Comput. Methods Appl. Mech. Engrg. 201–204 (2012) 65–81

Contents lists available at SciVerse ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Fast smoothing of mixed volume meshes based on the effective geometric element transformation method

Dimitris Vartziotis [a,b,c,*], Joachim Wipper [c]

[a] Institute of Structural Analysis & Antiseismic Research, National Technical University Athens (NTUA), Zografou Campus, 15780 Athens, Greece
[b] NIKI Ltd. Digital Engineering, Research Center, 205 Ethnikis Antistasis Street, 45500 Katsika, Ioannina, Greece
[c] TWT GmbH Science & Innovation, Research Department, Bernhäuser Straße 40–42, 73765 Neuhausen, Germany

ARTICLE INFO

Article history:
Received 13 December 2010
Received in revised form 13 September 2011
Accepted 15 September 2011
Available online 22 September 2011

Keywords:
Mesh smoothing
GETMe
Mesh quality
Hybrid mesh
Mixed mesh
Finite element mesh

ABSTRACT

The geometric element transformation method (GETMe) is an efficient geometry driven approach to mesh smoothing. It is based on regularizing element transformations which, if applied iteratively to a single element, improve its regularity and with this its quality. The smoothing method has already successfully been applied in the case of mixed surface meshes as well as all-tetrahedral and all-hexahedral meshes. In this paper, a GETMe-based approach for smoothing mixed volume meshes is presented. For this purpose, dual element-based regularizing transformations for tetrahedral, hexahedral, pyramidal, and prismatic elements are introduced and analyzed. Furthermore, it is shown that the general concept of GETMe smoothing also applies to mixed volume meshes requiring only minor modifications. Numerical results demonstrate that high quality meshes comparable to those obtained by a state of the art global optimization-based approach can be achieved within significantly shorter runtimes.

1. Introduction

The generation of quality meshes plays a fundamental role in computations based on the finite element method, since mesh quality affects solver convergence and solution accuracy. A large number of mesh generation methods for volume meshes have been proposed (e.g. [1–3]). Amongst them Delaunay-based approaches to tetrahedral mesh generation are quite popular, since they are comparatively well suited for automation, of reasonable complexity, and result in good quality meshes. However, the use of hexahedral meshes is preferred, for example, in elastic/elasto-plastic solid continuum analysis or CFD applications, since more accurate results can be obtained with a lower number of elements [4,5]. Compared to the case of tetrahedral mesh generation, meshing complex volume models by hexahedra is considerably harder and the achieved degree of generality and automation is still far from the state obtained in the case of tetrahedral meshing. Therefore, the usage of hybrid meshes and corresponding mesh generators has been proposed in order to combine the advantages of both element types [6–8]. In addition, pyramids or prisms serve as transition elements to connect triangular and quadrilateral element faces. Depending on the application, hybrid meshes combining other types of elements might be preferred such as tetrahedral meshes with prismatic boundary layers in computational biofluid dynamic computations [9].

Usually mesh improvement techniques are incorporated into the corresponding mesh generation processes or applied afterwards [7,10]. They can be based on mesh topology altering operations like face swapping, node insertion, and removal or smoothing techniques. The latter improve mesh quality by mesh topology preserving node movements only. The most noted approach of this kind is Laplacian smoothing, where node positions are iteratively updated by the arithmetic mean of neighboring node positions [11]. Hence, Laplacian smoothing is computationally inexpensive and well suited for meshes combining arbitrary element types. However, due to its mesh quality unaware node averaging scheme Laplacian smoothing can lead to mesh quality deterioration and the generation of inverted elements. This can be avoided by accomplishing node movements only if quality is improved or by using relaxation techniques. Such methods are denoted as smart or constrained Laplacian smoothing [12,13].

However, better results with respect to a given quality criterion can be obtained by local optimization. That is by placing nodes in order to maximize the quality of adjacent elements. Since this

* Corresponding author at: Institute of Structural Analysis & Antiseismic Research, National Technical University Athens (NTUA), Zografou Campus, 15780 Athens, Greece. Tel.: +30 21077 21694/+30 26510 85230 (NIKI Ltd.).
E-mail address: dimitris.vartziotis@nikitec.gr (D. Vartziotis).
URLs: http://www.nikitec.gr (D. Vartziotis), http://www.twt-gmbh.de.

0045-7825/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2011.09.008

results in a higher computational effort, a combined approach of Laplacian smoothing and local optimization was proposed in [12], where optimization is only accomplished in problematic regions of the mesh.

Focusing on overall mesh quality, a natural choice is to use a global optimization approach incorporating quality numbers of all mesh elements into one objective function [14,15]. Here, all free vertices are moved simultaneously within a single iteration. Again, this comes at the expense of a higher implementational and computational effort. Furthermore, mathematical aspects, like optimization techniques and the proper choice of quality metrics, play a fundamental role [16,17]. For example, in the case of mixed mesh smoothing, quality metrics have to be applicable and balanced in their behavior for all element types under consideration. In addition, optimization-specific additional requirements, like differentiability or practice-oriented requirements like evaluation efficiency, have to be taken into account.

This might be one of the reasons why the number of publications addressing mixed volume mesh smoothing is comparatively low. A combined global optimization approach improving element size as well as shape is presented in [18]. Here, smoothing is driven by minimizing Riemannian metric non-conformity. Since the objective function is not easily differentiable, a brute force approach is used for minimization.

A more generalized approach to the construction of appropriate and flexible objective functions with respect to optimization aims is given by the target-matrix paradigm [19]. Here, for each sample point of the mesh two matrices are required. One is the Jacobian matrix of the current mesh, the other a target-matrix representing the desired Jacobian matrix in the optimal mesh. Usually sample points are given by element nodes and the target matrices are derived from type dependent reference elements. Using objective functions based on these matrices allow mesh quality to be defined on a higher conceptual level.

In [20], a finite element method-based approach is proposed. In doing so, a large nonlinear algebraic system obtained by Laplace–Beltrami equations for an unstructured mixed element volume mesh has to be solved by using Newton–Krylov methods. This is a critical task as issues in solver convergence observed in numerical examples have shown.

Compared to the simple geometry-based approach of smart Laplacian smoothing, these methods result in meshes of higher quality at the expense of significantly higher computational and implementational effort. This is one of the reasons why smart Laplacian smoothing was able to retain its popularity even though resulting mesh quality is inferior. As a way out of this dilemma between mesh quality and computational effort the geometric element transformation method (GETMe) has been proposed as a novel geometry-based approach to mesh smoothing. Various numerical tests indicate that it results in high quality meshes comparable to those obtained by global optimization-based approaches within significantly shorter runtimes. Being based on regularizing single element transformations it is easy to implement and well suited for parallelization. Furthermore, due to its universal approach GETMe smoothing has proven its effectiveness and practicability for various kinds of mesh types including triangular and mixed element planar or surface meshes, as well as all-tetrahedral and all-hexahedral meshes [21–24].

The driving force behind GETMe smoothing are regularizing element transformations which, if applied iteratively, lead to more regular hence better quality elements. Such transformations for polygons with an arbitrary number of nodes have been proposed and analyzed in [25–27]. Transformations for tetrahedra and hexahedra are given in [23,24]. In the first GETMe smoothing step all elements are transformed simultaneously and new node positions are obtained as weighted means of the resulting transformed element nodes. This is repeated until overall mesh quality converges. In the second GETMe smoothing step, only the worst elements are iteratively transformed in order to improve their quality numbers. In both cases, mesh quality is incorporated into the smoothing process by simple, element quality dependent parameters and control mechanisms. Due to its general approach, GETMe smoothing can be applied to all types of meshes with elements for which regularizing transformations exist.

This holds in particular for volume meshes with mixed element types, as will be shown in this paper. For this purpose, a unified transformation scheme based on dual elements is introduced in Section 2, which regularizes tetrahedral, hexahedral, pyramidal, and prismatic elements. Properties of the transformation are analyzed and the regularizing effect is substantiated by numerical tests. Section 3 recapitulates the mean ratio quality criterion controlled GETMe smoothing algorithm and addresses the minor changes to be made in the case of mixed mesh smoothing. Numerical results for generic hybrid meshes are given in Section 4 and compared to those obtained by smart Laplacian smoothing, as well as a state of the art global optimization-based approach. In addition, a variant of GETMe smoothing combining the mean ratio and the warpage quality criterion is presented, which demonstrates the flexibility of this smoothing method with respect to smoothing objectives.

2. Regularizing element transformation

The mesh smoothing approach presented below is mainly based on regularizing element transformations. That is, transformations, which successively transform an arbitrary initial element into its regular counterpart if applied iteratively. In this section, such a transformation will be presented and analyzed for common element types.

2.1. Definition of the transformation

In the majority of cases mixed volume meshes are assembled of tetrahedra, hexahedra, quadrilateral pyramids, and/or triangular prisms. Hence, this paper is restricted to those types. In the case of the first two element types, different regularizing transformation schemes have already been presented by the authors. Whereas the tetrahedral transformation scheme is based on opposite face normals [23], the hexahedral transformation scheme is based on dual elements [24]. In this paper, the dual element approach is also introduced for the other three element types, in order to provide a unified transformation scheme.

All elements are represented by node matrices, i.e. $E^{\text{tet}} = (p_1,\dots,p_4)$, $E^{\text{hex}} = (p_1,\dots,p_8)$, $E^{\text{pyr}} = (p_1,\dots,p_5)$, and $E^{\text{pri}} = (p_1,\dots,p_6)$ with columns $p_k \in \mathbb{R}^3$ representing the nodes. The node numbering scheme for all elements is depicted in Fig. 1. The figure not only depicts the elements under consideration but also their dual elements marked by blue[1] edges with nodes d_k. In preparation of the definition of d_k let

$$
\begin{aligned}
F^{\text{tet}} &:= ((1,2,3),(1,2,4),(2,3,4),(3,1,4)),\\
F^{\text{hex}} &:= ((1,2,3,4),(1,2,6,5),(2,3,7,6),(3,4,8,7),\\
&\qquad (4,1,5,8),(5,8,7,6)),\\
F^{\text{pyr}} &:= ((1,2,3,4),(1,2,5),(2,3,5),(3,4,5),(4,1,5)),\\
F^{\text{pri}} &:= ((1,2,3),(1,2,5,4),(2,3,6,5),(3,1,4,6),(4,6,5))
\end{aligned}
$$

denote the element faces tuples. That is, the kth element of a faces tuple $F \in \{F^{\text{tet}}, F^{\text{hex}}, F^{\text{pyr}}, F^{\text{pri}}\}$ represents the tuple of node indices defining the kth face. The index of the ith node of the kth face will

[1] For interpretation of colour in Figs. 1 and 2, the reader is referred to the web version of this article.

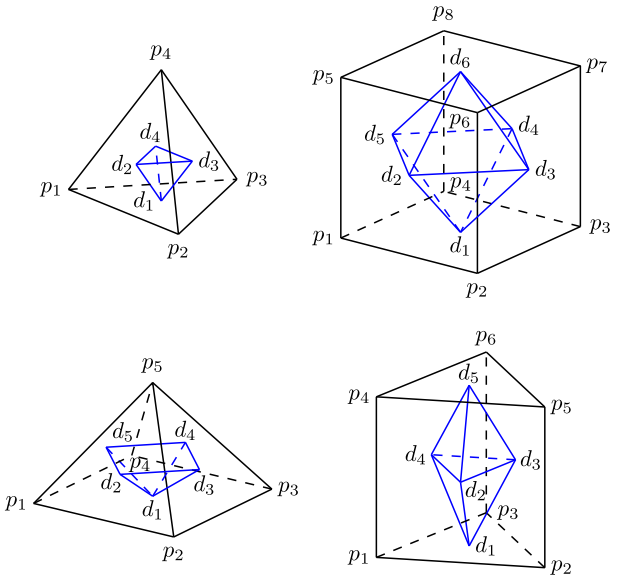

Fig. 1. Node numbering schemes for all element types and their dual elements.

be denoted as $F_{k,i}$. The nodes d_k of a dual element are defined as the arithmetic mean of the not necessarily coplanar face nodes, i.e.

$$d_k := \frac{1}{|F_k|}\sum_{i=1}^{|F_k|} p_{F_{k,i}}, \quad k \in \{1,\dots,|F|\}, \tag{1}$$

where $|F_k|$ denotes the number of nodes in the kth face and $|F|$ the number of element faces.

