Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

OLIVER SANDER, ROLF KRAUSE

Automatic Construction of Boundary Parametrizations for Geometric Multigrid Solvers

ZIB-Report 03-02 (February 2003)

Automatic Construction of Boundary Parametrizations for Geometric Multigrid Solvers [¶]

Oliver Sander[‖], Rolf Krause [**]

February 24, 2003

Abstract

We present an algorithm that constructs parametrizations of boundary and interface surfaces automatically. Starting with high-resolution triangulated surfaces describing the computational domains, we iteratively simplify the surfaces yielding a coarse approximation of the boundaries with the same topological type. While simplifying we construct a function that is defined on the coarse surface and whose image is the original surface. This function allows access to the correct shape and surface normals of the original surface as well as to any kind of data defined on it. Such information can be used by geometric multigrid solvers doing adaptive mesh refinement. Our algorithm runs stable on all types of input surfaces, including those that describe domains consisting of several materials. We have used our method with success in different fields and we discuss examples from structural mechanics and biomechanics.

1 Introduction

The last decades have shown a demand for solving partial differential equations with increasing precision. In many fields of applications, ranging from structural mechanics over computational fluid dynamics to medical computing, the occuring problems are usually formulated as partial differential equations that need to be solved numerically. Moreover, the geometries involved can be very complex.

Even nowadays, numerical simulations involving complex geometries are very time consuming. In particular, the demand on precision and thus on the size of the computation has risen much faster than the hardware capabilities offered by any type of computer. A great deal of emphasis has therefore been put on the development of fast algorithms. For elliptic partial differential equations, for example, multigrid and domain decomposition methods are widely used for the fast iterative solution of the arising discrete systems. These methods are of optimal and quasi-optimal complexity, respectively, see [8, 23, 36]. However, even algorithm of optimal complexity do not guarantee an efficient method. For example, in the case of linear finite elements in three space dimensions one

[¶]This work was supported by the DFG research center 'Mathematics for key technologies' (FZT 86) in Berlin

[‖]DFG research center 'Mathematics for key technologies' (FZT 86) in Berlin

[**]Freie Universität Berlin, Fachbereich Mathematik, Arnimallee 2–6, 14195 Berlin, Germany

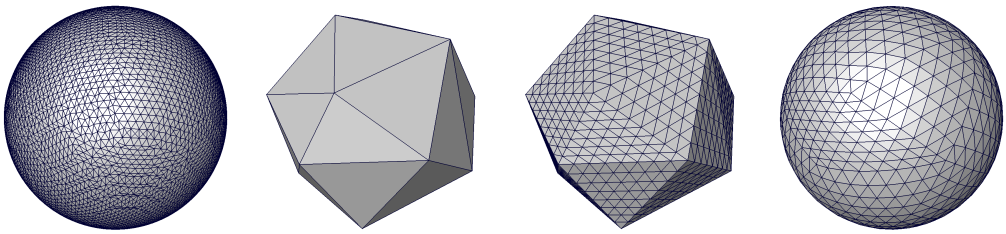

Figure 1: Using parametrized boundaries increases the geometric accuracy of local mesh refinements.

uniform refinement step only reduces the discretization error by a factor of 1/2 but increases the amount of memory needed by a factor of eight. Thus for complicated geometries, uniform refinement is not applicable. To overcome this difficulty, adaptive strategies can be used. Here, a sequence of discrete spaces is constructed by means of suitable a posteriori error estimators, see [1] for an overview. The spaces are designed to have good approximation properties while having as small a dimension as possible. Often for elliptic partial differential equations, this is achieved by refining those elements where some local error indicator hints at high errors.

Using adaptive mesh refinement techniques on complicated domains has a serious shortcoming, however. By definition, coarse base grids do only represent an approximation to the true geometry. Smooth boundaries and geometric details are sacrificed for the sake of computational speed. This is for example the case in Figure 1, where a sphere is approximated by a polyhedron. Mesh refinement techniques allow inserting an arbitrary number of additional vertices. Thus, the discretization error is diminished. However, standard methods such as edge bisection or red–green refinement do not change the geometry of the domain. The corresponding boundary value problem is therewith solved on a domain that differs considerably from the sphere, no matter how high the resolution.

This problem can be solved using parametrized boundaries. By storing the true boundary as a function on the discrete boundary, at each refinement step it is possible to insert the new boundary vertices on the true surface. This is available in many finite element software packages, as for example UG [4]. However, so far, such parametrizations had to be prescribed analytically and entered by hand. Thus, only simple geometric shapes such as circles and ellipses were available. For truly complicated geometries, as they occur for example in hyperthermia treatment planning, no boundary parametrizations were possible, even though the models are usually available in far higher resolution than the final computational grid.

Our paper addresses this issue. In Section 2, we propose an algorithm that automatically computes a coarse surface with a parametrized boundary taking as input a high-resolution surface. This high-resolution surface might for example be the output of a laser range scanner or the segmentation of a computer-tomograph scan. Following the approach by Lee et al. [28] known from computer graphics, we construct a coarse boundary surface by means of a surface simplification algorithm, and maintain a valid parametrization at each step.

In Section 3 we will show how this technique can be used to increase the

geometric accurracy of a standard multigrid solver. The method increases the efficiency of the solver since it allows to compute on coarser base grids, refining only where special error estimators decide that it is necessary. In particular, this is superior to local mesh simplification as introduced by Zachow [39], where the user has to choose the local levels of resolution. We give two examples from structural mechanics and biomechanics and show how the use of parametrized boundaries increases both overall precision and efficiency.

2 The Construction Algorithm

In this section we describe our algorithm for the automatic construction of parametrized boundary and interface surfaces. It is based on the successive removal of vertices of the triangulated surface. The resulting holes are then remeshed by a local Delaunay triangulation giving rise to a simplified surface with less surface vertices. In contrast to pure simplification algorithms, see, e.g., [11, 20], following our approach the original shape of the surface is stored. Moreover, each point on the simplified surface can be bijectively mapped to a point on the original surface by using a parametrization mapping constructed during the simplification process. Proceeding in that way, the shape of the original surface can be restored by repeated refinement of the simplified surface while moving the nodes inserted during the refinement process onto their position on the original surface. This simplification/refinement process is particularly useful for adaptive finite element calculations involving complex geometries, see Section 3 and, e.g., publications on medical computing [12, 37].

Our method originates in computer graphics, where it can be used for a variety of problems, such as surface remeshing and texture map generation [30]. Constructing a parametrization function by surface simplification has first been proposed by Lee et al. [28]. Other techniques have also been used, such as the automatic embedding of an explicitly described topological triangulation into the original surface by Praun et al. [32]. Eck et al. [15] use an approach based on Voronoi diagrams to partition their input surfaces into parameter patches. A completely different approach to multiresolution surface representations uses subdivision surfaces. Here, surfaces are stored as a coarse base grid together with interpolation rules and possibly detail offsets. Subdivision surfaces are a powerful technique that have been used successfully for such varied tasks as geometry compression [29], multiresolution editing [41], and level-of-detail rendering [33]. We refer the reader to [29, 40] for introductions to this topic.

