DISCRETE MESH OPTIMIZATION ON SURFACE AND VOLUME MESHES

Daniel Zint Roberto Grosso Florian Lunz

Chair of Computer Vision
Friedrich-Alexander-Universität, 91058 Erlangen, Germany
{daniel.zint, roberto.grosso, flo.lunz}@fau.de

ABSTRACT

State of the art algorithms in surface mesh smoothing rely on computing new vertex positions on approximated shapes and re-projecting the results back onto the real surface or having no internal surface representation at all, which leads inevitably to suboptimal results. Discrete Mesh Optimization (DMO) is a greedy approach to topology-consistent mesh quality improvement, which was initially designed to smooth triangle and quadrilateral meshes in two dimensions and tetrahedral meshes in three dimensions. We present a generalization of DMO which allows optimization on discretized surfaces, or more general d-dimensional manifolds. As the method is not bound to search directions, it is capable of finding the optimal vertex positions directly on a surface without any re-projection. Therefore, the proposed technique preserves the underlying surface or volume. We present examples for surface and volume meshes, showing the improvement-potential of considering boundary vertices in the smoothing process.

Keywords: mesh improvement, mesh smoothing, max-min optimization

1. INTRODUCTION

The discretization of some space $\Omega \subset \mathbb{R}^d$ in form of a mesh plays an important role in finite element based simulations and in computer graphics. For both cases a smooth mesh is preferred although the reasons may be different. Also the definition of a *smooth* mesh varies substantially depending on the field of application. As a consequence of this diversity a vast range of smoothing and optimization methods emerged within the last decades [1, 2, 3]. These methods aim to improve mesh quality by repositioning vertices while keeping the topology unchanged. In current research mesh smoothing is still a vividly discussed topic, especially for volume meshes [4, 5, 6, 7].

The method Discrete Mesh Optimization (DMO) [8] improves mesh quality iteratively while finding the optimal position for each vertex relatively to its neighborhood by evaluating a set of candidate positions. DMO does not rely on derivatives which allows quality metrics to be non-smooth or even discontinuous. This also means that DMO can optimize for any quality criterion, e.g. roundness, anisotropy, or rectangularity. Furthermore, quality improvement is assured within each iteration.

Being independent of derivatives and search directions brings another advantage. The set of candidate positions can be mapped onto any parameterizable space. Thus, optimization of manifold meshes can be done precisely on the surface. Together with the assurance of quality improvement follows that DMO will converge to an optimal result. Many other methods tend to deform a mesh to something spherical.

We present a generalization of DMO to extend the area of application to meshes discretizing a domain $\Omega \subset \mathbb{R}^{d-k}$ with $d = 2, 3$ and $k \geq 0$, i.e. volume and surface meshes in 2- and 3-dimensional space. Furthermore, we introduce a smooth and vertex-interpolating surface estimation method. This enables applying DMO on surface meshes where the underlying shape is unknown. An important field of application is tetrahe-

dral mesh improvement. Smoothing not just interior but also boundary vertices affects mesh quality significantly.

In this paper, we restrict ourselves to simplicial meshes. Just like DMO, this method can be easily adapted for other mesh types. As long as a quality metric for a mesh exists, the method can be applied.

In Section 1.1 we present the *max-min* problem (as presented in [8]) which we consider as the core issue of mesh smoothing. In Section 1.2 we display some widely-used smoothing methods. Section 1.3 gives a short introduction to DMO, according to [8]. It is followed by Section 2 where we generalize DMO and define the space of functions on which DMO is guaranteed to find the local optimum. Additionally, a surface estimation method is presented, which can be used if no surface description is available. In Section 3 we compare our method to other smoothing approaches. Conclusions are given in Section 4.

1.1 The max-min Problem of Element Quality Improvement

Assume for each element e_k of a mesh M in $\mathbb{R}^d$ an element quality $q_\mathrm{k}^{(e)}$ is obtained by evaluating a quality metric $q^{(e)}(e_\mathrm{k})$. The quality $q_\mathrm{i}^{(v)}$ of a vertex v_i that is positioned at $\mathbf{x}$ is defined as the minimal quality of its incident elements $e_\mathrm{k} \in N_\mathrm{e}(v_\mathrm{i})$,

$$q_\mathrm{i}^{(v)}(\mathbf{x}) = \min_{e_\mathrm{k} \in N_\mathrm{e}(v_\mathrm{i})} q_\mathrm{k}^{(e)}(\mathbf{x})\,. \tag{1}$$

From Equation (1) follows the local optimization problem for finding the maximum quality $q_\mathrm{i,max}^{(v)}$ for a vertex v_i,

$$q_\mathrm{i,max}^{(v)} = \max_{\mathbf{x}} q_\mathrm{i}^{(v)}(\mathbf{x}) = \max_{\mathbf{x}} \min_{e_\mathrm{k} \in N_\mathrm{e}(v_\mathrm{i})} q_\mathrm{k}^{(e)}(\mathbf{x})\,. \tag{2}$$

The optimal position $\mathbf{x}^*$ for v_i is given as

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} q_\mathrm{i}^{(v)}(\mathbf{x}) = \arg\max_{\mathbf{x}} \min_{e_\mathrm{k} \in N_\mathrm{e}(v_\mathrm{i})} q_\mathrm{k}^{(e)}(\mathbf{x})\,. \tag{3}$$

An iterative method which is meant to improve element quality should aim for finding $\mathbf{x}^*$ which is non-trivial as Equation (3) is only C^0 continuous, e.g. Figure 1.

Note that DMO actually improves vertex quality $q^{(v)}$ not element quality $q^{(e)}$. Using the definition in Equation (1) implies that by improving $q^{(v)}$ also the minimal element quality increases. Depending on the optimization goal, it might be reasonable to adjust Equation (1). This was done for example in [9] to adapt element size.

1.2 Related Work

Smoothing methods can be divided into three main groups, Laplacian-, physics-, and optimization-based.

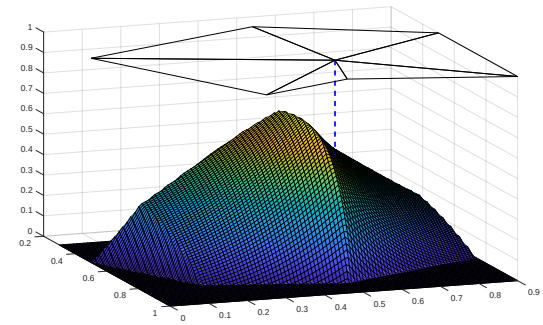


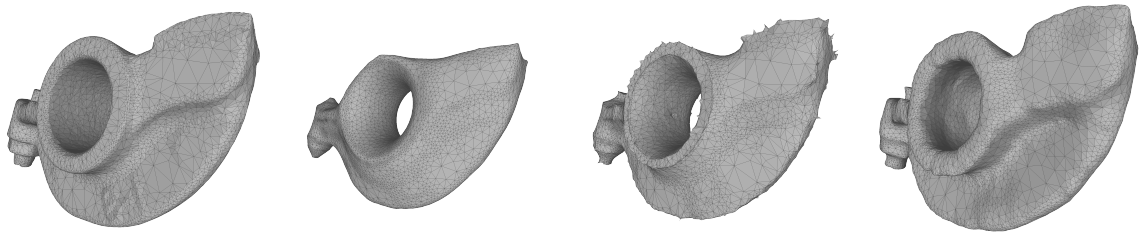
Figure 1: Vertex quality $q^{(v)}$ according to Equation (1) with $q^{(e)}$ defined as the mean ratio metric.

The classical Laplace-smoothing [10], developed for flat triangle meshes, is known to be fast but unstable in case of concave domains. A wide range of methods were proposed that modify the classical Laplace smoothing [11, 12, 13, 14, 15, 2, 3, 16, 17]. One representative is the "Smart" Laplacian Smoothing of Freitag [14] which only performs smoothing when mesh quality is increased. Using "Smart" Laplacian Smoothing without further processing steps does not lead to satisfying results as it does not improve mesh quality in concave regions. Freitag proposed to use it in combination with an optimization-based method. Laplacian-smoothers are fast but they also lead to suboptimal results. Many of these methods cannot guarantee that the quality will not decrease. Nevertheless, they are still used frequently.

Also for surface meshes Laplacian-based smoothing methods are standard. Here, they bring another disadvantage, namely they significantly change geometry when too many iterations are performed. Classical Laplace-smoothing drags the whole mesh towards its center of gravity, Figure 2(b). Applying cotangent weights makes the smoothing process unstable, Figure 2(c). A frequently used method is Taubin smoothing [18]. If only a few iterations are performed, the results look promising, as long as the parameters are chosen correctly. Changing the parameters or increasing the number of iterations also deforms the mesh, Figure 2(d).