Whereas the tetrahedron and the pyramid are self-dual, the dual elements of the hexahedron and the prism are an octahedron and a triangular dipyramid, respectively. In addition, among all dual element faces, the base quadrilateral of the dual pyramid is the only non-triangular face. Node indices of all dual element faces are given by

$$\begin{aligned}
\overline{F}^{\text{tet}} &:= ((1,2,4),(1,3,2),(1,4,3),(2,3,4)),\\
\overline{F}^{\text{hex}} &:= ((1,2,5),(1,3,2),(1,4,3),(1,5,4),(6,5,2),\\
&\qquad (6,2,3),(6,3,4),(6,4,5)),\\
\overline{F}^{\text{pyr}} &:= ((1,2,5),(1,3,2),(1,4,3),(1,5,4),(2,3,4,5)),\\
\overline{F}^{\text{pri}} &:= ((1,2,4),(1,3,2),(1,4,3),(5,4,2),(5,2,3),(5,3,4)).
\end{aligned}$$

The regularizing transformation scheme is based on erecting scaled versions of the dual element faces normals on suitable base points. Here, the outward directed face normals are defined as

$$n_k := \begin{cases} (d_{\overline{F}_{k,2}} - d_{\overline{F}_{k,1}}) \times (d_{\overline{F}_{k,3}} - d_{\overline{F}_{k,1}}) & \text{if } |\overline{F}_k| = 3,\\ \frac{1}{2}(d_{\overline{F}_{k,3}} - d_{\overline{F}_{k,1}}) \times (d_{\overline{F}_{k,4}} - d_{\overline{F}_{k,2}}) & \text{if } |\overline{F}_k| = 4, \end{cases} \tag{2}$$

where $k \in \{1,\dots,|\overline{F}|\}$ and $\overline{F}$ denoting a dual element faces tuple taken from $\{\overline{F}^{\text{tet}},\overline{F}^{\text{hex}},\overline{F}^{\text{pyr}},\overline{F}^{\text{pri}}\}$. That is, in the case of triangular faces, the normal is determined by the cross product of two vectors spanning the face. In the case of the quadrilateral base of the pyramid, the normal is defined as the cross product of the two diagonals of the quadrilateral scaled by 1/2. The base points, on which the normals will be erected, are defined with the aid of the dual element faces centroids

$$c_k := \frac{1}{|\overline{F}_k|}\sum_{i=1}^{|\overline{F}_k|} d_{\overline{F}_{k,i}}, \quad k \in \{1,\dots,|\overline{F}|\}. \tag{3}$$

For the triangular faces of the duals of non-Platonic elements, i.e. pyramids and prisms, the point

$$a_k(\tau) := (1-\tau)d_{\overline{F}_{k,1}} + \tau\frac{1}{2}\left(d_{\overline{F}_{k,2}} + d_{\overline{F}_{k,3}}\right)$$

with $\tau \in \mathbb{R}$ is used. It is located on the line connecting the triangular face apex and the midpoint of the opposing side. The following definition specifies the choice of τ with respect to a normals scaling factor $\sigma > 0$. The latter will be used in order to control the regularizing effect of the single element transformation.

Definition 1 (*Element transformation*). Let $E \in \{E^{\text{tet}},E^{\text{hex}},E^{\text{pyr}},E^{\text{pri}}\}$ denote an arbitrary element with $|E|$ nodes p_k and dual element face normals n_k. Furthermore, let $\sigma \in \mathbb{R}^+$ denote an arbitrary normals scaling factor. The nodes p'_k of the transformed element $E' = (p'_1,\dots,p'_{|E|})$ are given by

$$p'_k := b_k + \frac{\sigma}{\sqrt{|n_k|}} n_k, \quad k \in \{1,\dots,|E|\}, \tag{4}$$

with element type dependent base points

$$b_k := \begin{cases} c_k & \text{if } E \in \{E^{\text{tet}},E^{\text{hex}}\} \vee (E = E^{\text{pyr}} \wedge k = 5),\\ a_k(\tau), \tau := \frac{1}{2} + \sigma & \text{if } E = E^{\text{pyr}} \wedge 1 \leqslant k \leqslant 4,\\ a_k(\tau), \tau := \frac{4}{5}\left(1 - \frac{\sqrt{2}\sigma}{\sqrt[4]{39}}\right) & \text{if } E = E^{\text{pri}}. \end{cases} \tag{5}$$

A basic prerequisite for regularizing element transformations is that regular elements are transformed into regular elements. Due to symmetry reasons, this holds for tetrahedral and hexahedral elements for the specific choice of $b_k = c_k$ and arbitrary $\sigma > 0$. However, this does not hold for the non-Platonic pyramidal and prismatic elements. Hence, for these elements an additional degree of freedom had to be included. It is represented by the σ-dependent parameter τ defining the position of the base points b_k of triangular faces chosen on the line connecting the apex and the midpoint of the opposing side. The specific choice of τ given in (5) has been determined by the basic regularity prerequisite. That is, by solving an equation based on the equality of transformed element edge lengths.

The construction of the transformed element is depicted in Fig. 2 for the specific choice $\sigma = 1$. Here, the faces of the dual elements are marked gray with blue edges. The base points b_k are indicated by black markers, the normals n_k by black arrows, and the resulting transformed elements E' red. In the case of the pyramidal and prismatic element, the line connecting the triangular face apex with the midpoint of the opposite side is marked green. Here, the resulting base point b_k may lie outside the associated dual element face as can be seen in the case of the pyramidal element.

2.2. Transformation properties

Basic properties of the transformation presented in the previous section have been analyzed for hexahedral elements in [24]. In this section, this analysis is extended to all element types under consideration.

Lemma 1. *The transformation given by Definition 1 is invariant with respect to translation, rotation, and scaling for all element types.*

Proof. Following the proof in [24] it has to be shown that for arbitrary scaling factors $\zeta > 0$, rotation matrices R (i.e. matrices with $\det R = 1$ and $R^{-1} = R^{\text{t}}$) and translation vectors $t \in \mathbb{R}^3$ it holds that $(\zeta R p_k + t)' = \zeta R p'_k + t$. This will be accomplished by applying the geometric transformation given by Definition 1 to the nodes $\tilde{p}_k := \zeta R p_k + t$ of an element $E \in \{E^{\text{tet}},E^{\text{hex}},E^{\text{pyr}},E^{\text{pri}}\}$ with the corresponding faces tuple F.

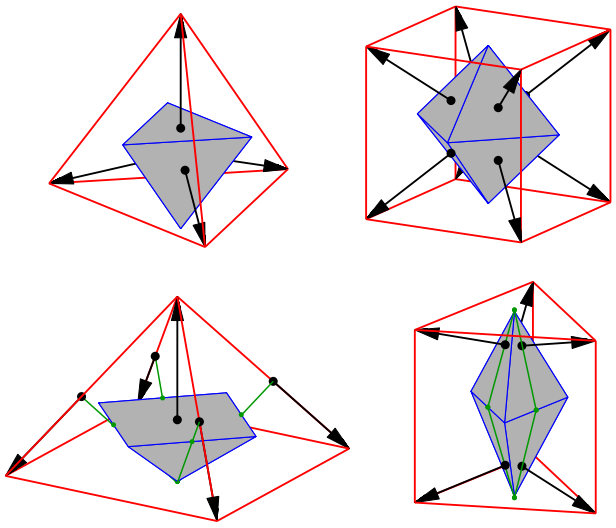

Fig. 2. Transformed element construction by erecting normals on the dual element.

According to (1) the nodes $\tilde{d}_k$ of the dual element associated to the element with nodes $\tilde{p}_k$ are given by

$$\tilde{d}_k := \frac{1}{|F_k|}\sum_{i=1}^{|F_k|}\tilde{p}_{F_{k,i}} = \frac{1}{|F_k|}\sum_{i=1}^{|F_k|}(\zeta R p_{F_{k,i}} + t) = \zeta R\left(\frac{1}{|F_k|}\sum_{i=1}^{|F_k|}p_{F_{k,i}}\right) + t$$
$$= \zeta R d_k + t,$$

with $k \in \{1,\dots,|F|\}$. Using (3) it can be shown analogously that the centroid $\tilde{c}_k$ of the kth dual element face with node indices given by $\overline{F}_k$ can be written as

$$\tilde{c}_k := \frac{1}{|\overline{F}_k|}\sum_{i=1}^{|\overline{F}_k|}\tilde{d}_{\overline{F}_{k,i}} = \zeta R c_k + t, \quad \text{for } k \in \{1,\dots,|\overline{F}|\}.$$

According to (5), the base node $\tilde{b}_k$ is either the centroid $\tilde{c}_k$ of the kth dual element face or given by the linear combination

$$\tilde{a}_k(\tau) := (1-\tau)\tilde{d}_{\overline{F}_{k,1}} + \frac{\tau}{2}\left(\tilde{d}_{\overline{F}_{k,2}} + \tilde{d}_{\overline{F}_{k,3}}\right)$$
$$= (1-\tau)(\zeta R d_{\overline{F}_{k,1}} + t) + \frac{\tau}{2}\left(\zeta R\left(d_{\overline{F}_{k,2}} + d_{\overline{F}_{k,3}}\right) + 2t\right)$$
$$= \zeta R\left((1-\tau)d_{\overline{F}_{k,1}} + \frac{\tau}{2}\left(d_{\overline{F}_{k,2}} + d_{\overline{F}_{k,3}}\right)\right) + t = \zeta R a_k(\tau) + t,$$

which implies $\tilde{b}_k = \zeta R b_k + t$ for $k \in \{1,\dots,|E|\}$. That is, the normal base nodes of the scaled, rotated, and translated element can be obtained by scaling, rotating, and translating the normal base nodes of the initial element.

For the normals $\tilde{n}_k$ of triangular dual element faces with nodes $\tilde{d}_{\overline{F}_{k,i}}$ it holds that

$$\tilde{n}_k := (\tilde{d}_{\overline{F}_{k,2}} - \tilde{d}_{\overline{F}_{k,1}}) \times (\tilde{d}_{\overline{F}_{k,3}} - \tilde{d}_{\overline{F}_{k,1}})$$
$$= (\zeta R(d_{\overline{F}_{k,2}} - d_{\overline{F}_{k,1}})) \times (\zeta R(d_{\overline{F}_{k,3}} - d_{\overline{F}_{k,1}}))$$
$$= \zeta^2 \det(R)(R^{-1})^{\mathrm{t}}\left[(d_{\overline{F}_{k,2}} - d_{\overline{F}_{k,1}}) \times (d_{\overline{F}_{k,3}} - d_{\overline{F}_{k,1}})\right] = \zeta^2 R n_k.$$

For normals of quadrilateral faces $\tilde{n}_k = \zeta^2 R n_k$ follows analogously.

These representations of $\tilde{b}_k$ and $\tilde{n}_k$ finally imply

$$(\zeta R p_k + t)' = \tilde{p}'_k = \tilde{b}_k + \frac{\sigma}{\sqrt{|\tilde{n}_k|}}\tilde{n}_k = \zeta R b_k + t + \frac{\sigma}{\zeta\sqrt{|n_k|}}\zeta^2 R n_k$$
$$= \zeta R\left(b_k + \frac{\sigma}{\sqrt{|n_k|}}n_k\right) + t = \zeta R p'_k + t,$$

thus proving the invariance property of the transformation as stated by the lemma. □

The last step in the proof of Lemma 1 also gives the motivation for the choice of the normalization factor $1/\sqrt{|n_k|}$ in the definition (4) of p'_k, since it ensures the scale invariance of the regularizing element transformation.

For an arbitrary element $E \in \{E^{\mathrm{tet}}, E^{\mathrm{hex}}, E^{\mathrm{pyr}}, E^{\mathrm{pri}}\}$ with $|E|$ nodes p_k let

$$q := \frac{1}{|E|}\sum_{k=1}^{|E|}p_k \tag{6}$$

denote the centroid of the initial element. Furthermore, let

$$q' := \frac{1}{|E|}\sum_{k=1}^{|E|}p'_k = \underbrace{\frac{1}{|E|}\sum_{k=1}^{|E|}b_k}_{=:q'_b} + \underbrace{\frac{\sigma}{|E|}\sum_{k=1}^{|E|}\frac{1}{\sqrt{|n_k|}}n_k}_{=:q'_n} \tag{7}$$

denote the centroid of the transformed element and its partition into the mean base node q'_b as well as the mean normal q'_n. The following lemma considers the relation between the centroids of initial and transformed elements.

Lemma 2 (Transformed element centroid). *For $E \in \{E^{tet}, E^{hex}, E^{pri}\}$ it holds that $q = q'_b$. In addition, for $E = E^{hex}$ it holds that $q = q'$, i.e. the transformation preserves the centroid of hexahedral elements.*

Proof. In the case of hexahedral elements, $q = q' = q'_b$ has been already shown in [24]. Here, $q = q'_b$ was deduced similarly to the proof for tetrahedral elements given in the following. Furthermore, it has been shown that the normals of opposite dual element faces only differ in their sign, thus resulting in $q'_n = (0,0,0)^{\mathrm{t}}$, which implies $q'_b = q'$.

In the case of tetrahedral elements successively substituting the expressions (3) and (1) in the representation of the base nodes $b_k = c_k$ implies

$$q'_b = \frac{1}{4}\sum_{k=1}^{4}c_k = \frac{1}{4}\sum_{k=1}^{4}\left(\frac{1}{3}\sum_{i=1}^{3}\left(\frac{1}{3}\sum_{j=1}^{3}p_{F_{\overline{F}_{k,i},j}}\right)\right) = \frac{1}{36}\sum_{k=1}^{4}9p_k = q.$$

Here, it has been used that each node p_k is involved in the three adjacent element face centroids d_k and with each of this in three dual element face centroids c_k.

Finally, in the case of prismatic elements the base nodes are given by $b_k = a_k(\tau)$ resulting in

$$q'_b = \frac{1}{6}\sum_{k=1}^{6}a_k(\tau) = \frac{1}{6}\sum_{k=1}^{6}\left[(1-\tau)d_{\overline{F}_{k,1}} + \frac{\tau}{2}(d_{\overline{F}_{k,2}} + d_{\overline{F}_{k,3}})\right]$$
$$= \frac{1}{6}[3(1-\tau)(d_1+d_5) + 2\tau(d_2+d_3+d_4)]$$
$$= \frac{1}{6}[(1-\tau)(p_1+\cdots+p_6) + \tau(p_1+\cdots+p_6)] = q.$$

Since $q'_b = q$ has been shown for arbitrary τ, this holds in particular for the specific choice of τ given in (5). □

Whereas Lemma 2 implies $q'_n = (0,0,0)^{\mathrm{t}}$ for arbitrary hexahedra, this does not generally hold for tetrahedral, pyramidal and prismatic elements as can be shown by simple counterexamples. However, centroid preservation can easily be achieved by an additional translation step after transforming the element. According to Lemma 2 the required translation vector is given by $-q'_n$ in the case of tetrahedral and prismatic elements and by $q - q'$ in the case of pyramidal elements. For the latter even $q = q'_b$ does not hold in general as can also be shown by simple counterexamples.