Let $\mathcal{S}^0$ be a triangulated surface. Intuitively, a triangulated surface is the union of a finite set of triangles in $\mathbb{R}^3$, together with a discrete topology. We will formalize this notion in Section 2.1. Our aim is to construct a second surface $\mathcal{S}$ consisting of less triangles and a continuous mapping $\phi : \mathcal{S} \to \mathbb{R}^3$ such that

$$\phi(\mathcal{S}) = \mathcal{S}^0$$

and ϕ is injective. We call $\mathcal{S}$ the *simplified surface* and ϕ the *parametrization* of $\mathcal{S}^0$ with respect to $\mathcal{S}$. Following this notation, the simplified surface $\mathcal{S}$ can be regarded as the *parameter domain* of the parametrization function ϕ.

With such a pair $(\mathcal{S}, \phi)$ available we can use standard refinement schemes but interpret them as refinements of the parameter domain $\mathcal{S}$. The finite element computation will then be done on a grid whose boundaries have been subjected

to the function ϕ. This assures that the boundaries of a repeatedly refined grid converge to the original boundaries of the model. Additionally, we can access other features of $\mathcal{S}^0$, such as the surface normals, as functions on $\mathcal{S}$.

The simplification algorithm consists of two steps. We first construct the base domain surface $\mathcal{S}$ and an initial valid parametrization function ϕ by means of a simplification algorithm. Then, as ϕ is not unique, we try to modify it such as to obtain refined surfaces of optimal quality. These somewhat vague notions will be formalized and explained in detail in Section 2.5.

2.1 Triangulated Surfaces

Our algorithm for constructing parametrizations is based on a simplification algorithm for triangular surfaces. We therefore formalize the notion of such surfaces. Unlike a surface in the differential geometry sense, a triangulated surface is a discrete combinatorial entity with an embedding into Euclidean space. This will allow us later on to speak of, for example, *vertices* and *neighbors of vertices* of a surface. Many concepts from the differential geometry case, such as the open neighborhood of a point, transfer easily to the discrete setting. We will define them when needed, and refer the more interested reader to [16] for details.

Definition 2.1 *An abstract simplicial complex $\mathcal{K}$ is an ordered pair of disjoint, finite sets (V, F), such that F is a set of subsets of V. For the set F, the following condition must hold:*

$$\text{If } \alpha \in F \text{ and } \beta \subseteq \alpha, \text{ then } \beta \in F. \tag{1}$$

We call a set $\alpha \in F$ a simplex of dimension $\dim \alpha = \operatorname{card} \alpha - 1$. *The dimension of the abstract simplicial complex is defined as the dimension of its highest-dimensional simplex*

$$\dim \mathcal{K} = \max_{\alpha \in F} \dim \alpha.$$

Intuitively, the set V is a set of *vertices*, and F a set of simplices on the vertices V. F is called the set of *faces*. Condition (1) then asserts that for each simplex α in F, its lower-dimensional faces are also part of the complex.

A triangulated surface is a special kind of an abstract simplicial complex, together with a geometric structure.

Definition 2.2 *A triangulated surface $\mathcal{S} = (V, F)$ is an abstract simplicial complex of dimension two such that*

$$\forall \alpha \in F, \quad \operatorname{card} \alpha < 3 \quad \exists \beta \in F \quad \textit{with} \quad \alpha \subsetneqq \beta$$

together with a function

$$\varphi : V \to \mathbb{R}^3.$$

An alternative way of describing a triangulated surface is as a triple of sets $\mathcal{K} = (V, E, F)$, where V is a set of vertex positions in $\mathbb{R}^3$, E is a set of *one-faces*, and F a set of *two-faces*. We call the one-faces *edges*, the two-faces *triangles* and require that every edge should be the subface of at least one triangle and that every vertex should be incident to at least one edge.

The above definition neatly separates the combinatorial and geometric aspects of triangulated surfaces. All information about adjacency is contained in the abstract simplicial complex, which in turn does not depend on the surrounding space. If we consider the union of the convex hulls

$$\bigcup_{\alpha \in F} \operatorname{conv} \alpha \subset \mathbb{R}^3$$

of all faces of F, we arrive at a surface in the classical sense, in particular, a finite union of compact C^0-manifolds. We call it the *straight-line geometric realization* of $\mathcal{S}$. Throughout this article, we sometimes use the term 'triangulated surface' when in fact the straight-line geometric realization of a triangulated surface is meant. The exact meaning will be clear from the context.

Two more concepts will be useful when dealing with triangulated surfaces.

Definition 2.3 *Given an abstract simplicial complex $K = (V, F)$, the* star $\operatorname{St} v$ *of a vertex $v \in V$ is the set of all faces $f \in F$ that contain v. The* link $\operatorname{Lk} v$ *of v is the set of all subfaces of the star of v that do* not *contain v [16].*

Thus, the term *star* formalizes the notion of the open neighborhood of a vertex v in a strictly combinatorial setting. Equivalently, the *link* of a vertex can be seen as the boundary of that neighborhood.

2.2 Surface Simplification

We now describe our algorithm that constructs a coarse surface and a parametrization function ϕ. Given a high-resolution surface $\mathcal{S}^0$ describing the boundaries of a computational domain, we initialize the algorithm by defining the embedding function ϕ^0 on all $\mathcal{S}^0$. If $\mathcal{S}^0 \hookrightarrow \mathbb{R}^3$ is the straight-line geometric realization of an abstract simplicial complex and $x \in \mathbb{R}^3$ is a point on $\mathcal{S}^0$, then $\phi^0(x) = x$. The algorithm then constructs a sequence of triangulated surfaces $\mathcal{S}^i$ and functions $\phi^i : \mathcal{S}^i \to \mathbb{R}^3$ such that $\mathcal{S}^i$ contains less triangles than $\mathcal{S}^{i-1}$ and $\phi^i(\mathcal{S}^i) = \mathcal{S}^0$ for all $0 \leq i \leq N$. For the construction of the surfaces $\mathcal{S}^i$, we use a surface simplification algorithm.

2.2.1 Geometric Operations

A surface simplification algorithm generally consists of two main parts. The first is a geometric operation that allows the controlled local reduction of surface complexity. Several of those operations have been proposed in the literature, such as half-edge contraction [26] and pair contraction [20]. We chose point removal since it doesn't lead to newly created vertices, it is easily extendable to non-manifold surfaces and it is easy to ensure the preservation of topology. Point removal works by removing a single vertex together with all triangles adjacent to it from the surface. The procedure leaves a hole which, when dealing with manifolds, is bounded by a simple polygonal loop. We flatten out this loop into the plane by using the polar map described in [14]. The flattened polygon is then triangulated with a constrained Delaunay triangulation algorithm which effectively patches the hole left in the surface.

2.2.2 Quality Measures

The second important ingredient is a scalar oracle that makes it possible to rank the different possible simplification steps according to the error they would introduce. Again, many different strategies have been described [26, 35]. We combine the following aspects:

- **Geometric Error:** In order to make sure that our simplified surface approximates the original surface well, we monitor a modified Hausdorff distance between different simplification stages. The Hausdorff distance defines a metric on subsets of $\mathbb{R}^3$ (see Alt [2] for a definition). It is, however, quite costly to compute. We therefore use a compromise. We define the geometric cost of removing a vertex v as the Hausdorff distance between the retriangulation of $\operatorname{Lk} v$ and the image of ϕ restricted on it. The retriangulation of $\operatorname{Lk} v$ consists only of very few triangles, and so the distance can be computed more efficiently.

