v_i)$ formed by $\mathcal{T}(v_i)$ which is the set of all the tets that share vertex v_i. Vertex v_i is the free vertex of submesh $\mathscr{M}(v_i)$. See Figure 3 for a submesh of a 2D planar triangle mesh. The quality of v_i is encoded with a displacement vector (Figure 3a):

$$\boldsymbol{\gamma}_i = \mathbf{v}_i - \mathbf{u}_i, \tag{2}$$

where $\mathbf{u}_i$ is the optimal position of v_i, which maximizes the overall quality measure for all tets in $\mathcal{T}(v_i)$, with the other vertices in $\mathscr{M}(v_i)$ held fixed. We call γ_i the *quality differential coordinate* (QDC) of v_i in this paper. Akin to the differential coordinates for geometric details, we write u_i as a weighted average of the 1-ring neighbors of v_i and rewrite QDC as follows:

$$\boldsymbol{\gamma}_i = (\gamma_i^{(x)}, \gamma_i^{(y)}, \gamma_i^{(z)})^{\mathrm{T}} = \mathbf{v}_i - \mathbf{X}_{N_1(i)} \mathbf{w}_i, \tag{3}$$

where $\mathbf{X}_{N_1(i)} \in \mathfrak{R}^{3 \times |N_1(i)|}$ is the Cartesian coordinate matrix of $\{v_j | j \in N_1(i)\}$, and $\mathbf{w}_i = (w_{ij})^{\mathrm{T}} \in \mathfrak{R}^{|N_1(i)| \times 1}$ $(j \in N_1(i))$ the weight vector. As a result, the computation of QDC is reduced to finding the *optimal weights* $\mathbf{w}_i$ satisfying $\mathbf{v}_i = \mathbf{X}_{N_1(i)} \mathbf{w}_i$:

$$\mathbf{w}_i = \arg\min_{\mathbf{w}_i} \sum_{t \in \mathcal{T}(v_i)} \tilde{q}_t. \tag{4}$$

The matrix that transforms a vector of Cartesian coordinates to the QDC vector is:

$$\mathbf{Q}\mathbf{x} = \boldsymbol{\gamma}^{(x)}, \tag{5}$$

where $\mathbf{x}$ and $\boldsymbol{\gamma}^{(x)}$are $|\mathcal{V}|$-vectors containing the x Cartesian and quality differential coordinates of all the vertices, respectively. The same goes for y and z coordinate vectors. We will simply use $\mathbf{x}$ and $\boldsymbol{\gamma}^{(x)}$ to cover all the three dimensions in the following discussion. Q is called *QDC Matrix* in this paper which has the form (assume the weights are normalized):

$$(\mathbf{Q})_{ij} = \begin{cases} 1 & i = j \\ -w_{ij} & (i, j) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases}$$

The local differential representation comes at the expense of a global reconstruction computation, i.e., the generation of Cartesian coordinates from differential coordinates requires one to solve a global PDE [33]. However, different from shape editing, we do not want to preserve the differential coordinates. Instead, we hope the reconstructed mesh would minimize the quality deviation for the purpose of quality improvement. This is the core of our algorithm, which we describe in Section 5.

4. Computing QDC

In this section, we describe how to compute QDC. As we have stated in Section 3.2, the key is to find a set of optimal

weights for each submesh (1-ring neighborhood of a vertex). Our main idea is to find approximately optimal weights such that minimizing the QDC of a free vertex in a submesh can untangle and smooth the submesh.

4.1. Weighting Scheme

The basic idea of our weighting scheme is to put larger weights on those vertices which have shorter distance to the optimal position of the free vertex in a submesh. However, computing the optimal position itself is a non-linear optimization problem. Therefore we compute it approximately based on the optimal position of the free vertex for each element. To make it clearer, we show the weighting scheme on the 2D planar submesh $\mathscr{M}(v_0)$ shown in Figure 3b. Let us take triangle T_1 formed by v_0, v_1 and v_2 for example. Let u_1 be the optimal position of v_0 which makes T_1 a regular triangle and hence optimizes T_1's quality. T_1's contribution to the weight of a 1-ring neighbor vertex, e.g., v_1, is inversely proportional to the square of the distance from v_1 to u_1. The same goes for other triangles in the submesh. Meanwhile, triangles with worse quality will contribute more to the weight computation in order to improve the worst tets preferentially. The final weight of a 1-ring neighbor is obtained by accumulating contributions from each of the triangles in $\mathscr{M}(v_0)$.

Thus the weighting scheme can be formularized as Eq. 6. In submesh $\mathscr{M}(v_i)$, the weight of v_j, a 1-ring neighbor of v_i, is:

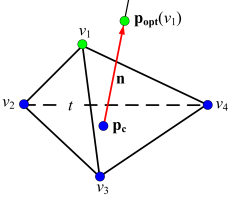
$$\tilde{w}_{ij} = \sum_{t \in \mathcal{T}(v_i)} \frac{\tilde{q}_t^2}{\left\| v_j - u_t \right\|^2}, \tag{6}$$

where $\mathcal{T}(v_i)$ is the set of tets in $\mathscr{M}(v_i)$ and $\tilde{q}_t$ is the quality of tet t. The weights should be normalized:

$$w_{ij} = \tilde{w}_{ij} / \sum_{k \in N_1(i)} \tilde{w}_{ik} \tag{7}$$

Although the estimation of optimal position of a vertex in a triangle (with the other vertices fixed) is obvious as mentioned above, it is not trivial in a tetrahedron. To achieve that, one can define an objective function of the position of the free vertex to measure the quality of the tet with some quality metric and optimize this objective function with a gradient-based algorithm. This expensive sub-optimization needs to be performed $4|\mathcal{T}|$ times for the entire mesh, where $|\mathcal{T}|$ is the number of tets.

We relax the requirement to find an approximate optimal position along a specific direction which we describe with a tet in the figure to the right. The tet t is composed of vertices v_1, v_2, v_3, and v_4. Suppose that v_1 is the free vertex (the other three vertices are fixed) and we want to find the position where v_1 optimizes t's quality. Instead of finding v_1's optimal position over the entire $\Re^3$, we search it along the ray, whose starting point is $\mathbf{p_c}$ (the barycenter of $\triangle v_2 v_3 v_4$) and whose direction is along $\mathbf{n}$ (the normal of $\triangle v_2 v_3 v_4$, pointing into $\mathscr{M}(v_1)$, the submesh around v_1).