Physics-based methods consider the mesh as a physical model. Some examples are spring-mass systems [19, 20], truss networks [21], or elasticity models [22, 23, 24]. Just like Laplace-based methods they do not provide any guarantee of mesh improvement.

Optimization-based methods are named after their approach of optimizing a quality metric. Some methods try to overcome the problem of non-differentiability by replacing Equation (2) with a smooth function



(a) Original (b) Laplace, 100 iterations (c) Laplace with cotangent weights, 50 iterations (d) Taubin, 500 iterations

Figure 2: Laplacian-based smoothing methods.

[25, 26, 27, 28, 29]. They run into the same problems as Laplacian-based methods when the domain is too complex.

To the authors knowledge, the currently most common smoothing algorithm is the method of Freitag. For example, it is implemented in the tetrahedral mesh improvement program Stellar [30]. While in [1, 14] the optimization is done with an analogue of the steepest descent method for smooth functions, later versions use the simplex algorithm to solve a linear programming problem [31, 32].

A derivative-free approach is done by Park and Shontz in [33]. They use pattern search in combination with backtracking line search to find a better vertex position. The convergence is suboptimal as it depends on search directions.

Rangarajan and Lew introduced the directional vertex relaxation (DVR) algorithm [7]. It solves the optimization problem by breaking it down to one dimension. This is achieved by providing a smoothing direction which can be chosen either randomly or by using previous knowledge. Within this one dimension the optimal solution can be found analytically. The major concern about this method is its randomness of relaxation directions as it leads to an inefficient smoothing with slow convergence.

Except Freitag's linear programming approach, all named optimization-based smoothing methods rely on linear search directions. For volume meshes this works fairly good, even though convergence is rather slow, but using search directions on surface meshes is not reasonable, because optimization is done along a line and not on the surface. A common way to deal with this problem is to perform the optimization on the tangent plane and re-project the vertex onto the surface. Finding the optimum cannot be guaranteed. Re-projection might even fail for complex geometries. Therefore, the search for optimal vertex positions should be restricted to the surface.

Zhang et al. [34] and Leng et al. [35, 36] present a geometric flow-driven approach for tetrahedral mesh smoothing. The methods use geometric partial differential equations to denoise the surface mesh and improve element quality while being volume preserving. For surface meshes emerging from geometry scanners, a wide range of feature preserving denoising methods exists [37, 38, 39, 40]. All these methods are not in scope of this work as we concentrate on element quality optimization, not on surface denoising. We expect an input mesh that represents its geometry correctly.

1.3 DMO

DMO solves the *max-min* problem in Equation (2) by evaluating the vertex quality function with a greedy algorithm on a uniform grid. The grid's center of gravity is set to the vertex that should be optimized. The grid size is defined by the axis aligned bounding box for the one-ring neighborhood and a scaling factor ω. Each grid-point is considered as candidate position where Equation (1) is evaluated, Figure 3. The vertex is repositioned at the best candidate if this increases its quality. After each iteration step the scaling factor ω is reduced such that the new grid size is twice the old grid spacing,

$$\omega \leftarrow \omega \cdot 2/(n-1), \tag{4}$$

where n is the number of grid points in one dimension. Furthermore, the grid is repositioned together with the vertex. Candidate evaluation and grid downscaling are repeated iteratively until the desired level of precision is reached.

2. GENERALIZATION OF DISCRETE MESH OPTIMIZATION

We reformulate DMO to cover surface and volume meshes by using an appropriate parameter space. We introduce the mapping of a uniform candidate grid from parametric domain Ξ^{d-k} to real space $\mathbb{R}^d$, with $d = 2, 3$ and $0 \leq k < d$. The mapping function is denoted by

$$\mathbf{x}(\xi): \quad \Xi^{d-k} \rightarrow \mathbb{R}^d\,. \tag{5}$$

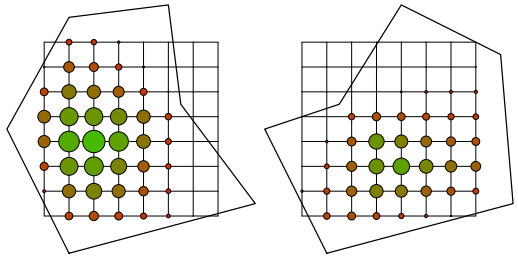


Figure 3: Uniform grid with quality metric evaluation for each candidate. A large green point represents good, a small red one bad quality

In general, the surface can be locally described as a smooth function,

$$s(\xi): \qquad \Xi^{d-k} \to \Xi^d\,, \tag{6}$$

where Ξ^d is the local coordinate system with Ξ^{d-k} being the parameter-subspace. The transformation from local to world coordinates is given as

$$\Phi(\xi): \quad \Xi^d \to \mathbb{R}^d\,, \tag{7}$$

and its inverse as

$$\Phi^{-1}(\mathbf{x}): \quad \mathbb{R}^d \to \Xi^d\,. \tag{8}$$

DMO works best if $\mathbf{x}(\xi)$ is known, i.e. a parametrization of the surface is given. Nevertheless, a surface description is often missing. We present a local surface estimation which is interpolating and tangential in all vertices. The advantage of converging towards an optimal mesh remains also on estimated surfaces.

Section 2.1 introduces our way of estimating smooth surfaces. Features, such as sharp edges, cannot be presented by smooth surfaces but also need to be preserved. Thus, Section 2.2 adds a feature detection approach. The application of DMO on surface meshes is explained in Section 2.3. Section 2.4 defines the function space on which DMO is guaranteed to converge locally towards the optimum.

2.1 Surface Estimation

The generalization of DMO allows smoothing on surfaces but often the surface is unknown, e.g. for meshes from point cloud reconstruction. In such a case, surface estimation is required. We present an approach based on moving least squares which produces smooth, vertex-interpolating surfaces.

Much research was done in estimating surface quantities [41, 42, 43, 44]. Unfortunately, these methods do not construct smooth surfaces. The transition between local surface estimations is often discontinuous. Moving least squares methods are commonly used for point cloud estimation [45, 46, 47, 48]. As we already have a mesh, we can use simpler techniques to get a surface estimation.

We base our approach on [49]. The method fits truncated Taylor expansions to the nearest neighbors of vertices. Its objective is actually to estimate differential quantities. We extend the method with a moving least squares approach inspired by [45] for creating a smooth surface estimation. Furthermore, we intentionally break with the smoothness property at feature edges to represent them correctly.

The surface is approximated at each vertex v_k with second order Taylor polynomials,

$$T_k(\xi) = \begin{pmatrix} \xi_1 \\ \xi_2 \\ \sum_{i=1}^{2}\sum_{j=i}^{2} a_{ij}\xi_i\xi_j \end{pmatrix}\,, \tag{9}$$

where we assume the normal and tangential vectors of the mesh in each vertex to represent the Monge coordinate system,

$$\begin{pmatrix} e_1' \\ e_2' \\ e_3' \end{pmatrix} = \begin{pmatrix} n \\ t_1 \\ t_2 \end{pmatrix}\,. \tag{10}$$

The transformations from local coordinate system (e_1', e_2', e_3') to world (e_1, e_2, e_3) and its inverse are

$$\Phi_k(\xi) = \mathbf{o}_k + \xi_1 e_1' + \xi_2 e_2' + \xi_3 e_3' \tag{11}$$

$$\Phi_k^{-1}(\mathbf{x}) = (e_1'|e_2'|e_3')^{\mathrm{T}}(\mathbf{x} - \mathbf{o}_k)\,, \tag{12}$$

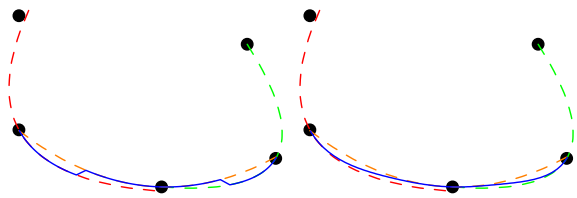
where $\mathbf{o}_k$ is the origin of the local coordinate system.

We chose second order polynomials in Equation (9) because we can determine the parameters a_{ij} with least squares using only the one ring neighborhood of a vertex. This results in a good approximation close to v_k which degrades with distance. Always using the local surface estimation of the nearest neighbor would be an easy way to overcome this problem, but doing so introduces discontinuous transitions on the Voronoi edges, Figure 4(a). Instead, we perform an interpolation between the local surfaces of one-ring-neighborhood-vertices N_k, with the center vertex v_k,

$$s_k(\mathbf{x}) = \frac{\sum_{v_i \in N_v} \omega_i(\mathbf{x})\Phi(T_i(\Phi_k^{-1}(\mathbf{x})))}{\sum_{v_i \in N_v} \omega_i(\mathbf{x})}\,, \tag{13}$$

where the inverse squared distance weighting $\omega_i(\mathbf{x}) = 1/\|\mathbf{x} - v_i + \varepsilon\|^2$ with $\epsilon > 0$ is applied to increase the influence of close vertices. The constant ε prohibits $\omega_i(v_i) \to \infty$ and should be chosen as small as possible. Furthermore, N_k also contains v_k itself. Thus, we have a smooth, vertex-interpolating surface estimation, Figures 4(b) and 5.