- **Triangle Aspect Ratio:** Our aim is to construct simplified surfaces that allow mesh generators to create tetrahedral meshes of high numerical quality. One commonly accepted measure for this quality is the *element aspect ratio* ρ, which is defined as the ratio R/r between circumsphere radius R and insphere radius r. This ratio ρ should be bounded from above. The aspect ratio of any tetrahedron is always bounded from below by the *triangle aspect ratios* of its four triangular faces. Therefore, the triangle aspect ratio $\rho_\Delta = R_\Delta/r_\Delta$ of the triangles in any boundary surface should stay below a reasonable limit. In our algorithm, we favour reduction steps that remove triangles with a high aspect ratio as well as steps that introduce triangles with a low aspect ratio.

- **Intersections:** Surfaces used for mesh generation divide space into compartments. Therefore it is of paramount importance that the surfaces do not self-intersect. Surfaces that contain self-intersections (or boundaries, for that matter) do not unambiguously separate space into regions and are therefore unsuitable as input for a mesh generator. To guarantee the absence of intersections we check which potential simplification operations generate intersecting triangles. Those that do are ranked as inadmissible.

- **Long Edges:** Many mesh generators have difficulties dealing with boundary surfaces with widely varying edge lengths. We therefore also limit the maximal length of the edges that are introduced into the surface.

2.2.3 Storing the Parametrization Function

The main goal of our procedure is to construct the parametrization ϕ of the original surface while we simplify it. Since the image of ϕ is piecewise linear, the function can be viewed as a graph $\mathcal{G}$ embedded without intersections in the coarse surface. Thus, $\mathcal{G}$ is a finite set of vertices on $\mathcal{S}$ and a set of paths on $\mathcal{S}$ that connect pairs of vertices. Different paths can only intersect at their endpoints and are shortest connections on $\mathcal{S}$, i.e., they are line segments on each triangle of $\mathcal{S}$. The graph $\mathcal{G}$ that represents the parametrization function ϕ is isomorphic to the *edge graph* of the original surface $\mathcal{S}^0$. Since all faces of $\mathcal{G}$ are triangles, the restriction of $\mathcal{G}$ to a base grid triangle has only triangular or quadrangular faces

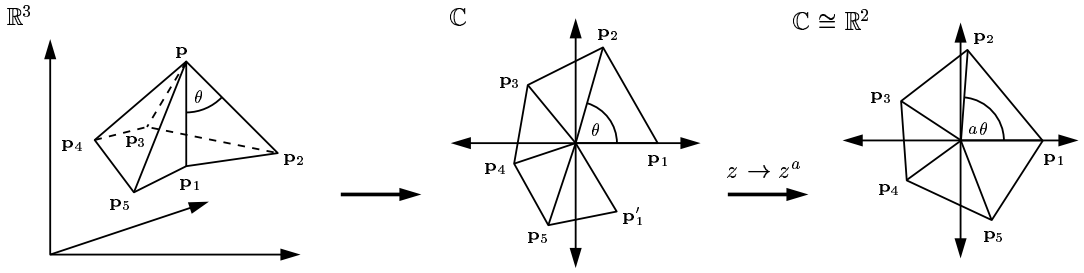

Figure 2: The polar map: The star is first cut along $\overline{\mathbf{p}\,\mathbf{p}_1}$ and flattened out into the complex plane. The conformal map $z \to z^a, \quad a = 2\pi/\Theta$, then stretches it out so that it covers one full circle.

(triangle stumps). For each vertex of $\mathcal{G}$, we store its *parameter position* x, i.e., its barycentric coordinates on a base grid triangle j, as well as its *image position* $\phi(x)$ in $\mathbb{R}^3$. The function ϕ can then be evaluated at any point on any triangle by a simple point location algorithm. For each point x on a base grid triangle, this algorithm returns the graph face x is in so that the return value $\phi(x)$ can be computed using linear interpolation. For the point location, we chose the randomized version of the algorithm proposed by Brown and Faigle [9] for its robustness against finite precision arithmetic and its ease of implementation. Since the Brown-Faigle algorithm only works on *triangulations*, we complete the graph $\mathcal{G}$ to a triangulation on each base grid triangle before querying ϕ for the first time.

2.2.4 The Simplification Algorithm

Remember from Section 2.2 that we initialize the algorithm by defining the embedding function ϕ^0 on the input surface $\mathcal{S}^0$. The graph that implements the identity function is a single triangle on each base grid triangle. We then compute the error function for each vertex and place all vertices in a priority queue. This queue always yields the current best vertex to remove.

Assume now that v is the current best vertex. We first take $\operatorname{St} v$, the star of v, and flatten it out into the plane. For this we use the *polar map* introduced in [14]. The polar map bijectively maps $\operatorname{St} v$ onto a star-shaped polygon in $\mathbb{C}$ and we identify $\mathbb{C}$ with $\mathbb{R}^2$. Unlike other schemes it has the advantage that it always exists. Also, it is *conformal*, i.e., angle-preserving.

The outer boundary of the flattened set of triangles forms a plane star-shaped polygon. In order to obtain a retriangulation, we use a special variant of the *constrained Delaunay triangulation*. We regard the set of all edges that separate the current polygon into a smaller polygon and a triangle and that are admissible in the Delaunay sense [16]. Of those edges, we do a greedy step and take the one that yields the triangle with the smallest aspect ratio when reinserted into the surface. We add this edge into the triangulation and iteratively repeat the above steps on the remaining polygon until we are left with a triangle. Even though this algorithm has a bad asymptotic run-time behaviour, it is usable without any difficulties, since the average number of polygon corners is a small

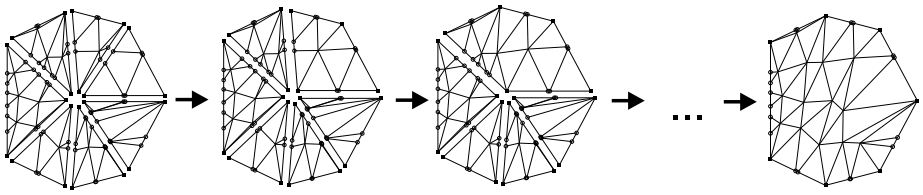

Figure 3: The flattened star of a vertex is merged into a single polygon.

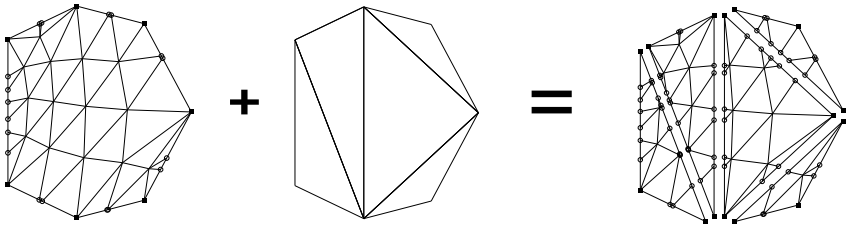

Figure 4: The polygon is cut along the new triangulation.

constant.