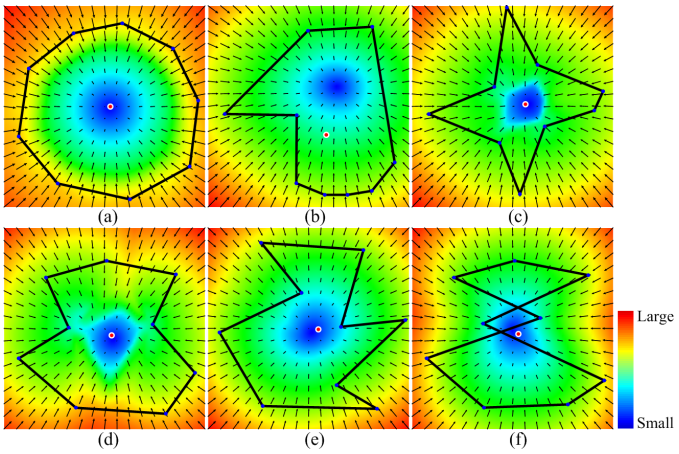


Figure 4: Visualization of the QDC vector fields for several 2D planar submeshes. The free vertex and all edges involved are hidden for clarity. In each submesh, the QDC vector field is visualized with both a scalar field to illustrate the magnitudes of all QDC vectors and arrows on a regular grid to demonstrate the directions of the QDC vectors. The barycenter of a 1-ring neighborhood in the figure is shown by a red point.

Therefore, the problem is reduced to a standard Armijo line search problem which is easy to solve:

$$\begin{aligned} \mathbf{p}_{\text{opt}}(v_1) &= \mathbf{p}_c + \alpha^* \cdot \mathbf{n} \\ \alpha^* &= \arg\min_{\alpha} \tilde{q}_t(\mathbf{p}_c + \alpha \cdot \mathbf{n}) \ \text{ s.t. } \ \alpha \geq 0 \ \cdot \end{aligned} \tag{8}$$

4.2. Further Analysis

For a different position of the free vertex in a submesh, there is a different set of optimal weights and hence a different QDC vector. A QDC vector field, defined over the space in which the submesh is embedded, can be constructed with all QDC vectors for different positions of the free vertex. To observe and analyze the behavior of our weighting scheme, we visualize the QDC vector field in a 2D planar submesh (see Figure 4), in which the magnitudes of all QDC vectors are illustrated as a color-coded scalar field while the directions of QDC vectors are demonstrated with arrows. We also depict the position of the barycenter of a 1-ring neighborhood (shown with a red point in figure 4) of the free vertex to compare our QDC approach with Laplacian smoothing for improving element quality.

The weighted average of 1-ring vertices is located at the position where the free vertex approximately optimizes the quality of the submesh. We call this position the approximate optimal position of the free vertex. The smaller the magnitude of a QDC vector is, the closer the free vertex is to its approximate optimal position. In consequence, the position where the QDC magnitude reaches the minimum can be seen as the approximate optimal position of the free vertex.

If the submesh is convex (Figure 4a), the approximate optimal position almost coincides with the barycenter. Otherwise, if the submesh is noncovex (Figure 4b), the two positions can be quite different. As shown in Figure 4b, the free vertex can be safely moved to the approximate optimal position without introducing any tangled triangle. This is not true for the barycenter

however. That is why Laplacian smoothing can cause element tangling for a noncovex mesh whereas our method does not.

The feasible region of a submesh is an interior region of the submesh, within which moving the free vertex will not introduce any tangled elements [13]. By observing the scalar fields in Figure 4c and d, it can be found that the distribution of QDC magnitudes roughly reflects the feasible region of a submesh (See the blue regions in the two figures). Specifically, the megnitude of a QDC vector is relatively small when the free vertex moves within the feasible region. When it moves out of the feasible region, the magnitude increases quickly. Meanwhile, QDC magnitude reaches its minimum in the feasible region. These properties meet approximately the requirements of mesh untangling and quality improving.

Another important feature of QDC is that if there is no feasible region in a submesh (Figure 4e and f), the approximate optimal position again coincides approximately with the barycenter. This feature leads to stability of our method for optimizing meshes of very low quality: our QDC-based optimization first smoothes a low-quality mesh, in which almost no feasible region is available, like Laplacian smoothing. As the smoothing proceeds, more feasible regions appear on which our method performs both smoothing and untangling simultaneously.

5. Variational Optimization

The computation of QDC is essentially a process of mesh quality encoding. The Cartesian coordinates of mesh vertices can be reconstructed from the QDC under certain boundary conditions. As QDC measures the quality deviation of a submesh from its counterpart with the total element quality maximized, in order to optimize the quality of the reconstructed mesh, the QDC vectors in the reconstruction should be set to zero. In Laplacian mesh editing, one must fix at least one vertex as additional constraint to guarantee that the reconstruction has a unique solution [33]. In tetrahedral mesh optimization, the boundary vertices will serve as the positional constraints as we hope to keep the boundary unchanged.

Our optimization solves the following quadric minimization problem:

$$\min_{\mathbf{x}}(\mathbf{G}(\mathbf{x})\mathbf{x}-\mathbf{g}(\mathbf{x}))^{\mathrm{T}}(\mathbf{G}(\mathbf{x})\mathbf{x}-\mathbf{g}(\mathbf{x})), \tag{9}$$

where $\mathbf{G}(\mathbf{x}) = [\mathbf{Q}(\mathbf{x}), \mathbf{B}]^{\mathrm{T}}$ and $\mathbf{g}(\mathbf{x}) = [\mathbf{0}, \mathbf{b}]^{\mathrm{T}}$. $\mathbf{B}$ is the positional constraint matrix for boundary mesh, and $\mathbf{b}$ is the original positions of boundary vertices of input mesh. The resulting mesh will reduce the above-mentioned quality deviation while preserving the positions of boundary vertices in a least-square sense. The quadric energy is defined globally over the entire mesh. All the inner vertices are repositioned simultaneously.

5.1. Harmonic-Guided Optimization

Most previous tetrahedral mesh optimization methods run into issues with boundary optimization: boundary tets often benefit less from the optimization than interior ones since the positions of boundary vertices are fixed as positional constraint. Our optimization by solving (Eq. 9) has the same problem.

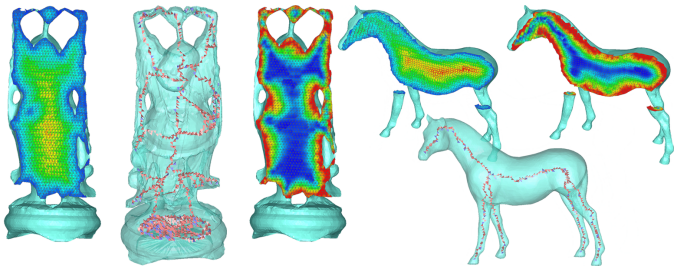


Figure 5: The cut-away views of the medial axis transform (left) and harmonic fields (right) of two tetrahedral meshes. The harmonic fields are computed using boundary surface constraint and the medial axes (middle).

We propose harmonic-guided optimization to overcome this inconsistency problem. To allow the boundary tets to benefit more from the optimization, we employ weighted least-squares where we weight the residuals for each vertex according to their distances to the boundary. We compute a harmonic scalar field for the tetrahedral mesh to be optimized, which reaches maximum at its boundary and minimum at its medial axis. Using the values in the harmonic scalar field as weights, both the boundary and interior tets can be optimized more consistently.