Due to its geometric construction and depending on the choice of the normals scaling factor σ given in Definition 1, the size of a transformed element can differ significantly from the size of the

initial element. This can be avoided by an element scaling step, which can be combined with the centroid preservation step described in the previous paragraph. Let $Q = (q,\dots,q)$ and $Q' = (q',\dots,q')$ denote $3 \times |E|$ matrices with each column representing the initial and transformed element centroid q and q', respectively. For an arbitrary scaling factor $\zeta > 0$ the scaled transformed element E'_s preserving the initial element centroid q is given by

$$E'_s := Q + \zeta(E' - Q'). \tag{8}$$

A natural choice of ζ applicable to all element types is, for example, the mean length of all edges of E divided by the mean length of all edges of E'. This choice of ζ will be denoted as mean edge length preserving scaling. Other scaling schemes could for example be based on preserving the mean volume of all node tetrahedra T_k.

In the following, the regularizing property of the element transformation will be analyzed, based on the well established mean ratio quality criterion [14,16]. As a prerequisite of its definition, specific element validity conditions have to be stated first. Element validity will be assessed based on node tetrahedra, that is tetrahedra which are spanned by the three edges emanating from one node of the element. Here, node indices of such node tetrahedra are given by the tuples

$$\begin{aligned}
N^{\text{tet}} &:= ((1,2,3,4)),\\
N^{\text{hex}} &:= ((1,4,5,2),(2,1,6,3),(3,2,7,4),(4,3,8,1),(5,8,6,1),\\
&\qquad (6,5,7,2),(7,6,8,3),(8,7,5,4)),\\
N^{\text{pyr}} &:= ((1,2,4,5),(2,3,1,5),(3,4,2,5),(4,1,3,5)),\\
N^{\text{pri}} &:= ((1,2,3,4),(2,3,1,5),(3,1,2,6),(4,6,5,1),(5,4,6,2),\\
&\qquad (6,5,4,3)).
\end{aligned}$$

That is, the kth element of $N \in \{N^{\text{tet}},N^{\text{hex}},N^{\text{pyr}},N^{\text{pri}}\}$ represents the four node indices $N_{k,i}$, $i \in \{1,\dots,4\}$, of the node tetrahedron associated with p_k. The number of node tetrahedra, i.e. the number of elements in the tuple N, is denoted as $|N|$. It should be noticed that in the case of E^{tet} each of the four node tetrahedra represents E^{tet} itself. Hence it suffices to use only the node tetrahedron associated with p_1 resulting in $|N^{\text{tet}}| = 1$. Furthermore, the apex p_5 of the pyramidal element is the only node with an emanating edge number being not equal to three. However, this node is omitted, since only the four base node tetrahedra will be used in order to define the mean ratio quality number of a pyramidal element. Consequently, for the tetrahedral as well as the pyramidal element it holds that $|E| \neq |N|$.

Definition 2 (*Element validity*). Let $E \in \{E^{\text{tet}},E^{\text{hex}},E^{\text{pyr}},E^{\text{pri}}\}$ denote an arbitrary element with its associated node tetrahedra indices tuple $N \in \{N^{\text{tet}},N^{\text{hex}},N^{\text{pyr}},N^{\text{pri}}\}$. For each node p_k, $k \in \{1,\dots,|N|\}$, let

$$T_k := (p_{N_{k,1}},\dots,p_{N_{k,4}}) \tag{9}$$

denote the node tetrahedron associated with the node p_k, and

$$D(T_k) := (p_{N_{k,2}} - p_{N_{k,1}}, p_{N_{k,3}} - p_{N_{k,1}}, p_{N_{k,4}} - p_{N_{k,1}}),$$

the (3×3)-matrix of its spanning edge vectors. The element is called *valid* if $\det(D(T_k)) > 0$ holds for all $k \in \{1,\dots,|N|\}$.

The given validity criterion excludes in particular specific cases of degenerated elements, like for example tetrahedra with three identical nodes, for which the geometric element transformation according to Definition 1 is not defined. However, the prerequisites of the mean ratio quality criterion or additional prerequisites imposed by the subsequent application, like for example a positive interior Jacobian in the case of the finite element approach, are stronger than those of the geometric transformation in order to be well defined.

Introduced in the context of optimization-based mesh smoothing techniques, the mean ratio quality criterion is based on measuring the deviation of an arbitrary valid element from its ideal counterpart, represented by a target matrix W [16]. This is done by taking the average deviation of the node tetrahedra according to the following definition.

Definition 3 (*Mean ratio quality criterion*). Let $E \in \{E^{\text{tet}},E^{\text{hex}},E^{\text{pyr}},E^{\text{pri}}\}$ denote an arbitrary valid element and N its associated indices tuple of node tetrahedra T_k according to (9). The *mean ratio quality number* of the element is given by

$$q(E) := \frac{1}{|N|}\sum_{k=1}^{|N|}\frac{3\det(S_k)^{2/3}}{\operatorname{trace}(S_k^{\text{t}}S_k)}, \quad S_k := D(T_k)W^{-1}, \tag{10}$$

with an element type dependent target matrix

$$W := \begin{cases}
\begin{pmatrix} 1 & 1/2 & 1/2 \\ 0 & \sqrt{3}/2 & \sqrt{3}/6 \\ 0 & 0 & \sqrt{2/3} \end{pmatrix} & \text{if } E = E^{\text{tet}},\\
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \text{if } E = E^{\text{hex}},\\
\begin{pmatrix} 1 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 0 & 0 & \sqrt{2}/2 \end{pmatrix} & \text{if } E = E^{\text{pyr}},\\
\begin{pmatrix} 1 & 1/2 & 0 \\ 0 & \sqrt{3}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \text{if } E = E^{\text{pri}}.
\end{cases}$$

Here, the columns of W represent the spanning edge vectors of a node tetrahedron in the associated regular element.

In the definition of the mean ration quality criterion given in [14] it is assumed that each node of the element is connected to the same number of edges. This does not hold for the pyramidal element where the apex is connected to four edges. However, in accordance to [28], in this case only the node tetrahedra associated to the four base nodes are used for quality computation. It should also be noticed that the denominator in (10) represents the squared Frobenius norm of the matrix S_k. Furthermore, it holds that $q(E) \in [0,1]$ with small quality numbers representing low quality elements and $q(E) = 1$ representing ideal regular elements.

Valid initial elements of all types under consideration defined by randomly chosen nodes and their mean ratio quality numbers can be seen in the leftmost column of Fig. 3. In addition, each row contains the elements obtained by successively applying the same element transformation using $\sigma = 1$ and mean edge length preserving scaling to prevent the successive growth of the elements.

Mean ratio quality numbers are successively improved resulting in nearly regular elements after applying the transformation five times, as can be seen by the elements depicted in the rightmost column. In particular, the first transformation step leads to a considerable improvement of element regularity and with this in element quality. In the context of mesh smoothing, such a rapid change of shape is not always favorable since it also increases the risk of invalid neighbor element generation. Therefore, relaxation is applied, that is the linear combination

$$E'_r := (1 - \varrho)E + \varrho E'_s \tag{11}$$

of the initial element E and the transformed and scaled element E'_s is used, where $\varrho \in (0,1]$ represents the relaxation parameter. The smaller ϱ gets, the smaller are the geometric changes compared to the initial element E. For the choice $\varrho = 1$ it holds that $E'_r = E'_s$.

Due to its recursive nature and the involvement of cross products and vector normalization, proving the regularizing effect of the non-linear transformation given by Definition 1 is not straight-

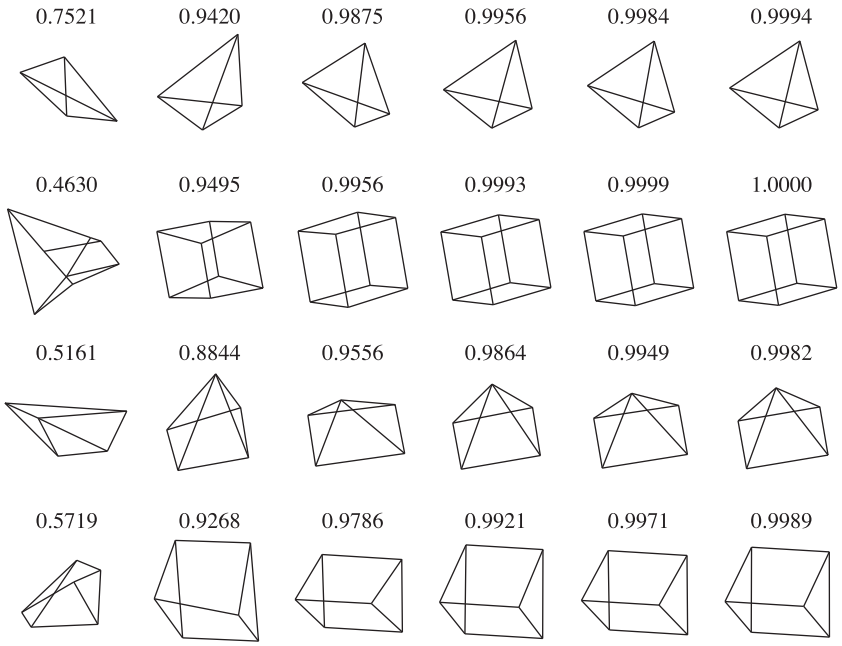

Fig. 3. Elements and their mean ratio quality numbers obtained by successively applying the regularizing transformation using $\sigma = 1$ and mean edge length preserving scaling.

forward. Therefore, it has been substantiated by numerical tests, which also provide more insight into the regularizing behavior of the transformation with respect to the scaling parameter σ.

The tests are based on generating 100,000 random initial elements of each type and taking 101 equidistant values for $\sigma \in [0,2]$. For each pair (E_j, σ_k) taken from the Cartesian product of all elements and all σ-values, the geometric transformation using the scaling factor σ_k has been iteratively applied starting with the initial element E_j until the mean ratio quality number of the resulting element deviated less than 10^{-6} from the ideal value one. After each iteration step the resulting element has been scaled with ζ being set to the inverse of the arithmetic mean of all edge lengths to avoid numerical instabilities caused by the successive element growth, which would have occurred otherwise. Due to the scaling invariance of the transformation according to Lemma 1 this has no influence on the speed of regularization. In addition, relaxation has been disabled by the specific choice $\varrho = 1$. For each element type this test resulted in 10,100,000 transformation cycles, each represented by its number of iterations, thus providing an indication of the speed of regularization.

Results of this test are depicted in Fig. 4 (a). Here, for each element type and specific choice of σ_k the according graph gives the average iteration number of all 100,000 transformation cycles performed. In addition, for selected values of σ the standard deviation is marked by error markers. As can be seen, all graphs show a similar behavior. To be precise, the average iteration number increases if σ tends to zero, since the transformation tends to become a node averaging scheme without regularizing effect. For $\sigma \in (1/2, 2)$, that is after reaching its minimum, the average iteration number depends almost linearly on σ. Thus, the parameter σ provides an easy control for the speed of regularization.

Due to the usage of random initial nodes, the probability of an initial element being invalid increases with its complexity, i.e. with its number of nodes. This is reflected by the valid initial element percentage amounting to 50.15% for tetrahedral, 0.37% for hexahedral, 9.36% for pyramidal, and 5.33% for prismatic elements. That is, in the majority of cases random generated elements are invalid. However, the transformation also reliably regularizes invalid elements, as can be seen by the average iteration numbers given in Fig. 4 (a). Furthermore, as is shown by the standard deviation error markers the dispersion of iteration numbers is moderate.

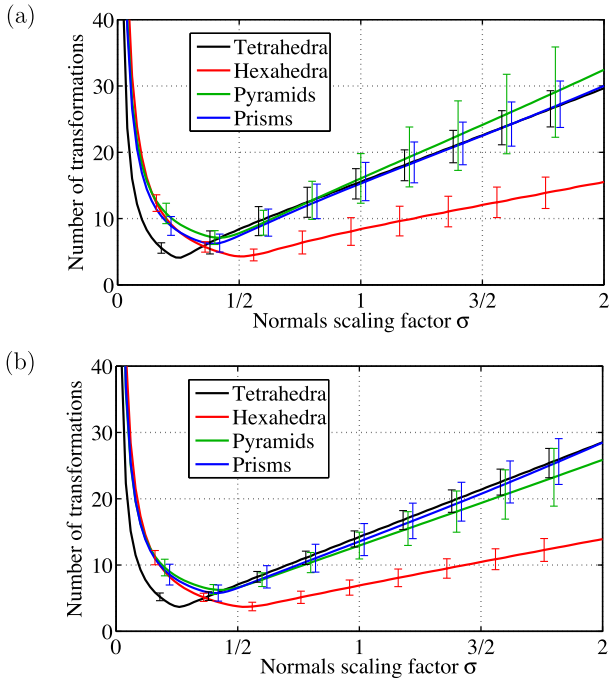

Fig. 4. Regularization speed with respect to scaling parameter σ for arbitrary initial elements (a) and valid initial elements (b). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Regularization results are even more uniform, if only valid elements are used as is common in the case of mesh smoothing. Results based on using 100,000 valid initial elements for each element type are depicted in Fig. 4 (b). Compared to the test case of arbitrary initial elements the average iteration number becomes

smaller while preserving its qualitative behavior with respect to σ. Furthermore, the standard deviation is decreased significantly. That is, for a given element type and normals scaling factor σ_k elements are regularized quite uniformly. This is an important fact with respect to the mesh smoothing approach presented below, assuring a consistent smoothing result.