Given the star of v and the new triangulation, we have to transfer the parametrization defined on $\operatorname{St} v$. For that we merge the graphs defined on the triangles of $\operatorname{St} v$ into a single one defined on the polygon bounded by the flattened $\operatorname{Lk} v$ (Figure 3). This polygon implements that part of the parametrization function ϕ that had formerly been implemented by the triangles in $\operatorname{St} v$. We then repartition the graph according to the new triangulation obtained above (Figure 4). This leaves us with a set of triangles that can be inserted into the base grid.

There are two problems which can occur at this point. The first one is the following. The polar map which maps $\operatorname{St} v$ onto a plane polygon is a bijection. However, it does not map straight lines onto straight lines. Therefore, the unfolding of the plane graphs on $\operatorname{St} v$ may contain triangles that are 'flipped over'. As a remedy, we decided to reposition the graph vertices on the polygon according to Floater's shape-preserving mapping [17], if such flips occur. This guarantees a intersection-free embedding of the graph into the polygon.

The second possible problem is also due to properties of the polar map. The polygon yielded by it is not necessarily convex. In rare cases it can therefore occur that an edge of the graph on the polygon is partly *outside* the bounding polygon. If that happens, it is impossible to consistently repartition the graph on any retriangulation of the polygon. In those cases, we move the concave corner away from the coordinate center far enough to make it convex.

2.3 Preservation of Topology

When constructing a low-resolution triangulated surface $\mathcal{S}$ as the parameter domain of another surface $\mathcal{S}^0$, it is essential that both surfaces have the same

topological type. This is because identical topological type is a necessary condition for the existence of a continuous bijection between embeddings in $\mathbb{R}^3$ of $\mathcal{S}$ and $\mathcal{S}^0$. Therefore, any geometrical operation that is used to construct $\mathcal{S}$ from $\mathcal{S}^0$ has to leave the topological type unchanged.

Equivalence of topology of abstract simplicial complexes is a global property. It is defined combinatorially by introducing *normal forms* and by asking that they be *trivially isomorphic* (cf., e.g., [31]). However, in order to guarantee topology preservation while simplifying a surface, we have to find some local conditions. These will depend on the particular geometrical reduction operation used. Edelsbrunner [16] has stated such conditions for surface simplification with edge contraction. We found the following simple condition for point removal algorithms to work well in practice and we state its correctness as a conjecture.

Conjecture 2.1 *Let $\mathcal{K} = (V, F)$ be a two-dimensional abstract simplicial complex. Remove a vertex v together with its star $\operatorname{St} v$ from $\mathcal{K}$ and choose a retriangulation of the resulting hole. That retriangulation will add one-faces to the complex which are not part of $\operatorname{Lk} v$. Call the set of those one-faces E. Then the following holds:*

If no $e \in E$ is a one-face in $F \setminus \operatorname{St} v$ then $\mathcal{K}' = (V \setminus \{v\}, F \setminus \operatorname{St} v)$ together with the retriangulation is of the same topological type as $\mathcal{K}$.

More details can be found in [35].

2.4 Handling Non-Manifold Surfaces

When working with triangulated surfaces for finite element solvers it is important that the simplification algorithm can handle surfaces that are not manifolds. When models are considered that consist of more than a single material, the interface surfaces contain points that do not have neighborhoods homeomorphic to a disk. Those are the lines where more than two materials meet. Many simplification algorithms neglect the systematic treatment of such cases. However, for our algorithm to be truly applicable, multi-domain models have to be handled.

In order to properly define the term *manifold* for triangulated surfaces we first state what we mean by a *path* on a surface.

Definition 2.4 *A path P on a surface $\mathcal{S}$ is an alternating sequence of vertices and edges, say $x_0, e_1, x_1, e_2, \ldots, x_l$, where $e_i = \{x_{i-1}, x_i\}$. The* length *of P is l, the number of edges in the path. If $x_i \neq x_j \quad \forall i \neq j$ we call P a simple path. If $x_0 = x_l$, we call the path a* cycle*, and if in addition all its edges and all vertices $x_1, \ldots, x_{l-1}$ are distinct, we call it a simple cycle.*

We can now define the manifoldness of a vertex v by looking only at its link. Note that we arrive at a strictly combinatorial definition of manifoldness!

Definition 2.5 *A vertex v is called a manifold vertex if its link forms a simple cycle. The vertex v is called a boundary vertex, if its link forms a simple path. In all other cases, v is called a non-manifold vertex.*

If a triangulated surface $\mathcal{S}$ contains only manifold vertices, it is called a manifold surface, or simply a manifold. If it contains boundary vertices as well

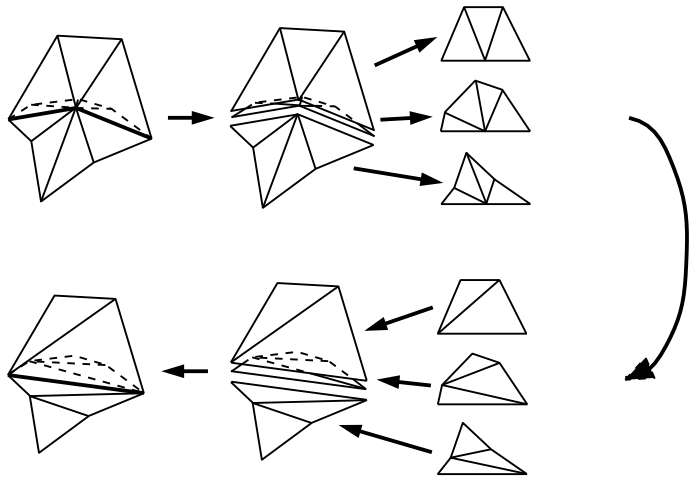

Figure 5: Removal of a vertex on a feature line of third order. The three adjacent half-stars are separately mapped to the plane and retriangulated. The result still contains the feature line.

as manifold vertices, it is called a manifold with boundary. In all other cases, it is called a non-manifold.

This is equivalent to saying that a geometric realization of a triangulated surface $\mathcal{S}$ is a manifold in the standard sense.

In order to simplify and parametrize triangulated surfaces that are not manifold everywhere, we have implemented a strategy inspired by the handling of *feature edges* in [28]. By looking at all triangles touching a given vertex, we put that vertex into one of three categories:

- **Regular Vertices:** The vertices are manifold and can be removed as described in Section 2.2.4.
- **Feature Line Vertices:** This is the important case for non-manifold handling. The vertices in this class sit on a path where three or more sheets touch and form a t-junction. We can handle them in the following manner (Figure 5). The neighborhood of those vertices is equivalent to a collection of n half-disks which are glued together at two edges. All quality measures described in Section 2.2.2 are therefore computed *per half-disk*, and the arithmetic mean of all of them is taken as the priority of the vertex. The point removal follows along the same lines. We map each half-disk onto the positive half-plane using a modified polar map, and retriangulate each polygon separately. The result is a consistent retriangulation of the neighborhood of the removed vertex, with the two feature edges now replaced by a single one.
- **Feature Vertices:** These vertices have a neighborhood that is so garbled up that we don't know what to do with it. We simply don't remove them.