A harmonic function h defined on mesh vertices satisfies Laplace's equation $\nabla^2 h=0$. We prescribe the value 1 as boundary conditions for surface vertices acting as sources, and 0 for the vertices located at the medial axis serving as sinks. By solving $\mathbf{Lh} = \mathbf{0}$ (where $\mathbf{h} = (h_i)^{\mathrm{T}}_{i=1,2,\ldots,|\mathcal{V}|}$) with respect to these boundary conditions, we obtain a harmonic function that smoothly blends between 0 and 1. Therefore the weight matrix we use in the weighted least-squares is composed of two parts, harmonic weights and positional weights:

$$\mathbf{W} = \mathrm{diag}(\omega_{\mathrm{I}}h_1, \omega_{\mathrm{I}}h_2, \ldots, \omega_{\mathrm{I}}h_{|\mathcal{V}|}, \underbrace{\omega_{\mathrm{B}}, \ldots, \omega_{\mathrm{B}}}_{|\hat{\mathcal{V}}|}),$$

where ω_{B} is the positional weight which is used to tweak the importance of positional constraints of boundary vertices. Larger ω_{B} leads to more accurate boundary shape preservation but less interior mesh quality improvement. We find by experiments that $|\mathcal{V}|/|\hat{\mathcal{V}}|$ is a relatively good tradeoff. ω_{I} is the maximum weight for interior vertices, which can be seen as the decreasing rate of the harmonic weights from the boundary to the medial axis. The larger the decreasing rate we use, the larger the weight difference between boundary and interior and hence the more optimization the boundary tets can obtain. However, large ω_{I} weakens the boundary constraint (weighted by ω_{B}) and introduces more errors in boundary mesh at the same time. In our experiments, we use $\omega_{\mathrm{I}} = 0.4\omega_{\mathrm{B}}$. The new energy is thus:

$$\min_{\mathbf{x}}(\mathbf{W}(\mathbf{G}(\mathbf{x})\mathbf{x}-\mathbf{g}(\mathbf{x})))^{\mathrm{T}}(\mathbf{W}(\mathbf{G}(\mathbf{x})\mathbf{x}-\mathbf{g}(\mathbf{x}))). \tag{10}$$

The extraction of medial axis for a mesh model is often a time-consuming task. Fortunately, a tetrahedral mesh provides a natural voxelization (tets stuffed in its boundary) on which the medial axis transform (MAT) [31] can be performed to obtain an approximate medial axis (see Fig. 5 for some results). In practice, the tets in a low quality input mesh are often too

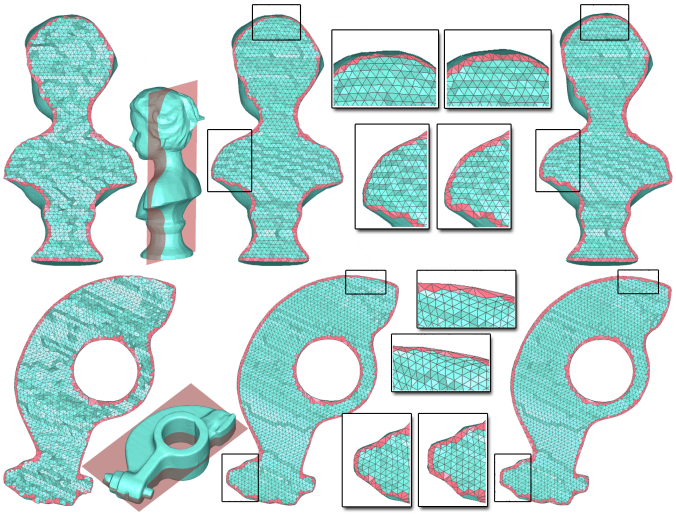


Figure 6: Effects of harmonic-guided optimization. For different input tetrahedral meshes (left), optimization results w/o (middle) and with (right) harmonic field guidance are compared. Note the comparison of boundary tets (highlighted in red) especially.

badly shaped to obtain an accurate MAT. To remedy this, we first use Eq. 9 to pre-optimize the initial mesh to obtain a mesh with adequate quality, on which a satisfactory MAT can be evaluated. The approximate medial axes computed using the above process are sufficient for our task.

Figure 6 gives the cut-away view of two tetrahedral meshes to compare the effect boundary optimization of our method with and w/o harmonic field guidance. The boundaries of these meshes are firstly faired using Laplacian smoothing, which is not a feature-preserving method and can introduce much surface degradation. We use this method for surface fairing only for a more obvious demonstration. It can be observed from the figure that, with the guidance of harmonic fields, our method can better optimize the squished boundary tets.

6. Numerics

This section discusses some details on solving the QDC-based variational problem. Since QDC depends nonlinearly on vertex coordinates, minimization of the energy function (Eq. 10) is a nonlinear least squares (NLS) problem. We derive a robust and efficient solver based on the gradient flow approach [3]. In order to achieve robustness for low quality input meshes, the quality metric in computing the QDC should satisfy the smoothness conditions, which we discuss in this section.

6.1. Semi-implicit Gradient Flow Solver

The main difficulty of using the standard methods, e.g., Gauss-Newton (GN) and Levenberg-Marquardt (LM), to solve our problem lies in the estimation of the Jacobian of the residual vector $\mathbf{r}(\mathbf{x}) = \mathbf{G}(\mathbf{x})\mathbf{x} - \mathbf{g}(\mathbf{x})$. The derivatives of QDC with respect to vertex Cartesian coordinates are hard to compute since the estimation of the optimal weights for QDC adopts a sub-optimization (line search) and hence QDC cannot be expressed analytically as a function of vertex coordinates.

We linearize our problem and derive an inexact solver from the point of view of gradient flow. The gradient flow method [3] solves the NLS problem by means of integration of a first order ordinary differential equation (ODE). A necessary condition for point $\mathbf{x}^*$ to be an optimal solution for the NLS problem $\min_{\mathbf{x}\in\Re^n} F(\mathbf{x}) = 1/2 \cdot \mathbf{r}(\mathbf{x})^{\mathrm{T}}\mathbf{r}(\mathbf{x})$ is:

$$\nabla F(\mathbf{x}^*) = \mathbf{J}_{\mathbf{r}}^{\mathrm{T}}(\mathbf{x}^*) \cdot \mathbf{r}(\mathbf{x}^*) = 0, \tag{11}$$

where $\mathbf{J}_{\mathbf{r}}$ is the Jacobian of $\mathbf{r}(\mathbf{x})$. To fulfill this optimality condition, we rely on a reformulation of the continuous gradient flow of the NLS problem. Specifically, we solve the ODE

$$\frac{d\mathbf{x}(t)}{dt} = -\nabla F(\mathbf{x}(t)), \tag{12}$$

with the initial condition

$$\mathbf{x}(0) = \mathbf{x}_0. \tag{13}$$

The optimal solution can be obtained by following the trajectory of a system of ODEs. We employ a semi-implicit scheme to discretize the right hand side of Eq. 12, where we use $\mathbf{G}_k \cdot \mathbf{x}_{k+1} - \mathbf{g}_k$ to approximate $\mathbf{r}(\mathbf{x}_{k+1})$. Thus $\mathbf{J}_{\mathbf{r}}(\mathbf{x}_{k+1})$ is approximated by $\mathbf{G}_k$. The resulting discretized form of Eq. 12 is:

$$(\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k + \mathbf{I}) \cdot \mathbf{x}_{k+1} = \mathbf{G}_k^{\mathrm{T}}\mathbf{g}_k + \mathbf{x}_k. \tag{14}$$

