Abstract

In this paper we review the novel meccano method. We summarize the main stages (subdivision, mapping, optimization) of this automatic tetrahedral mesh generation technique and we concentrate the study to complex genus-zero solids. In this case, our procedure only requires a surface triangulation of the solid. A crucial consequence of our method is the volume parametrization of the solid to a cube. We construct volume T-meshes for isogeometric analysis by using this result. The efficiency of the proposed technique is shown with several examples. A comparison between the meccano method and standard mesh generation techniques is introduced.

Keywords: Tetrahedral mesh generation, adaptive refinement/derefinement, nested meshes, mesh smoothing, mesh untangling, surface and volume parametrization.

1 Introduction

Many authors have devoted great effort to solving the automatic mesh generation problem in different ways [4, 9, 21, 22, 37, 38, 39, 41], but the 3-D problem is still open [2, 3]. In the past, the main objective has been to achieve high quality adaptive meshes of complex solids with minimal user intervention and low computational cost. At present, it is well known that most mesh generators are based on Delaunay triangulation and advancing front technique. However, problems related to mesh quality, mesh adaptation and mesh conformity with the solid boundary, still remain.

We have recently introduced the meccano technique in [6, 7, 33, 34] for constructing adaptive tetrahedral meshes of solids. The method requires a surface triangulation of the solid, a meccano and a tolerance that fixes the desired approximation of the solid surface. The name of the method stems from the fact that the process starts from an outline of the solid, i.e. a meccano composed by connected polyhedral pieces.

The method builds a 3-D triangulation of the solid as a deformation of an appropriate tetrahedral mesh of the meccano. The meccano can be made up of different types of pieces (cuboids, pyramids, prisms, etc). A particular case is a meccano consisting only of connected cubes, i.e. a polycube [28, 40, 44].

The main idea of the new mesh generator is to combine an automatic parametrization of surface triangulations [14], a local refinement algorithm for 3-D nested triangulations [26] and a simultaneous untangling and smoothing procedure [10]. Those previous procedures (mapping, refinement, untangling and smoothing) are not in themselves new, but the overall integration is an original contribution.

In this paper, we present significant advances in the method. We define an automatic parametrization of a solid surface triangulation to the meccano boundary. For this purpose, we first divide the surface triangulation into patches with the same topological connection as the meccano faces. Then, a discrete mapping from each surface patch to the corresponding meccano face is constructed by using the parameterization of surface triangulations proposed in [14, 15, 16, 17]. The shape-preserving parametrizations, which are planar triangulations on the meccano faces, are the solutions of linear systems based on convex combinations. Specifically, we describe the procedure for a solid whose boundary is a surface of genus 0; i.e. a surface that is homeomorphic to the surface of a sphere. In this case, the meccano is a single cube, and the global mapping is the combination of six patch-mapping. The solution to several compatibility problems on the cube edges will be discussed.

The extension to more general solids is possible if the construction of an appropriate meccano is assumed. In the near future, more effort should be made in an automatic construction of the meccano when the genus of the solid surface is greater than zero. Currently, several authors are working on this aspect in the context of polycube-maps, see for example [28, 40, 44]. They are analyzing how to construct a polycube for a generic solid and, simultaneously, how to define a conformal mapping between the polycube boundary and the solid surface. Although surface parametrization has been extensively studied in the literature, only a few works deal with volume parametrization and this problem is still open. A meshless procedure is presented in [27] as one of the first tentative approaches to solve the problem. In addition, [18] gives a simple counterexample to show that convex combination mappings over tetrahedral meshes are not necessarily one-to-one. Our method automatically obtains a volume parametrization of the solid. In this paper we introduce this result for isogeometric analysis [3, 8].

In the following section we present a brief description of the main stages of the method for a generic meccano composed of polyhedral pieces. In Section 3 we analyze the algorithm in the case that the meccano is formed by a simple cube. In Section 4 we show test problems and practical applications that illustrate the efficiency of this strategy. Finally, the conclusions and future research are presented in Section 5.

2 The Algorithm of the Meccano Method

The main steps of the *meccano tetrahedral mesh generation algorithm* are summarized in this section. A first approach of this method can be found in [33, 6, 7, 34]. The input data of the algorithm are the definition of the solid boundary (for example a surface triangulation) and a given precision (corresponding to the approximation of the solid boundary). The following algorithm describes the mesh generation approach.

Meccano tetrahedral mesh generation algorithm

1. Construct a meccano approximation of the 3-D solid formed by polyhedral pieces.
2. Define an admissible mapping between the meccano boundary faces and the solid boundary.
3. Construct a coarse tetrahedral mesh of the meccano.
4. Generate a local refined tetrahedral mesh of the meccano, such that the mapping (according step 2) of the meccano boundary triangulation approximates the solid boundary for a given precision.
5. Move the boundary nodes of the meccano to the solid surface according to the mapping defined in 2.
6. Relocate the inner nodes of the meccano.
7. Optimize the tetrahedral mesh by applying the simultaneous untangling and smoothing procedure.

The first step of the procedure is to construct a meccano approximation by connecting different polyhedral pieces. The meccano and the solid must be equivalent from a topological point of view, i.e., their surfaces must have the same genus. Once the meccano is assembled, we have to define an *admissible* one-to-one mapping between the boundary faces of the meccano and the boundary of the solid. In step 3, the meccano is decomposed into a coarse tetrahedral mesh by an appropriate subdivision of its initial polyhedral pieces. This mesh is locally refined and its boundary nodes are *virtually* mapped to the solid surface until it is approximated to within a given precision. Then, we construct a mesh of the domain by mapping the boundary nodes from the meccano faces to the true surface and by relocating the inner nodes at a *reasonable* position. After those two steps, the resulting mesh is generally tangled. Therefore, a simultaneous untangling and smoothing procedure is applied and a valid adaptive tetrahedral mesh of the solid is obtained.

3 The Meccano Method for Genus-Zero Solids

In this section, we present the application of the meccano algorithm in the case of the solid surface being genus-zero and the meccano being formed by one cube. We assume a triangulation of the solid surface as a datum. We introduce an automatic parametrization between the surface triangulation of the solid and the cube boundary. To that end, we divide the surface triangulation into six patches, with the same topological connection that cube faces, so that each patch is mapped to a cube face.

We note that the meccano method constructs a high quality tetrahedral mesh even when the initial triangulation has poor quality.

3.1 Meccano

A simple cube, $\mathcal{C}$, is defined as meccano. We associate a planar graph, $\mathcal{G}_{\mathcal{C}}$, to the meccano in the following way:

- Each face of the meccano corresponds to a vertex of the graph.
- Two vertices of the graph are connected if their corresponding meccano faces share an edge.

Figure 1 shows the numbering of cube faces and their connectivities, and Figure 2 represents the corresponding planar graph.

The position of the cube is crucial to define an *admissible* mapping between the cube and solid boundary, as we analyze later. However, its size is less important, because it only affects the efficiency of the mesh optimization step. If the center of the cube is placed inside the solid, the existence of an admissible mapping is ensured.

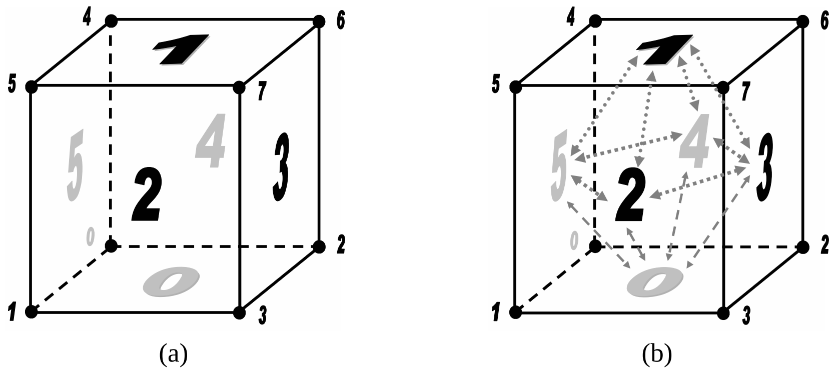


Figure 1: Meccano formed by one cube: (a) notation of nodes and faces of the cube and (b) connectivities of faces

3.2 Mapping from Cube Faces to Solid Surface Patches

Once the cube is fixed, we have to determine a mapping between the cube faces and the solid surface triangulation. First, we define the concept of admissible mapping for a cube. Let $\Sigma_{\mathcal{C}}$ be the boundary of the cube and $\Sigma_{\mathcal{S}}$ the boundary of the solid,

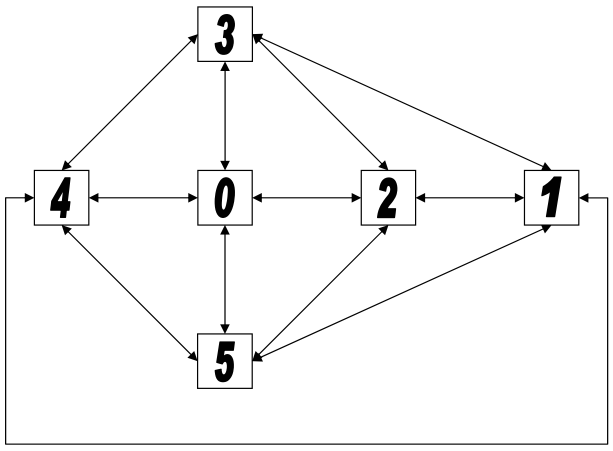


Figure 2: Planar graph $\mathcal{G}_{\mathcal{C}}$ associated to the cube

given by a surface triangulation $\mathcal{T}_{\mathcal{S}}$. We denote by $\Sigma_{\mathcal{C}}^{i}$ the i-*th* face of the cube, i.e. $\Sigma_{\mathcal{C}} = \bigcup_{i=0}^{5} \Sigma_{\mathcal{C}}^{i}$. Let $\Pi : \Sigma_{\mathcal{C}} \to \Sigma_{\mathcal{S}}$ be a piecewise function, such that $\Pi_{|\Sigma_{\mathcal{C}}^{i}} = \Pi^{i}$ where $\Pi^{i} : \Sigma_{\mathcal{C}}^{i} \to \Pi^{i}(\Sigma_{\mathcal{C}}^{i}) \subset \Sigma_{\mathcal{S}}$. Then, Π is called an *admissible* mapping if it satisfies:

a) Functions $\{\Pi^{i}\}_{i=0}^{5}$ are compatible on $\Sigma_{\mathcal{C}}$. That is $\Pi^{i}_{|\Sigma_{\mathcal{C}}^{i} \cap \Sigma_{\mathcal{C}}^{j}} = \Pi^{j}_{|\Sigma_{\mathcal{C}}^{j} \cap \Sigma_{\mathcal{C}}^{i}}$, $\forall i, j = 0, \ldots, 5$, with $i \neq j$ and $\Sigma_{\mathcal{C}}^{i} \cap \Sigma_{\mathcal{C}}^{j} \neq \emptyset$.

b) Global mapping Π is continuous and bijective between $\Sigma_{\mathcal{C}}$ and $\Sigma_{\mathcal{S}}$.

Note that admissible mapping is not unique. We define an automatic admissible mapping in the following sections. For this purpose, we first construct a partition of the solid surface triangulation into six patches, maintaining the topology of the graph of the Figure 2, and then we parametrize each patch to a cube face.

3.2.1 Partition of the Solid Surface Triangulation

In the following, we call *connected subtriangulation* a set of triangles of $\mathcal{T}_{\mathcal{S}}$ whose interior is a connected set. Given a decomposition of the surface triangulation $\mathcal{T}_{\mathcal{S}}$ in any set of connected subtriangulations, we can associate a planar graph, $\mathcal{G}_{\mathcal{S}}$, to this partition in the following way:

- Each subtriangulation corresponds to a vertex of the graph.
- Two vertices of the graph are connected if their corresponding subtriangulations have at least one common edge.