2.5 Parametrization Smoothing

The parametrization function ϕ as it has been introduced in the first part of Section 2 is not unique. It is therefore reasonable to ask whether the particular choice of ϕ has any impact on the quality of the computational grid. That this is indeed the case can be seen by remeshing a base grid using a function ϕ as yielded by the algorithm described so far. The result is a grid of extremely poor quality (see Figure 7, center). We will now shortly describe an algorithm that considerably improves the quality of the refined grids.

The straight line geometric realization of our base grid is a finite union of two-dimensional C^0-manifolds embedded in $\mathbb{R}^3$. Since it is compact, it can be covered by a finite set of coordinate patches. For each patch U there exists a homeomorphism $\theta : U \to \mathbb{R}^2$ such that $V = \theta(U)$ is a simply connected open subset of $\mathbb{R}^2$. The functions θ are called *coordinate functions*. For any U, the parametrization function ϕ restricted to U can also be seen as a function $\phi' = \phi \circ \theta^{-1}$ defined on $V \subset \mathbb{R}^2$.

This case has been intensively studied in the literature. We would like the parametrization function $\phi|_U$ to be as free of distortion as possible. Different definitions of distortion of a function are possible, see for example [35]. It follows from Gauss's *Theorema Egregium* that ϕ' cannot always be an isometry [18]. Note, however, that if θ is an isometry, then finding a suitable ϕ' is equivalent to finding a suitable ϕ.

For the case that the image of ϕ' is the embedding of a triangulated surface, Floater [17] suggested a compromise. He showed that it is always possible to construct a map $\mathcal{P} : \mathcal{S} \to V$ that fulfills the following *reproduction property*. He calls a mapping $\mathcal{P}$ *shape-preserving* if it fulfills the following condition:

Definition 2.6 *Let $\mathcal{S}$ be a triangulated surface with n interior nodes $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and $N - n$ boundary nodes $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N$ and let $\mathcal{P}$ be a parametrization of $\mathcal{S}$ over a convex polygon V with the boundary points $\mathbf{v}_{n+1}, \ldots, \mathbf{v}_N$ such that $\mathcal{P}(\mathbf{x}_i) = \mathbf{v}_i, \quad n+1 \leq i \leq N$. Then $\mathcal{P}$ is called shape-preserving if the following condition holds:*

If $\mathcal{S}$ is planar and if the mapping from $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N$ to $\mathbf{v}_{n+1}, \ldots, \mathbf{v}_N$ is an affine one, then $\mathcal{P}$ is an affine mapping of $\mathcal{S}$.

This type of mapping is widely used in surface parametrization algorithms [21, 22]. For details on its construction see Floater's article [17].

We now hope that iteratively rearranging ϕ on different coordinate patches according to Floater's algorithm will increase the overall quality of the parametrization. For the coordinate patches U we choose the quadrangles formed by pairs of triangles T_1, T_2 that share a common edge. That way, an isometric coordinate function $\theta : T_1 \cup T_2 \to \mathbb{R}^2$ can always be found.

We then consider the two triangles as a single quadrangle. In many cases, this quadrangle is convex. If it is not, there will be exactly one inward corner, and it will be one of the endpoints of the common edge E. We take this endpoint and move it along the continuation of E away from the other endpoint until the corner is convex.

Floater's parametrization scheme is now applied to the new quadrangle. This moves the preimages of the vertices of $\mathcal{S}^0$. Some of them wander from one triangle onto the other. The quadrangle is then recut into the original two triangles. The whole procedure is visualized in Figure 6.

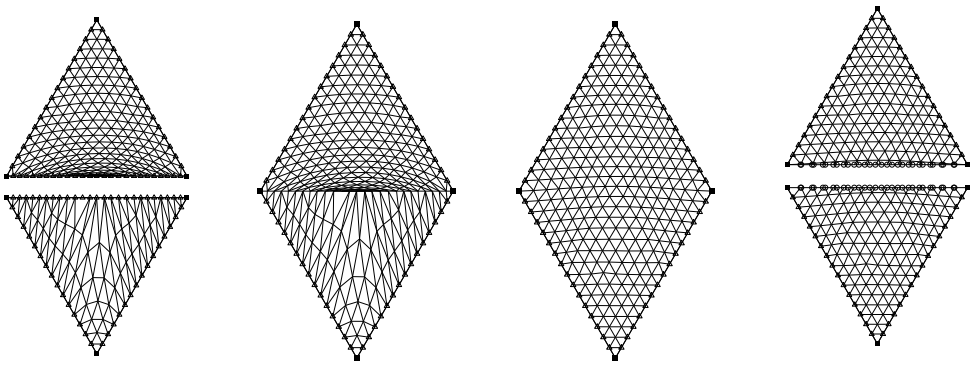

Figure 6: The edge relaxation procedure stretches out the plane graph representing the parametrization function ϕ evenly across both triangles. The triangulations on the two triangles on the far left actually do match.

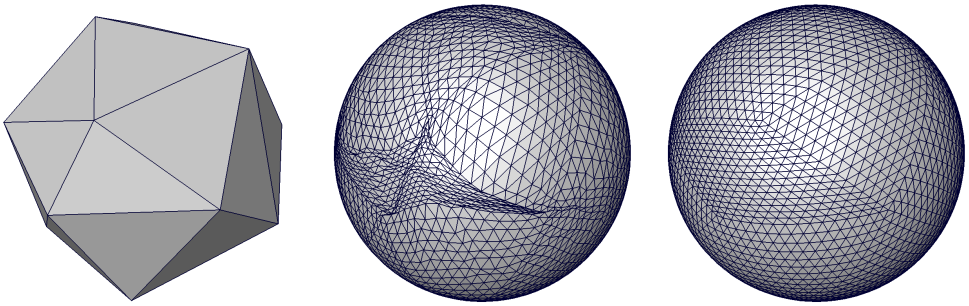

Figure 7: The edge relaxation algorithm significantly increases the quality of the refined mesh. The figure shows a base grid (left) and a refined surface without (center) and with (right) smoothing.

In order to improve the quality of ϕ on all $\mathcal{S}$ we apply this procedure successively to all adjacent triangle pairs. After repeating this a few times the parametrization function converges to a stable limit of greatly improved quality, as can be seen from Figure 7.

3 Application to Multigrid Methods

In this section we apply the surface parametrization developed in Section 2 within the context of finite element methods. Particularly, we combine our parametrization with adaptive multigrid methods on unstructured tetrahedral grids and discuss the resulting iterative scheme.

For the numerical simulation of problems involving complex geometries in, e.g., biomechanics or continuum mechanics, two main goals have to be achieved. These are accuracy and efficiency. Accuracy can be provided by constructing a suitable finite–dimensional space, which in the context of finite element methods is done by choosing a suitable grid and appropriate ansatz functions. For well–chosen spaces, the larger the number of unknowns, the better the discrete approximation. On the other hand, particularly for problems in three space dimensions, a large number of unknowns also requires a high computational effort for solving the arising systems of equations. Thus, to obtain an efficient numerical method it is desirable to use grids with as few unknowns as possible. To achieve this, in case of local errors as may be caused by, e.g., corners in the domain or local surface loads at contact interfaces, it is reasonable to use meshes with varying meshsize. Using a posteriori error estimators, (see [38] for an overview) these grids can be constructed by successively refining a given coarse grid.