The computation for each iteration only amounts to solving a sparse linear system and no evaluation of the Jacobian matrix is needed. The approximation to $\mathbf{J}_{\mathbf{r}}(\mathbf{x}_{k+1})$ with $\mathbf{G}_k$ is valid only when $\mathbf{r}(\mathbf{x})$ is quasi-linear with respect to $\mathbf{x}$, where the two following conditions hold: (1) $\mathbf{G}(\mathbf{x})$ is a nearly constant matrix which changes slowly at each iteration and (2) $\mathbf{g}(\mathbf{x})$ is a vector whose Jacobian is small, i.e., $\|\mathbf{J}_{\mathbf{g}}\| \ll \|\mathbf{G}\|$.

The first condition reveals that the success of the semi-implicit solver highly depends on the changes between $\mathbf{G}_k$ and $\mathbf{G}_{k+1}$. We give a simple analysis for the accuracy of our linearization by estimating the matrix variance $\|\mathbf{G}_{k+1} - \mathbf{G}_k\|$. According to the Mean Value theorem in calculus, the variance satisfies the following inequality:

$$\|\mathbf{G}_{k+1} - \mathbf{G}_k\| \leq \left\|\mathbf{G}'(\xi)\right\| \|\mathbf{x}_{k+1} - \mathbf{x}_k\|,$$

where $\mathbf{G}'(\mathbf{x})$ is the derivative of the matrix function $\mathbf{G}(\mathbf{x})$ and $\xi = (1-\lambda)\mathbf{x}_{k+1} + \lambda\mathbf{x}_k$ $(0 < \lambda < 1)$ is the mean value vector. To ensure that the solver converges with a relatively large step size, $\mathbf{G}'(\mathbf{x})$ must have a small upper bound. Although the derivative of $\mathbf{G}(\mathbf{x})$ is hard to compute analytically, it is sufficient to evaluate it along the direction of the vector $\mathbf{x}_{k+1} - \mathbf{x}_k$ as we only need to know how fast $\mathbf{G}(\mathbf{x})$ changes along that direction. We use the finite difference method to evaluate the directional derivative. In particular, we solve for $\mathbf{x}_k$ in each iteration and compute the difference quotient using the forward Euler scheme:

$$\left\|\mathbf{G}'_k\right\| \approx \frac{\|\mathbf{G}_{k+1} - \mathbf{G}_k\|}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}.$$

We find in the numerical experiments that for any given initial unknown $\mathbf{x}_0$, the directional derivative converges to 0 as the iterations proceed. Table 1 shows the numerical results.

Input mesh			Iteration Steps				
Name	$\lvert\mathcal{V}\rvert$		k=0	k=1	k=2	k=3	k=4
Fertility	21,213	d_k	1.24*e*–1	2.03*e*–3	2.89*e*–5	4.75*e*–7	3.01*e*–7
		r_k	3.96*e*–3	5.10*e*–5	2.21*e*–6	1.93*e*–7	1.15*e*–7
Pegasus	46,212	d_k	3.19*e*–1	2.44*e*–3	6.47*e*–5	1.08*e*–6	7.91*e*–7
		r_k	2.12*e*–2	1.52*e*–4	8.95*e*–5	6.25*e*–6	3.73*e*–6

Table 1: Values of $d_k = \|\mathbf{G}'(\mathbf{x}_k)\|$ and $r_k = \|\mathbf{J_g}(\mathbf{x}_k)\|/\|\mathbf{G}_k\|$ at the first 4 iterations of Eq. 14. Although $\|\mathbf{G}'\|$ and $\|\mathbf{J_g}\|/\|\mathbf{G}\|$ are initially large, they decrease to no more than 1.0×10^{-6} and 1.0×10^{-5} respectively within 4 iterations.

Input mesh			Iteration Steps				
Name	$\lvert\mathcal{V}\rvert$		k=0	k=1	k=2	k=3	k=4
Cube	379	κ_k^{ex}	5.72*e*+4	1.61*e*+4	5.67*e*+3	4.93*e*+3	1.23*e*+3
		κ_k^{si}	2.19*e*+2	6.35*e*+1	4.22*e*+1	3.35*e*+1	1.67*e*+1
Sphere	993	κ_k^{ex}	2.73*e*+6	9.35*e*+5	6.18*e*+5	3.79*e*+5	7.20*e*+4
		κ_k^{si}	1.91*e*+3	4.63*e*+2	3.16*e*+2	1.37*e*+2	4.03*e*+1

Table 2: Comparison of the conditioning of the system matrix using explicit and semi-implicit methods in their first 4 iterations. The condition number of our semi-implicit method, denoted by κ_k^{si}, is smaller than that of the explicit method, denoted by κ_k^{ex}, throughout the iteration steps.

The second condition can also be satisfied asymptotically for our problem: $\left\|\mathbf{J_g}\right\|$ (evaluated with the finite difference method) becomes small as compared to $\|\mathbf{G}\|$ after 2 to 3 iterations of Eq. 14 (Table 1). This is owed to the smooth definition of QDC: the weighted sum definition of the weights in Eq. 6 and the smoothness of the quality metric adopted (Section 4.1). Our experiment shows that the latter factor is more crucial to the second condition and numerical convergence. If a nonsmooth quality metric, e.g., the minimum sine of dihedral angles [32], is used in QDC, our solver can hardly converge for low quality meshes. Moreover, the rate of change of $\mathbf{g}(\mathbf{x})$ with respect to $\mathbf{x}$ decreases rapidly during the first few iterations (Table 1). This is because our QDC-based optimization behaves like a Laplacian smoother on a low quality mesh at the beginning of optimization (recall the analysis in Section 4.2), which helps to reduce the nonlinearity of QDC.