We say that a solid surface partition and the meccano are *compatible* if their graphs are isomorphic, $\mathcal{G}_{\mathcal{S}} = \mathcal{G}_{\mathcal{C}}$. In our case, since the solid surface is isomorphic to a sphere, it is clear that a compatible partition exists.

We now propose an algorithm to obtain a decomposition of the given solid surface triangulation $\mathcal{T}_{\mathcal{S}}$ into six subtriangulations $\{\mathcal{T}_{\mathcal{S}}^{i}\}_{i=0}^{5}$. We distinguish three steps:

a) *Subdivision in connected subtriangulations*. We construct the Voronoi diagram associated to the centers of the six cube faces. We consider that a triangle $F \in \mathcal{T}_{\mathcal{S}}$ belongs to the i*-th* Voronoi cell if its barycenter is inside this cell. We generate a partition of $\mathcal{T}_{\mathcal{S}}$ in maximal connected subtriangulations with this criterion, i.e. two subtriangulations belonging to the same cell can not be connected. We denote as $\mathcal{T}_{\mathcal{S}}^{ij}$ the j*-th* connected subtriangulation belonging to the i*-th* Voronoi cell, and n_i is the total number of subtriangulation in the i*-th* cell. The number of subtriangulation depends only on the position of the cube center. If this point is placed inside the solid and the surface triangulation is fine enough, there is a subtriangulation in each cell. If any $n_i = 0$ the process is aborted and the center of the meccano must be modified. For a complex solid the value of n_i is usually greater than 1, therefore, a modification of the partition is necessary to obtain a compatible decomposition of $\mathcal{T}_{\mathcal{S}}$.

b) *Construction of the graph*. We associate a planar graph, $\mathcal{G}_{\mathcal{S}}$ to the partition generated in the previous step. If the center of the cube is inside the solid and the surface triangulation is fine enough, there is one compatible subtriangulation for each Voronoi cell, i.e. there is one *head* subtriangulation $\mathcal{T}_{\mathcal{S}}^{i0}$, vertex of the graph $\mathcal{G}_{\mathcal{S}}$, with the same connection as the vertex associated to the i*-th* cube face in $\mathcal{G}_{\mathcal{C}}$. In other case, the subtriangulation with the greatest number of elements is chosen as $\mathcal{T}_{\mathcal{S}}^{i0}$.

c) *Reduction of the graph*. In order to achieve a decomposition of $\mathcal{T}_{\mathcal{S}}$, we propose an iterative procedure to reduce the current graph $\mathcal{G}_{\mathcal{S}}$. In each step all triangles of $\mathcal{T}_{\mathcal{S}}^{jk}$ are included in the head subtriangulation $\mathcal{T}_{\mathcal{S}}^{i0}$ if:

   – $\mathcal{T}_{\mathcal{S}}^{i0}$ is the head subtriangulation with the fewest triangles.
   – $\mathcal{T}_{\mathcal{S}}^{jk}$ and $\mathcal{T}_{\mathcal{S}}^{i0}$ are connected.
   – k is higher than zero.

   Then, the vertex $\mathcal{T}_{\mathcal{S}}^{jk}$ is removed from the graph and its connectivities are inherited by $\mathcal{T}_{\mathcal{S}}^{i0}$. The connectivity of the graph is updated.

   After this process, $\mathcal{T}_{\mathcal{S}}^{i0}$ could be connected to other subtriangulations $\mathcal{T}_{\mathcal{S}}^{il}$ of the same i*-th* cell. In this case, the triangles of all $\mathcal{T}_{\mathcal{S}}^{il}$ are included in $\mathcal{T}_{\mathcal{S}}^{i0}$, the graph vertices $\mathcal{T}_{\mathcal{S}}^{il}$ are removed from the graph, their connectivities are inherited and the graph connectivities are updated. Therefore, the connected subtriangulations are always maximal in all algorithm steps.

   This procedure continues iteratively until the graph comprises only six head vertices.

Although the proposed algorithm obtains a six vertex graph $\mathcal{G}_{\mathcal{S}}$, its compatibility with the cube graph $\mathcal{G}_{\mathcal{C}}$ can not be ensured. As the computational cost of this algorithm is low, a movement in the cube center in order to obtain a compatible partition $\{\mathcal{T}_\mathcal{S}^i\}_{i=0}^5$, does not affect the efficiency of the meccano technique. In fact, this procedure could be guided by a genetic algorithm [43].

In what follows $\Sigma_\mathcal{S}^i$ is the solid surface patch defined by the triangles of $\mathcal{T}_\mathcal{S}^i$.

3.2.2 Parametrization of the Solid Surface Triangulation

Once the given solid surface $\Sigma_\mathcal{S}$ is decomposed into six patches $\Sigma_\mathcal{S}^0, \ldots, \Sigma_\mathcal{S}^5$, we map each surface patch $\Sigma_\mathcal{S}^i$ to the corresponding cube face $\Sigma_\mathcal{C}^i$ by using the parametrization of surface triangulations proposed by M. Floater; in [14] the author gives a method to compute a planar triangulation on a convex domain that is isomorphic to a simply connected surface triangulation. In our case the convex domain is a cube face $\Sigma_\mathcal{C}^i$, and the surface triangulation is $\mathcal{T}_\mathcal{S}^i$. Then, we define

$$\left(\Pi^i\right)^{-1} : \Sigma_\mathcal{S}^i \to \Sigma_\mathcal{C}^i$$

as the parametrization introduced by Floater, and we denote $\tau_F^i = \left(\Pi^i\right)^{-1}\left(\mathcal{T}_\mathcal{S}^i\right)$ as the planar triangulation of $\Sigma_\mathcal{C}^i$ associated to $\mathcal{T}_\mathcal{S}^i$. To obtain τ_F^i, Floater parametrization fixes their boundary nodes and the position of their inner nodes is given by the solution of a linear system based on convex combinations. Let $\{P_1^i, \ldots, P_n^i\}$ be the inner nodes and $\{P_{n+1}^i, \ldots, P_N^i\}$ be the boundary nodes of $\mathcal{T}_\mathcal{S}^i$, respectively, where N denotes the total number of nodes of $\mathcal{T}_\mathcal{S}^i$. Once the boundary nodes $\{Q_{n+1}, \ldots, Q_N\}$ of τ_F^i are fixed, the position of the inner nodes $\{Q_1^i, \ldots Q_n^i\}$ is given by the solution of the system:

$$Q_k^i = \sum_{l=1}^{N} \lambda_{kl} Q_l^i, \qquad k = 1, \ldots, n.$$

The values of $\{\lambda_{kl}\}_{k=1,\ldots,n}^{l=1,\ldots,N}$ are the weights of the convex combinations, such that

$$\begin{aligned} \lambda_{kl} &= 0, && \text{if } P_k \text{ and } P_l \text{ are not connected} \\ \lambda_{kl} &> 0, && \text{if } P_k \text{ and } P_l \text{ are connected} \\ \sum_{l=1}^{N} \lambda_{kl} &= 1, && \text{for } k = 1, \ldots n. \end{aligned}$$

In [14] three alternatives are analyzed: uniform parametrization, weighted least squares of edge lengths and shape preserving parametrization. Another choice, called mean value coordinate, is presented in [16]. The goal is to obtain an approximation of a conformal mapping.

In order to ensure the compatibility of $\{\Pi^i\}_{i=0}^5$, the boundary nodes of $\{\tau_F^i\}_{i=0}^5$ must coincide on their common cube edges. The six transformations $\{\Pi^i\}_{i=0}^5$ define an admissible mapping between $\Sigma_\mathcal{C}$ and $\Sigma_\mathcal{S}$, i.e. the cube boundary triangulation $\tau_F = \bigcup_{i=0}^5 \tau_F^i$ is a global parametrization of the solid surface triangulation $\mathcal{T}_\mathcal{S}$.

Two important properties of mapping Π are:

(a) the triangulations τ_F and $\mathcal{T}_S$ have the same topology,

(b) each triangle of τ_F is completely contained in one face of the cube.

We note that usual polycube-maps [40, 28] verify property (a), but they do not verify property (b), i.e., a triangle belonging to $\mathcal{T}_S$ can be transformed by a polycube-map into a *triangle* whose vertices are placed on different faces of the polycube.

The proposed mapping Π is used in a following step of the meccano algorithm to map a new triangulation τ_K (obtained on $\Sigma_{\mathcal{C}}$ by application of the refinement algorithm of Kossaczky [26]) to the solid boundary. Several problems can appear in the application of this transformation and their solutions will be commented on next. Basically, these problems are due to the fact that a valid triangulation $\tau_K \neq \tau_F$ on $\Sigma_{\mathcal{C}}$ can be transformed by Π into a non-valid one on the solid surface.

3.3 Coarse Tetrahedral Mesh of the Meccano

We build a coarse and high quality tetrahedral mesh by splitting the cube into six tetrahedra [26]. For this purpose, it is necessary to define a main diagonal on the cube and corresponding diagonal on its faces, see Figure 3(a). The resulting mesh can be recursively and globally bisected [26] to fix a uniform element size in the whole mesh. Three consecutive global bisections for a cube are presented in Figures 3 (b), (c) and (d). The resulting mesh of Figure 3(d) contains 8 cubes similar to the one shown in Figure 3(a). Therefore, the recursive refinement of the cube mesh produces similar tetrahedra to the initial ones.

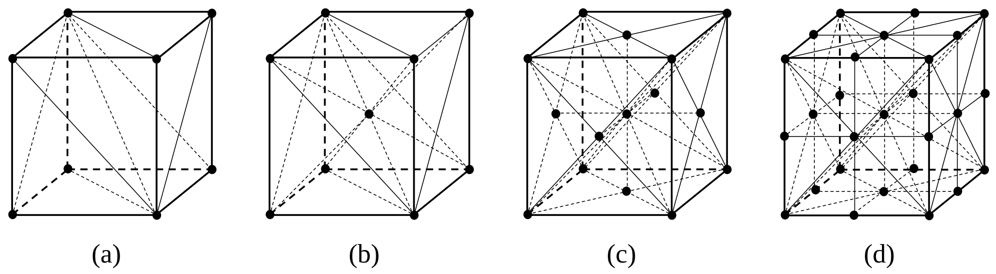


Figure 3: Refinement of a cube by using Kossaczky's algorithm: (a) cube subdivision into six tetrahedra, (b) bisection of all tetrahedra by inserting a new node in the cube main diagonal, (c) new nodes in diagonals of cube faces and (d) global refinement with new nodes in cube edges

3.4 Local Refined Tetrahedral Mesh of the Meccano

The next step in the meccano mesh generator includes a recursive adaptive local refinement strategy, by using Kossaczky's algorithm [26], of those tetrahedra with a face

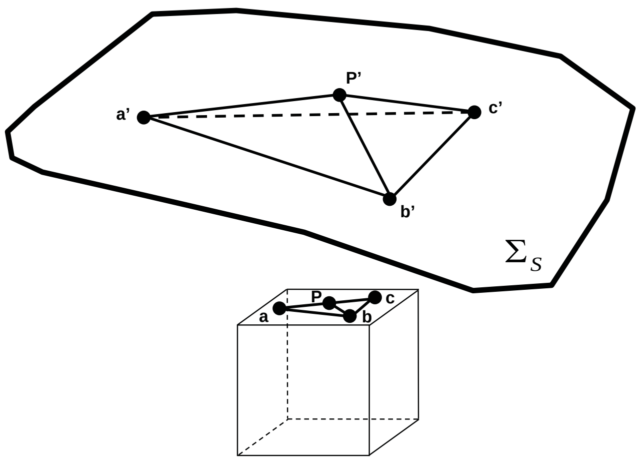


Figure 4: Mapping Π from the boundary nodes of the meccano a, b, c, P to the points a', b', c', P' of the solid surface

placed on a boundary face of the initial coarse tetrahedral mesh of the cube. The refinement process is done in such a way that the given solid surface triangulation $\mathcal{T}_S$ is approximated by a new triangulation within a given precision. That is, we seek an adaptive triangulation τ_K on the cube boundary $\Sigma_{\mathcal{C}}$, so that the resulting triangulation after node mapping $\Pi(\tau_K)$ is a good approximation of the solid boundary. The user has to introduce as input data a parameter ε, which is a tolerance to measure the separation allowed between the linear piecewise approximation $\Pi(\tau_K)$ and the solid surface defined by the triangulation $\mathcal{T}_{\mathcal{S}}$. At present, we have considered two criteria: the first related to the Euclidean distance between both surfaces and the second attending to the difference in terms of volume.