Often, local refinement techniques are used in combination with multigrid methods. For linear elliptic partial differential equations, multigrid methods are well known to be of optimal complexity for the solution of the arising linear systems of equations, see [8, 23]. In contrast, using direct solvers leads to a computational complexity of $\mathcal{O}(n^{(d+1)/2})$, n the number of unknowns and d the spatial dimension. Thus, in three space dimensions multigrid methods are clearly superior to direct solvers in terms of computational effort. Multigrid methods require a hierarchy of nested grids, which in the context of adaptive refinement is provided by the successive refinement process. Unfortunately, in case of complicated geometries this procedure cannot be applied without any modifications.

For the representation of complicated geometries, a large number of elements (tetrahedrons, hexahedrons, prisms, pyramids) are necessary, whereas multigrid methods require a coarse grid having as few unknowns as possible. Such a coarse grid can be constructed by first simplifying a given complicated surface and then using a mesh generator [35]. Then, a mesh hierarchy can be established by successively subdividing elements of the coarse grid. As can be seen in Figures 1, 8, and 11, during this process shape information of the original surface is lost, if standard subdivision techniques are applied. This motivates combining adaptive multigrid techniques with the parametrization proposed in the previous section. Starting from a coarse approximation of the geometry under consideration, a sequence of meshes is constructed by successively refining the coarse grid locally on the basis of an a posteriori estimate of the local

errors. Additionally, the new nodes on the boundary are moved to their original position, which is provided by our parametrization. Proceeding in that way, we can meet the requirement of accuracy and efficiency by applying a multigrid method on the resulting hierarchy of grids.

In the following, we present some numerical results illustrating the application of the boundary parametrization for a linear elasticity problem in structural mechanics and a nonlinear contact problem in biomechanics. The numerical experiments have been carried out in the framework of the finite element toolbox UG [4]. For the nonlinear contact problem considered in Section 3.2, we use the monotone multigrid for contact problems developed and implemented in [27]. All results have been visualized using AMIRA [3].

Let us finally remark that, when applying boundary parameterizations, using standard a posteriori error estimators are not sufficient. This is due to discretization errors caused by the approximation of the boundary. Since the boundary of the domain is not assumed to be resolved by the coarse grid, additional effort has to be spent to also estimate the geometric error. We refer the reader to [13] for the construction and analysis of an a posteriori error estimator that includes control of the boundary approximation. However, this error estimator requires local meshes which resolve the original boundary. In three space dimensions, this is possible in principle but requires a lot of implementational effort.

3.1 Structural Mechanics

Our first example is a problem in linear elasticity. The geometry under consideration is the crank shaft depicted in the right picture of Figure 8. In the middle of the crank shaft, where the connecting rods are mounted, Dirichlet boundary data corresponding to a uniform displacement in vertical direction are prescribed. On the circular areas at both ends of the crank shaft, homogeneous Dirichlet boundary conditions are applied. The original surface consists of $13,133$ vertices. We do not use the original geometry for our numerical computation. Instead, by applying the simplification algorithm described in Section 2, the number of surface vertices is reduced to 897. The corresponding simplified surface can be seen on the left side of Figure 8. Using the mesh generator provided by AMIRA, a grid is created which serves as coarse grid for an adaptive linear multigrid method. As described above, the grids of the multigrid hierarchy are constructed successively by adaptive refinement.

The adaptive refinement process is controlled using a standard residual based a posteriori error estimator. During the refinement process, the newly inserted boundary nodes are moved to their original position on the original boundary which is provided by our parametrization. The adaptive refinement process is stopped, if the estimated discretization error is below 5%. On the coarsest level, the discrete problem is solved up to machine precision by Gaussian elimination. On subsequent levels, $k \geq 1$, we use a conjugate gradient method preconditioned by a linear multigrid $\mathcal{W}(3,3)$–cycle to solve the arising system of linear equations. The iterate $\boldsymbol{u}_l^\nu$ on level l is accepted, if the stopping criterion

$$\|\boldsymbol{f}_l - \boldsymbol{A}_l \boldsymbol{u}_l^\nu\|_2, \leq 10^{-6} \tag{2}$$

is satisfied, where $\|\boldsymbol{f}_l - \boldsymbol{A}_l \boldsymbol{u}_l^\nu\|_2$ is the Euclidean norm of the residual $\boldsymbol{f}_l - \boldsymbol{A}_l \boldsymbol{u}_l^\nu$ of the ν-th iteration step on level l. Here, $\boldsymbol{f}_l$ and $\boldsymbol{A}_l$ are the right hand side

level	# vertices	# dof	# iterations
0	897	2 691	–
1	3 903	11 709	14
2	7 763	23 289	20
3	26 094	78 282	26
4	37 017	111 051	29
5	69 415	208 245	26
6	89 669	269 007	25

Table 1: Number of iterations

and the stiffness matrix on level l, respectively. We use nested iteration, i.e.,

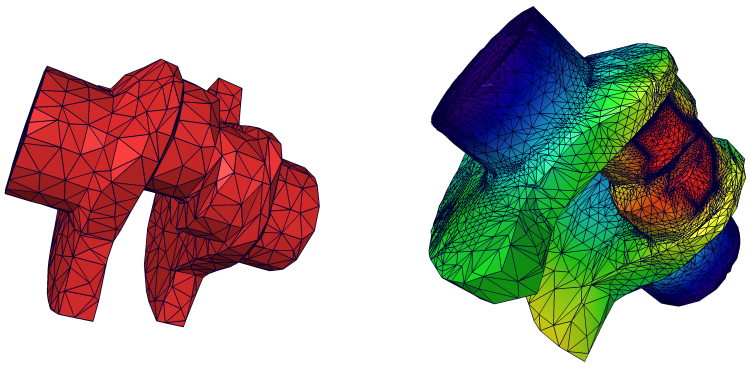

Figure 8: Crank shaft: Coarse grid (left) and adaptively refined grid (right).

the initial iterates on level $l, l > 0$, are given by $\boldsymbol{u}_l^0 = \mathcal{I}_{l-1}^l \boldsymbol{u}_{l-1}^{\nu_{l-1}}$ for $l = 0, \ldots, 6$. Here, $\mathcal{I}_{l-1}^l$ is the natural injection from the finite element space on level $l-1$ into the finite element space on level l and $\boldsymbol{u}_{l-1}^{\nu_{l-1}}$ is the final iterate on level $l-1$.

In Table 1, the number of nodes and the degrees of freedom on the levels $l = 0, \ldots, 6$ and the number of iterations necessary to fulfill the stopping criterion (2) are shown. As can be seen, the number of iterations necessary to achieve the prescribed tolerance does not increase with the number of degrees of freedom.

The resulting displacement field can be seen in the right picture of Figure 8. Here, light color reflects large displacements and dark color small displacements. In the right picture of Figure 9 the isosurface of the norm of the solution $\boldsymbol{u}_5$ on level 5 for $|\boldsymbol{u}_5| = 0.0815$ is shown.