A key factor to the numerical stability of an iterative solver is the finite condition number of the associated system matrix, denoted by $\kappa(\bullet)$; it is the ratio between the largest and the smallest non-zero eigenvalues of the matrix. Compared to explicit integration of Eq. 12, which results an inexact variant of the GN solver whose system matrix is $\mathbf{G}^{\mathrm{T}}\mathbf{G}$, our semi-implicit method is more robust as $\kappa(\mathbf{G}^{\mathrm{T}}\mathbf{G}+\mathbf{I})$ is much smaller than $\kappa(\mathbf{G}^{\mathrm{T}}\mathbf{G})$. Note that $\kappa(\mathbf{G}^{\mathrm{T}}\mathbf{G})$ is dominated by $\kappa(\mathbf{Q}^{\mathrm{T}}\mathbf{Q}) = (\lambda^{\mathbf{Q}}_{\max}/\lambda^{\mathbf{Q}}_{\min})^2$, where $\mathbf{Q}$ is QDC matrix with $\lambda^{\mathbf{Q}}_{\max}$ and $\lambda^{\mathbf{Q}}_{\min}$ as its the largest and the smallest non-zero eigenvalues, respectively. Following [32], the lower bound of $\lambda^{\mathbf{Q}}_{\min}$ is proportional to the volume of the smallest tet and $\lambda^{\mathbf{Q}}_{\max}$ to $l_{\max}/\sin(\theta_{\min})$, where $l_{\max}$ is the length of the longest edge and $\theta_{\min}$ is the smallest dihedral angle in the tetrahedral mesh. For a mesh of low quality, $\lambda^{\mathbf{Q}}_{\min}$ is small and $\lambda^{\mathbf{Q}}_{\max}$ is large. It follows that $\left((\lambda^{\mathbf{Q}}_{\max}+1)/(\lambda^{\mathbf{Q}}_{\min}+1)\right)^2 \ll (\lambda^{\mathbf{Q}}_{\max}/\lambda^{\mathbf{Q}}_{\min})^2$, indicating that $\kappa(\mathbf{G}^{\mathrm{T}}\mathbf{G}+\mathbf{I}) \ll \kappa(\mathbf{G}^{\mathrm{T}}\mathbf{G})$. In addition, the conditioning of $\mathbf{G}^{\mathrm{T}}\mathbf{G}+\mathbf{I}$ can be further improved by quality improvement as the iterations proceed (see Table 2).

To obtain a better-conditioned system matrix, we can introduce a reasonably chosen parameter to $\mathbf{G}^{\mathrm{T}}\mathbf{G}+\mathbf{I}$ and solve the following linear system instead:

$$(\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k + \mu_k\mathbf{I}) \cdot \mathbf{x}_{k+1} = \mathbf{G}_k^{\mathrm{T}}\mathbf{g}_k + \mu_k\mathbf{x}_k, \tag{15}$$

where μ_k is a positive parameter. Eq. 15 is essentially an inexact variant of the LM method, where $\mathbf{J_r}(\mathbf{x}_{k+1})$ is approximated with $\mathbf{G}_k$ under the aforementioned quasi-linear conditions. To improve the conditioning of the system matrix, μ_k should be set to a large value for a mesh with low quality as $\kappa(\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k+\mu_k\mathbf{I})$ is non-increasing with respect to μ_k. We follow the choice of [36] and use $\mu_k = \left\|\mathbf{G}_k^{\mathrm{T}}\mathbf{g}_k\right\|^2$ (with the same approximation as above), which has all of our desirable features. If $\mathbf{x}_k$ is far away from the optimal solution, which is the case when the mesh quality is low, μ_k is large and thereby $\kappa(\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k + \mu_k\mathbf{I})$ is small. When the mesh quality is improved and $\mathbf{x}_k$ is close to the optimal solution, μ_k can be quite small while $\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k + \mu_k\mathbf{I}$ remains well-conditioned since a high quality mesh has a small $\kappa(\mathbf{G}_k^{\mathrm{T}}\mathbf{G}_k)$.

At each iteration, we solve Eq. 15 using Cholesky factorization. Although the system matrix is sparse, it depends on $\mathbf{x}$ and thus changes at each iteration. Therefore, the full factorization cannot be reused. However, the non-zero structure remains unchanged since our optimization dose not change the mesh connectivity. As a result, a fill-reducing permutation of the system matrix and symbolic factorization based only on its non-zero structure can be precomputed and reused [35].

7. Results and Discussions

We have implemented our algorithm with the three quality metrics described in Section 3.1. We find that the MIMR and MICN metrics have similar optimization behavior, e.g., in their resulting distribution of dihedral angles. The MIVL metric, however, works somewhat differently. The running times presented are recorded on a PC with 1.83 GHz AMD Sempron processor and 512MB RAM.

7.1. Different Metrics on Different Kinds of Meshes

First, we test our algorithm on meshes generated by three different meshing algorithms. Table 5 shows the meshes before and after quality improvement by our variational optimization using the MIMR and MIVL metrics, respectively. Histograms of dihedral angle distributions are also shown. Meshes 1 and 2 are generated using the Nuages software [20] which takes parallel cross sectional contours as input. Tetrahedral meshes reconstructed from cross sectional data are widely used in biomedicine since the 3D data of human organs are often obtained from CT and MRI data. Since any two adjacent contours can be arbitrarily different in shape and/or vertex distribution, this approach usually generates boundary tets of very low quality. The tetrahedral meshes 1 and 2 are obtained by cutting a surface mesh into a stack of cross sectional contours and then

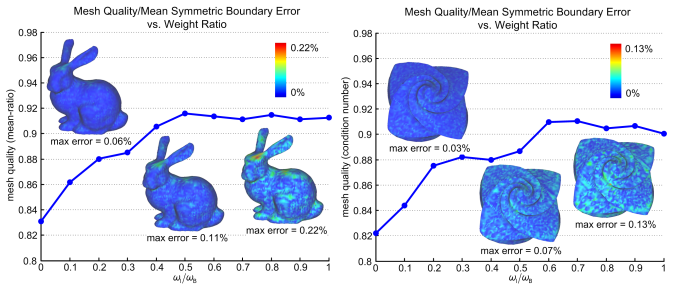


Figure 7: Mesh quality measures and boundary error (symmetric Hausdorff) plotted against the ratio between interior and boundary weights.

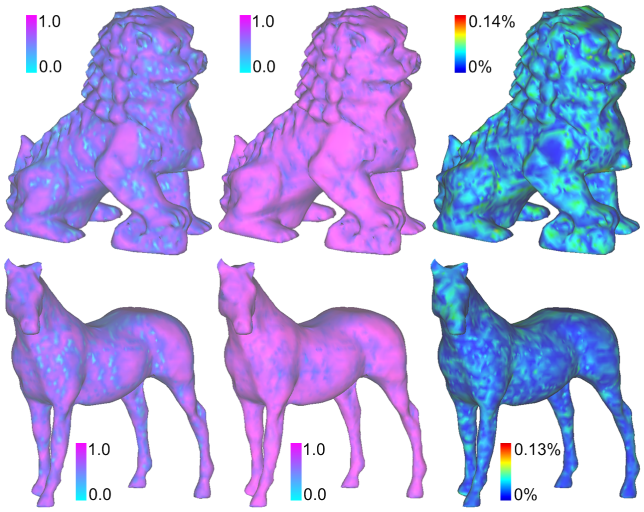


Figure 8: Color-coded illustration of boundary quality (mean-ratio metric) for both input (left) and optimized meshes (middle), as well as that of boundary error (right) introduced. The latter is again measured by symmetric Hausdorff distance using the Metro tool.

feeding them to the Nuages software. Meshes 3 and 4 are generated with the variational tetrahedral meshing algorithm of Alliez et al. [2]. They produce better boundary tets due to a global optimization of mesh connectivity (including boundary connectivity). However, sliver tets can still appear in these meshes due to boundary conformation. Isosurface stuffing [25] can provide tetrahedral meshes with guaranteed good dihedral angles. For the resulting meshes, 5 and 6, the tets with the worst dihedral angles are also the boundary ones.