To illustrate these criteria, let abc be a triangle of τ_K placed on the *meccano* boundary, and $a'b'c'$ the resulting triangle of $\Pi(\tau_K)$ after mapping the nodes a, b and c on the given solid surface $\Sigma_{\mathcal{S}}$, see Figure 4. We define two different criteria to decide whether it is necessary to refine the triangle (and consequently the tetrahedron containing it) in order to improve the approximation.

For any point Q in the triangle abc, we define $d_1(Q)$ as the euclidean distance between the mapping of Q on $\Sigma_{\mathcal{S}}$, Q', and the plane defined by $a'b'c'$. This definition is an estimate of the distance between the surface of the solid and the current piecewise approximation $\Pi(\tau_k)$.

We also introduce a measure in terms of volume and then, for any Q in the triangle abc, we define $d_2(Q)$ as the volume of the *virtual* tetrahedron $a'b'c'Q'$. In this case, $d_2(Q)$ is an estimate of the lost volume in the linear approximation by the face $a'b'c'$ of the solid surface.

The threshold of whether to refine the triangle or not is given by a tolerance ε_i fixed by the user. With the previous definition, $d_1(Q) < \varepsilon_1$ for all Q in the boundary on the cube implies that the distance between the surface of the solid and its new piecewise linear approximation $\Pi(\tau_K)$ is less than ε_1. On the other hand, $d_2(Q) < \varepsilon_2$ for all Q in the boundary on the cube would mean that the lost volume per triangle of $\Pi(\tau_K)$ is bounded by ε_2. Alternatively, ε_2 could be defined as the allowed difference of volumes and we could use an equidistribution strategy as is usual in *a-posteriori* error estimates. Nevertheless, we prefer to use here a local version of ε_2, so the difference of volumes is estimated by multiplying ε_2 by the number of triangles of the final approximation $\Pi(\tau_K)$.

Obviously, other measures could be introduced in line with the desired approximation type (curvature, point properties, etc.). What is more, the user could consider the combination of several measures simultaneously.

Once we have defined separation measures d_i and the corresponding tolerances ε_i, we propose the following strategy to achieve our objective.

Starting from the coarse tetrahedral mesh of the cube, we construct a sequence of tetrahedral nested meshes by recursive bisection of a subset of the tetrahedra that contain a face located on the cube faces. The *refinement criterion* decides whether a tetrahedron should be refined attending to the current node distribution of triangulation τ_K on the cube boundary $\Sigma_{\mathcal{C}}$ and their *virtual* mapping $\Pi(\tau_K)$ on the solid boundary $\Sigma_{\mathcal{S}}$. The separation between triangulations $\Pi(\tau_K)$ and $\Pi(\tau_F) = \mathcal{T}_{\mathcal{S}}$ is used in the refinement criterion for tetrahedron T:

Refinement criterion

Tetrahedron T is marked to be refined if it satisfies the following two conditions:

1. T has a face $F \in \tau_K$ on the cube boundary.
2. $d_i(Q) \geq \varepsilon_i$ for some node $Q \in \tau_F$ located on face F of T.

From a numerical point of view, the number of points Q (analyzed in this strategy) is reduced to the set of nodes of the triangulation τ_F (defined by the parametrization of Floater) that are contained in face F. The analysis must be done every time that a face is subdivided into its two *son* faces. The subdivision criterion should stop for a particular face when all the nodes of τ_F included on this face have been analyzed and all of them verify the approximation criterion. This aspect has the inconvenience that each node of τ_F could be studied many times, but we use the nested mesh *genealogy* to implement the refinement criterion efficiently.

Finally, the *refinement procedure* for constructing a local refined tetrahedral mesh of the meccano is summarized in the following algorithm:

Refinement procedure

1. Given the coarse tetrahedral mesh of the meccano.
2. Set a tolerance ε_i.
3. Do

```
    (a) Mark for refinement all tetrahedra that satisfy the *refinement
        criterion* for a distance $d_i$ and a tolerance $\varepsilon_i$.
    (b) Refine the mesh.
While any tetrahedron $T$ is marked.
```

We denote by n_b the number of levels of the nested tetrahedral mesh sequence and τ_K the resulting triangulation of the cube boundary associated to the finest level of the sequence. We note that the refinement procedure automatically concludes according to a single parameter, i.e. ε_i.

We introduced in [34] another method for obtaining τ_K based on the application of a derefinement procedure (which is a generalization of the strategy developed in [13]) to a sequence of tetrahedral nested meshes. In this case, the number of bisections n_b was determined by the user as a function of the desired resolution. We have discarded this strategy because it could lead to problems with memory requirements if any nodes of τ_F are very close.

We note that other local refinement algorithms, see for example [23, 29, 35], can be used in the meccano method. However, the Kossaczky's algorithm [26] has several important advantages related with simplicity and mesh quality.

3.5 External Node Mapping on Solid Boundary

Once we have defined the local refined tetrahedral mesh by using the method proposed in the previous section, the nodes of the triangulation τ_K are mapped to the solid surface. Therefore, the triangulation $\Pi(\tau_K)$ is the new approximation of the solid surface.

After this process, due to the properties of Floater's parametrization, $\Pi(\tau_K)$ is generally a valid triangulation. However, unacceptable triangulations can appear. We comment on this problem and its solution in Section 3.8.

In addition, a tangled tetrahedral mesh is generated because the position of the inner nodes of the cube tetrahedral mesh has not changed.

3.6 Relocation of Inner Nodes

Even if $\Pi(\tau_K)$ is an acceptable triangulation, an optimization of the solid tetrahedral mesh is necessary. Since it is better that the optimization algorithm starts from a mesh with as good a quality as possible, we propose to relocate the inner nodes of the cube tetrahedral mesh in a reasonable position before the mesh optimization.

Although this node movement does not solve the tangle mesh problem, it normally reduces it. In other words, the resulting number of inverted elements is lower and the mean quality of valid elements is greater.

There would be several strategies for defining an appropriate position for each inner node of the cube mesh. The relocation procedure should modify their relative position as a function of the solid surface triangulation before and after their mapping Π. However, an ideal relocation of inner nodes requires a volume mapping from the cube to the complex solid. Obviously, this information is not known *a priori*. In fact, we will reach an approximation of this volume mapping at the end of the mesh generation. Therefore, an interesting idea is to use an specific discrete volume mapping that is defined by the transformation between a cube tetrahedral mesh and the corresponding solid tetrahedral mesh. A point P that belongs to tetrahedron T of the cube is mapped to point P' of the tetrahedron T' of the solid with the same barycentric coordinates.

If we construct a volume mapping once, we can use it for meshing the same solid with different grades of discretization, i.e. different values of tolerance ε. In practice, a good strategy is to mesh the solid by using a high value of ε (a coarse tetrahedral mesh of the solid is obtained) and gradually decrease ε. In the first step of this strategy, no relocation is applied. In this case, the number of nodes of the resulting mesh is low and the mesh optimization algorithm is fast. In the following steps a relocation of inner nodes is applied by using the mapping that is defined by the previous iteration.

3.7 Solid Mesh Optimization: Untangling and Smoothing

The proposed relocation procedure, based on volumetric parametrization, is efficient but does not solve the tangling problem completely. Therefore, it is necessary to optimize the current mesh. This process must be able to smooth and untangle the mesh and is crucial in the proposed mesh generator.

The most usual techniques to improve the quality of a *valid* mesh, that is, a mesh with no inverted elements, are based upon local smoothing. In short, these techniques consist of finding the new positions that the mesh nodes must hold, in such a way that they optimize an objective function. Such a function is based on a certain measurement of the quality of the *local submesh*, $N(v)$, formed by the set of elements connected to the *free node* v, whose coordinates are given by $\mathbf{x}$. We have considered the following objective function derived from an *algebraic mesh quality metric* studied in [25],

$$K(\mathbf{x}) = \left[\sum_{m=1}^{M} \left(\frac{1}{q_{\eta_m}}\right)^p (\mathbf{x})\right]^{\frac{1}{p}}$$

where M is the number of elements in $N(v)$, q_{η_m} is an algebraic quality measure of the m-th element of $N(v)$ and p is usually chosen as 1 or 2. Specifically, we have considered the mean ratio quality measure, which for a tetrahedron is $q_\eta = \frac{3\sigma^{\frac{2}{3}}}{|S|^2}$ and for a triangle is $q_\eta = \frac{2\sigma}{|S|^2}$, $|S|$ being the Frobenius norm of matrix S associated to the affine map from the *ideal* element (usually equilateral tetrahedron or triangle) to the physical one, and $\sigma = \det(S)$. Other algebraic quality measures can be used as, for example, the metrics based on the condition number of matrix S, $q_\kappa = \frac{\rho}{|S||S^{-1}|}$, where $\rho = 2$ for triangles and $\rho = 3$ for tetrahedra. It would also be possible to use other objective functions that have barriers like those presented in [24].

We have proposed in [10] an alternative to the procedure of [19, 20], so the untangling and smoothing are carried out in the same stage. For this purpose, we use a suitable modification of the objective function such that it is regular all over $\mathbb{R}^3$. It consists of substituting the term σ in the quality metrics with the positive and increasing function $h(\sigma) = \frac{1}{2}(\sigma + \sqrt{\sigma^2 + 4\delta^2})$. When a feasible region (subset of $\mathbb{R}^3$ where v could be placed, $N(v)$ being a valid submesh) exists, the minima of the original and modified objective functions are very close and, when this region does not exist, the minimum of the modified objective function is located in such a way that it tends to untangle $N(v)$. With this approach, we can use any standard unconstrained optimization method [1] to find the minimum of the modified objective function.

In addition, a smoothing and aligning of the boundary surface triangulation could be applied before the movement of inner nodes of the domain by using the new procedure presented in [11, 12] and [32]. This surface triangulation smoothing and aligning techniques are also based on a vertex repositioning defined by the minimization of a suitable objective function. The original problem on the surface is transformed into a two-dimensional one on the *parametric space*.

3.8 Comments and Remarks about the Procedure

We now discuss several problems that can singularly arise in the application of the proposed mesh generation algorithm. Basically, these problems occur due to the mapping $\Pi : \Sigma_{\mathcal{C}} \to \Sigma_{\mathcal{S}}$.