3.2 Biomechanics

In our second example, we consider a nonlinear contact problem in biomechanics. The geometry under consideration can be seen in the right picture of Figure 10 and consists of a skull and a mandible. On the upper part of the skull and on the lower part of the mandible Dirichlet boundary conditions corresponding to a displacement in direction towards the grey block between the teeth, see Figure 11, are applied. This grey block is assumed to be a rigid obstacle,

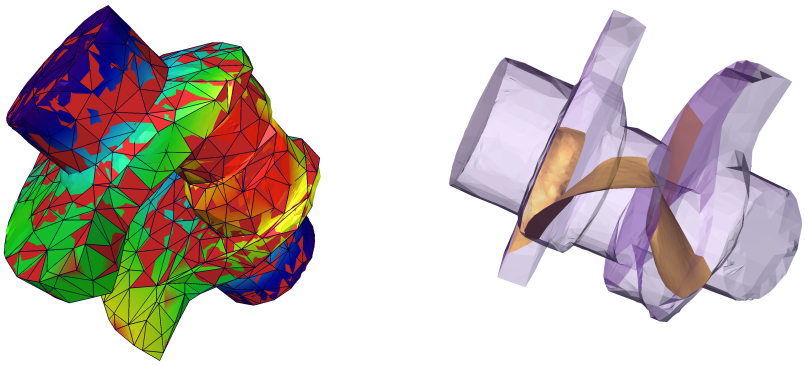

Figure 9: Final grid and coarse grid (left) and isosurface of the norm of the displacements on level 5 for $|\boldsymbol{u}_5| = 0.0815$ (right).

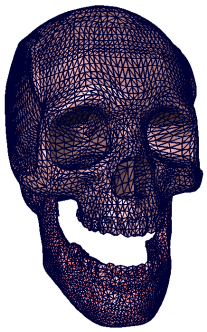

Figure 10: Original surfaces of skull and mandible

thus leading to a Signorini-type contact problem. Linear elastic behavior of the material is assumed. Again, the original surface is simplified applying the algorithm given ins Section 2. For the

resulting simplified surface see the left picture of Figure 11. By means of a mesh generator, a coarse grid is created. Again, the grids of the multigrid hierarchy are constructed successively by adaptive refinement controlled by a standard residual error estimator. On each level, a nonlinear contact problems is solved by means of a $\mathcal{W}(3,3)$–cycle of the non-linear monotone multigrid method for contact problems developed in [27]. For this example, the refinement process is stopped after 5 refinement steps. As can be seen in the right picture of Figure 11, the refinement is highly local and is restricted to the vicinity of the contact area. Near the contact area, the original geometry is restored completely. In contrast, on the upper part of the skull the grid remains coarse. Again, light color reflects large displacements and dark color reflects small displacements.

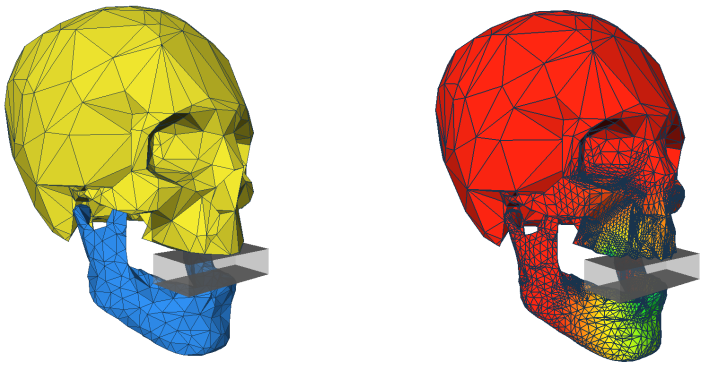

Figure 11: Biting skull: Coarse grid (left) and adaptively refined grid (right).

3.3 Preserving the Quality of the Elements

In the preceding section we have shown that multigrid techniques can be combined successfully with the parametrizations developed in Section 2. We now discuss shortly the main difficulty associated with this approach which is the possibly deteriorating quality of elements near the boundary.

The shape of the elements is of great importance for the quality of the numerical approximation. On the one hand, the standard discretization error estimates for finite element spaces rely on the shape regularity of the mesh, see, e.g., [7, 10]. On the other hand, the convergence of the multigrid method does also depend on the shape regularity. In particular, in case of geometric multigrid methods there arise even more constraints with respect to grid quality than do appear in case of a single grid method. This is due to standard multigrid methods being based on a sequence of nested grids, all of which have to be of suitable quality.

For this reason, often so called red–green refinement techniques are used. These refinement techniques preserve the shape properties of the coarse grid, see, e.g., [5, 6]. However, using red–green refinement, newly inserted boundary vertices remain on the surface of the initial grid, see Figure 1 For the numerical examples presented above, we use red–green refinement, but we move the newly inserted boundary vertices to their original positions which is provided by the parametrization. This can lead to elements having negative volume or to extremely flat elements, so-called 'slivers'. For an example, we consider the case of a locally non-convex boundary. As can be seen in in Figure 12, moving newly inserted vertices to their original positions on the surface can result in elements with negative volume.

To overcome this difficulty, grid improvement techniques can be used. Since geometric multigrid methods rely on a hierarchy of *nested* grids, we can only consider mesh improvement algorithms which preserve the discrete topology. Grid smoothing algorithms that preserve the discrete topology usually work by solving either a global nonlinear problem [34] or a sequence of local nonlinear problems [19, 24, 25]. In both cases, the aim is to improve the quality of the mesh under consideration by moving interior vertices.

For our numerical computations, we use local strategies. After refinement

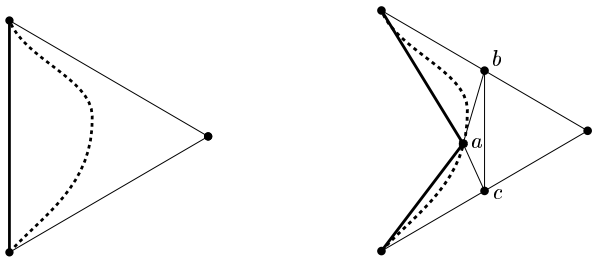

Figure 12: Refinement at a concave surface (dotted) leads to a badly shaped element (a, b, c). If the surface was even more concave, a would pass beyond b and c, leading to an element with negative volume.

of the mesh the smoothing and untangling techniques from [19] are applied. Since for our examples, these techniques do not always lead to a mesh without elements with negative volumes, we additionally used the following simple recovering technique to obtain positive volume for all elements of the mesh. Let T be a tetrahedron created by red–green refinement of its father element T' and let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4$ denote the vertices of T. We assume T' to have positive volume. Let us further assume that T has one vertex $\boldsymbol{x} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4\}$ on the surface of the domain. Since we use red–green refinement, during the refinement process the boundary vertex $\boldsymbol{x}$ is constructed by subdividing an edge $(\boldsymbol{x}'_{\text{left}}, \boldsymbol{x}'_{\text{right}}) \subset T'$ of the father element T', i.e., we have $\boldsymbol{x} = 1/2 \cdot (\boldsymbol{x}'_{\text{left}} + \boldsymbol{x}'_{\text{right}})$. After refinement, the boundary vertex $\boldsymbol{x}$ is moved to its original position $\boldsymbol{v} = \phi(\boldsymbol{x})$ which is provided by our parametrization ϕ. This can cause the element T to have a negative volume. In this case, we choose some $0 < \lambda < 1$ such that the convex combination $\boldsymbol{v} = \lambda\boldsymbol{x} + (1-\lambda)\phi(\boldsymbol{x})$ yields an element with positive volume. This works since the choice $\lambda = 0$ corresponds to standard red–green refinement and the determinant is a continuous function.