Note that $s_k(\mathbf{x})$ is stated in world coordinates, not in local coordinates as in Equation (6). This is just a



(a) Using nearest neighbor surface estimation. (b) Interpolating between local surface estimations.

Figure 4: Surface estimation with second order polynomials (dashed lines) and different transitions(blue line).

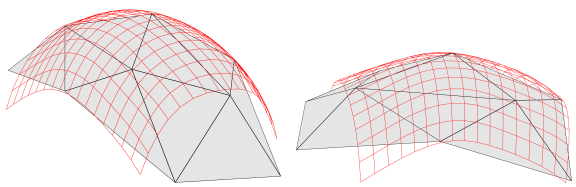


Figure 5: Surface estimation on a triangle mesh using interpolation between local surfaces.

matter of formulation. Equation (13) could also be stated in local coordinates but that would just make it more tedious to read.

The surface estimation is computed locally for each vertex of the original mesh. As vertices move during optimization, we always use the surface estimation of the closest vertex in the original mesh. In the beginning, each vertex will be its own nearest neighbor but due to mesh optimization, vertices might move quite far.

The surface estimation presented so far is only reasonable for smooth meshes. Sharp edges cannot be represented by second order Taylor polynomials. Therefore, we add a simple feature detection which takes care of boundaries and sharp edges, see Section 2.2. At feature vertices, a smooth surface estimation is not reasonable. Instead, the local surface estimation needs to be discontinuous. We estimate the surface at a feature vertex with tangential planes, separated by the feature edge, Figure 6. If a vertex is part of a feature edge, this edge splits the incident faces of the vertex in two sections. The normal of a tangential plane is the average normal of the incident faces on a section.

2.2 Feature Detection

In geometry processing, feature detection is a core issue and therefore well studied, [50, 51, 52, 53, 54]. Plenty of methods exist that can also handle noisy input data, [55, 56, 57]. We expect our input mesh to represent the geometry correctly and therefore do not require such sophisticated techniques.

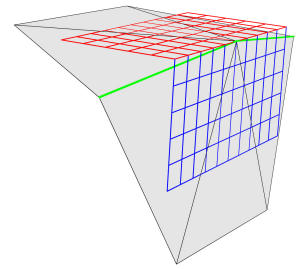


Figure 6: Surface estimation for a vertex on a feature edge (green).

We detect feature vertices by evaluating the angle between normals of two adjacent faces. If the angle is above a certain threshold, we consider the edge and its incident vertices as features. In our tests we used a threshold angle between 30 and 50 degrees, depending on the input mesh.

Geometric consistency does not necessarily require feature vertices to be static. A feature vertex v_k can be smoothed if it has exactly two incident feature edges which are not shared by one face. For example, in the mesh *cube*, Figure 7(a), only the corner vertices need to remain fixed. All other feature vertices can move along the edges without changing the geometry. Feature edge estimation is analogue to the surface estimation method in Section 2.1 but with $k = 2$, and the Taylor polynomial

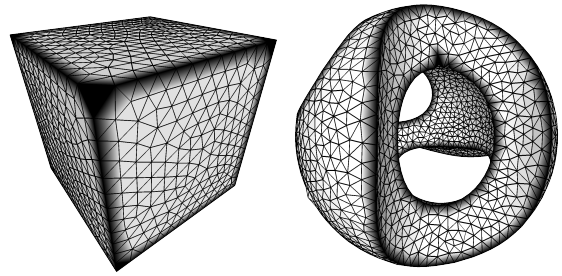
$$T_k(\xi) = \begin{pmatrix} \xi_1 \\ 0 \\ a\,\xi_1^2 \end{pmatrix}, \tag{14}$$

where e_1' and e_3' must lie in the plane described by v_k and its adjacent feature vertices. With feature edge estimation, smoothing is also possible on more complex feature edges, e.g. Figure 7(b).

In the special case that one face contains only feature vertices, none of these vertices may be repositioned. This special case appears when two feature edges intersect, e.q. the faces in the corners of *cube*, Figure 7(a).

2.3 DMO on Surface Meshes

The local surface around a vertex v_k is assumed to be either given or estimated, e.g. with the method in Section 2.1. Vertex v_k is positioned at $\mathbf{x}_k$ and has the local coordinates ξ_k, Figure 8(a). The candidate points for DMO are positioned around v_k in parametric space on a uniform grid with its center at ξ_k, Figure 8(b). For evaluating the quality metric the candidates are mapped onto the surface with Equation (5). The uniform grid is moved to the current optimum and scaled down, Figure 8(c). This is repeated until the desired level of precision is reached. Scaling down the grid



(a) *cube* (b) *sculpt*

Figure 7: Feature edges for different meshes shaded in black.

more than two times did not have significant impact on quality in our test cases. Finally, the vertex is repositioned at the best candidate, Figure 8(d). Algorithm 1 states the vertex optimization routine adapted for surface meshes.

The resulting mesh highly depends on the chosen quality metric. We got satisfying results using the mean ratio metric, see Section 3, but for other cases alternate metrics might be more suitable. As DMO's implementation is independent of the quality metric, users can plug in a different quality metric in the algorithm to obtain more appropriate results for their specific application.

2.4 Definition of Optimizable Function Space

We define the function space $\mathcal{F}^h$ containing all functions that can be optimized by DMO using the initial grid size h.

Proposition 1. *A function $f(\xi) : \Xi^{d-k} \to \mathbb{R}$ with $0 \leq k < d$ is in $\mathcal{F}^h$ if it satisfies:*

- *$f(\xi)$ contains exactly one maximum $f_{\max} = f(\xi_{\max})$ and is strictly decaying from it in all directions.*
- *Given nested balls*

$$S_i = \{\xi \in \mathbb{R}^d : \|\xi - \mathbf{o}\| = r_i\,, \quad i = 1, 2\} \tag{15}$$

with $r_1 = h/\sqrt{d}$, $r_2 = 3r_1$, and their shared origin $\mathbf{o}$ satisfying

$$\|\xi_{\max} - \mathbf{o}\| \leq r_1\,. \tag{16}$$

Assume a grid cell defined by its set of vertices G_h, with each vertex lying on the ball S_1, Figure 9. If for any orientation of the cell it holds,

$$\max_{\xi \in G_h} f(\xi) \geq \max_{\xi \in S_2} f(\xi), \tag{17}$$

then DMO will reach the optimum.

Proof. Assume $f_{\max}$ inside a grid cell defined by its set of vertices G_h, then G_h must contain the maximal value of all grid vertices to ensure convergence towards $f_{\max}$. Equation (17) states that the maximal value of G_h is larger than all values on S_2. Considering, that $f(\xi)$ is strictly decaying, it follows that no vertex $\|\mathbf{v} - \mathbf{o}\| \geq r_2$ has a value greater than the maximum of $Q_{\max}$. □

For the practical application of DMO it is not reasonable to check a function to satisfy Proposition 1 each time an optimization should be performed. Instead we include an additional step in the DMO algorithm. Prior to reducing the grid size it is checked, if the candidate node is on the grid-boundary. In this case, the grid size is kept unchanged and the greedy search is repeated but the center of the grid is moved to the last found optimum, Algorithm 1 Line 19-20. Thus, if the initial grid size was too large, the grid is iteratively moved towards the optimum. This heuristics produced good results in all cases we have applied our method.

As vertices never leave the estimated surface, DMO is point-wise surface preserving. In extreme cases, it might be possible that an edge "cuts" through the surface but this behavior was never experienced in any example. The same holds for volume preservation.

3. RESULTS

We compare the generalized DMO to other smoothing methods. In Section 3.1 the method is applied to triangle surface meshes as they are common in computer graphics. In Section 3.2 generalized DMO is compared to Stellar on tetrahedral meshes.