3.8.1 Dividing Edges

Floater's parametrization does not define a one-to-one mapping if there are *dividing edges* [15] in a patch triangulation $\mathcal{T}_{\mathcal{S}}^i$. A dividing edge is an interior edge of $\mathcal{T}_{\mathcal{S}}^i$ whose vertices are on the boundary of $\mathcal{T}_{\mathcal{S}}^i$, see Figure 5. A triangulation without dividing edges is called *strongly connected*. In order to obtain a strongly connected triangulation $\mathcal{T}_{\mathcal{S}}^i$ we introduce a new node in the midpoint of the dividing edge, and bisect the two triangles sharing it. We note that the boundary nodes of $\mathcal{T}_{\mathcal{S}}^i$ are not modified; therefore, this local procedure does not change the patch boundary edges.

3.8.2 Distortion of the Parametrization

As the position of the boundary nodes are fixed in Floater's parametrization, the shape-preserving property is missed and, therefore, high distortions of the parametrization can be produced, especially in regions close to patch boundaries.

In fact, we have to take into account that a valid triangulation $\tau_K \neq \tau_F$ on $\Sigma_{\mathcal{C}}$ can be transformed by Π into a non-valid one on the solid surface. Therefore the new approximation of the solid boundary $\Pi(\tau_K)$ can be erroneous locally. These unacceptable situations can produce inverted or poor quality tetrahedra. We have confirmed that most of these problems are minimal in relation to the number of mesh elements; they are

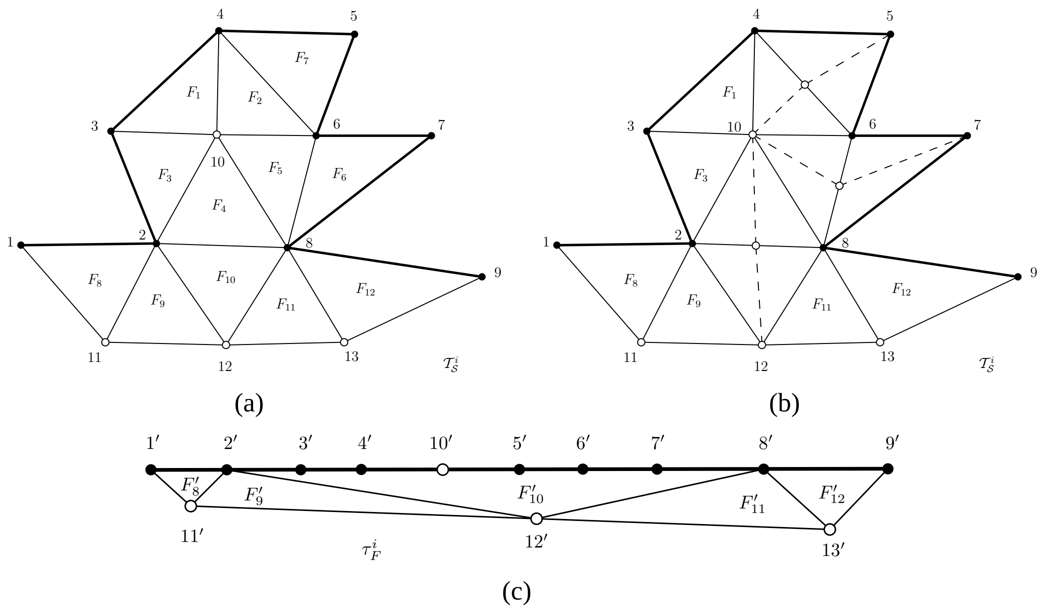


Figure 5: The edges $[2, 8]$, $[4, 6]$ and $[6, 8]$ are dividing edges (a). The set of triangles $\{F_i\}_{i=1}^{7}$ of $\mathcal{T}_S^i$ is mapped into the line segment $[2', 8']$ because of dividing edges (c). The dividing edges are avoided by bisecting triangles $\{F_2, F_4, F_5, F_6, F_7, F_{10}\}$ (b)

placed close to the boundaries of $\mathcal{T}_S^i$ and are accentuated when the patch boundaries of $\mathcal{T}_S^i$ have strong zigzag shape. In order to minimize this effect we propose several alternatives:

- Smoothing *zigzag shape* of the patch boundaries. Let us consider a simple example to explain this problem. In Figure 6 we show a parametrization of a triangulation composed of two triangles $A'_F D'_F B'_F$ and $B'_F D'_F C'_F$ of the same patch. The boundary of the patch is given by the zigzag line $[A'_F, B'_F, C'_F]$. We note that the mapping of triangulation $A_K D_K B_K$ and $B_K D_K C_K$ produces a tangled mesh in the physical space. We also note that in Figure 6 the boundary nodes of triangulation τ_K coincide with nodes of τ_F and therefore, the patch boundary is captured exactly by triangulation $\Pi(\tau_K)$. In practice, the boundary nodes of the two triangulations do not match up, which can aggravate the problem.

  A simple method to smooth the zigzag shapes is presented in Figure 7. Triangle $F \in \mathcal{T}_S^i$, which has two neighboring triangles belonging to $\mathcal{T}_S^j$, is moved to the neighboring patch $\mathcal{T}_S^j$.

- Tetrahedral mesh optimization. The situation of Figure 6 can produce tangled tetrahedra, which requires a movement of the nodes $\Pi(\tau_K)$ to obtain a valid mesh. When it occurs, we applied the optimization procedure of section 3.7 considering the boundary nodes of tangled elements as free nodes. Once the tetrahedral mesh is untangled and smoothed, the nodes of $\Pi(\tau_K)$ can be projected on surface $\mathcal{T}_S$.

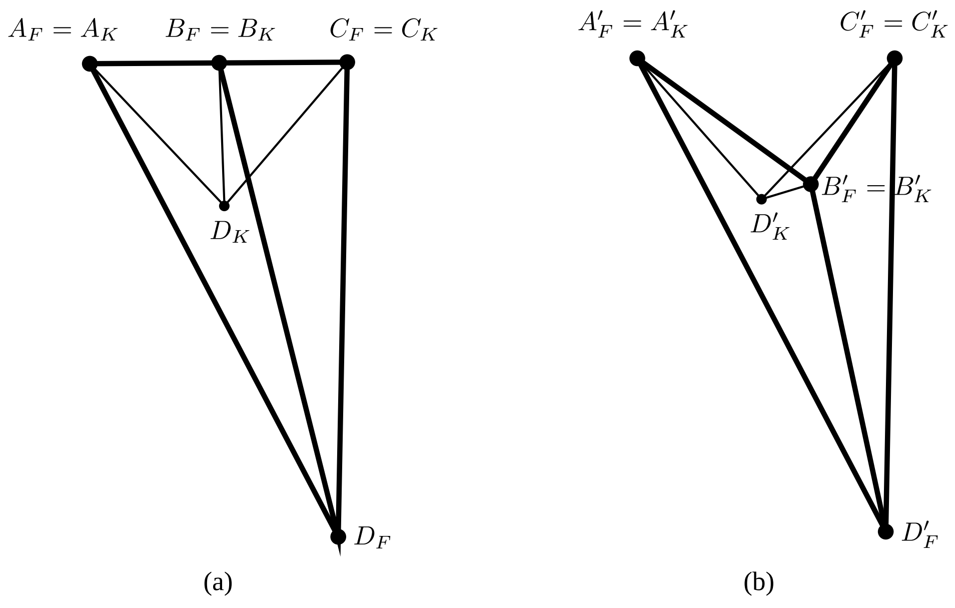


Figure 6: (a) Triangulations τ_K (bold lines) and τ_F on the parametric space and (b) triangulations $\Pi(\tau_K)$ (bold lines) and $\Pi(\tau_F)$ on the physical space. Boundary zigzag line produces an inverted triangle $D'_K C'_K B'_K$

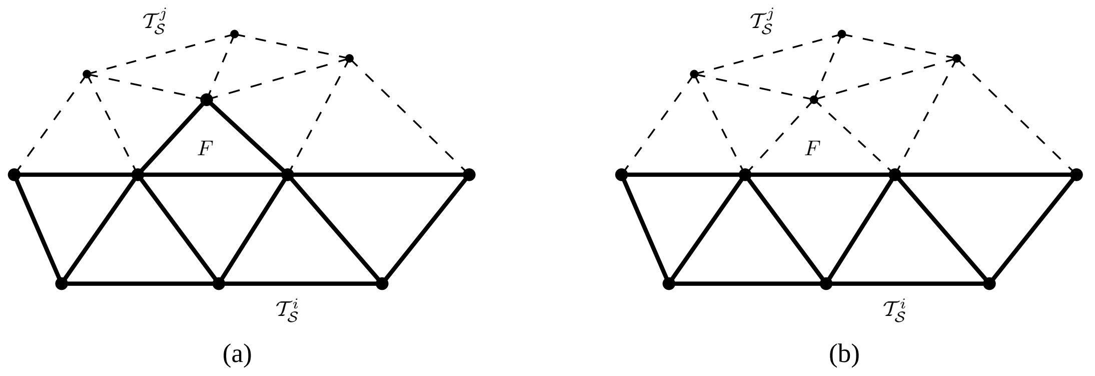


Figure 7: To smooth the interface between patches, the triangle $F \in \mathcal{T}_S^i$ is moved to patch $\mathcal{T}_S^j$

- Surface smoothing. Our algorithm can include a smoothing of surface triangulation $\Pi(\tau_k)$. The zigzag shape can produce poor quality elements close to interface between solid surface patches, which the surface smoothing improves.

We have checked that all these problems appear only when the mesh size of surface approximation $\Pi(\tau_K)$ is the same order as the mesh size of $\mathcal{T}_S$. Therefore, if a more precise approximation of the solid surface by the meccano method is required, a simple solution is to refine the given solid surface triangulation $\mathcal{T}_S$.

In the future, we will focus on improving the quality of global parametrization. One alternative is to optimize the initial parametrization τ_F, allowing the node movement between cube faces, as is usual in polycube maps applications [40, 28, 44].

4 Applications

We have implemented the meccano technique using:

- The parametrization toolbox of the geometry group at SINTEF ICT, Department of Applied Mathematics.
- The module of 3D refinement of ALBERTA code.
- Our optimization mesh procedure describes in section 3.7.

The parametrization of a surface triangulation patch $\mathcal{T}_S^i$ to a cube face Σ_C^i is done with GoTools core and parametrization modules from SINTEF ICT, available on the website *http://www.sintef.no/math_software*. This code implements Floater's parametrization in C++. Specifically, in the following applications we have used the mean value method for the parametrization of the inner nodes of triangulation, and the boundary nodes are fixed with chord length parametrization [14, 16].

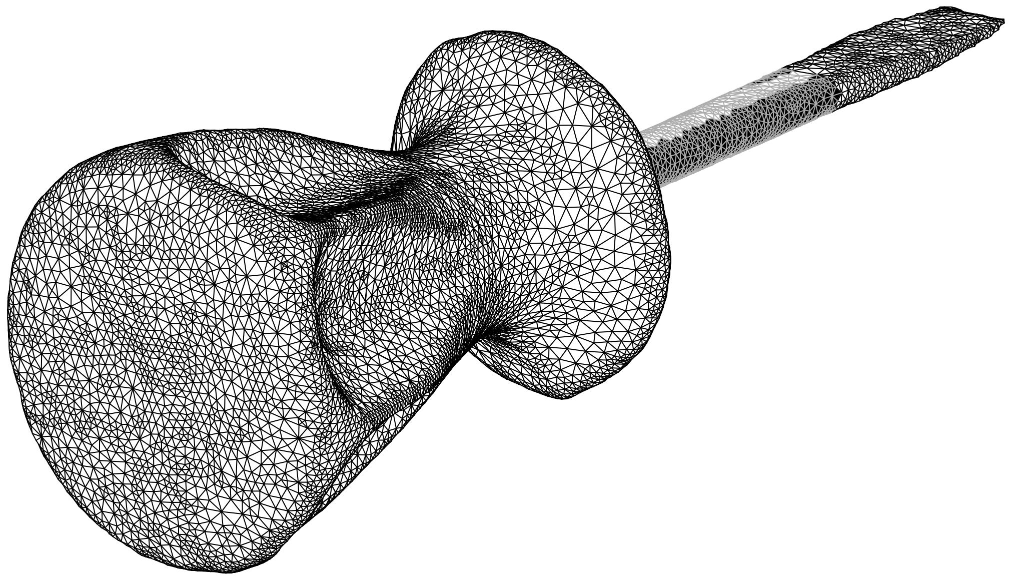
ALBERTA is an adaptive multilevel finite element toolbox [36] developed in C. This software can be used for solving several types of 1-D, 2-D or 3-D problems. ALBERTA uses the Kossaczky refinement algorithm [26] and requires an initial mesh topology [36]. The recursive refinement algorithm could not terminate for general meshes. The meccano technique constructs meshes that verify the imposed restrictions of ALBERTA relative to topology and structure. They can be refined by its recursive algorithm, because they are loop-free, and the degeneration of the resulting triangulations after successive refinements is avoided. The minimum quality of refined meshes is function of the initial mesh quality [30, 42].