The transformation of hexahedral elements, not only differs in the centroid preservation property, but also in its increased speed of regularization as can be seen in both cases. Hence, for mixed mesh smoothing it is advisable to use element type dependent values for σ to obtain a uniform regularization for all elements under consideration.

3. The GETMe smoothing scheme

This section describes, how the regularizing element transformation is used in order to smooth mixed volume meshes. Due to its approach, the resulting smoothing scheme is denoted as the geometric element transformation method (GETMe). A sequential GETMe approach based on successively improving elements with the lowest quality has already been introduced for triangular surface meshes in [21]. Alternatively, a simultaneous GETMe approach for smoothing mixed planar meshes was presented in [22]. It is based on transforming all elements simultaneously. Here, updated node positions are obtained as weighted means of the transformed element nodes.

Whereas the simultaneous GETMe approach efficiently improves the overall mesh quality, the sequential GETMe approach is particularly well suited to resolve local mesh quality problems. Consequently, a combined method of applying both steps subsequently, resulting in a smoothing scheme simply denoted as GETMe smoothing, was introduced for all-tetrahedral and all-hexahedral meshes in [23,24]. It requires only minimal modifications in order to smooth mixed meshes, thus confirming the generality of the GETMe approach. Here, the changes only affect the element specific regularizing transformation and its parameters. However, for completeness, a detailed description of the GETMe smoothing algorithm is given in the following.

In preparation, let M denote a conforming mesh with $n_p \in \mathbb{N}$ nodes $p_i \in \mathbb{R}^3$, where $i \in I := \{1,\dots,n_p\}$ representing the index set of all nodes. The latter are connected by $n_e \in \mathbb{N}$ valid tetrahedral, hexahedral, pyramidal, and/or prismatic elements $E_j = (p_{j_1},\dots,p_{j_\ell})$, where $j \in J := \{1,\dots,n_e\}$. Each element E_j is uniquely defined by its node indices $j_1,\dots,j_\ell \in I$ with ℓ denoting the element type specific number of nodes. GETMe smoothing, in its presented form, is based on using the mean ratio quality criterion for transformation and termination control. Due to the requirements of this quality criterion, the initial mesh has to be valid. Otherwise, untangling techniques, similar to those presented in [29,30], have to be applied first.

Overall mesh quality is assessed with the aid of the minimal and mean mesh quality numbers given by

$$q_{\min} := \min_{j\in J} q(E_j) \quad \text{and} \quad q_{\text{mean}} := \frac{1}{n_e}\sum_{j=1}^{n_e} q(E_j). \tag{12}$$

For the sake of simplicity and comparability with other smoothing approaches boundary nodes of the mesh will be kept fixed during the smoothing process.

3.1. Simultaneous GETMe smoothing

Before defining the single steps of the simultaneous GETMe algorithm, a rough sketch of its procedure will be given. As mentioned before, this scheme is based on transforming all mesh elements E_j simultaneously. Here, an individual element quality and element type dependent normals scaling factor σ_j contained in a predefined interval $[\sigma^{\text{type}}_{\min}, \sigma^{\text{type}}_{\max}]$ is used, where type $\in \{\text{tet}, \text{hex}, \text{pyr}, \text{pri}\}$ denotes the element type of E_j.

Let $J(i) \subseteq J$ denote the index set of all mesh elements adjacent to the node p_i. For each node p_i and its adjacent elements E_j, $j \in J(i)$, this results in $|J(i)|$ new temporary nodes $p'_{i,j}$. Out of this, new node positions p'_i are derived as quality weighted means. Due to its geometric approach the validity of the resulting new mesh elements cannot be guaranteed so far. Therefore, a simple repair step for invalid elements is applied subsequently. All steps are repeated until the resulting mesh improvement is below a given tolerance. This results in the following simultaneous GETMe algorithm:

Step 1: *Simultaneous element transformation*
Perform the following steps for all $j \in J$:

Step 1.1: Compute E'_j according to (4) using the normals scaling factor

$$\sigma_j := \sigma^{\text{type}}_{\min} + \left(\sigma^{\text{type}}_{\max} - \sigma^{\text{type}}_{\min}\right)(1 - q(E_j)).$$

Step 1.2: Scale E'_j while restoring its centroid according to (8) and apply relaxation according to (11) using a fixed relaxation value $\varrho \in (0,1]$ resulting in the new temporary nodes $p'_{i,j}$.

Step 2: *Update inner nodes*

Step 2.1: For each inner node p_i of the mesh compute its new position

$$p'_i := \frac{\sum_{j\in J(i)} w_j p'_{i,j}}{\sum_{j\in J(i)} w_j} \quad \text{with } w_j := (1 - q(E_j))^\eta,$$

where $\eta \in \mathbb{R}^+_0$ denotes a fixed user defined exponent.

Step 2.2: Store old node positions $p^{\text{old}}_i := p_i$ and update all inner nodes, i.e. $p_i := p'_i$.

Step 3: *Invalid element handling*

Step 3.1: Determine index set $J_i \subseteq J$ of all invalid elements.

Step 3.2: If $J_i = \emptyset$ go to step 4, otherwise revert nodes of invalid elements to their previous position p^{old}_i and go to step 3.1.

Step 4: *Termination control*

Step 4.1: Compute quality numbers $q_{\min}$ and q_{mean}.

Step 4.2: Terminate if maximal number of iterations reached or quality improvement is below a given tolerance. Otherwise go to step 1.

As has been shown numerically in Section 2.2, the average convergence rate of the transformation not only depends on the normals scaling factor σ_j but also on the element type. Therefore, element type dependent ranges for σ_j are used in step 1.1 to synchronize the speed of regularization. The concrete choice of σ_j is based on the individual element quality $q(E_j)$, which allows low quality elements to be smoothed more carefully than high quality elements. Within the mesh smoothing context the element growth, centroid movement, and rapid change of shape caused by the element transformation are undesirable and hence corrected in step 1.2.

In step 2 new coordinates of inner nodes are computed as quality weighted means of the intermediate nodes obtained by step 1. Again, quality weighted averaging puts an emphasis on low quality elements. This can additionally be enforced by an appropriate choice of the user defined exponent $\eta \geqslant 0$. If all elements adjacent to p_i are regular, the denominator in the representation of p'_i becomes zero. However, this is no problem, since from a local point of view, the original node position is already optimal. Whereas according to step 1.2 element centroid changes are prevented on a single element level, they are enabled on the complete mesh level due to weighted node averaging.

It should also be noted that according to step 2 new node coordinates are not immediately updated, since this would influence the computation of neighboring nodes. Therefore, new node computation and applying the new coordinates are separated, ensuring that the computation does not depend on the numbering scheme of nodes and elements. This is of particular interest for parallelized implementations of the GETMe approach, in order to obtain reproducible results. Furthermore, using appropriate data structures as described in [24], the sum of the weighted new nodes $p'_{i,j}$ as well as the backup nodes p_i^{old} can be stored with only little memory overhead.

Due to the geometry driven approach, the steps accomplished so far may lead to the generation of invalid elements. Such elements are identified and removed in step 3. For reasons of simplification and due to good results in various numerical tests, this is done by iteratively resetting the associated nodes to their previous position p_i^{old}. More elaborate techniques could, for example, be based on a dynamic adjustment of parameters and re-computation of the steps 1 and 2 for the problematic elements and nodes.

A likewise simple approach is used for the termination control of the algorithm accomplished in step 4. It is based on the number of iterations performed so far and the mesh quality numbers of the current and previous iteration. Numerical tests showed that, in particular, the first few steps of simultaneous GETMe smoothing are highly efficient. Hence, in practice, termination criteria should be based on the rate of improvement assessed on the basis of all preceding iterations and the obtained mesh quality.

3.2. Sequential GETMe smoothing

Although the sequential approach is likewise based on improving element quality by applying the regularizing element transformation, the way in which transformed element nodes are handled is much simpler. Here, the elements with the lowest quality are selected iteratively and transformed within the mesh. That is, new node coordinates are set directly. This is repeated until specific termination criteria are met. In order to allow an additional control over element selection, each element quality number $q(E_j)$ is corrected by a quality penalty value $\pi_j \geqslant 0$, which is initialized to zero. This results in the following sequential GETMe algorithm:

Step 1: *Transform element with lowest quality number*

Step 1.1: Determine index j of the element with lowest corrected quality $q(E_j) + \pi_j$.

Step 1.2: Transform E_j according to (4) using a type dependent fixed normals scaling factor σ^{type}.

Step 1.3: Scale E'_j while restoring its centroid according to (8) and apply relaxation according to (11) using a fixed relaxation value $\varrho \in (0,1]$ resulting in the new temporary nodes p'_i.

Step 2: *Set nodes of the transformed element*

Store initial node positions $p_i^{\text{old}} := p_i$ of all nodes of E_j and assign new node positions $p_i := p'_i$.

Step 3: *Invalid element and penalty handling*

Step 3.1: In the case that any of the neighboring elements becomes invalid, revert to the initial node positions.

Step 3.2: Increase quality penalty value π_j of element E_j by fixed value $\Delta\pi_{\text{i}} > 0$ if an invalid element was generated, by $\Delta\pi_{\text{r}} > 0$ if the same element was picked in previous step, and decrease by $\Delta\pi_{\text{s}} > 0$ if transformation was successful.

Step 4: *Termination control*

Terminate if maximal number of iterations is reached or quality improvement is below a given tolerance. Otherwise go to step 1.

In step 1 the element E_j with the lowest quality number has to be determined first, which is an $\mathcal{O}(1)$ operation if a min heap of quality numbers and associated element indices is used. Subsequently, E_j is transformed and adjusted similar to the steps 1.1 and 1.2 of the simultaneous approach. However, since only one element is transformed, there is no need for a quality dependent control of the transformation and averaging parameters. Hence, it suffices to use a fixed set of conservatively chosen σ^{type} values to slightly improve element quality while reducing the risk of invalid element generation.

Since only one element is transformed, its new nodes can immediately be set after the original node coordinates have been stored for invalid element handling purposes. This is done in step 2. Likewise handling of invalid elements becomes simple, since only the element E_j and its direct neighbor elements are affected by the node updates. In the case that one of these elements becomes invalid, the initial node coordinates are restored.

Without further modifications, this could result in an infinite loop of picking, transforming and reverting the same element and its nodes. This is prevented by a simple element quality penalty mechanism based on adjusting the quality penalty value π_j associated to the selected element E_j. In this, π_j is increased by $\Delta\pi_{\text{i}}$ if invalid elements have been generated. This leads to a successively increased corrected quality number $q(E_j) + \pi_j$ of E_j until a different element is picked first in step 1.1. The increase of the penalty term in case of a repeated selection of the same element can be accelerated by the choice of $\Delta\pi_{\text{r}} > 0$. Finally, if an element E_j has been successfully transformed and applied, its quality penalty value π_j is decreased by $\Delta\pi_{\text{s}}$. Using a min heap for determining the lowest quality element also requires the update of changed element qualities, which is an $\mathcal{O}(\log n_e)$ operation for the selected element E_j itself as well as its neighboring elements if node updates have been successful. Hence, the selection of the elements, which have to be transformed, can be handled efficiently.

Since only one element is transformed, it is not sensible to evaluate the comparatively expensive overall mesh quality q_{mean} after each iteration of the sequential approach for termination control. Instead, q_{mean} is only evaluated after a prescribed number of sequential iterations. In between this mean mesh quality evaluation cycles, iteration number-based and $q_{\min}$-based termination criteria can be used in step 4.

3.3. GETMe smoothing

By deriving new node positions as weighted means of transformed element nodes for the complete mesh, the simultaneous GETMe approach focuses on the overall mesh quality. In contrast, transforming only low quality elements in the case of the sequential approach puts an emphasis on improving the lowest element quality. Numerical tests in [21,22] have demonstrated that both methods lead to promising results with respect to $q_{\min}$ and q_{mean}. However, even better results with respect to quality and runtime behavior can be obtained by combining the characteristic strengths of both methods. This is done by applying GETMe simultaneous first, which improves overall mesh quality and with this problematic parts of the mesh containing the element with the lowest quality. Therefore, the subsequently applied GETMe sequential approach can improve the remaining low quality elements more effectively. The resulting combined approach is simply denoted as GETMe smoothing.

For all-tetrahedral and all-hexahedral meshes numerical tests have shown that the basic GETMe smoothing as presented here results in quality numbers at least comparable to those of a state of the art global optimization approach within significantly shorter runtimes [23,24]. The same holds for mixed volume meshes, as will be shown in the following section.