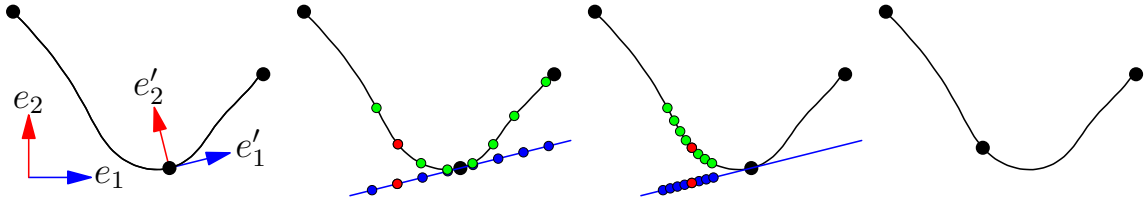
A quality comparison is given in terms of the mean ratio metric [58, 59, 60, 61, 31, 7], which is defined for triangles,

$$q_{\mathrm{m_{tri}}} = 4\sqrt{3}\frac{A}{\sum_{i=1}^{3} l_i^2}\,, \tag{18}$$

and tetrahedrons,

$$q_{\mathrm{m_{tet}}} = 12\sqrt[3]{9}\frac{V^{2/3}}{\sum_{i=1}^{6} l_i^2}\,, \tag{19}$$

where A is the signed area of the triangle, V the signed volume of the tetrahedron, and l_i is the length of their incident edges. Note that this quality measure does not depend on free parameters that have to be input by the user. Therefore, no user interaction is required. We display mesh-quality by lexicographically ordering the elements according to their quality. The element index is given in logarithmic scaling as the elements with lowest quality are the most interesting ones.



(a) Vertex with world and local coordinate system and estimated surface.

(b) Candidate positions in parametric space and on the surface.

(c) Refinement around optimal candidate.

(d) Smoothed mesh.

Figure 8: DMO on surface mesh. The blue line represents the parameter space on which the vertices are distributed uniformly.

Algorithm 1 Discrete optimization of vertex position

1: **function** OPTIMIZEVERTEXPOSITION(v, $N_\mathrm{e}(v)$, n, n_greedy)
2: $\omega \leftarrow 0.5$ ▷ Scaling-factor for grid
3: **for** *counter* $= 0$ **to** n_greedy **do**
4: $(\xi_{1,\min}, \xi_{2,\min}, \xi_{1,\max}, \xi_{2,\max}) = \mathrm{GETGRID}(v, N_\mathrm{e}(v), n, \omega)$
5: Create *quality_grid*[n][n]
6: **for** $i, j = 0$ **to** $n-1$ **do**
7: $\xi_1 \leftarrow \frac{i}{n-1}\xi_{1,\min} + \frac{n-1-i}{n-1}\xi_{1,\max}$
8: $\xi_2 \leftarrow \frac{j}{n-1}\xi_{2,\min} + \frac{n-1-j}{n-1}\xi_{2,\max}$
9: $\xi_3 \leftarrow s(\xi_1, \xi_2)$ ▷ Evaluate Equation (13)
10: $v' \leftarrow \Phi(\xi_1, \xi_2, \xi_3)$ ▷ Map from local to world coordinates
11: *quality_grid*[i][j] $\leftarrow \mathrm{VERTEXQUALITY}(v', N_\mathrm{e}(v))$
12: $q_\max \leftarrow -\infty,\ i_\max \leftarrow 0,\ j_\max \leftarrow 0$
13: **for** $i, j = 0$ **to** $n-1$ **do**
14: **if** *quality_grid*[i][j] $> q_\max$ **then**
15: $q_\max \leftarrow$ *quality_grid*[i][j]
16: $i_\max \leftarrow i,\ j_\max \leftarrow j$
17: **if** $q_\max > \mathrm{VERTEXQUALITY}(v, N_\mathrm{e}(v))$ **then**
18: $\xi_{1,\mathrm{opt}} \leftarrow \frac{i_\max}{n-1}\xi_{1,\min} + \frac{n-1-i_\max}{n-1}\xi_{1,\max}$
19: $\xi_{2,\mathrm{opt}} \leftarrow \frac{j_\max}{n-1}\xi_{2,\min} + \frac{n-1-j_\max}{n-1}\xi_{2,\max}$
20: $\xi_{3,\mathrm{opt}} \leftarrow s(\xi_{1,\mathrm{opt}}, \xi_{2,\mathrm{opt}})$
21: $v \leftarrow \Phi(\xi_{1,\mathrm{opt}}, \xi_{2,\mathrm{opt}}, \xi_{3,\mathrm{opt}})$
22: **if** $i == 0$ **or** $j == 0$ **or** $i == n-1$ **or** $j == n-1$ **then**
23: go to line 4
24: $\omega \leftarrow \omega \cdot 2/(n-1)$ ▷ Reduce scaling factor
return

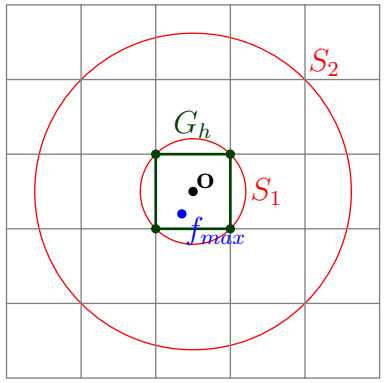


Figure 9: The best candidate of grid cell G_h must have a larger value then any point on S_2 to guarantee convergence towards $f_{\max}$.

3.1 Surface Meshes

We compare our method to Taubin smoothing [18]. We use the implementation in Meshlab [62] with default parameters. Other smoothing methods were also tested in Meshlab but they either did not improve quality or changed the geometry so significantly that a comparison is pointless. For the presented examples the real surface is unknown. Therefore, DMO uses surface estimation of Section 2.1.

The first comparison is done on mesh *tweety* with 6 752 vertices and a minimal quality 0.048, Figure 10(a). Taubin and DMO show both visually good improvements, Figures 10(b) and 10(c). However, Taubin smoothing leaves a significant amount of bad shaped triangles with a minimal quality of 0.089. DMO instead improves quality considerably to 0.484.

Similar behavior of the smoothing methods is observed on mesh *rocker-arm* with 10 044 vertices and a minimal quality of 0.077, Figure 11(a). Taubin smoothing improves quality overall but cannot get rid of low quality triangles resulting in a minimal quality of 0.241, Figure 11(b). DMO eliminates bad shaped elements completely giving a minimal quality of 0.599, Figure 11(c). Also, even with default parameters, the mesh becomes visibly more round at corners when applying Taubin smoothing. The shape remains unchanged by DMO.

On the mesh *hand* with 50 085 vertices, Figure 12(a), Taubin cannot deal with the more complex geometry, reducing minimal element quality from 0.343 to 0.033, Figure 12(b). DMO improves quality to 0.468, Figure 12(c).

The mesh *filigree* with 514 300 vertices results from marching cubes and therefore contains very thin triangles with a minimal element quality of 0.104, Figure 13. In this example, Taubin smoothing reaches a higher minimal element quality than DMO, namely

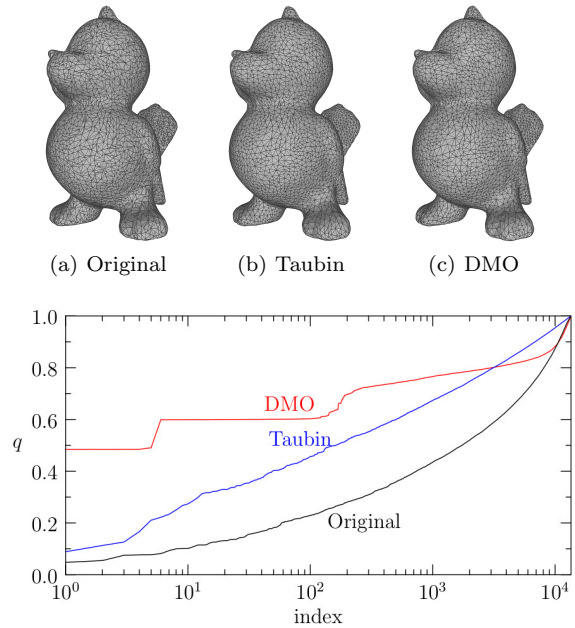


Figure 10: Comparison of Taubin smoothing and DMO on surface mesh *tweety*.

0.218 in comparison to 0.143. Considering that Taubin smoothing may leave the surface, this is not a surprising result. The mesh is mostly regular and Laplace-based smoothers are highly efficient in such cases. DMO instead is bound to its surface estimation. If geometric consistency is not required, then Taubin smoothing is preferable here.

Also mesh *shoe* has some thin triangles which cannot be improved, Figure 14. Therefore, the minimal element quality remains 0.018. Also Taubin smoothing does not improve quality. Instead it reduces it to 0.017. Besides the worst elements, DMO has a significant impact.

The last surface mesh example is *augustus* with 9 758 vertices. Again, DMO improves minimal element quality from 0.094 to 0.241. Taubin also improves minimal quality to 0.145. Taking a look at Figure 15 Taubin smoothing only improves the worst element but decreases quality of other bad shaped elements, whereas DMO increases quality of the worst 1 000 elements.

3.2 Volume Meshes

We compare DMO to Stellar [30], a tetrahedral mesh improvement software. We only allow topology-consistent optimizations to create a reasonable comparison. Stellar can also smooth boundary vertices. A quadric error metric is added to the optimization problem to keep the boundary mostly unchanged.