The boundaries of the meshes generated by Nuages are often noisy and of pool quality. We first perform a feature-preserving surface fairing on the boundaries. Note that not all feature-preserving fairing algorithms are suitable for this task. For example, fairing via mean curvature flow [9] can remove surface noise along the normal direction while not attempting to improve the quality of the triangles; the latter would require tangential movement of the mesh vertices. We have found by experiments that Taubin's $\lambda|\mu$ filtering [34] and the Min-Dist flow algorithm [37] are more suitable and they have been applied to meshes 1 and 2, respectively.

As shown in Table 5, our method significantly improves the dihedral angles of the meshes generated with Nuages. Dihedral angles of the high-quality meshes generated by the variational tetrahedral meshing and the isosurface stuffing algorithms can be further improved by our algorithm thanks to our harmonic-guided optimization step. Another observation is that our variational approach presents different optimization behavior when different quality metrics are integrated. The MIMR metric often leads to more centralized distribution of dihedral angles and a sharper peak near 60° in the histograms than the MIVL metric, indicating the tet shapes in the resulting meshes are more regular. However meshes by MIVL have better minimum and maximum dihedral angle bounds. This can be explained intuitively as follows. Mean-ratio directly measures the deviation of a tet from a regular one. As a result, the energy built with MIMR penalizes more the deviation while pays relatively less attention on improving dihedral angles. On the contrary, MIVL penalizes more the tets with undesirable dihedral angles.

7.2. Comparison with Other Methods

Table 3 compares our algorithm with three other optimization techniques. The first one is the Opt-MS package developed by Freitag [15]. This software adopts a local optimization algorithm and uses a two-stage method to smooth and untangle a mesh. The second one, which we refer to as SUS, is proposed by Escobar et al. [13]. SUS is also a local algorithm, using steepest descent for optimization. In addition, we extend SUS and implement a global version by building a global objective function based on the modified quality measurements proposed in [13]. Wherein, the inexact Newton method is employed to perform an all-vertex global optimization. Note that all of the three methods achieve smoothing and untangling. In our algorithm, the MIVL metric is used. To speed up the optimization on large meshes, we utilize the out-of-core factorization provided by TAUCS [35]. For each method, iteration continuous until convergence or the maximum execution time (200 min.) is exceeded. We use the mean-ratio metric to measure the qualities of meshes. Both the mean and standard deviation of the quality measures are computed for comparison.

As all-vertex algorithms, QDC and global SUS can obtain better mesh quality than Opt-MS and local SUS. Our algorithm can achieve even better quality measures than global SUS due to improved boundary optimization. Also shown in Table 3, although the global SUS improves the convergence rate (obtaining better mesh quality with much fewer iterations in contrast to Opt-MS and SUS), it does not save the computational cost since each iteration is more costly. Our method, however, is more efficient thanks to not only the fast convergence but also the low computational cost per iteration. In addition, we obtain smaller standard deviation of the quality measures than global SUS and hence more uniform tet shapes in the resulting meshes.

7.3. Simultaneous Untangling and Smoothing

To compare the mesh untangling behavior of our QDC algorithm with Opt-MS and SUS, we show in Table 4 the minimum execution time needed to untangle all the tangled tets in the input meshes and the resulting quality measures. By observing the quality measures, it can be found that our algorithm can improve the mesh quality during untangling. Therefore, our algo-

rithm achieves simultaneous smoothing and untangling as SUS since the QDC-based energy function penalizes tangled tets.

7.4. Boundary Optimization and Boundary Error

Our algorithm does not ensure exact boundary conformation, since boundary preservation is achieved only in a least-square sense. As we pointed out in Section 5.1, the ratio between interior and boundary weights can be used to tradeoff between mesh quality and boundary error. The larger the ratio, the higher the mesh quality, while larger boundary errors can be introduced at the same time. This is confirmed by Figure 7, in which we plot the mesh quality (measured by both the mean-ratio and condition number) against the weight ratio, as well as a color-coded illustration of the boundary error, computed by the Metro tool [8]. Note that the boundary errors shown in Figure 7 does not include those introduced by surface fairing.

Figure 8 demonstrates the boundary optimization results of our method. To visualize the quality of boundary tets, we compute for each boundary vertex the average quality measures of its 1-ring tets and color-code the average measures on the boundary. The figure shows that our method significantly improves the quality of boundary tets with only very small boundary error introduced. Regions with large boundary errors roughly agree to those with low boundary quality. Since the distribution of boundary errors does not appear to depend on surface geometry, our method typically do not lead to more shape degradation at sharp features than over flat regions.

Other limitations. Another limitation of our current method is that it does perform well on tetrahedral meshes with many thin regions where almost all the tets are boundary ones. Since most vertices in these regions serve as the boundary condition, little or no optimization can be performed. This case necessitates an optimization for mesh connectivity.

8. Conclusion

We have presented a variational tetrahedral optimization approach based on per-vertex quality encoding. Our approach leaves the boundary smoothing to feature-preserving surface fairing and focuses on how to improve the boundary and interior tets consistently. The QDC-based representation provides our method with robustness (for optimizing meshes of very low quality, effectiveness (for respecting dihedral angle and other quality measures), efficiency (fast convergence rate and low computational cost per iteration) and flexibility (allowing for different optimization behavior with different quality metrics integrated). Our method also significantly improves the quality of boundary tets through harmonic-guided optimization.

Our optimization procedure is performed through vertex repositioning without considering topological transformation, such as mesh connectivity. We believe that even higher mesh quality can be achieved by coupling topology transformation with our energy minimization, at the expense of higher computational cost. We wish to investigate this in future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. We are also grateful to Andrea Tagliasacchi for proofreading the manuscript. Meshes 3 and 4 in Table 5 are taken from the AIM@SHAPE shape repository, 5 and 6 are courtesy of Bryan Matthew Klingner. This work was supported in part by grants from the 863 program of China (No. 2007AA01Z313 and No. 2008AA09Z124049), National Natural Science Foundation of China (No. 60773022 and No. 60707030) and NSERC (No. 611370).