The performance of our novel tetrahedral mesh generator is shown in the following applications. All of them are complex genus-zero solid that are classical test in mesh generation analysis.

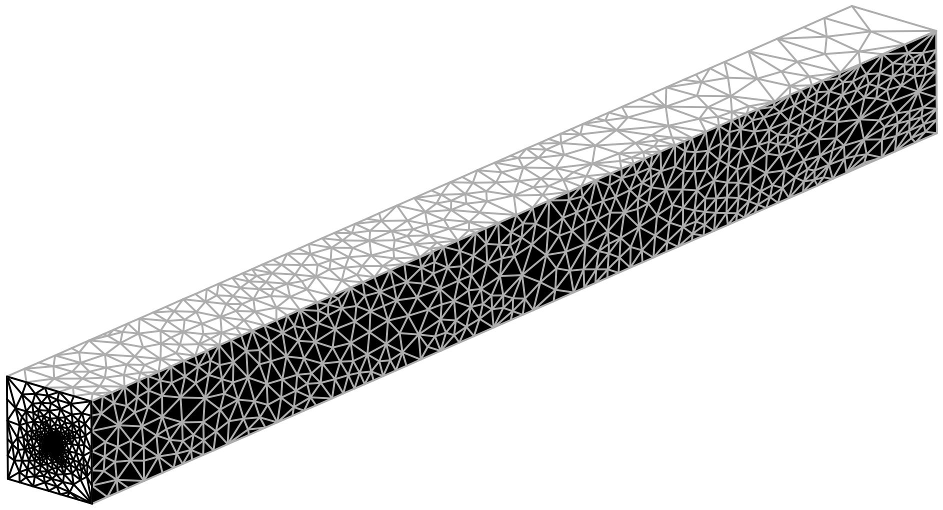
The first example corresponds to a screwdriver and the second to the armadillo. In addition, we present an application of the meccano method to construct volume T-meshes for isogeometric analysis. Particularly, we consider the volume parametrization of the Stanford bunny. We have obtained a surface triangulation of these objects from internet.

4.1 Screwdriver

The original surface triangulation of the screwdriver has been obtained from CNR IMATI Shapes Repository, *http://shapes.aim-at-shape.net*, and it is shown in Figure 8 (a). It has 27150 triangles and 13577 nodes. The bounding box of the solid is defined by the points $(x, y, x)_{min} = (-15, -26, -15)$ and $(x, y, z)_{max} = (15, 44, 15)$.



(a)



(b)

Figure 8: (a) Compatible partition of the original surface triangulation, (b) Screwdriver meccano, the Floater's parametrization is showed on the prism faces

We consider a regular prism as a meccano (see Figure 8(b)). Its dimensions are $4 \times 48 \times 4$ and its center is placed inside the solid at the point $(0, 14, 0)$. Although a unit cube could be considered as meccano, we have decided to use a narrow meccano because this choice slightly improves the quality of the final mesh. In any case, both meccanos (cube or prism) have associated the same planar graph.

We obtain an initial subdivision of screwdriver surface in six connected subtriangulations using Voronoi diagram of prism face centers. It is a compatible decomposition of the surface triangulation, according 3.2.1. We then remove the dividing edges and smooth the interfaces between patches by minimizing the zigzag shapes. Figure 8(a) shows the resulting compatible partition $\{\mathcal{T}_{\mathcal{S}}^i\}_{i=0}^5$.

We map each surface patch $\Sigma_{\mathcal{S}}^i$ to the prism faces $\Sigma_{\mathcal{C}}^i$ by using the Floater parametrization [14]. The definition of the one-to-one mapping between the cube and screwdriver boundaries is straightforward once the parametrization of the screwdriver surface triangulation is built, see Figure 8(b). A detail of the two more significant parametrizations are shown in Figure 9.

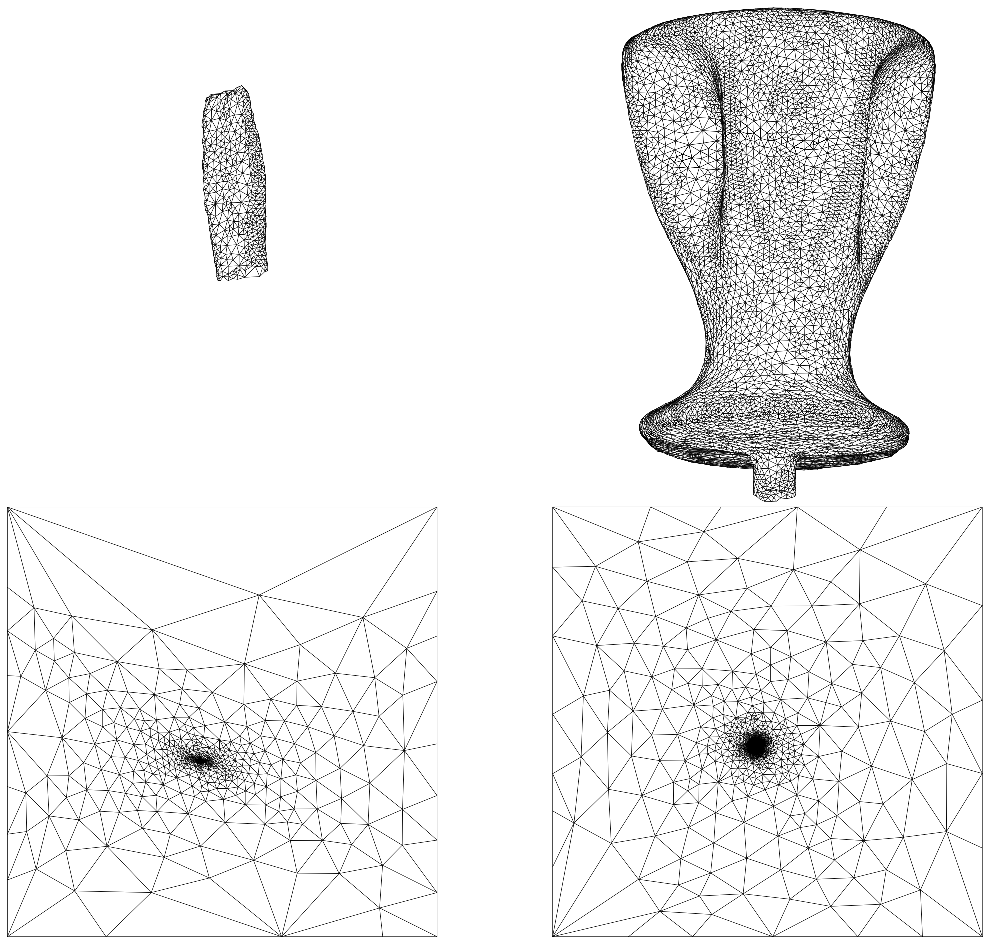

(a) (b)

Figure 9: Detail of Floater's parametrization of two screwdriver patches to square meccano faces

The prism is divided into 12 cubes, and then in 72 tetrahedra. We now fix several tolerances: $\varepsilon_2 = 1, 0.1, 0.01, 0.001$ and generate the corresponding tetrahedral meccano meshes. In Table 1 we report the main features of them. For example, for a tolerance $\varepsilon_2 = 0.001$, our method applies 51 Kossaczky recursive bisections to generate a local refined mesh that contains 168834 tetrahedra and 39617 nodes, with 36968 triangles and 18486 nodes on its boundary.

	ε_2= 1	ε_2=0.1	ε_2= 0.01	ε_2= 0.001
# nodes	1955	4430	12783	39617
# tetrahedra	8118	18814	54276	168834
# nodes on boundary	941	2047	5979	18486
# triangles on boundary	1878	4090	11954	36968
Kossaczky refinement	42	45	48	51

Table 1: Main features of the screwdriver tetrahedral meshes generated by meccano method for tolerances $\varepsilon_2 = 1, 0.1, 0.01, 0.001$

In order to obtain a tetrahedral mesh of the screwdriver we have to apply the procedures: external node mapping, relocation on inner nodes and mesh optimization.

The Floater's parametrizations allow us to map the meccano external nodes to the screwdriver surface, but the resulting tetrahedral mesh is complety tangled. For example, in Figure 10(a) we show the tangled mesh for the tolerance $\varepsilon_2 = 0.001$.

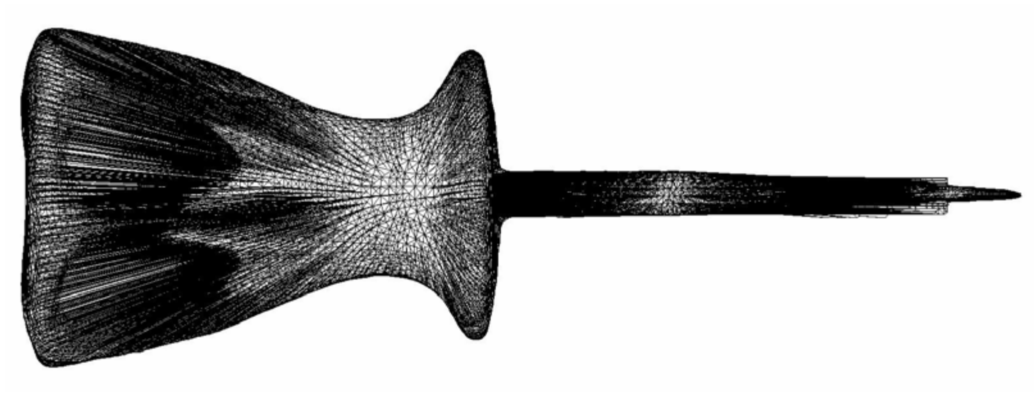
No relocation is applied for the coarser tolerance ($\varepsilon_2 = 1$), and we use the volume parametrization (from the screwdriver to the meccano) given by this coarse approximation to relocate the inner nodes in the other cases. The relocation procedure significatively reduces the number of inverted tetrahedra but it does not solve the problem: in the case $\varepsilon_2 = 0.001$ the number of inverted tetrahedra decreases from 20875 to 4961 (see Table 2).

We now use the tetrahedral mesh optimization. Although no relocation is applied in the coarse mesh, the optimization procedure untangles it in only 13 iterations. The other meshes are untangled in no more than 6 iterations. All data are reported in Table 2. We could improve the behaviour of this procedure, if we relocate the inner nodes using the volume parametrization defined by the previous value of the tolerance, i.e the inner nodes of mesh $\varepsilon_2 = 0.01$ relocated with the volume parametrization obtained for $\varepsilon_2 = 0.1$. However, we have decided to use the coarser approximation in order to show the robustness of the optimization algorithm introduced in [10].