4. Examples

In this section, three meshes of different type will be considered to illustrate the effectiveness of GETMe smoothing of mixed volume meshes. Results will be compared to those obtained by the well known geometry-based smart Laplacian smoothing approach. Here, node updates are only applied if this improves the arithmetic mean of the mean ratio quality numbers of all adjacent elements. Since the definition of the mean ratio quality number requires elements to be valid, this additionally avoids the generation of invalid elements. Mesh smoothing is terminated, if the q_{mean} values of two consecutive meshes deviate less than 10^{-6}.

Due to its simple averaging scheme, results of the geometry-based smart Laplacian smoothing are usually inferior if compared to those using a global optimization-based approach. Therefore, GETMe results are also compared to those obtained by the shape improvement wrapper of the mesh quality improvement toolkit Mesquite [28]. Here, smoothing is based on minimizing the mean of the inverse mean ratio numbers of all mesh elements using a feasible Newton approach. The shape improvement wrapper has been applied repeatedly using its default settings until the mean mesh quality could not be further improved. As in the case of smart Laplacian smoothing, the mesh with the best overall mean quality is used for comparison.

In the case of GETMe smoothing a common parameter set was used for all examples. It has been determined numerically by assessing the results of various meshes and parameter sets. For the simultaneous substep this resulted in $\sigma^{\text{tet}} \in [0.77, 0.84]$, $\sigma^{\text{hex}} \in [2.57, 3.45]$, $\sigma^{\text{pyr}} = 1.86$, $\sigma^{\text{pri}} = 1.59$, mean element edge length preserving scaling, $\varrho = 2/3$, and $\eta = 1/4$. The same termination criterion was applied as in the case of smart Laplacian smoothing. In the case of the sequential GETMe substep $\sigma^{\text{tet}} = 0.81$, $\sigma^{\text{hex}} = 2.74$, $\sigma^{\text{pyr}} = 1.82$, $\sigma^{\text{pri}} = 0.85$, average element edge length preserving scaling, and the more conservative relaxation parameter $\varrho = 0.01$ was used. Penalty values have been set to $\Delta\pi_{\text{i}} = 0.01$, $\Delta\pi_{\text{r}} = 0.0005$, and $\Delta\pi_{\text{s}} = 0.01$.

4.1. Hexahedral in tetrahedral mesh embedding

The first example considers a hexahedral mesh of a plate with two drill holes depicted in Fig. 5 (a) embedded into a tetrahedral mesh of a cuboid. The plate mesh consists of 5748 hexahedral elements and is refined towards the drill hole with the smaller diameter. This mesh has been wrapped by one layer of 2220 pyramidal elements in order to embed it into the tetrahedral mesh consisting of 26,082 elements. A clipped version of the resulting hybrid mesh is depicted in Fig. 5 (b). Here, different element types are marked by different shades of gray and the cuboid is marked by thick black lines.

In total, the mesh consists of 34,050 elements and 11,371 nodes of which 1022 are boundary nodes on the outer cuboid faces kept fixed during smoothing. The boundary nodes of the plate model surface have not been fixed and no constraints have been used in order to allow an unhampered smoothing of adjacent elements of different type. Hence, the shape of the embedded plate is not preserved. In practice, smoothing methods are usually combined with surface and feature line back-projection techniques or constrained node movement techniques in order to preserve external and internal boundaries.

Although the lowest element quality $q_{\text{min}} = 0.1625$ is comparatively low, the mean mesh quality of $q_{\text{mean}} = 0.7302$ indicates that the overall mesh quality is already reasonable. The initial mesh has been smoothed using the methods and parameters as described at the beginning of this section resulting in the runtime information and quality numbers given in Table 1. Cross sections of the initial, as well as the smoothed meshes, are depicted in Fig. 6.

Table 1 also provides the iteration numbers and smoothing time in seconds for all methods. In the case of GETMe smoothing the iteration numbers of both substeps are given. Smart Laplacian smoothing terminated after 21 iterations improving q_{mean} by 12.2% but also decreasing q_{min} significantly by 84.1%. This generation of low quality elements is also obvious from the cross section depicted in Fig. 6 (b) if compared to the initial mesh depicted in (a). Here, all elements are colored according to their mean ratio quality number, where reddish colors indicate low quality elements and bluish colors elements of high quality. Since meshes usually have to meet specific minimal requirements in subsequent applications, such a loss in minimal element quality is a drawback. However, no invalid elements have been generated due to the smart approach as they would have been generated by standard Laplacian smoothing.

The global optimization-based approach performed a total of 52 feasible Newton iterations within eight shape improvement wrapper cycles of Mesquite. Here, both quality numbers have been improved, namely q_{min} by 109.2% and q_{mean} by 14.8%, which can also be seen by the cross section depicted in Fig. 6 (c).

The GETMe simultaneous approach terminated after 26 iterations resulting in $q_{\text{mean}} = 0.8351$, However, the mean mesh quality slightly decreased during the subsequently applied GETMe sequential approach, performing 11,500 iterations. This is caused by directly setting new node positions obtained by transforming the worst elements, which led to a slight decrease in neighboring elements quality.

Since only one element is transformed during one iteration step of GETMe sequential, the total number of element transformations

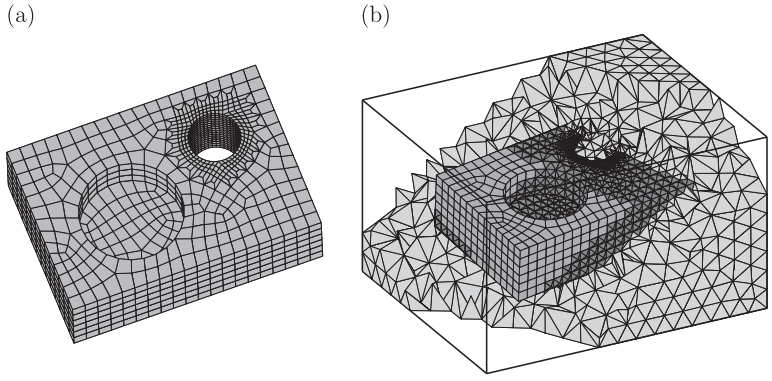

Fig. 5. Hexahedral mesh of a plate (a) and its embedding into a tetrahedral mesh using one layer of pyramidal elements (b). Different element types are marked by different shades of grey.

Table 1
Embedded plate smoothing results.

Method/Criterion	Iter	Time (s)	q_{min}	q_{mean}
Initial	–	–	0.1625	0.7302
Smart Laplace	21	5.17	0.0258	0.8195
Global Optimization	52	9.72	0.3400	0.8382
GETMe	26/11,500	2.70	0.4476	0.8300

corresponds to those performed in less than one GETMe simultaneous step, transforming all 34,050 mesh elements at once. Compared to the initial mesh the combined GETMe approach improved q_{min} by 175.4% and q_{mean} by 13.7%.

As can be seen by the element quality histogram depicted in Fig. 7 and the minimal element quality results given in Table 1, GETMe smoothing significantly reduces the number of low quality elements. To be precise, the number of elements with a mean ratio value below 0.44 amounts to 2150, 1138, 106, and 0 in the case of the initial mesh and those obtained by smart Laplacian smoothing, global optimization-based smoothing, and GETMe smoothing, respectively. The peak in the histogram of GETMe smoothing near q_{min} is caused by the sequential substep, which successively improves low quality elements, leading to an accumulation of elements near this threshold.

GETMe smoothing also has a favorable runtime behavior, as can be seen in Fig. 8 depicting the mean mesh quality q_{mean} with respect to smoothing runtime in seconds. Here, each circular marker indicates the results of one iteration. Results have been obtained by a straightforward C++implementation of GETMe smoothing and smart Laplacian smoothing. They are compared to the results of the feasible Newton-based global optimization approach of Mesquite [28], which is also implemented in C++. Runtimes have been measured on a notebook with an Intel® Core™ i5-540M CPU (dual core, 3 MB cache, 2.53 GHz), 4 GB RAM, 64bit Linux operating system with kernel 2.6.34.4, and the GNU C++ compiler version 4.5.1.

The complete smoothing time amounts to 5.17 s, 9.72 s, and 2.70 s in the case of smart Laplace, global optimization and GETMe smoothing respectively. In the case of GETMe smoothing, the simultaneous substep took 2.18 s and the sequential 0.52 s. In all cases, due to the comparatively tight termination criterion, the majority of iterations only led to little improvements. In practice, smoothing would have been stopped at an earlier stage. For example, the mean mesh quality of 0.83 was reached by global optimization after 12 iterations taking 1.7 s and by GETMe simultaneous after three iterations taking 0.2 s. That is, even a not specifically optimized version of GETMe smoothing can achieve quality meshes comparable to those obtained by the global optimization approach within significantly shorter runtimes. In contrast, smart Laplacian smoothing was not able to reach this level of mesh quality.

4.2. Tetrahedral in hexahedral mesh embedding

The second example considers the embedding of a ring into a hexahedral mesh. The ring surface mesh depicted in Fig. 9 (a) is provided by the Gamma project mesh database [31] and consists

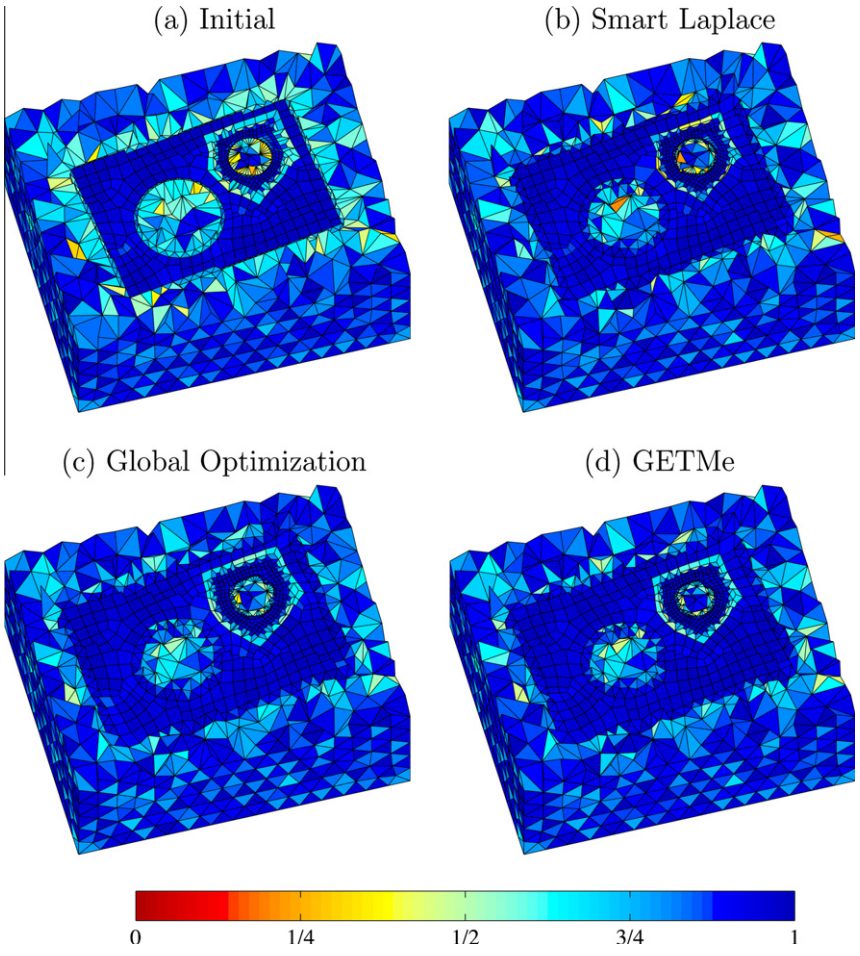

Fig. 6. Initial and smoothed plate embedding meshes with elements colored according to their mean ratio quality number.

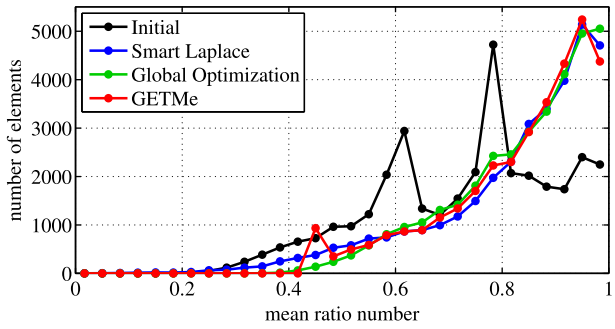

Fig. 7. Embedded plate element quality histogram. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of 5226 triangular elements. It has been embedded into a hexahedral mesh of $24 \times 25 \times 16$ regular elements. Three layers of hexahedral elements have been removed around the ring model and the resulting space was filled with tetrahedral elements. One layer of pyramidal elements connects the tetrahedral mesh with the hexahedral mesh. A clipped version of the resulting mesh is depicted in Fig. 9 (b). Again, different element types are marked by different shades of gray and the outline of the regular hexahedral mesh is marked by thick black lines.

The mesh consists of 74,106 tetrahedra, 7486 hexahedra, and 1618 pyramids resulting in a total of 83,210 volume elements. All nodes of the outer cuboid surface as well as all nodes of the ring surface have been kept fixed resulting in 14,246 movable nodes out of total of 25,531 nodes. Since the resulting initial mesh again was of already good quality, it has been additionally distorted by element validity preserving random node movements to complicate smoothing. The resulting initial mesh quality, runtime information, as well as the results obtained by the smoothing approaches described at the beginning of this section, are given in Table 2.