4 Conclusion

We have presented an algorithm for the automatic construction of boundary and interface parametrizations for finite element grids. This algorithm is an adaption of the MAPS-algorithm [28] known from computer graphics. It consists of constructing a coarse boundary surface $\mathcal{S}$ by simplifying a high-resolution surface and keeping the original surface as the image of a parametrization function ϕ defined on $\mathcal{S}$. This function allows access to the correct shape and surface normals of the original surface as well as to any kind of data defined on it. Unlike standard simplification schemes, our algorithm correctly handles nonmanifold surfaces. It can thus deal with surfaces that describe several domains. We have shown that these boundary parametrizations can be used effectively by adaptive multigrid solvers. They lead to an increase in efficiency, since computations can be started on coarser grids without sacrificing geometric accuracy. The changing boundary frequently causes a decrease in the quality of the three-dimensional grid. This problem is a topic of ongoing research, however, even simple heuristics for improving grid quality render the system quite well usable. We showed

this giving two examples from structural mechanics and biomechanics.

References

[1] Mark Ainsworth and John Tinsley Oden. *A Posterior Error Estimation in Finite Element Analysis*. Wiley, 2000.

[2] Hans Wilhelm Alt. *Lineare Funktionalanalysis*. Springer Verlag, 1985.

[3] Amira. Amira visualization and modeling system . `http://www.AmiraVis.com`, 2003.

[4] Peter Bastian, Klaus Birken, Klaus Johannsen, Stefan Lang, Nicolas Neuß, Henrik Rentz–Reichert, and Christian Wieners. UG – a flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science*, 1:27–40, 1997.

[5] Jürgen Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.

[6] Folkmar Bornemann, Bodo Erdmann, and Ralf Kornhuber. Adaptive multilevel–methods in three space dimensions. *Int. J. Numer. Methods in Engng.*, 36:3187–3203, 1993.

[7] Dietrich Braess. *Finite Elements*. Cambridge University Press, 2001.

[8] James Bramble. *Multigrid methods*, volume 294 of *Pitman Research Notes in Mathematics Series*. Longman Scientific, 1993.

[9] Peter J.C. Brown and Christopher T. Faigle. A robust efficient algorithm for point location in triangulations. Technical report, Cambridge University, February 1997.

[10] Philippe G. Ciarlet. *Mathematical Elasticity; Volume 1: Three–Dimensional Elasticity*, volume 20 of *Studies in Mathematics and its Applications*. North–Holland, Amsterdam, 1988.

[11] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, 1998.

[12] Peter Deuflhard and Martin Seebass. Adaptive multilevel FEM as decisive tools in the clinical cancer therapy hyperthermia. In *Proc. Eleventh International Conference on: Domain Decomposition Methods in Sciences and Engineering*, pages 403–414, 1999.

[13] Willy Dörfler and Martin Rumpf. An adaptive strategy for elliptic problems including a posteriori controlled boundary approximation. *Mathematics of Computation*, 67(224):1361–1382, 1998.

[14] Tom Duchamp, A. Certain, Antony DeRose, and Werner Stuetzle. Hierarchical computation of PL harmonic embeddings. Technical report, University of Washington, July 1997.

[15] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. *Computer Graphics*, 29(Annual Conference Series):173–182, 1995. URL `citeseer.nj.nec.com/eck95multiresolution.html`.

[16] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.

[17] Michael S. Floater. Parametrization and smooth approximations of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[18] Theodore Frankel. *The Geometry of Physics*. Cambridge University Press, 1997.

[19] Lori Freitag. *Users Manual for Opt-MS: Local Methods for Simplicial Mesh Smoothing and Untangling*. Argonne National Laboratory, Illinois, March 1999.

[20] Michael Garland and Paul Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH*, pages 209–218, 1997.

[21] Igor Guskov, Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Hybrid meshes. under revision, 2002.

[22] Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. Normal meshes. In *Proceedings of SIGGRAPH*, 2000.

[23] Wolfgang Hackbusch. *Multigrid Methods and Applications*, volume 4 of *Computational Mathematics*. Springer–Verlag, Berlin, 1985.

[24] Patrick M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. I: A framework for surface mesh optimization. *Int. J. Numer. Methods Eng. 48*, 48(3):401–420, 2000.

[25] Patrick M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. II: A framework for volume mesh optimization and the condition number of the Jacobian matrix. *Int. J. Numer. Methods Eng. 48*, 48(8):1165–1185, 2000.

[26] Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. A general framework for mesh decimation. In *Graphics Interface Proceedings*, pages 43–50, 1998.

[27] Rolf Krause. *Monotone Multigrid Methods for Signorini's Problem with Friction*. PhD thesis, FU Berlin, 2000. http://www.diss.fu-berlin.de/2001/240/indexe.html.

[28] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. MAPS: Multiresolution adaptive parametrization of surfaces. In *Proceedings of SIGGRAPH*, pages 95–104, 1998.

[29] John Michael Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, University of Washington, Seattle, 1993.

[30] Makoto Maruya. Generating a texture map from object-surface texture data. *Computer Graphics Forum*, 14(3):397–406, 1995.

[31] Colm Ó'Dúnlaing, Colum Watt, and David Wilkins. Homeomorphism of 2-complexes is equivalent to graph isomorphism. *International Journal of Computational Geometry and Applications*, 10(5):453–476, 2000.

[32] Emil Praun, Wim Sweldens, and Peter Schröder. Consistent mesh parametrizations. In *Proceedings of SIGGRAPH*, 2001.

[33] Kari Pulli and Mark Segal. Fast Rendering of Subdivision Surfaces. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 61–70, New York, NY, 1996. Springer-Verlag/Wien.

[34] Martin Rumpf. A variational approach to optimal meshes. *Numer. Math.*, 72(4):523–540, 1996.

[35] Oliver Sander. Constructing boundary and interface parametrizations for finite element solvers. Master's thesis, Institute of Computer Science, Freie Universität Berlin, 2001.

[36] F. Smith, Petter Bjørstad, and William Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[37] Detlev Stalling, Martin Seebass, Malte Zöckler, and Hans-Christian Hege. Hyperthermia treatment planning with HyperPlan – User's Manual. Technical Report ZR 00-27, Konrad-Zuse-Zentrum für Informationstechnik, October 2000.

[38] Rüdiger Verführt. *A review of a posteriori error estimation and adaptive mesh-refinement techniques*. Series Advances in Numerical Mathematics. Wiley-Teubner, Chichester: John Wiley & Sons. Stuttgart: B. G. Teubner, 1996.

[39] Stefan Zachow. 3D-Osteotomieplanung in der Mund-, Kiefer- und Gesichtschirurgie unter Berücksichtigung der räumlichen Weichgewebeanordnung. Work in progress, Zuse Institute Berlin (ZIB), 2003.

[40] Denis Zorin and Peter Schröder. Subdivision for modeling and animation. SIGGRAPH Course Notes, 2000.

[41] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. *Computer Graphics*, 31(Annual Conference Series): 259–268, 1997.