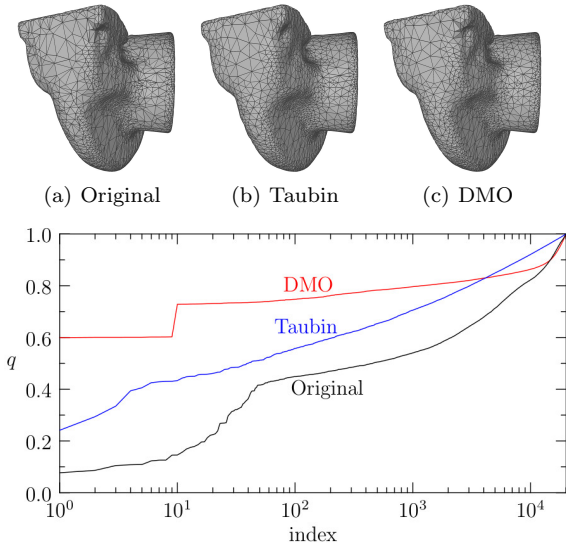


Figure 11: Comparison of Taubin smoothing and DMO on surface mesh *rocker-arm*.

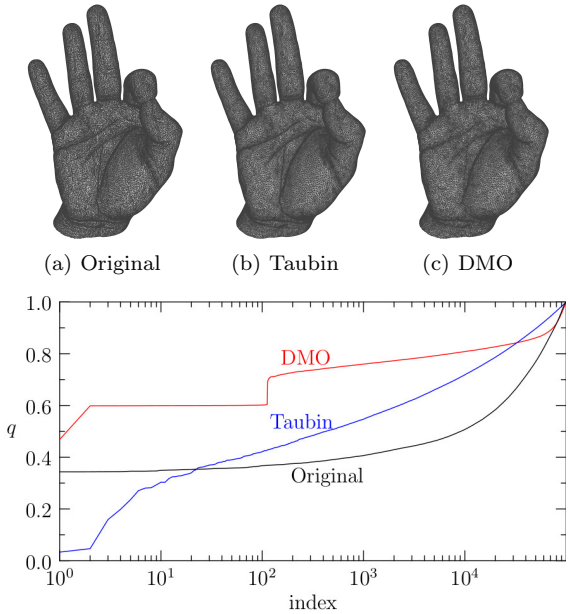


Figure 12: Comparison of Taubin smoothing and DMO on surface mesh *hand*.

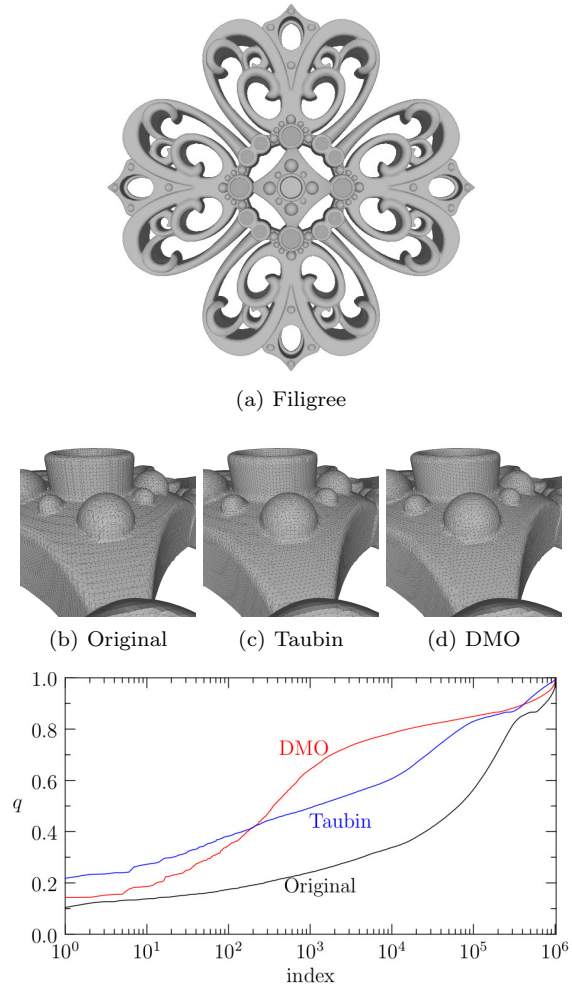


Figure 13: Comparison of Taubin smoothing and DMO on surface mesh *filigree*.

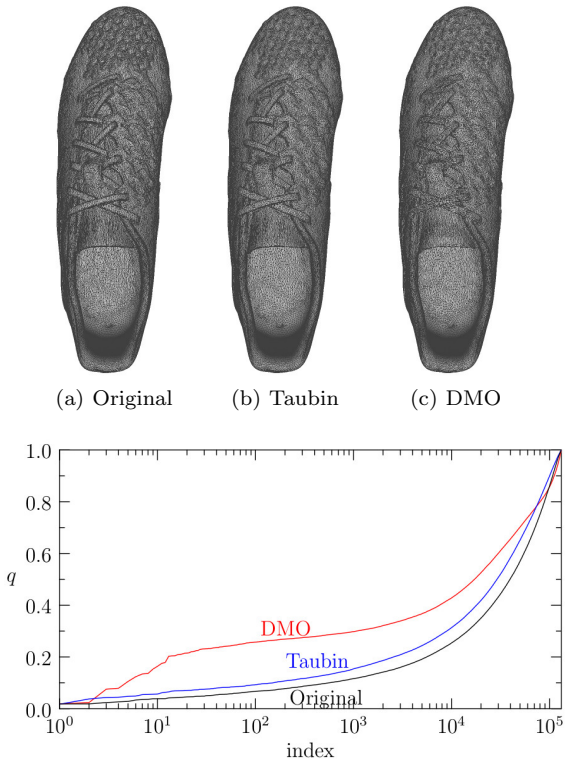


Figure 14: Comparison of Taubin smoothing and DMO on surface mesh *shoe*.

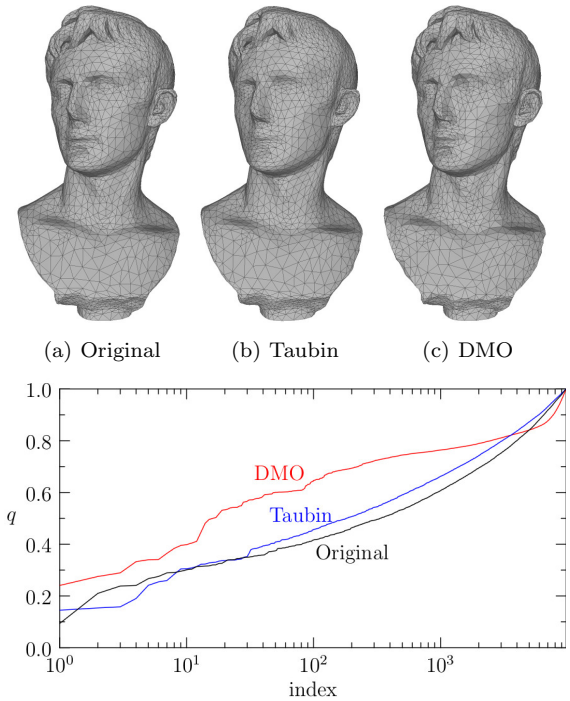


Figure 15: Comparison of Taubin smoothing and DMO on surface mesh *augustus*.

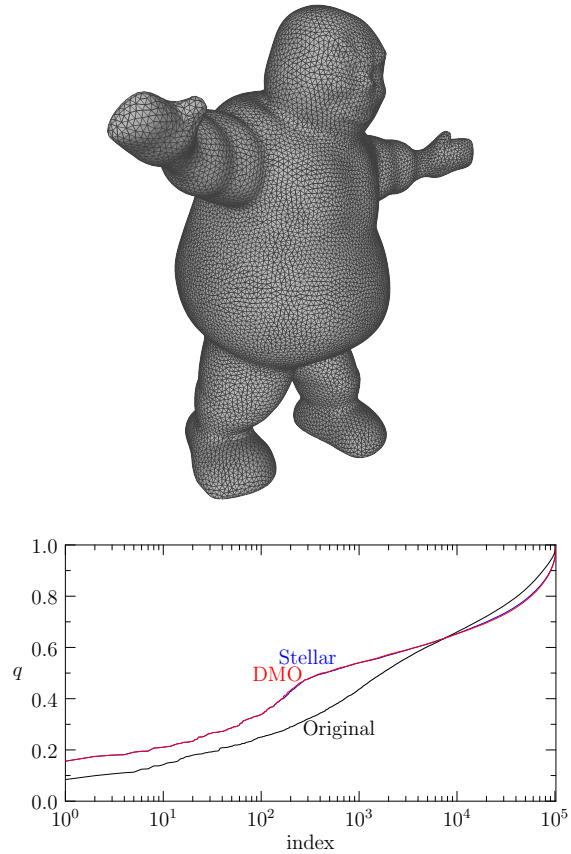


Figure 16: Volume mesh *staypuft* [30].