As in the case of the first example, smart Laplacian smoothing was not able to improve the minimal element quality. However, the mean mesh quality was increased by 45.4%. In contrast, global optimization not only improved q_{mean} by 67.5% but also increased the minimal element quality significantly, although the method is mainly geared towards improving q_{mean}. Finally, GETMe smoothing even further increased q_{min} by 17.1% if compared to the value obtained by global optimization, and the mean mesh quality by 65.3% if compared to the initial mesh.

The achieved mesh quality is also reflected by the cross sections of the initial and smoothed versions of the mesh depicted in Fig. 10. In the case of smart Laplacian smoothing several clusters of low quality elements remain, since node updates are only performed if quality is improved. Standard Laplacian smoothing would have been able to resolve some of these clusters, but on the other hand would have led to a lower mean mesh quality

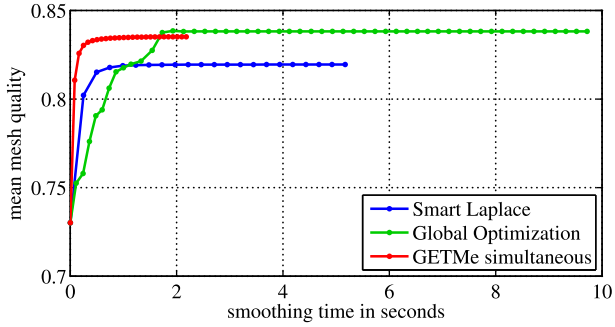

Fig. 8. Embedded plate mean mesh quality with respect to smoothing runtime.

and the generation of invalid elements. In contrast, global optimization and GETMe smoothing yielded valid meshes of high quality.

Compared to the first example, the number of low quality elements is significantly higher in the case of smart Laplacian smoothing, as can be seen in the element quality histogram depicted in Fig. 11. For example, the number of elements with a mean ratio quality number below 0.23 amounts to 12,634, 3269, 3, and 0 in the case of the initial mesh and those obtained by smart Laplacian smoothing, global optimization and GETMe smoothing, respectively. Although GETMe smoothing is a geometry-based method not using optimization techniques, its histogram resembles that of the global optimization approach.

Finally, Fig. 12 depicts the mean mesh quality with respect to smoothing runtime. For the example under consideration, smart Laplacian smoothing performed 24 iterations within 9.8 s. Four cycles of the shape improvement wrapper performing 72 feasible Newton iterations in the case of the global optimization approach took 14.0 s in total. The simultaneous GETMe substep accomplished 30 iterations within 7.6 s, followed by the sequential substep performing 6700 iterations within 0.2 s, resulting in a total GETMe smoothing time of 7.8 s. Again, a slightly lower mean mesh quality could be obtained much faster in the case of GETMe smoothing. For example, the threshold $q_{\text{mean}} = 0.73$ was obtained after 8 iterations of the simultaneous substep taking 2.0 s in total. In contrast, global optimization required 44 iterations taking 8.6 s in total to achieve this quality threshold.

In addition, the simultaneous GETMe approach is well suited for parallelization, since steps like the single element transformation and new node computation can be accomplished in parallel. For example, using the OpenMP [32] directive "`#pragma omp parallel for`" in order to parallelize these two steps resulted in an already good speedup factor of 1.7 for the named test system with a theoretical speedup limit of 2.0.

4.3. Tetrahedral mesh wrapped with prismatic boundary layer mesh

Tetrahedral meshes wrapped with layers of prisms are used for example in computational biofluid dynamic applications [9]. As an example, a part of an aorta will be considered in this section. The model, as well as its initial tetrahedral mesh, is provided courtesy of MB-AWG by the AIM@SHAPE Shape Repository [33]. It consists of 35,551 tetrahedral elements and has been wrapped with six layers of prisms with a constant height of 0.012. Here, the height was chosen small to avoid the intersection of layer elements as well as not to change the model surface too much. The wrapped model is depicted in part (b) of Fig. 13. In this, part of the prismatic boundary layer has been removed for illustration purposes. The part of the aorta represented by the model is marked by a blue rectangle in the scheme (a) depicted on the left. The scheme itself was created by J. Heuser and is provided by the Wikimedia Commons media archive [34].

Due to the minimal and average element quality numbers given by 0.2677 and 0.8098 respectively, the initial tetrahedral mesh is of already good quality. However, adding six layers of thin prisms reduces these numbers significantly as can be seen by the q_{min} and q_{mean} values of the initial mesh given in Table 3. The number of prisms amounts to 80,724 resulting in an initial mesh with 49,903 nodes and 116,275 elements in total.

The low quality of the prismatic boundary layers can also bee seen in the cross section of the initial mesh depicted in Fig. 14 (a). Although smart Laplacian smoothing is able to widen up the prismatic layers resulting in a mean mesh quality improvement of 86.4%, the minimal element quality halves if compared to the initial mesh. In particular, results obtained by this approach are not satisfactory if compared to the other two smoothing methods resulting in significantly better quality numbers. This is because

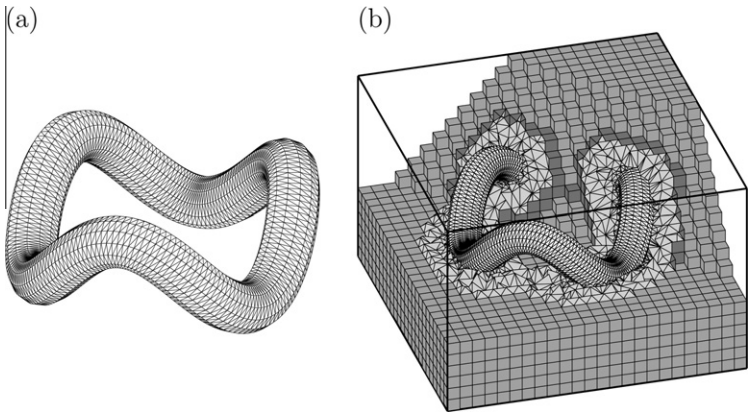

Fig. 9. Triangular mesh of a ring (a) and its embedding into a hexahedral mesh using a tetrahedral wrapper mesh and one layer of pyramidal elements (b). Different element types are marked by different shades of grey.

Table 2
Embedded ring smoothing results.

Method/Criterion	Iter	Time (s)	$q_{\min}$	q_{mean}
Initial	–	–	0.0005	0.4499
Smart Laplace	24	9.80	0.0005	0.6542
Global Optimization	72	14.00	0.1984	0.7534
GETMe	30/6700	7.78	0.2324	0.7437

smart Laplacian smoothing is based on a simple node averaging scheme being not quality driven. Here, quality only comes into account in the decision if node movements are applied depending on the mean quality improvement of adjacent elements. Due to the uniform prism heights, widening the prismatic boundary layers is mainly induced by the movement of the interface nodes shared by both element types. In addition, overall mesh improvement is hampered by the interior quality tetrahedral mesh, as can also be seen in Fig. 14. This does not hold for standard Laplacian smoothing leading to a significantly better mean mesh quality of 0.7542 but also to the generation of 372 invalid elements, thus resulting in an unusable mesh for most applications.

In contrast, since GETMe smoothing is based on regularizing element transformations and quality weighted node averaging, the prismatic layers are widened up by the transformation induced node movements. Hence, applying GETMe simultaneous resulted

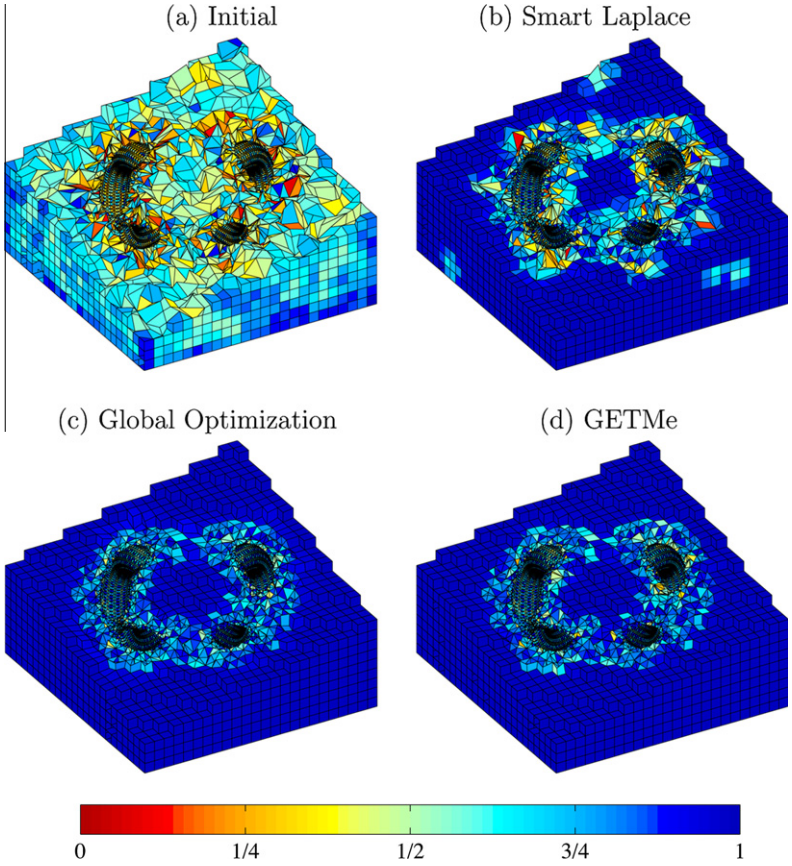

Fig. 10. Initial and smoothed ring embedding meshes with elements colored according to their mean ratio quality number.

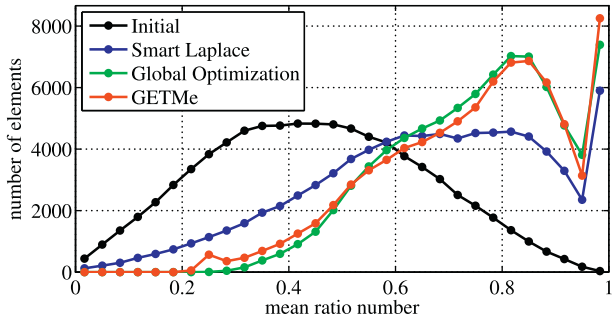

Fig. 11. Embedded ring element quality histogram.

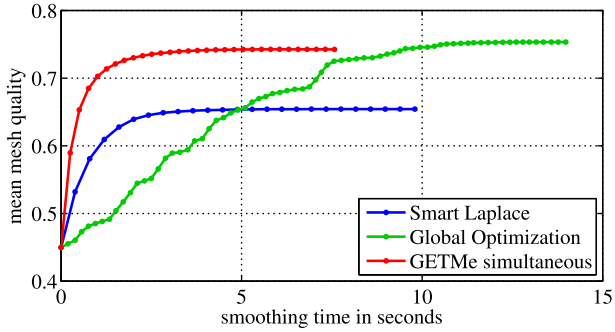

Fig. 12. Embedded ring mean mesh quality with respect to smoothing runtime.

in a significantly better mean mesh quality of $q_{\text{mean}} = 0.8253$ representing an improvement by 168.2% if compared to the initial mesh quality. The subsequently applied sequential substep of GETMe smoothing slightly reduced this mean mesh quality number to $q_{\text{mean}} = 0.8231$ while improving the minimal element quality by factor 13.4, if compared to the initial mesh.

Due to its global optimization-based approach, the shape improvement wrapper of Mesquite also widened the prismatic boundary layer significantly, resulting in a minimal quality improvement by factor 9.4 and a mean mesh quality improvement by 172.7% if compared to the initial mesh.

The big difference in mesh quality between smart Laplacian smoothing on the one hand, and global optimization-based and GETMe smoothing on the other hand is also obvious from the quality histogram depicted in Fig. 15. Here, the mean initial prisms quality of 0.0866 is reflected by a peak of the initial mesh histogram. For sake of clarity, this peak representing 49,604 elements

Table 3
Aorta smoothing results.

Method/Criterion	Iter	Time (s)	q_{min}	q_{mean}
Initial	–	–	0.0413	0.3077
Smart Laplace	98	120.84	0.0211	0.5737
Global Optimization	140	105.04	0.3895	0.8392
GETMe	251/66,300	95.65	0.5522	0.8231

in the quality bin [1/15, 1/10) is not visible. The mean quality of all prisms is improved by smart Laplacian smoothing to 0.4796. However, the prisms mean quality numbers 0.8440 and 0.8586 achieved by GETMe smoothing and global optimization are nearly twice as large. Furthermore, there are a large number of low quality elements in the case of smart Laplacian smoothing. For example, the number of elements with a mean ratio number below 0.5522 amounts to 81,381, 58,385, 587 and 0 in the case of the initial mesh, and those obtained by smart Laplacian smoothing, global optimization, and GETMe smoothing respectively.

For the given example, smart Laplacian smoothing terminated after 98 iterations taking 121 s in total. Due to the local oriented approach of GETMe smoothing and the thin initial prismatic boundary-layers, GETMe simultaneous performed a comparatively large number of 251 iterations taking 93 s in total. That is, one iteration of GETMe simultaneous smoothing is about 3.3 times faster if compared to one step of smart Laplacian smoothing. This is caused by the lower number of element quality evaluations, which, in the case of the mean ratio quality criterion, are comparatively expensive. The subsequently applied sequential GETMe approach performed 66,300 iterations within 3 s resulting in a total GETMe smoothing time of about 96 s. Mesquite performed 140 feasible Newton iterations within three cycles of the shape improvement wrapper, taking 105 s in total. Since one iteration of GETMe simultaneous smoothing is about two times faster compared to one feasible Newton iteration, the overall smoothing time of GETMe smoothing is shorter, although a larger number of iterations was performed.