References

[1] Alexa, M., 2001. Local control for mesh morphing. In: Proc. IEEE Conf. on Shape Modeling and Applications. pp. 209–215.

[2] Alliez, P., Cohen-Steiner, D., Yvinec, M., Desbrun, M., 2005. Variational tetrahedral meshing. ACM Trans. on Graphics 24 (3), 617–625.

[3] Andrei, N., 2004. Gradient flow method for nonlinear least squares minimization.

[4] Botsch, M., Pauly, M., Rössl, C., Bischoff, S., Kobbelt, L., 2006. Geometric modeling based on triangle meshes. Course Notes ACM SIGGRAPH 2006.

[5] Botsch, M., Sorkine, O., 2008. On linear variational surface deformation methods. IEEE Trans. Vis. & Comp. Graphics 14 (1), 213–230.

[6] Chen, L., 2004. Mesh smoothing schemes based on optimal delaunay triangulations. In: Proceedings of 13th International Meshing Roundtable. pp. 109–120.

[7] Cheng, S.-W., Dey, T. K., Edelsbrunner, H., Facello, M. A., Teng, S.-H., 1999. Sliver exudation. In: Proc. 15th Annu. ACM Sympos. Comput. Geom. pp. 1–13.

[8] Cignoni, P., Rocchini, C., Scopigno, R., 1998. Metro: measuring error on simplified surfaces. Computer Graphics Forum 17 (2), 167–174.

[9] Desbrun, M., Meyer, M., Schröder, P., Barr, A. H., 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of ACM SIGGRAPH 99. pp. 317–324.

[10] Du, Q., Wang, D., 2003. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. International Journal on Numerical Methods in Engineering 56 (9), 1355–1373.

[11] Elcott, S., Tong, Y., Kanso, E., Schröder, P., Desbrun, M., 2007. Stable, circulation-preserving, simplicial fluids. ACM Trans. on Graphics 26 (1), 4:1–4:12.

[12] Eppstein, D., 2001. Global optimization of mesh quality. In: Tutorial at 10th International Meshing Roundtable.

[13] Escobar, J. M., Rodrìguez, E., Montenegro, R., Montero, G., Gonzàlez-Yuste, J. M., 2003. Simultaneous untangling and smoothing of tetrahedral meshes. Comput. Methods Appl. Mech. Engrg. 192 (25), 2775–2787.

[14] Fleishman, S., Drori, I., Cohen-Or, D., 2003. Bilateral mesh denoising. ACM Trans. on Graphics 22 (3), 950–953.

[15] Freitag, L. A., 1999. Users manual for opt-ms: local methods for simplicial mesh smoothing and untangling. Tech. Rep. ANL/MCS-TM-239, Argonne National Laboratory, Argonne, IL.

[16] Freitag, L. A., Knupp, P. M., 2002. Tetrahedral mesh improvement via optimization of the element condition number. International Journal for Numerical Methods in Engineering 53 (6), 1377–1391.

[17] Freitag, L. A., Knupp, P. M., Munson, T., Shontz, S., 2006. A comparison of inexact newton and coordinate descent mesh optimization techniques. Engineering with Computers 22 (2), 61–74.

[18] Freitag, L. A., Ollivier-Gooch, C., 1996. A comparison of tetrahedral mesh improvement techniques. In: Proceedings of 5th International Meshing Roundtable. pp. 87–106.

[19] Freitag, L. A., Ollivier-Gooch, C., 1997. Tetrahedral mesh improvement using swapping and smoothing. International Journal for Numerical Methods in Engineering 40 (21), 3979–4002.

[20] Geiger, B., 1996. Nuages: a package for 3d reconstruction. URL `ftp://ftp-sop.inria.fr/prisme/NUAGES/`

[21] Jones, T. R., Durand, F., Desbrun, M., 2003. Non-iterative, feature-preserving mesh smoothing. ACM Trans. on Graphics 22 (3), 943–949.

Input Tetrahedral mesh			Initial Quality		Optimization Method	Optimized Quality		Cost	
Name	$\|\mathcal{V}\|$	$\|\mathcal{T}\|$	$\overline{q}_\eta$	σ_η		$\overline{q}_\eta$	σ_η	I	T(min.)
Fertility	21,213	108,120	0.417	0.280	Opt-MS	0.796	0.113	5	3.1
					SUS	0.878	0.102	6	8.7
					Global SUS	0.938	0.100	4	10.7
					QDC	0.945	0.065	3	0.5
Pegasus	46,212	234,512	0.635	0.401	Opt-MS	0.761	0.233	12	8.9
					SUS	0.842	0.203	15	19.7
					Global SUS	0.916	0.121	7	41.9
					QDC	0.923	0.089	3	1.7
Stanford Dragon	91,201	483,394	0.701	0.455	Opt-MS	0.722	0.318	26	58.2
					SUS	0.853	0.197	28	138.7
					Global SUS	—	—	–	>200
					QDC	0.892	0.174	4	12.4
Chinese Dragon	108,596	601,355	0.577	0.413	Opt-MS	0.628	0.300	37	140.2
					SUS	—	—	–	>200
					Global SUS	—	—	–	>200
					QDC	0.867	0.136	4	29.5

Table 3: Comparing quality (measured by the mean-ratio metric) improvements of four algorithms: Opt-MS [15], SUS [13], Global SUS and our QDC. $\overline{q}_\eta$ and σ_η denote the average and standard deviation of the mean-ratio measures, respectively. I and T are the number of iterations and total running time (in minutes), respectively.

Input Tetrahedral mesh			Opt-MS		SUS		ODC	
Name	$\overline{q}_\eta$	N_{tangle}	$\overline{q}_\eta$	T(min.)	$\overline{q}_\eta$	T(min.)	$\overline{q}_\eta$	T(min.)
Fertility	0.437	3,555	0.441	1.74	0.807	4.62	0.812	0.17
Pegasus	0.537	6,754	0.534	3.87	0.761	15.6	0.758	0.57
Happy Buddha	0.353	9,801	0.358	8.96	0.713	38.7	0.693	0.75

Table 4: Comparison of mesh untangling of Opt-MS [15], SUS [13] and our QDC. N_{tangle} is the number of tangled tets in the input mesh. T is the minimum running time (in minutes) needed to untangle all the tangled tets present in the input meshes. $\bar{q}_\eta$ is the average of mean-ratio quality measure at time T.

[22] Klingner, B. M., Shewchuk, J. R., 2007. Aggressive tetrahedral mesh improvement. In: Proceedings of 16th International Meshing Roundtable. pp. 3–23.

[23] Knupp, P. M., 2000. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii: a framework for volume mesh optimization and the condition number of the jacobian matrix. International Journal for Numerical Methods in Engineering 48 (8), 1165–1185.

[24] Knupp, P. M., 2001. Algebraic mesh quality metrics. SIAM Journal on Scientific Computing 23 (1), 193–218.

[25] Labelle, F., Shewchuk, J. R., 2007. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. ACM Trans. on Graphics 26 (3), 57:1–57:10.