When deactivating boundary smoothing, DMO and Stellar converge towards the same result, Figure 16. This is expected as both solve the same optimization problem. The mesh *staypuft* from [30] was generated by Adam Bargteil's implementation of variational tetrahedral meshing [63].

Including boundaries, DMO has more freedom to move vertices along the boundary. Stellar penalizes boundary movement. Instead, DMO uses boundary information to put hard constraints on the optimization. This leads to much better quality, e.g. on mesh *cube*, Figure 17(a), from [30] using NETGEN [64]. Stellar can only slightly improve the minimal element quality from 0.628 to 0.635 in comparison to fixed boundaries. DMO reaches a minimal quality of 0.758. Furthermore, Stellar changed the boundary, e.g. the corner vertices. For numerical simulations this might be a critical issue.

The tetrahedral mesh *sphere*, Figure 18(a), created with *TetGen* [5], benefits also from boundary smoothing. Even though, the surface meshes of Stellar, Figure 18(b), and DMO, Figure 18(c) look disturbed, both improve tetrahedral mesh quality significantly,

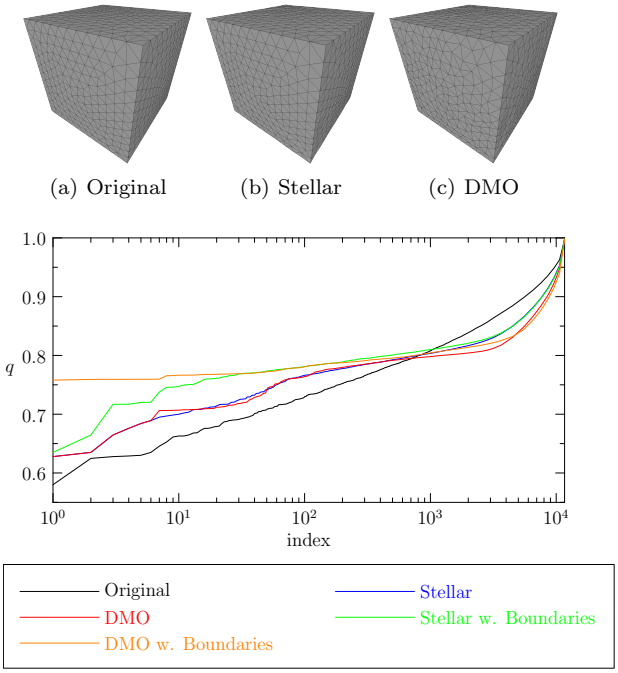


Figure 17: Volume mesh *cube*.

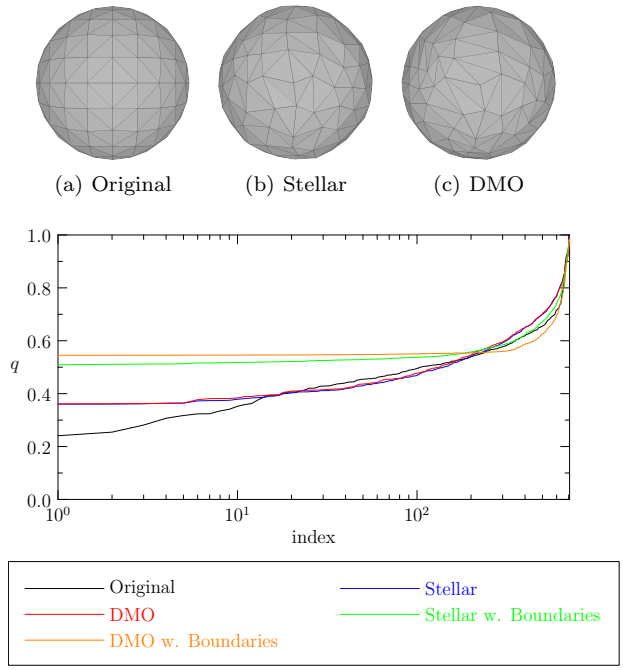


Figure 18: Volume mesh *sphere*.

Figure 18. Without boundary smoothing Stellar and DMO reach a minimal quality of 0.361. With boundary smoothing Stellar creates a minimal quality of 0.509, DMO of 0.545. Again, it has to be considered that some boundary vertices of Stellar do not lie precisely on the sphere anymore.

Observing Figures 18(b) and 18(c) one might realize that the surface meshes of DMO and Stellar look worse than the original. This is caused by optimizing for tetrahedral quality instead of triangle quality. Low quality elements are improved by decreasing the quality of their surrounding elements. Therefore, a former good looking surface mesh becomes disturbed, even though the minimal element quality increases.

Surface preservation constraints optimization. The mesh *sculpt*, Figure 19, from [30] was generated by Pierre Alliez's implementation of variational tetrahedral meshing [63]. On this mesh, Stellar creates a minimal element quality of 0.518, DMO 0.386. Slight movements on feature edges may have severe impact on element quality. DMO restricts movement to the feature edge, whereas Stellar has more freedom.

More complex shapes are presented with mesh *fandisk*, *bracket-2*, and *bracket-3*, Figures 20 to 22. The surface meshes are provided by and generated with *JIGSAW* [65]. The tetrahedral meshes were generated with *TetGen*. For uncertain reasons, Stellar cannot handle these meshes well. DMO performs as expected and

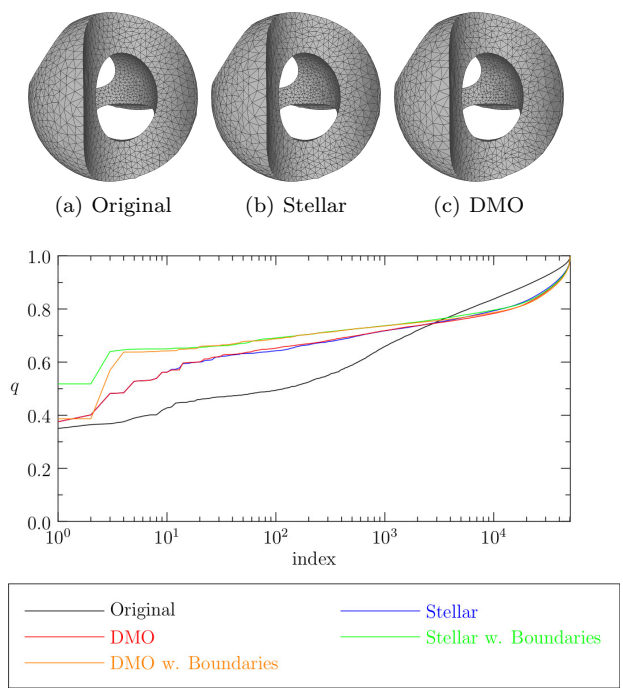


Figure 19: Volume mesh *sculpt*.

	DMO / Original	Stellar / Original
cube	1.0000	0.9995
sphere	0.9926	1.0044
sculpt	1.0002	0.9995
fandisk	1.0040	0.9998
bracket-2	1.0034	1.0000
bracket-3	1.0043	1.0000

Table 1: Relative volume of optimized meshes.

improves quality, when boundary vertices are also optimized. For fixed boundaries, DMO cannot improve minimal element quality.

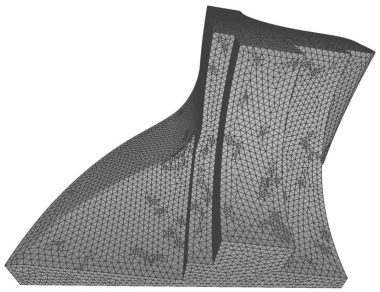
Our method preserves the described volume very well. Table 1 compares the mesh volume before and after optimization. The volumes never differ more than 1%.

DMO was developed for being executed on a GPU, which allows smoothing in the range of a few seconds. For the optimization of only interior vertices in tetrahedral meshes, DMO repositions about $1.1 \cdot 10^5$ vertices per second, e.g. DMO requires one second to perform 10 iterations on a mesh with 10 000 vertices. In most cases, 10 iterations are enough to converge towards the optimum. Unfortunately, so far the surface estimation is only implemented on CPU. Therefore, a valid performance statement cannot be done for surface optimization. We expect the performance to be comparable as surface estimation is mostly a pre-processing step. On CPU the performance for tetrahedral meshes drops to 230 vertices per second. Therefore, using a GPU is highly recommended.