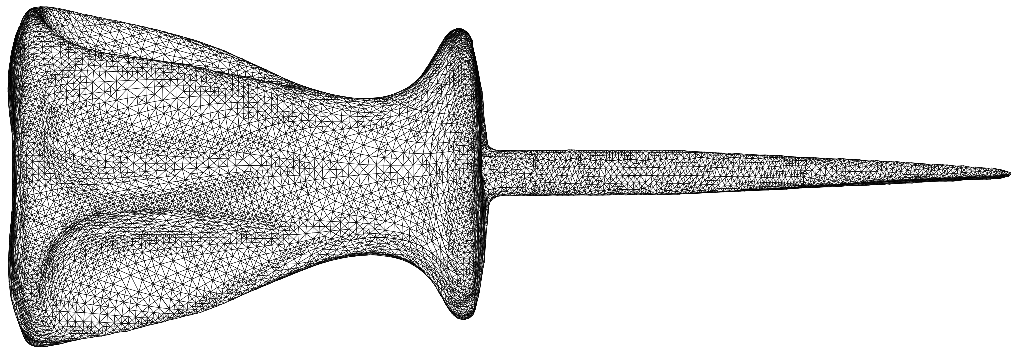
Finally, we apply 5 iterations to smooth the meshes. In all cases, the resulting mesh quality is improved to a minimum value about 0.2 and an average about $\overline{q}_\kappa = 0.7$. We note that the meccano technique generates a high quality tetrahedral mesh, see Figure 10(b). In order to show the efficiency of the mesh optimization technique inside the screwdriver we display in Figure 11 two sections before (a) and after (b) its application for $\varepsilon_2 = 0.001$. In addition, we show the sequence of screwdriver approximation for different values of ε_2 in Figure 12.

We also note that, due to the high quality surface triangulation obtained with our method, the mesh improvement is not significant if we previously apply the smoothing surface triangulation algorithm introduced in [11].

The computations have been done on a Dell precision 690, 2 Dual Core Xeon processor and $8\ Gb$ RAM memory. The CPU times for constructing the final meshes of the screwdriver are also reported in Table 2. The most demanding example requires approximately 42 second. More precisely, the CPU time in this case of each step of the meccano algorithm is: 0.5 seconds for the subdivision of the initial surface triangulation into six patches, 0.9 seconds for the Floater parametrization, 18.6 seconds for the Kossaczky recursive bisections, 2.3 seconds for the external node mapping and inner node relocation, and 19.7 seconds for the mesh optimization.



(a)



(b)

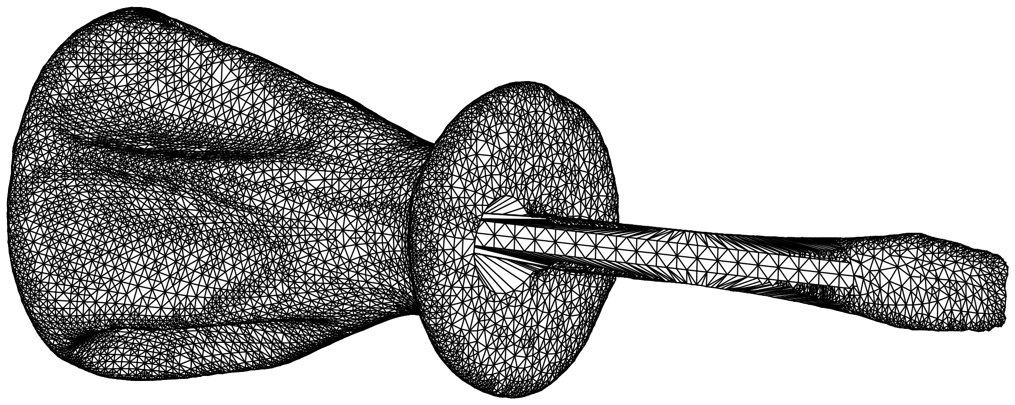
Figure 10: Screwdriver tetrahedral meshes after (a) external node mapping and after (c) the application of the mesh optimization procedure for a tolerance $\varepsilon_2 = 0.001$

4.2 Armadillo

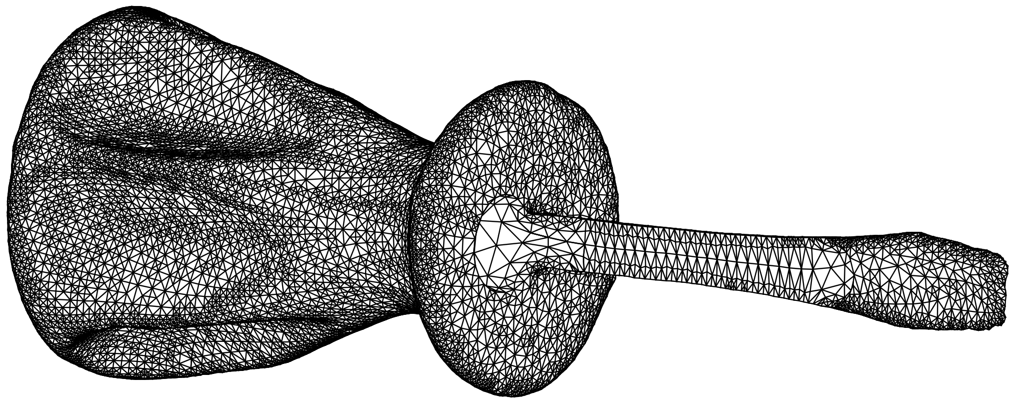
We now consider the Armadillo. The original surface triangulation has been obtained from *http://graphics.stanford.edu/data/3Dscanrep/* , i.e. the Stanford Computer Graphics Laboratory. It has 30000 triangles and 15002 nodes. The bounding box of the solid is defined by the points $(x, y, x)_{min} = (-60, -50, -26)$ and $(x, y, z)_{max} = (68, 66, 90)$.

	$\varepsilon_2 = 1$	$\varepsilon_2 = 0.1$	$\varepsilon_2 = 0.01$	$\varepsilon_2 = 0.001$
Inverted elements before/after relocation	1263	$2407/11$	$6358/417$	$20875/4961$
Untangling iterations	13	1	3	6
Smoothing iterations	5	5	5	5
Minimum quality	0.16	0.21	0.21	0.18
Mean quality	0.68	0.69	0.71	0.73
Optimization time (seconds)	2	1	5	19
Total time (seconds)	14	15	21	42

Table 2: Relocation and optimization data for the screwdriver meshes based on the volume parametrization obtained with $\varepsilon_2 = 1$



(a)



(b)

Figure 11: Section of the screwdriver tetrahedral meshes before (a) and after (b) the application of the mesh optimization procedure for a tolerance $\varepsilon_2 = 0.001$

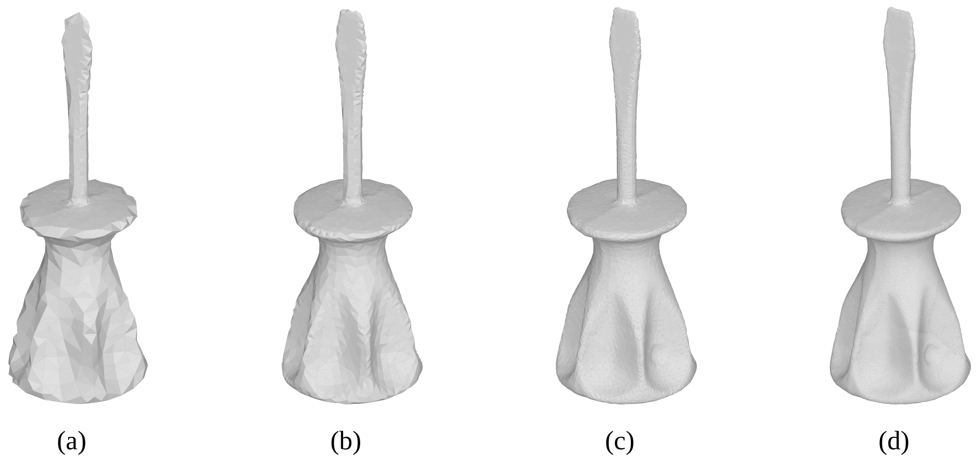


Figure 12: Screwdriver approximations for tolerances (a) 1, (b) 0.1, (c) 0.01, (d) 0.001

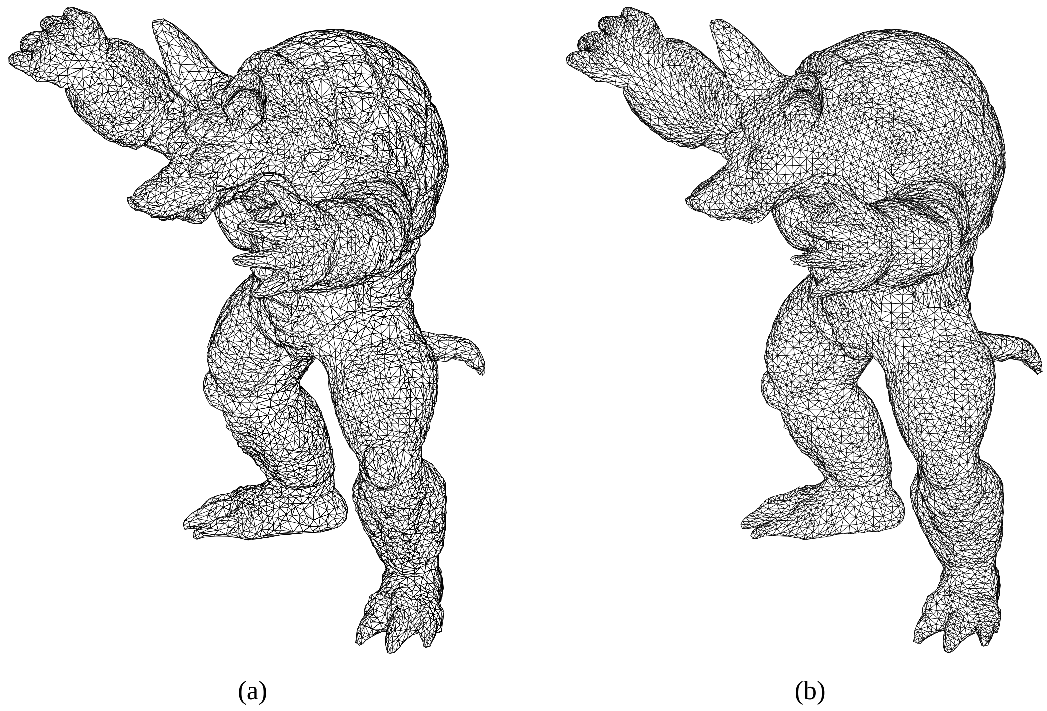


Figure 13: (a) Original surface triangulation of the Armadillo, (b) resulting valid tetrahedral mesh generated by the meccano method

We consider a unit cube as meccano. Its center is placed inside the solid at the point $(7.5, 17.5, 55.5)$. We obtain an initial subdivision of Armadillo surface in eleven maximal connected subtriangulations using Voronoi diagram. We reduce the surface partition to six patches and construct the Floater parametrization from each surface patch to the corresponding cube face.