As in the previous examples, the first iterations of the simultaneous GETMe substep led to a sharp rise in mesh quality as is visible in Fig. 16. Hence, practically good meshes can be obtained in significantly shorter runtimes. For example, GETMe simultaneous achieved $q_{\text{mean}} = 0.6$ within 14 iterations taking 6 s, hence being about 15.7 times faster than the global optimization approach, which required 128 iterations taking 94 s. This is also caused by a slow mean quality improvement of the first shape improvement wrapper cycle of Mesquite, which achieved $q_{\text{mean}} = 0.5956$ after 127 iterations. The second cycle led to a more rapid improvement, resulting in $q_{\text{mean}} = 0.8392$ after ten additional

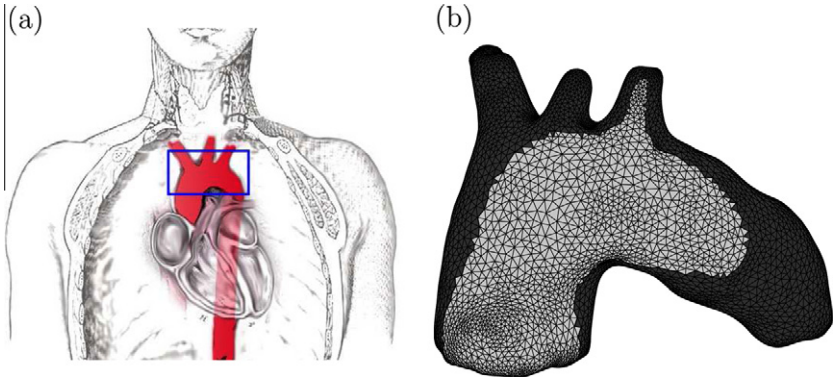

Fig. 13. Schematic representation of an aorta (a), and its tetrahedral mesh with six prismatic boundary layers (b). Different element types are marked by different shades of grey.

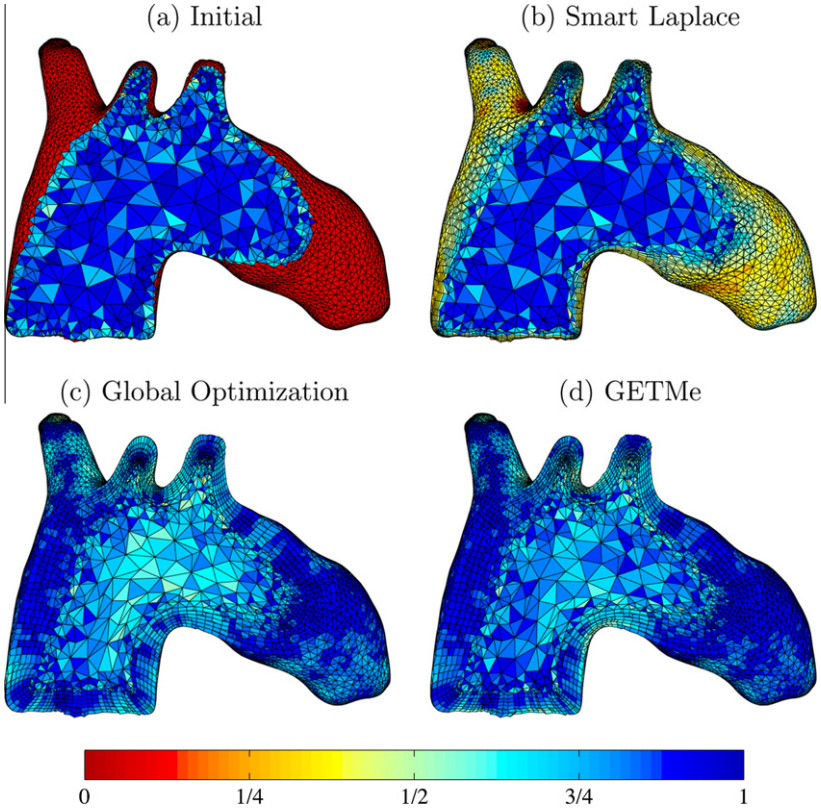

Fig. 14. Initial and smoothed aorta meshes with elements colored according to their mean ratio quality number.

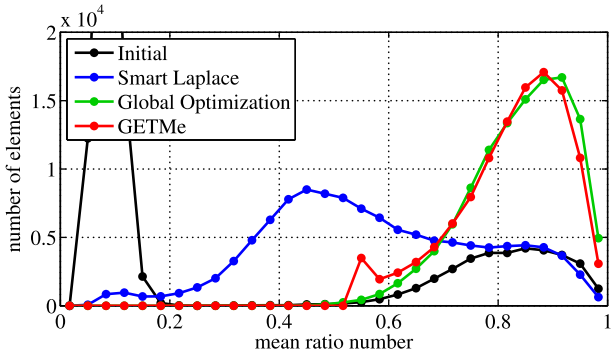

Fig. 15. Aorta element quality histogram. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

iterations. However, even the quality threshold $q_{\text{mean}} = 0.8$ was achieved about 2.4 times faster by GETMe smoothing.

4.4. Mean ratio and warpage-based GETMe smoothing

In the previous sections, mesh quality was assessed using the mean ratio quality criterion, which was introduced in the context of optimization-based smoothing methods [14,16]. By measuring the distance of an arbitrary valid element from a prescribed reference element, the mean ratio criterion not only satisfies basic requirements, such as being well defined for all element types under consideration and the invariance under affine transformations, but it also provides flexibility by the choice of the reference shape. Furthermore, since it is normalized, element quality can be easily incorporated as a control parameter. Due to these advantages, and in order to allow an equitable comparison with the results of the mean ratio-based global optimization approach of Mesquite [28], GETMe smoothing, in its presented form, has adopted the mean ratio quality criterion for smoothing control.

However, depending on the application, additional requirements might be imposed, which can be measured by more specialized quality metrics. One example is the warpage of quadrilateral faces, for which an exemplary modification of the sequential GETMe approach will be considered in the following. The resulting algorithm focuses on improving the lowest element quality with respect to the mean ratio and the warpage criterion. This shows that GETMe smoothing is not only flexible with respect to mesh element types, but also with respect to quality metrics and smoothing objectives.

Let $Q := (p_0, \ldots, p_3)$ denote a quadrilateral face of a volumetric element. The normal of the triangle defined by a node p_k and its connected neighbors is given by $n_k := (p_k - p_{(k+3)\text{mod}4}) \times (p_{(k+1)\text{mod}4} - p_k)$, $k \in \{0, \ldots, 3\}$. According to [35], for $\hat{n}_k := \frac{1}{\|n_k\|} n_k$ the warpage $w(Q) \in [0,2]$ of Q is defined as

$$w(Q) := 1 - \min\{(\hat{n}_0 \cdot \hat{n}_2)^3, (\hat{n}_1 \cdot \hat{n}_3)^3\}.$$

Here, $w(Q) = 0$ indicates non-warped quadrilaterals consisting of coplanar face nodes and large values indicate strongly warped low quality quadrilaterals.

Let $w_{\max}$ denote the maximal warpage of all quadrilateral faces in a given mesh and w_{mean} the arithmetic mean of the warpage values of all unique quadrilateral faces. Since iteratively applying the geometric transformation according to Definition 1 leads to regular elements, the warpage of quadrilateral element faces tends to zero. Hence, GETMe smoothing is also suitable for mesh smoothing with respect to the warpage criterion.

This is reflected by the results given in Table 4. In this, the first four rows of each example contain the mean ratio and warpage values for the meshes obtained in the previous examples. As can

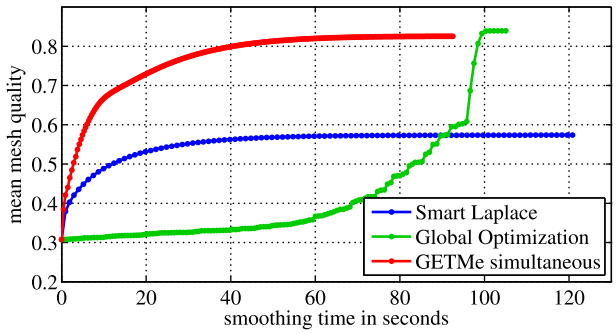

Fig. 16. Aorta mean mesh quality with respect to smoothing runtime.

be seen, global optimization as well as the mean ratio-based GETMe smoothing led to comparable mean warpage values. Using the example of the embedded ring model, Fig. 17 depicts a considerable reduction of w_{mean} within the first few iterations of GETMe simultaneous. As a result, GETMe smoothing reached the threshold $w_{\text{mean}} = 0.01$ more than eight times faster than the global optimization-based approach, whereas smart Laplacian smoothing did not reach this threshold. In addition, smart Laplace generated strongly warped faces in the case of the ring and aorta model. However, compared to the results of the global optimization-based approach, GETMe smoothing led to increased $w_{\max}$ values for the embedded plate and ring model.

Such low quality elements can effectively be improved by a mean ratio and warpage-based GETMe sequential substep. In doing so, only the element selection scheme of Step 1.1 of the sequential algorithm described in Section 3.2 has to be adjusted as is described in the following.

For a given element $E \in \{E^{\text{hex}}, E^{\text{pyr}}, E^{\text{pri}}\}$ let I_Q denote the index set of all its quadrilateral faces Q_i. The warpage of an element E is defined as

$$w(E) := \begin{cases} \max\limits_{i \in I_Q} w(Q_i) & \text{if } E \in \{E^{\text{hex}}, E^{\text{pyr}}, E^{\text{pri}}\}, \\ 0 & \text{otherwise.} \end{cases}$$

This element quality number and the mean ratio criterion are not compatible, since large values of $w(E) \in [0,2]$ indicate low quality elements, whereas large values of $q(E) \in [0,1]$ indicate high quality elements. Therefore, $w(E)$ is transformed by $(1 - w(E)/2)$ and element quality in the modified GETMe sequential substep is assessed by the combined criterion

$$q_\gamma(E) := \min(q(E), \gamma(1 - w(E)/2)).$$

Here, the fixed weight $\gamma > 0$ allows either to balance the two quality criteria or to put an emphasis on one of them. For example, large values of γ focus on mean ratio quality improvements, whereas small values of γ focus on warpage quality improvements. In all cases, low values of q_γ indicate low quality elements. Consequently, the element with the lowest penalty corrected $q_\gamma(E)$ value in the mesh is transformed in order to improve its quality. If the lowest combined quality value is defined by a warped quadrilateral face connecting two elements, the one with the lower mean ratio quality number is selected for transformation.

Results obtained for such a q_γ-based GETMe sequential approach applied to the GETMe smoothed meshes of the previous examples are given in the rows of Table 4 denoted by GETMe q_γ. As can be seen, this method led to improved results with respect to $q_{\min}$ as well as $w_{\max}$ in 4 of 6 cases if compared to global optimization. In addition, results with respect to the mean quality numbers q_{mean} and w_{mean} are comparable. Here, for all three examples, the transformation and penalty parameters of the q_γ-based GETMe sequential smoothing substep have been set to those of the mean ratio-based sequential substep given in the beginning of Section 4. In addition, the quality weight $\gamma = 0.25$ has been used.

Due to the incorporation of two quality criteria, the q_γ-based sequential substep has an increased computational complexity. Furthermore, mesh quality was assessed after each iteration, since quality changes more rapidly if compared to the mean ratio-based GETMe sequential approach. This resulted in the smoothing runtimes 0.44 s, 2.10 s, and 0.01 s for the plate, ring and aorta example requiring 2555, 7068, and 218 iterations, respectively. Nevertheless, the overall runtimes of the GETMe q_γ approach amounting to 3.14 s, 9.88 s, and 95.66 s are significantly lower than those of the global optimization-based approach requiring 9.72 s, 14.00 s, and 105.04 s, respectively.

Differences between the two quality criteria are evident for the aorta example. Here, the initial mesh has been constructed by wrapping a tetrahedral mesh with six prismatic layers of constant height. Due to this, the nodes of the quadrilateral faces are almost coplanar, which results in an excellent initial mean warpage of $w_{\text{mean}} = 0.0092$. In contrast, due to the low heights, these prisms are strongly non-regular yielding a very low average mean ratio value of $q_{\text{mean}} = 0.3077$. This is is also visible by the initial mesh cross section depicted in Fig. 14 (a). Applying global optimization and GETMe smoothing widened up these layers of prisms resulting in a considerable improvement of its mean ratio quality. At the same time, the warpage values increased, since the prismatic layer heights and with this the quadrilateral element faces became less

Table 4
Mesh quality with respect to the mean ratio (q) and warpage (w) quality criterion for previous examples and results after applying the combined GETMe approach (denoted as GETMe q_γ). Larger mean ratio values indicate elements of better quality, whereas lower warpage values indicate quadrilateral faces of better quality.