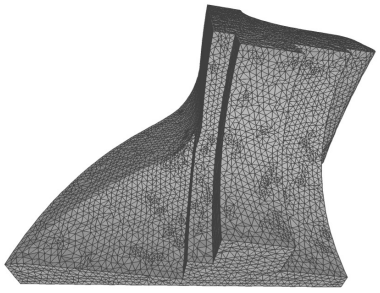
4. CONCLUSION

We presented a method to improve surface and volume meshes. Combining DMO's geometry- and topology-consistency with its local optimality, results in a method which produces high quality meshes. Surface mesh quality can be significantly improved without the need of user interaction as it is required in other methods. This is due to the mean ratio metric used for the examples presented above. We demonstrated the effectiveness of our method for different representative examples including surface and volume meshes. Furthermore, we showed that smoothing boundary vertices of tetrahedral meshes has significant impact on their quality.

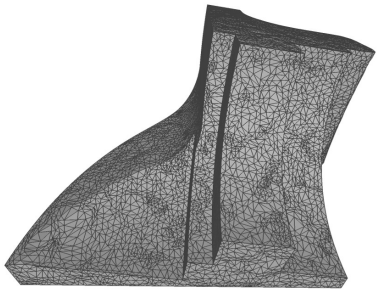
In future work we want to extend DMO with topological optimizations [30], [4]. DMO is perfectly suited for running on GPUs. We plan to develop a GPU implementation and compare performance to other smoothing techniques.



(a) Original



(b) Stellar



(c) DMO

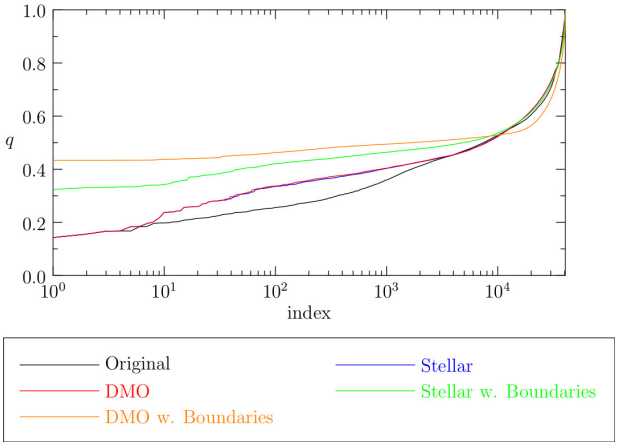


Figure 20: Volume mesh *fandisk*.

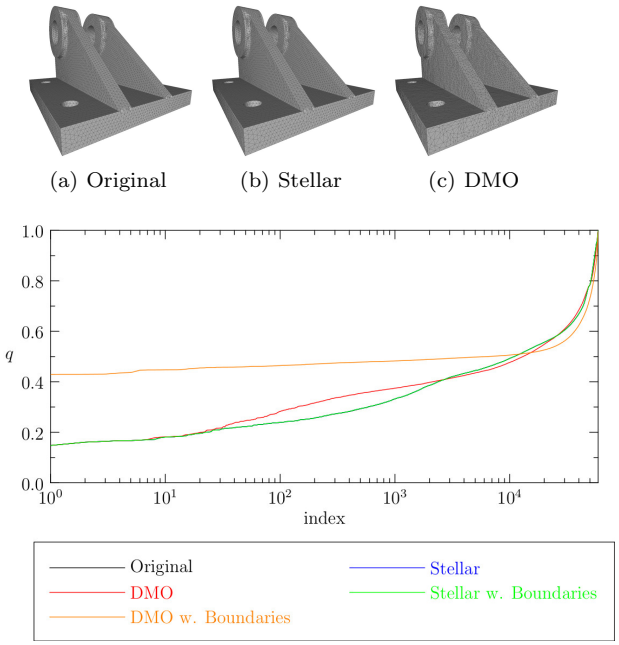


Figure 21: Volume mesh *bracket-2*.

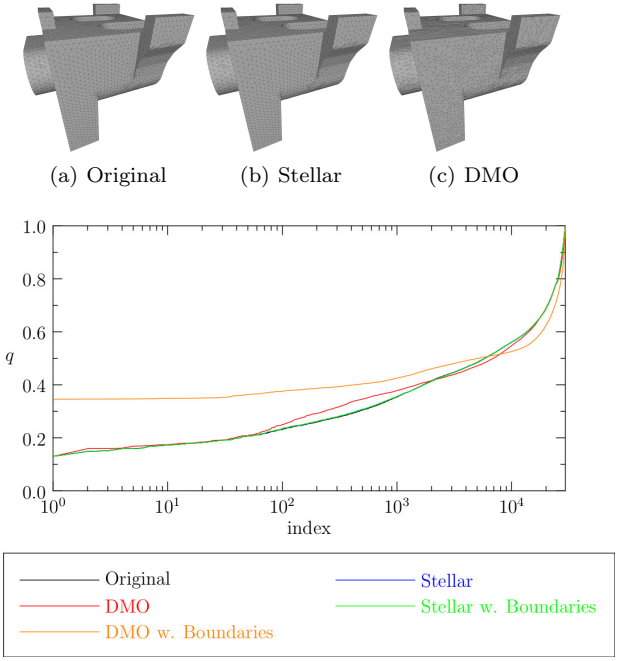


Figure 22: Volume mesh *bracket-3*.

ACKNOWLEDGMENT

This work has been supported by the DFG grant 'Rechenleistungsoptimierte Software-Strategien für auf unstrukturierten Gittern basierende Anwendungen in der Ozeanmodellierung' (GR 1107/3-1).

References

[1] Freitag L., Plassmann P., Jones M. "An efficient parallel algorithm for mesh smoothing." Tech. rep., Argonne National Lab., IL (United States), 1995

[2] Herrmann L.R. "Laplacian-isoparametric grid generation scheme." *Journal of the Engineering Mechanics Division*, vol. 102, no. 5, 749–907, 1976

[3] Jones R. "QMESH: A self-organizing mesh generation program." Tech. rep., Sandia Labs., Albuquerque, N. Mex.(USA), 1974

[4] Dassi F., Kamenski L., Si H. "Tetrahedral mesh improvement using moving mesh smoothing and lazy searching flips." *Procedia engineering*, vol. 163, 302–314, 2016

[5] Si H. "TetGen, a Delaunay-based quality tetrahedral mesh generator." *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 2, 11, 2015

[6] Wicke M., Ritchie D., Klingner B.M., Burke S., Shewchuk J.R., O'Brien J.F. "Dynamic local remeshing for elastoplastic simulation." *ACM Transactions on graphics (TOG)*, vol. 29, no. 4, 49, 2010

[7] Rangarajan R., Lew A.J. "Provably Robust Directional Vertex Relaxation for Geometric Mesh Optimization." *SIAM Journal on Scientific Computing*, vol. 39, no. 6, A2438–A2471, 2017

[8] Zint D., Grosso R. "Discrete Mesh Optimization on GPU." *27th International Meshing Roundtable*, 2018

[9] Zint D., Grosso R., Aizinger V., Köstler H. "Generation of Block Structured Grids on Complex Domains for High Performance Simulation (accepted)." *Numerical Geometry, Grid Generation and Scientific Computing*, 2019

[10] Field D.A. "Laplacian smoothing and Delaunay triangulations." *International Journal for Numerical Methods in Biomedical Engineering*, vol. 4, no. 6, 709–712, 1988

[11] Blacker T.D., Stephenson M.B. "Paving: A new approach to automated quadrilateral mesh generation." *International Journal for Numerical Methods in Engineering*, vol. 32, no. 4, 811–847, 1991

[12] Blacker T.D., Stephenson M.B., Canann S. "Analysis automation with paving: a new quadrilateral meshing technique." *Advances in engineering software and workstations*, vol. 13, no. 5-6, 332–337, 1991

[13] Canann S.A., Liu Y.C., Mobley A.V. "Automatic 3D surface meshing to address today's industrial needs." *Finite Elements in Analysis and Design*, vol. 25, no. 1-2, 185–198, 1997

[14] Freitag L.A. "On combining Laplacian and optimization-based mesh smoothing techniques." *ASME applied mechanics division-publications-amd*, vol. 220, 37–44, 1997

[15] George P., Borouchaki H. *Delaunay Triangulation and Meshing: Application to Finite Elements*. Hermès, 1998

[16] Knupp P.M. "Winslow smoothing on two-dimensional unstructured meshes." *Engineering with Computers*, vol. 15, no. 3, 263–268, 1999

[17] Zhou T., Shimada K. "An Angle-Based Approach to Two-Dimensional Mesh Smoothing." *IMR*, pp. 373–384. 2000

[18] Taubin G. "A signal processing approach to fair surface design." *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 351–358. ACM, 1995

[19] Blom F.J. "Considerations on the spring analogy." *International journal for numerical methods in fluids*, vol. 32, no. 6, 647–668, 2000