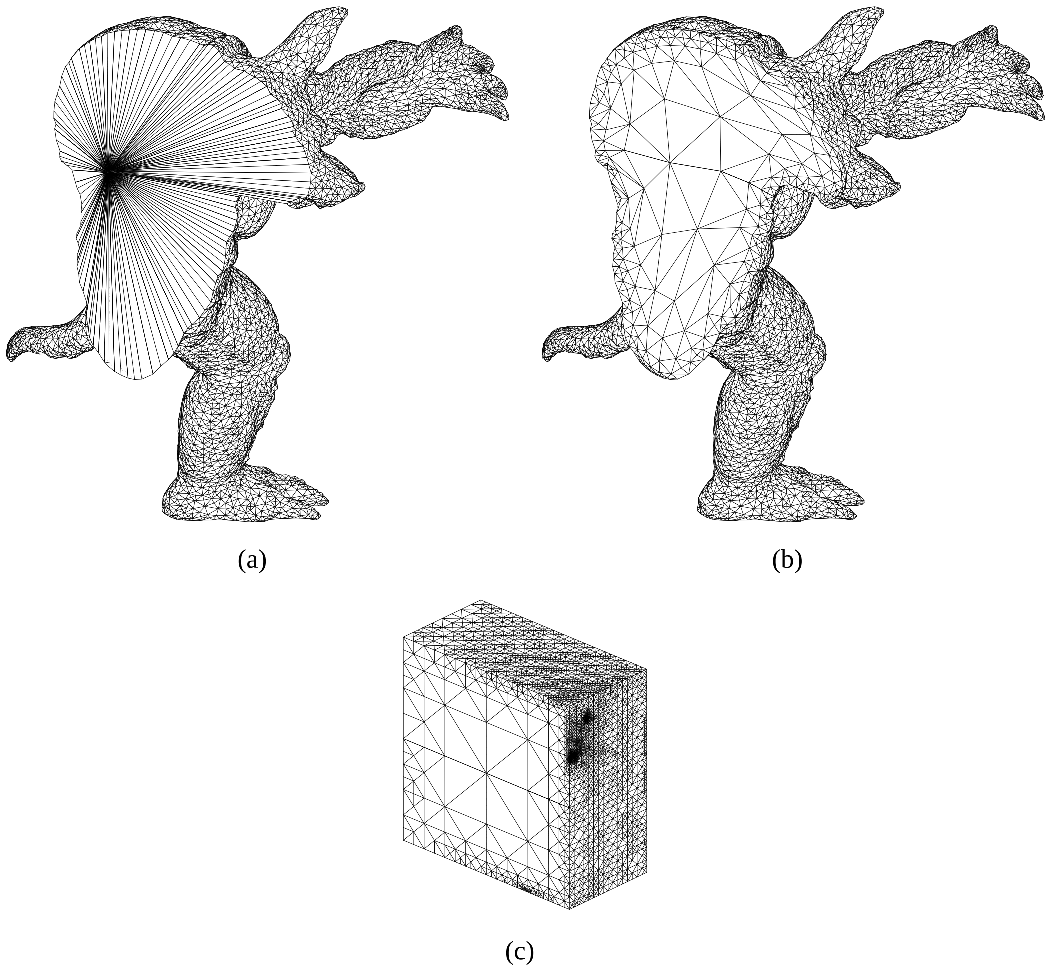


Figure 14: Cross sections of the Armadillo tetrahedral meshes before (a) and after (b) mesh optimization, and (c) same section of the meccano mesh

Fixing a tolerance $\varepsilon_2 = 0.1$, the meccano method generates a tetrahedral mesh with 144964 tetrahedra and 33889 nodes; see Figure 13(b). This mesh has 31316 triangles and 15660 nodes on its boundary and it has been reached after 68 Kossaczky refinements from the initial subdivision of the cube into six tetrahedra. The mapping of the cube external nodes to the Armadillo surface produces a 3-D tangled mesh with 10871 inverted elements. The relocation of inner nodes by using volume parametrizations reduces the number of inverted tetrahedra to 413. We use the tetrahedral mesh optimization, presented in [10], such that the mesh is untangled in 3 iterations. The mesh quality is improved to a minimum value of 0.08 and an average $\overline{q}_\kappa = 0.71$ after 6 smoothing iterations. In this case, we also note that the meccano technique generates a high quality tetrahedral mesh, Figure 13(b): only 4 tetrahedron has a quality lower than 0.1, 150 lower than 0.2 and 1046 lower than 0.3. In Figure 14, we display cross sections of the cube and Armadillo meshes before and after the mesh optimization. The location of the cube is shown in Figure 14(a).

The CPU time for constructing the final mesh of the Armadillo is approximately 54 seconds on a Dell precision 690, 2 Dual Core Xeon processor and 8 Gb RAM memory. More precisely, the CPU time of each step of the meccano algorithm is: 5 seconds for the subdivision of the initial surface triangulation into six patches, 1 seconds for the Floater parametrization, 31 seconds for the Kossaczky recursive bisections, 2 seconds for the external node mapping and inner node relocation, and 15 seconds for the mesh optimization.

Finally, we summarize a comparison between our method and standard tetrahedral mesh generation techniques [37, 38, 39]. On the one hand, one of the most important contributions of the meccano method is the resulting volume parametrization of the solid. It can have interesting applications in solid modeling and numerical simulation. For example, the application of isogeometric analysis [2, 8, 3] can be easier. On the other hand, our volume meshes can be utilized in adaptive finite element applications by using the Kossaczky's algorithm. The local refinement steps are very fast because the sequence of solid meshes is defined from the coarse mesh of the meccano, i.e., the dividing edge for tetrahedron bisection is known straightforward. In addition, we note that the minimum mesh quality is bounded during all the mesh adaptation process, because similar elements appear after three consecutive bisections.

For a given solid surface triangulation, we have checked that a constrained Delaunay tetrahedralization [39] can be faster than our method, but the resulting meshes have lower quality. If the admissible minimum quality is increased, many points can be added and the number of tetrahedra increases to reach the objective. If only a conforming Delaunay tetrahedralization is desired, the number of tetrahedra can be much higher than in our method. A great advantage of our method is that the adaptive node distribution on the boundary and the inner of the solid is more structured, because the positions are fixed with a sequence of nested meshes. In the case of advancing front [37], the resulting mesh depends on the quality of the given surface triangulation. We have seen that the optimization of this triangulation, changing the mesh topology, can be too costly.

4.3 Isogeometric Model of the Bunny

The volume parametrization presented in this paper has applications in other fields different from tetrahedral mesh generation. For example, it can be used to construct a volume T-mesh for isogeometric analysis [8]. The key lies in using the mapping, provided by the volume parametrization, to transform a T-mesh defined on the parametric domain (a unit cube in our case) into the physical domain. The T-mesh of the parametric domain is the parametric space in which the set of T-splines are defined [3].

The technique to construct a T-mesh starts by dividing the parametric cube in lower cubes by using an octree in such a way that each leaf of the octree is divided in eight children (eight cubes). The division continues until each terminal cube of the octree does not contain a node of the Kossaczky's mesh in its inner. The octree defines a

T-mesh in the parametric space that it is used to determine the *local knot vector* and the *anchors* of the T-splines following the description of [3]. Thus, the image of a point (u, v, w) in the parametric domain is given by

$$\mathbf{S}(u, v, w) = \sum_{\alpha \in A} \mathbf{P}_\alpha R_\alpha(u, v, w)$$

where $R_\alpha(u, v, w) = \frac{B_\alpha(u,v,w)}{\sum_{\beta \in A} B_\beta(u,v,w)}$ is the T-spline blending function and B_α the corresponding B-spline associated to the $\mathbf{s}_\alpha$ *anchor*. The index set A runs over all the anchors of the T-mesh. The control points $\mathbf{P}_\alpha$ are calculated by imposing the conditions $\mathbf{S}(\mathbf{s}_\beta) = \sum_{\alpha \in A} \mathbf{P}_\alpha R_\alpha(\mathbf{s}_\beta)$ for all the anchors of the T-mesh. Here, we have also used the anchors as interpolation points. The image of each interpolation point $\mathbf{s}_\beta$ in the physical space, $\mathbf{S}(\mathbf{s}_\beta)$, is determined by the volume parametrization.

We have applied our technique to the Stanford bunny. The original surface triangulation has been obtained from *http://graphics.stanford.edu/data/3Dscanrep/,* i.e. the Stanford Computer Graphics Laboratory. Figure 15(a) shows the cube tetrahedral mesh obtained by the meccano method. Figure 15(b) shows the parametric T-mesh. Figure 16(a) shows the tetrahedral mesh of the bunny constructed by the meccano method. Figure 16(b) shows the T-mesh of the Standford bunny generated by the application of the mapping $\mathbf{S}(u, v, w)$ to the parametric T-mesh.

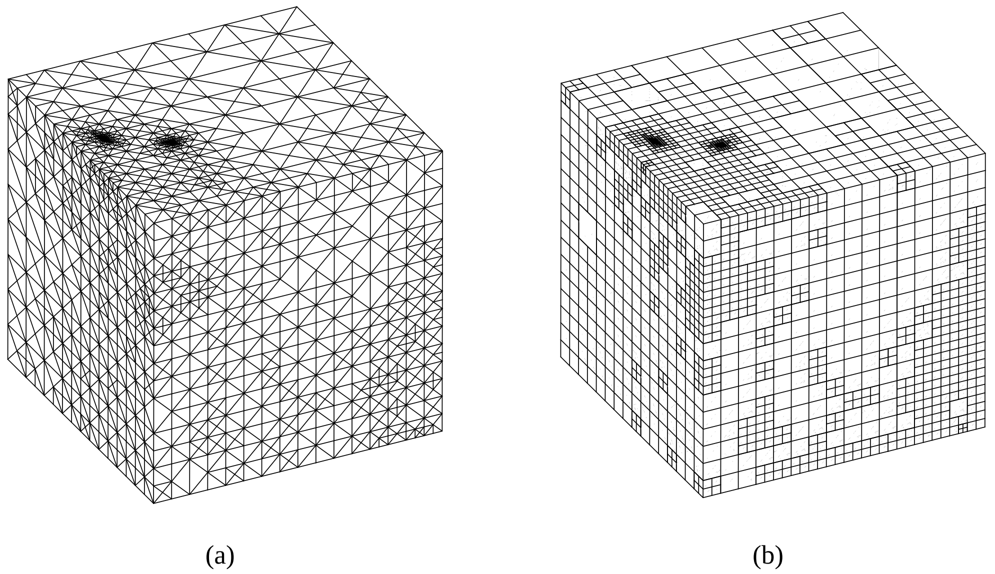
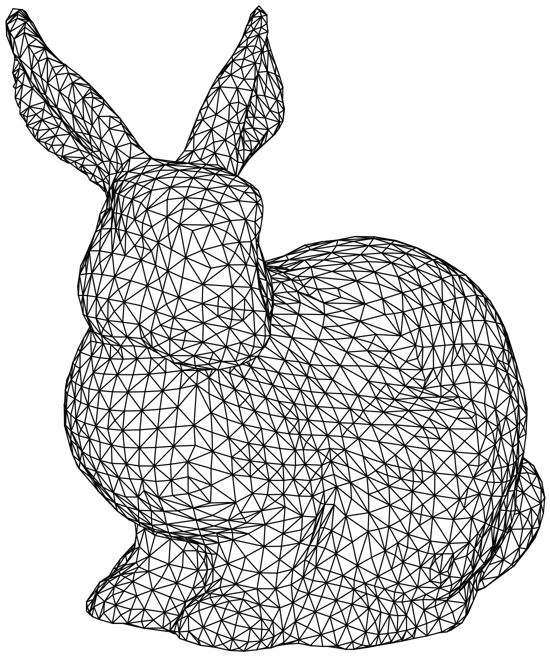


Figure 15: Meccano tetrahedral mesh (a) and corresponding parametric T-mesh (b)


(a)

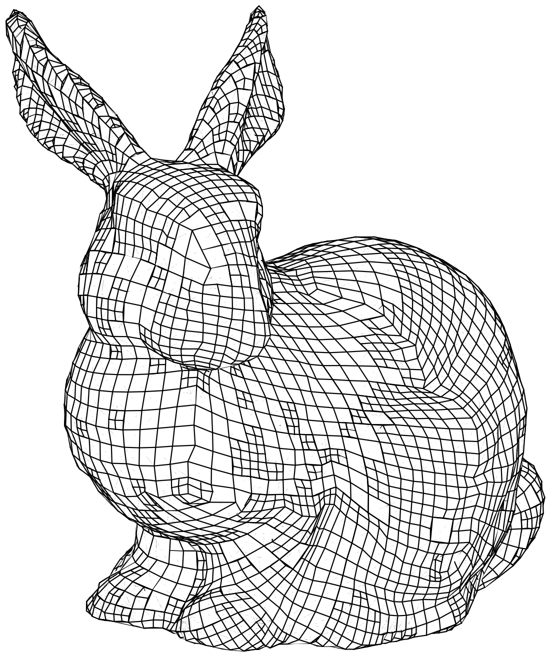

(b)

Figure 16: Tetrahedral mesh of the Stanford bunny (a) and corresponding physical T-mesh (b)

5 Conclusions and Future Research

The meccano method is an efficient mesh generator for creating adaptive tetrahedral meshes of a solid whose boundary is a surface of genus 0. We highlight the fact that the method requires minimum user intervention and has a low computational cost. The procedure is fully automatic and it is only defined by a surface triangulation of the solid, a cube and a tolerance ε that fixes the desired approximation of the solid surface. We note that the method simultaneously constructs a volume parametrization of the complex solid. On this direction, we have introduced an application of the meccano method in isogeometric analysis.