Mesh	Method	$q_{\min}$	q_{mean}	$w_{\max}$	w_{mean}
Plate	Initial	0.1625	0.7302	0.7368	0.0100
	Smart Laplace	0.0258	0.8195	0.7752	0.0139
	Global Optimization	0.3400	0.8382	0.6529	0.0154
	GETMe	0.4476	0.8300	0.9192	0.0194
	GETMe q_γ	0.2346	0.8298	0.2972	0.0179
Ring	Initial	0.0005	0.4499	1.8157	0.4498
	Smart Laplace	0.0005	0.6542	1.5461	0.0566
	Global Optimization	0.1984	0.7534	0.8013	0.0095
	GETMe	0.2324	0.7437	1.0856	0.0071
	GETMe q_γ	0.2324	0.7435	0.1173	0.0047
Aorta	Initial	0.0413	0.3077	0.6028	0.0092
	Smart Laplace	0.0211	0.5737	1.9245	0.1203
	Global Optimization	0.3895	0.8392	0.9915	0.0247
	GETMe	0.5522	0.8231	1.0117	0.0355
	GETMe q_γ	0.4359	0.8231	1.0000	0.0355

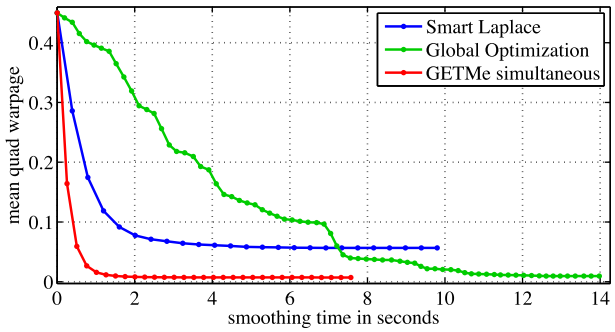

Fig. 17. Mean quadrilateral warpage w_{mean} with respect to smoothing time for embedded ring example. Lower values indicate better quality quadrilateral faces.

uniform. This shows that both quality criteria can result in opposing smoothing objectives. Nevertheless, in the case of q_γ-based GETMe smoothing the quality weight γ allows a good compromise to be found.

For the given examples, numerical tests have shown that applying the sequential q_γ-based GETMe approach to the results of GETMe smoothing leads to better results than applying it to the results of the mean ratio-based simultaneous substep. Since the same transformation and penalty parameters have been applied during the two sequential substeps, this can be interpreted as one sequential substep incorporating a change of the quality criterion.

Further improvements are expected by an adaptive choice of transformation parameters. Here, numerical tests based on parameter variations have shown that similar GETMe q_γ results as given in Table 4 can be achieved within significantly shorter runtimes. Furthermore, $w_{\max}$ values have been further reduced by 12.1%, 63.3%, and 2.7% for the plate, ring and aorta model by using individual smoothing parameters. It is noted however, that these results should not be compared with those of smart Laplacian smoothing and the global optimization-based approach, since the latter are tuned to improve mesh quality with respect to the mean ratio criterion only.

This first test demonstrates the flexibility of the GETMe sequential smoothing approach with respect to the quality criterion used for smoothing control. A similar potential for improvement is expected for an accordingly adjusted GETMe simultaneous approach. However, since in this case element quality is also incorporated in the computation of element transformation parameters and the weighted node averaging scheme, adjustments are more closely bound to the characteristics of the quality criterion. Further research regarding these topics has to be accomplished within the context of specific applications and their individual requirements for mesh quality.

5. Conclusion

The geometric element transformation method for smoothing mixed volume meshes consisting of tetrahedral, hexahedral, pyramidal, and/or prismatic elements, has been presented. It is based on successively transforming single mesh elements leading to more regular, hence better quality elements. Such a regularizing transformation scheme based on the dual element has been presented for all element types under consideration. Here, the construction of the transformation had to be slightly adjusted in the case of non-Platonic element types due to their specific geometry. Nevertheless, as numerical tests have shown, the transformation reliably and efficiently regularizes even invalid elements of all types within a low number of steps. Additional mechanisms, like a normals scaling factor and relaxation, allow the speed of regularization to be controlled, which is reasonable with respect to mesh smoothing.

Basic properties of the transformation regarding the preservation of the centroid as well as the invariance with respect to scaling, rotation, and translation have been discussed. However, an analytic proof of the regularizing effect remains open. Although the transformation scheme is geometrically simple, involving cross products and normalization in a recursive approach leads to complex nonlinear expressions. As has been shown by the authors for linear polygon transformations, such a proof can lead to a deep insight into the regularizing mechanisms and result in additional fields of application. Therefore, further analysis of the given and similar transformations for polyhedral elements is a worthwhile goal.

Two different approaches using the presented transformation, in order to smooth mixed volume meshes, have been presented. The first is the simultaneous GETMe approach combining two driving forces. One is the regularizing element transformation, the other is a Laplacian smoothing like node averaging scheme. However, unlike Laplacian smoothing, which is based on computing arithmetic means of neighboring nodes, simultaneous GETMe smoothing is based on weighted means of temporary nodes obtained by transforming adjacent elements. Hence, quality improvement is mainly caused by the element quality controlled regularizing transformation and the quality weighted node averaging scheme. The second approach, named sequential GETMe, is based on successively improving the lowest quality element of the mesh by applying the transformation directly. Whereas the strong point of the simultaneous approach is to improve the overall mesh quality, the sequential approach efficiently improves the minimal element quality. Therefore, GETMe smoothing consists of applying both methods consecutively, leading to high quality results with respect to both quality numbers.

This has been demonstrated by numerical examples of smoothing mixed volume meshes comprising elements of different type. Using results of smart Laplacian smoothing and a state of the art global optimization-based smoothing approach, it has been shown that GETMe smoothing leads to convincing results with respect to both quality and runtime. That is to say, it results in mesh qualities comparable to those obtained by the global optimization approach, while being significantly faster than the other two methods. Here, a common GETMe parameter set has been used for all examples, which has been derived numerically. However, similar to the case of single element transformations, a mathematical analysis of the smoothing approach regarding quality improvement and convergence behavior might also lead to a deeper insight into the effect of the given parameters and their optimal choice.

This paper concludes a series of papers presenting regularizing element transformations for polygonal surface elements and polyhedral elements, as well as the GETMe approach being based on such transformations. Here, the basic principles of GETMe smoothing have been proven to be generally applicable to all kinds of meshes. Various numerical tests have shown that even using simple transformation schemes and control mechanisms lead to convincing results. Nevertheless, GETMe smoothing offers great potential for further improvements. For example, in the case of the simultaneous approach one can think of applying the algorithm to arbitrary submeshes with dynamically adjusted parameters. In addition, whether other types of element transformations or node averaging schemes also lead to better results should be analyzed. The same holds for the sequential approach, where more elaborate control mechanisms for element selection and an adaptive control of the transformation parameters could be used. Beyond that, in order to reach its full potential, GETMe smoothing should be combined with shape preservation techniques, boundary smoothing,

and topology modification techniques to result in a powerful method for the improvement of arbitrary meshes.

In addition, specific applications might impose additional requirements like, for example, constraints on the height of boundary layers in computational fluid dynamics applications. Such modifications can be based on adjusted element transformations resulting in non-regular polyhedra or an adaptive scaling of transformed elements with respect to a prescribed direction. Similar modifications can be applied in terms of alternative control mechanisms and quality criteria. Here, a simple test of smoothing with respect to the warpage quality criterion led to already promising results and demonstrated the flexibility of the GETMe approach.

References

[1] P.J. Frey, P.-L. George, Mesh Generation, second ed., Wiley-ISTE, 2008.
[2] V.D. Liseikin, Grid Generation Methods, second ed., Springer, 2010.
[3] S.J. Owen, A survey of unstructured mesh generation technology, in: Proceedings of the 7th International Meshing Roundtable, 1998, pp. 239–267.
[4] A.O. Cifuentes, A. Kalbag, A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis, Finite Elem. Anal. Des. 12 (3–4) (1992) 313–318.
[5] S.E. Benzley, E. Perry, K. Merkley, B. Clark, G. Sjaardama, A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis, in: Proceedings of the 4th International Meshing Roundtable, 1995, pp. 179–191.
[6] D. Wake, K. Lilja, V. Moroz, A hybrid mesh generation method for two and three dimensional simulation of semiconductor processes and devices, in: Proceedings of the 7th International Meshing Roundtable, 1998, pp. 159–166.
[7] N. Flandrin, H. Borouchaki, C. Bennis, 3D hybrid mesh generation for reservoir simulation, Int. J. Numer. Methods Engrg. 65 (10) (2006) 1639–1672.
[8] Y. Ito, K. Nakahashi, Improvements in the reliability and quality of unstructured hybrid mesh generation, Int. J. Numer. Methods Fluids 45 (1) (2004) 79–108.
[9] V. Dyedov, D.R. Einstein, X. Jiao, A.P. Kuprat, J.P. Carson, F. del Pin, Variational generation of prismatic boundary-layer meshes for biomedical computing, Int. J. Numer. Methods Engrg. 79 (8) (2009) 907–945.
[10] Y. Zhang, T.J.R. Hughes, C.L. Bajaj, An automatic 3D mesh generation method for domains with multiple materials, Comput. Methods Appl. Mech. Engrg. 199 (5–8) (2010) 405–415.
[11] D.A. Field, Laplacian smoothing and Delaunay triangulations, Commun. Appl. Numer. Methods 4 (6) (1988) 709–712.
[12] L.A. Freitag, On combining Laplacian and optimization-based mesh smoothing techniques, in: Trends in Unstructured Mesh Generation, 1997, pp. 37–43.
[13] S.A. Canann, J.R. Tristano, M.L. Staten, An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes, in: Proceedings of the 7th International Meshing Roundtable, 1998, pp. 479–494.
[14] L.A. Freitag Diachin, P.M. Knupp, T. Munson, S.M. Shontz, A comparison of two optimization methods for mesh quality improvement, Engrg. Comput. 22 (2) (2006) 61–74.
[15] M. Brewer, L.A. Freitag Diachin, P.M. Knupp, T. Leurent, D. Melander, The Mesquite mesh quality improvement toolkit, in: Proceedings of the 12th International Meshing Roundtable, 2003, pp. 239–250.
[16] P.M. Knupp, Algebraic mesh quality metrics, SIAM J. Sci. Comput. 23 (1) (2001) 193–218.
[17] P.M. Knupp, Remarks on mesh quality, in: Proceedings of the 45th AIAA Aerospace Sciences Meeting and Exhibit, 2007.
[18] Y. Sirois, J. Dompierre, M.-G. Vallet, F. Guibault, Hybrid mesh smoothing based on Riemannian metric non-conformity minimization, Finite Elem. Anal. Des. 46 (1–2) (2010) 47–60.
[19] P.M. Knupp, Introducing the target-matrix paradigm for mesh optimization via node-movement, in: Proceedings of the 19th International Meshing Roundtable, Springer, 2010, pp. 67–83.
[20] G. Hansen, A. Zardecki, D. Greening, R. Bos, A finite element method for three-dimensional unstructured grid smoothing, J. Comput. Phys. 202 (1) (2005) 281–297.
[21] D. Vartziotis, T. Athanasiadis, I. Goudas, J. Wipper, Mesh smoothing using the geometric element transformation method, Comput. Methods Appl. Mech. Engrg. 197 (45–48) (2008) 3760–3767.
[22] D. Vartziotis, J. Wipper, The geometric element transformation method for mixed mesh smoothing, Engrg. Comput. 25 (3) (2009) 287–301.
[23] D. Vartziotis, J. Wipper, B. Schwald, The geometric element transformation method for tetrahedral mesh smoothing, Comput. Methods Appl. Mech. Engrg. 199 (1–4) (2009) 169–182.
[24] D. Vartziotis, J. Wipper, A dual element based geometric element transformation method for all-hexahedral mesh smoothing, Comput. Methods Appl. Mech. Engrg. 200 (9–12) (2011) 1186–1203.
[25] D. Vartziotis, J. Wipper, Classification of symmetry generating polygon-transformations and geometric prime algorithms, Math. Pannon. 20 (2) (2009) 167–187.
[26] D. Vartziotis, J. Wipper, Characteristic parameter sets and limits of circulant Hermitian polygon transformations, Linear Algebra Appl. 433 (5) (2010) 945–955.
[27] D. Vartziotis, S. Huggenberger, Iterative geometric triangle transformations, Elemente der Mathematik, in press.
[28] Mesh Quality Improvement Toolkit (Mesquite), Version 2.1.2, <http://www.cs.sandia.gov/optimization/knupp/Mesquite.html>, Accessed September 29, 2010.
[29] P.M. Knupp, Hexahedral Mesh Untangling & Algebraic Mesh Quality Metrics, in: Proceedings of the 9th International Meshing Roundtable, 2000, pp. 173–183.
[30] X.-Y. Li, L.A. Freitag, Optimization-based Quadrilateral and Hexahedral Mesh Untangling and Smoothing Techniques, Technical report, Argonne National Laboratory, 1999.
[31] Gamma Project Mesh Database, <http://www-roc.inria.fr/gamma/download/>, Accessed September 9, 2010.
[32] The OpenMP API Specification for Parallel Programming, <http://www.openmp.org>, Accessed August 27, 2010.
[33] Aim@Shape Mesh Repository, <http://shapes.aim-at-shape.net/>, Accessed September 21, 2010.
[34] J. Heuser, Aorta scheme, <http://de.wikipedia.org/w/index.php?title=Datei:Aorta_scheme.jpg&filetimestamp=20060328102541>, Accessed November 10, 2010.
[35] C.J. Stimpson, C.D. Ernst, P. Knupp, P.P. Pébay, D. Thompson, The Verdict Geometric Quality Library, Technical Report, SAND2007-1751, Sandia National Laboratories, 2007.