[20] Farhat C., Degand C., Koobus B., Lesoinne M. "Torsional springs for two-dimensional dynamic unstructured fluid meshes." *Computer methods in applied mechanics and engineering*, vol. 163, no. 1-4, 231–245, 1998

[21] Persson P.O., Strang G. "A simple mesh generator in MATLAB." *SIAM review*, vol. 46, no. 2, 329–345, 2004

[22] Baker T.J. "Mesh movement and metamorphosis." *Engineering with Computers*, vol. 18, no. 3, 188–198, 2002

[23] De Almeida V.F. "Domain deformation mapping: application to variational mesh generation." *SIAM Journal on Scientific Computing*, vol. 20, no. 4, 1252–1275, 1999

[24] Rumpf M. "A variational approach to optimal meshes." *Numerische Mathematik*, vol. 72, no. 4, 523–540, 1996

[25] Freitag L.A., Knupp P.M. "Tetrahedral mesh improvement via optimization of the element condition number." *International Journal for Numerical Methods in Engineering*, vol. 53, no. 6, 1377–1391, 2002

[26] Kim J. "A Multiobjective Mesh Optimization Algorithm for Improving the Solution Accuracy of PDE Computations." *International Journal of Computational Methods*, vol. 13, no. 01, 1650002, 2016

[27] Knupp P. "Updating meshes on deforming domains: An application of the target-matrix paradigm." *International Journal for Numerical Methods in Biomedical Engineering*, vol. 24, no. 6, 467–476, 2008

[28] Xu K., Gao X., Chen G. "Hexahedral mesh quality improvement via edge-angle optimization." *Computers & Graphics*, vol. 70, 17–27, 2018

[29] Zavattieri P.D., Dari E.A., Buscaglia G.C. "Optimization strategies in unstructured mesh generation." *International Journal for Numerical Methods in Engineering*, vol. 39, no. 12, 2055–2071, 1996

[30] Klingner B.M., Shewchuk J.R. "Aggressive tetrahedral mesh improvement." *Proceedings of the 16th international meshing roundtable*, pp. 3–23. Springer, 2008

[31] Freitag L., Jones M., Plassmann P. "A parallel algorithm for mesh smoothing." *SIAM Journal on Scientific Computing*, vol. 20, no. 6, 2023–2040, 1999

[32] Freitag L.A., Plassmann P., et al. "Local optimization-based simplicial mesh untangling and improvement." *International Journal for Numerical Methods in Engineering*, vol. 49, no. 1, 109–125, 2000

[33] Park J., Shontz S.M. "Two derivative-free optimization algorithms for mesh quality improvement." *Procedia Computer Science*, vol. 1, no. 1, 387–396, 2010

[34] Zhang Y., Xu G., Bajaj C. "Quality meshing of implicit solvation models of biomolecular structures." *Computer Aided Geometric Design*, vol. 23, no. 6, 510–530, 2006

[35] Leng J., Zhang Y., Xu G. "A novel geometric flow-driven approach for quality improvement

of segmented tetrahedral meshes." *Proceedings of the 20th international meshing roundtable*, pp. 347–364. Springer, 2011

[36] Leng J., Zhang Y., Xu G. "A novel geometric flow approach for quality improvement of multi-component tetrahedral meshes." *Computer-Aided Design*, vol. 45, no. 10, 1182–1197, 2013

[37] Fleishman S., Drori I., Cohen-Or D. "Bilateral mesh denoising." *ACM transactions on graphics (TOG)*, vol. 22, no. 3, 950–953, 2003

[38] Jones T.R., Durand F., Desbrun M. "Non-iterative, feature-preserving mesh smoothing." *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, 943–949, 2003

[39] Yu Y., Zhou K., Xu D., Shi X., Bao H., Guo B., Shum H.Y. "Mesh editing with poisson-based gradient field manipulation." *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, 644–651, 2004

[40] Choudhury P., Tumblin J. "The trilateral filter for high contrast images and meshes." *ACM SIGGRAPH 2005 Courses*, p. 5. ACM, 2005

[41] Boissonnat J.D., Flototto J. "A Local Coordinate System on a Surface." *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, SMA '02, pp. 116–126. ACM, New York, NY, USA, 2002. URL `http://doi.acm.org/10.1145/566282.566302`

[42] Goldfeather J., Interrante V. "A novel cubic-order algorithm for approximating principal direction vectors." *ACM Transactions on Graphics (TOG)*, vol. 23, no. 1, 45–63, 2004

[43] Kalogerakis E., Nowrouzezahrai D., Simari P., Singh K. "Extracting lines of curvature from noisy point clouds." *Computer-Aided Design*, vol. 41, no. 4, 282–292, 2009

[44] Demarsin K., Vanderstraeten D., Volodine T., Roose D. "Detection of closed sharp edges in point clouds using normal estimation and graph theory." *Computer-Aided Design*, vol. 39, no. 4, 276–283, 2007

[45] Lancaster P., Salkauskas K. "Surfaces generated by moving least squares methods." *Mathematics of computation*, vol. 37, no. 155, 141–158, 1981

[46] Levin D. "The approximation power of moving least-squares." *Mathematics of Computation of the American Mathematical Society*, vol. 67, no. 224, 1517–1531, 1998

[47] Fleishman S., Cohen-Or D., Silva C.T. "Robust moving least-squares fitting with sharp features." *ACM transactions on graphics (TOG)*, vol. 24, no. 3, 544–552, 2005

[48] Guennebaud G., Gross M. "Algebraic point set surfaces." *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, 23, 2007

[49] Cazals F., Pouget M. "Estimating differential quantities using polynomial fitting of osculating jets." *Computer Aided Geometric Design*, vol. 22, no. 2, 121–146, 2005

[50] Darom T., Keller Y. "Scale-invariant features for 3-D mesh models." *IEEE Transactions on Image Processing*, vol. 21, no. 5, 2758–2769, 2012

[51] Ohtake Y., Belyaev A., Seidel H.P. "Ridge-valley lines on meshes via implicit surface fitting." *ACM transactions on graphics (TOG)*, vol. 23, no. 3, 609–612, 2004

[52] Watanabe K., Belyaev A.G. "Detection of salient curvature features on polygonal surfaces." *Computer Graphics Forum*, vol. 20, no. 3, 385–392, 2001

[53] Hubeli A., Gross M. "Multiresolution feature extraction for unstructured meshes." *Proceedings of the Conference on Visualization'01*, pp. 287–294. IEEE Computer Society, 2001

[54] Zaharescu A., Boyer E., Varanasi K., Horaud R. "Surface feature detection and description with applications to mesh matching." *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 373–380. IEEE, 2009

[55] Kim H.S., Choi H.K., Lee K.H. "Feature detection of triangular meshes based on tensor voting theory." *Computer-Aided Design*, vol. 41, no. 1, 47–58, 2009

[56] Wang X.c., Cao J.j., Liu X.p., Li B.j., Shi X.q., Sun Y.z. "Feature detection of triangular meshes via neighbor supporting." *Journal of Zhejiang University Science C*, vol. 13, no. 6, 440–451, 2012

[57] Zhihong M., Guo C., Mingxi Z. "Robust detection of perceptually salient features on 3D meshes." *The Visual Computer*, vol. 25, no. 3, 289–295, 2009

[58] Amenta N., Bern M., Eppstein D. "Optimal point placement for mesh smoothing." *Journal of Algorithms*, vol. 30, no. 2, 302–322, 1999

[59] Bank R.E., Smith R.K. "Mesh smoothing using a posteriori error estimates." *SIAM Journal on Numerical Analysis*, vol. 34, no. 3, 979–997, 1997

[60] Bank R. "A Software Package for Solving Elliptic Partial Differential Equations–Users' Guide 7.0." *Frontiers in Applied Mathematics*, vol. 15, 1998

[61] Canann S.A., Tristano J.R., Staten M.L., et al. "An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes." *IMR*, pp. 479–494. Citeseer, 1998

[62] Cignoni P., Callieri M., Corsini M., Dellepiane M., Ganovelli F., Ranzuglia G. "Meshlab: an open-source mesh processing tool." *Eurographics Italian chapter conference*, vol. 2008, pp. 129–136. 2008

[63] Alliez P., Cohen-Steiner D., Yvinec M., Desbrun M. "Variational tetrahedral meshing." *ACM SIGGRAPH 2005 Courses*, p. 10. ACM, 2005

[64] Schöberl J. "NETGEN An advancing front 2D/3D-mesh generator based on abstract rules." *Computing and visualization in science*, vol. 1, no. 1, 41–52, 1997

[65] Engwirda D. "Conforming restricted Delaunay mesh generation for piecewise smooth complexes." *Procedia engineering*, vol. 163, 84–96, 2016