[26] Liu, L., Tai, C.-L., Ji, Z., Wang, G., 2007. Non-iterative approach for global mesh optimization. Computer-Aided Design 39 (9), 772–782.

[27] Müller, M., Dorsey, J., McMillan, L., Jagnow, R., Cutler, B., 2002. Stable real-time deformation. In: Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation. pp. 49–54.

[28] Munson, T. S., 2004. Mesh shape-quality optimization using the inverse mean-ratio metric: tetrahedral proofs. Tech. Rep. ANL/MCS-TM-275, Argonne National Laboratory, Argonne, IL.

[29] Nealen, A., Igarashi, T., Sorkine, O., Alexa, M., 2006. Laplacian mesh optimization. In: Proceedings of ACM GRAPHITE. pp. 381–389.

[30] Parthasarathy, V. N., Graichen, C. M., Hathaway, A. F., 1994. A comparison of tetrahedron quality measures. Finite Elements in Analysis and Design 15 (3), 255–261.

[31] Sherbrooke, E. C., Patrikalakis, N. M., Brisson, E., 1996. An algorithm for the medial axis transform of 3d polyhedral solids. IEEE Trans. Vis. & Comp. Graphics 2 (1), 44–61.

[32] Shewchuk, J. R., 2002. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures. In: Proceedings of 11th International Meshing Roundtable. pp. 115–126.

[33] Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C., Seidel, H.-P., 2004. Laplacian surface editing. In: Proc. Eurographics Symp. on Geometry Processing. pp. 179–188.

[34] Taubin, G., 1995. A signal processing approach to fair surface design. In: Proceedings of ACM SIGGRAPH 95. pp. 351–358.

[35] Toledo, S., 2003. Taucs: a library of sparse linear solvers, version 2.2. URL `http://www.tau.ac.il/~stoledo/taucs/`

[36] Yamashita, N., Fukushima, M., 2000. On the rate of convergence of the levenberg-marquardt method. Tech. rep., Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University.

[37] Zhang, H., Fiume, E. L., 2002. Mesh smoothing with shape or feature preservation. In: Advances in Modeling, Animation, and Rendering. pp. 167–182.

[38] Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.-Y., 2005. Large mesh deformation using the volumetric graph laplacian. ACM Trans. on Graphics 24 (3), 496–503.

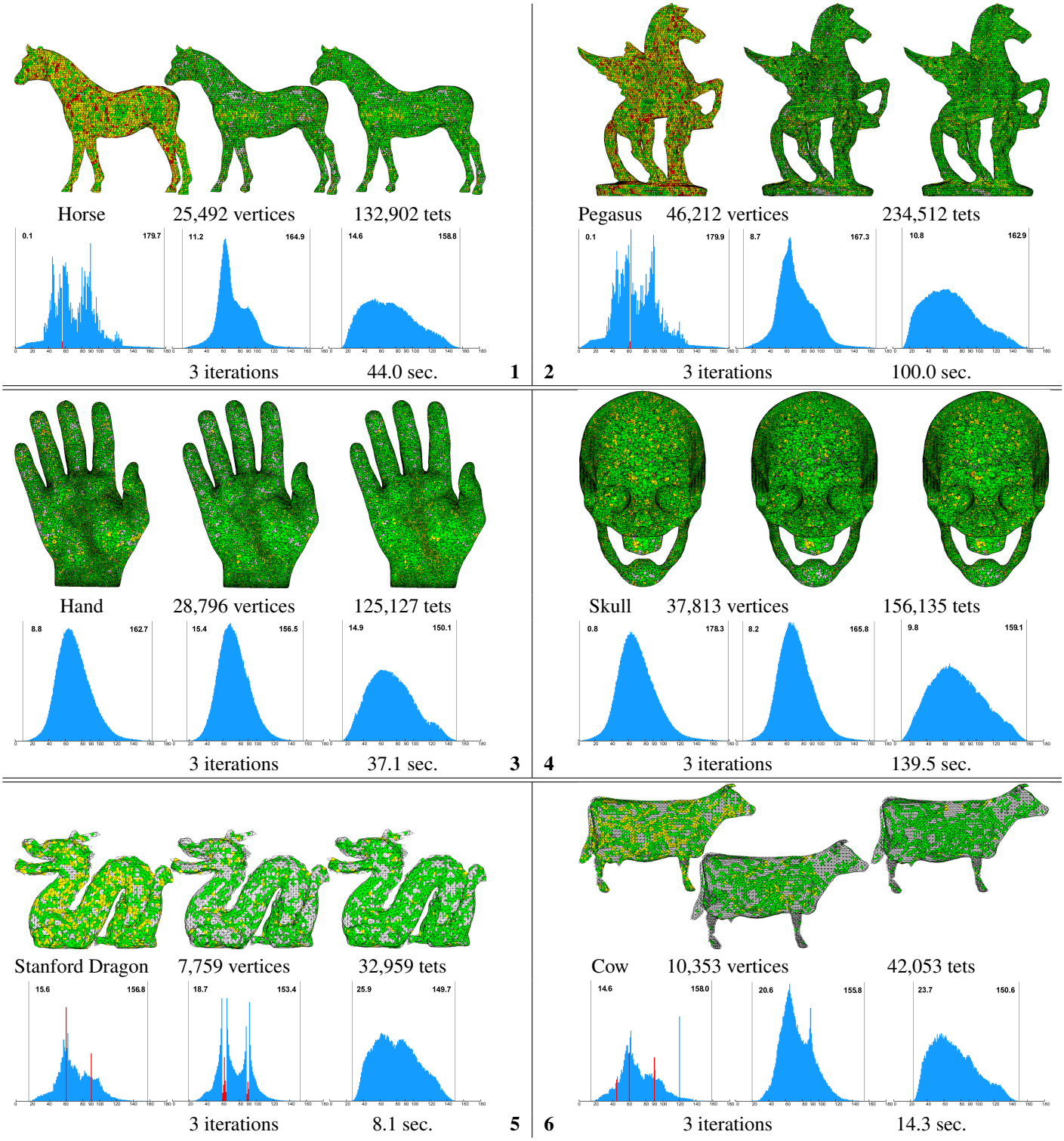


Table 5: Six meshes before and after quality improvement. In each box, the left mesh is the input, the middle mesh is optimized with the MIMR metric integrated, and the right one with the MIVL metric. Red tets have dihedral angles under 5° or over 175°, orange ones have angles under 15° or over 165°, yellow ones have angles under 30° or over 150°, green ones have angles under 40° or over 140°, and the remaining tets are not colored. The histograms (in 1° intervals) show the distributions of dihedral angles in each mesh, where the minimum and maximum angles are indicated at the two top corners. Note that in order to plot the distributions in the same scale while revealing them better, we down-scale the plot over certain intervals. Specifically, each red bar should have its height multiplied by 20 to reflect the true distribution.