In addition, we have introduced an automatic partition of the given solid surface triangulation for fixing an admissible mapping between the cube faces and the solid surface patches, such that each cube face is the parametric space of its corresponding patch.

The mesh generation technique is based on sub-processes (subdivision, mapping, optimization) which are not in themselves new, but the overall integration using a simple shape as starting point is an original contribution of the method and has some obvious performance advantages. Another interesting property of the new mesh generation strategy is that it automatically achieves a good mesh adaption to the geometrical characteristics of the domain. Moreover, the quality of the resulting meshes is high.

The main ideas presented in this paper can be applied for constructing tetrahedral or hexahedral meshes of complex solids. Different local refinement algorithms can be considered. In future works, the meccano technique can be extended for meshing a complex solid whose boundary is a surface of genus greater than zero. In this case, the meccano can be a polycube or built by polyhedral pieces with compatible connections. At present, the user has to define the meccano associated to the solid, but we have implemented a special CAD package for more general input solid.

Acknowledgements

This work has been supported by the Spanish Government, “Secretaría de Estado de Universidades e Investigación”, “Ministerio de Ciencia e Innovación”, and FEDER, grant contracts: CGL2008-06003-C03 and UNLP08-3E-010. The authors are also grateful to the “Junta de Castilla y León”, grant number SA124A08, and the “Instituto Tecnológico de Canarias” for their support.

References

[1] M.S. Bazaraa, H.D. Sherali and C.M. Shetty, “Nonlinear Programing: Theory and Algorithms”, John Wiley and Sons Inc., New York, 1993.

[2] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott and T.W. Sederberg, “Isogeometric analysis: Toward unification of com-

puter aided design and finite element analysis", in: Trends in Engineering Computational Technology, Saxe-Coburg Publications, Stirling, 1-16, 2008.

[3] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott and T.W. Sederberg, Isogeometric analysis using T-splines, "Comput. Meth. Appl. Mech. Eng.", 199, 229-263, 2010.

[4] G.F. Carey, "Computational Grids: Generation, Adaptation, and Solution Strategies", Taylor & Francis, Washington, 1997.

[5] G.F. Carey, A Perspective on Adaptive Modeling and Meshing (AM&M), "Comput. Meth. Appl. Mech. Eng.", 195, 214-235, 2006.

[6] J.M. Cascón, R. Montenegro, J.M. Escobar, E. Rodríguez and G. Montero, "A new *meccano* technique for adaptive 3-D triangulation" in: "Proceedings of 16th International Meshing Roundtable", Springer, Berlin, 103-120, 2007.

[7] J.M. Cascón, R. Montenegro, J.M. Escobar, E. Rodríguez and G. Montero, "The meccano method for automatic tetrahedral mesh generation of complex genus-zero solids" in: "Proceedings of 18th International Meshing Roundtable", Springer-Verlag, Berlin, 463-80, 2009.

[8] J.A. Cottrell, T.J.R. Hughes and Y. Bazilevs, "Isogeometric Analysis: Towad Integration of CAD and FEA", John Wiley & Sons, Chichester, 2009.

[9] J.M. Escobar and R. Montenegro, "Several Aspects of Three-dimensional Delaunay Triangulation", Adv. Eng. Soft., 27, 27-39, 1996.

[10] J.M. Escobar, E. Rodríguez, R. Montenegro, G. Montero and J.M. González-Yuste, "Simultaneous Untangling and Smoothing of Tetrahedral Meshes", Comput. Meth. Appl. Mech. Eng., 192, 2775-2787, 2003.

[11] J.M. Escobar, G. Montero, R. Montenegro and E. Rodríguez, "An Algebraic Method for Smoothing Surface Triangulations on a Local Parametric Space", Int. J. Num. Meth. Eng., 66, 740-760, 2006.

[12] J.M. Escobar, R. Montenegro, E. Rodríguez and G. Montero, "Simultaneous Aligning and Smoothing of Surface Triangulations", Engineering with Computers, published on-line, DOI 10.1007/s00366-010-0177-7, 2010.

[13] L. Ferragut, R. Montenegro and Plaza, "Efficient Refinement/Derefinement Algorithm of Nested Meshes to Solve Evolution Problems", Comm. Num. Meth. Eng., 10, 403-412, 1994.

[14] M.S. Floater, "Parametrization and Smooth Approximation of Surface Triangulations", Comput. Aid. Geom. Design, 14, 231-250, 1997.

[15] M.S. Floater, "One-to-one Piece Linear Mappings over Triangulations", Mathematics of Computation, 72, 685-696, 2002.

[16] M.S. Floater, "Mean Value Coordinates", Comput. Aid. Geom. Design, 20, 19-27, 2003.

[17] M.S. Floater and K. Hormann, "Surface parameterization: a tutorial and survey, Advances in Multiresolution for Geometric Modelling", Mathematics and Visualization. Springer, Berlin, 157-186, 2005.

[18] M.S. Floater and V. Pham-Trong, "Convex Combination Maps over Triangulations, Tilings, and Tetrahedral Meshes", Advances in Computational Mathematics, 25, 347-356, 2006.

[19] L.A. Freitag and P. Plassmann, “Local Optimization-based Simplicial Mesh Untangling and Improvement”, Int. J. Num. Meth. Eng. 49, 109-125, 2000.

[20] L.A. Freitag and P.M. Knupp, “Tetrahedral Mesh Improvement Via Optimization of the Element Condition Number”, Int. J. Num. Meth. Eng., 53, 1377-1391, 2002.

[21] P.J. Frey and P.L. George, “Mesh Generation”, Hermes Science Publishing, Oxford, 2000.

[22] P.L. George and H. Borouchaki, “Delaunay Triangulation and Meshing: Application to Finite Elements”, Editions Hermes, Paris, 1998.

[23] J.M. González-Yuste, R. Montenegro, J.M. Escobar, G. Montero and E. Rodríguez, “Local Refinement of 3-D Triangulations Using Object-oriented Methods”, Adv. Eng. Soft., 35, 693-702, 2004.

[24] P.M. Knupp, “Achieving Finite Element Mesh Quality Via Optimization of the Jacobian Matrix Norm and Associated Quantities. Part II-A Frame Work for Volume Mesh Optimization and the Condition Number of the Jacobian Matrix”, Int. J. Num. Meth. Eng., 48, 1165-1185, 2000.

[25] P.M. Knupp, “Algebraic Mesh Quality Metrics”, SIAM J. Sci. Comput., 23, 193-218, 2001.

[26] I. Kossaczky, “A Recursive Approach to Local Mesh Refinement in Two and Three Dimensions”, J. Comput. Appl. Math., 55, 275-288, 1994.

[27] X. Li, X. Guo, H Wang, Y. He, X. Gu and H. Qin, “Harmonic Volumetric Mapping for Solid Modeling Applications”, in: “Proceedings of ACM Solid and Physical Modeling Symposium”, Association for Computing Machinery, Inc., 109-120, 2007.

[28] J. Lin, X. Jin, Z. Fan and C.C.L. Wang, “Automatic PolyCube-Maps”, Lecture Notes in Computer Science, 4975, Springer, Berlin, 3-16, 2008.

[29] R. Löhner and J.D. Baum, “Adaptive H-Refinement on 3-D Unstructured Grids for Transient Problems”, Int. J. Num. Meth. Fluids, 14, 1407-1419, 1992.

[30] J. Maubach, Local Bisection Refinement for N-Simplicial Grids Generated by Reflection, SIAM J. Sci. Comput., 16, 210-227, 1995.

[31] W.F. Mitchell, “A Comparison of Adaptive Refinement Techniques for Elliptic Problems”, ACM Trans. Math. Soft., 15, 326-347, 1989.

[32] R. Montenegro, J.M. Escobar, G. Montero and E. Rodríguez, “Quality improvement of surface triangulations”, in: “Proceedings of 14th International Meshing Roundtable”, Springer, Berlin, 469-484, 2005.

[33] R. Montenegro, J.M. Cascón, J.M. Escobar, E. Rodríguez and G. Montero, “Implementation in ALBERTA of an automatic tetrahedral mesh generator”, in: “Proceedings of 15th International Meshing Roundtable”, Springer, Berlin, 325-338, 2006.

[34] R. Montenegro, J.M. Cascón, J.M. Escobar, E. Rodríguez and G. Montero, “An Automatic Strategy for Adaptive Tetrahedral Mesh Generation”, Applied Numerical Mathematics, 59, 2203-2217, 2009.

[35] M.C. Rivara and C. Levin, “A 3-D Refinement Algorithm Suitable for Adaptive Multigrid Techniques”, J. Comm. Appl. Numer. Meth., 8, 281-290, 1992.

[36] A. Schmidt and K.G. Siebert, Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA, “Lecture Notes in Computer Science and Engineering”, Vol. 42, Springer, Berlin, 2005.

[37] J. Schöberl, “NETGEN - An advancing front 2D/3D-mesh generator based on abstract rules”, Comput. Visual. Sci., 1, 41-52, 1997.

[38] H. Si, “Adaptive tetrahedral mesh generation by constrained Delaunay refinement”, Int. J. Num. Meth. Eng., 75, 856-880, 2008.

[39] H. Si, “TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator”, http://tetgen.berlios.de, version 1.4.3, 2009.

[40] M. Tarini, K. Hormann, P. Cignoni and C. Montani, “Polycube-maps”, ACM Trans. Graph., 23, 853-860, 2004.

[41] J.F. Thompson, B. Soni and N. Weatherill, “Handbook of Grid Generation”, CRC Press, London, 1999.

[42] C.T. Traxler, “An Algorithm for Adaptive Mesh Refinement in N Dimensions”, Computing, 59, 115-137, 1997.

[43] M. Vose, “The Simple Genetic Algorithm”, MIT Press, Cambridge, Massachusetts, 1999.

[44] H. Wang, Y. He, X. Li, X. Gu and H. Qin, Polycube Splines, Comput. Aid. Geom. Design, 40, 721-733, 2008.