The Meccano Method for Automatic Tetrahedral Mesh Generation of Complex Genus-Zero Solids

J.M. Cascón[1], R. Montenegro[2], J.M. Escobar[2], E. Rodríguez[2] and G. Montero[2]

[1] Department of Economics and Economic History, Faculty of Economics and Management, University of Salamanca, Spain, casbar@usal.es

[2] Institute for Intelligent Systems and Numerical Applications in Engineering, University of Las Palmas de Gran Canaria, Campus Universitario de Tafira, Las Palmas de Gran Canaria, Spain, {rmontenegro,jmescobar,erodriguez,gmontero}@siani.es, http://www.dca.iusiani.ulpgc.es/proyecto2008-2011

Summary. In this paper we introduce an automatic tetrahedral mesh generator for complex genus-zero solids, based on the novel meccano technique. Our method only demands a surface triangulation of the solid, and a coarse approximation of the solid, called meccano, that is just a cube in this case. The procedure builds a 3-D triangulation of the solid as a deformation of an appropriate tetrahedral mesh of the meccano. For this purpose, the method combines several procedures: an automatic mapping from the meccano boundary to the solid surface, a 3-D local refinement algorithm and a simultaneous mesh untangling and smoothing. A volume parametrization of the genus-zero solid to a cube (meccano) is a direct consequence. The efficiency of the proposed technique is shown with several applications.

Key words: Tetrahedral mesh generation, local refinement, nested meshes, mesh untangling and smoothing, surface and volume parametrization.

1 Introduction

Many authors have devoted great effort to solving the automatic mesh generation problem in different ways [3, 13, 14, 26], but the 3-D problem is still open [1]. 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 adaption and mesh conformity with the solid boundary, still remain.

We have recently introduced the meccano technique in [21, 2, 22] 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. A particular case is a meccano consisting only of connected cubes, i.e. a polycube [25, 19, 27]. The method generates the solid mesh as a deformation of an appropriate tetrahedral mesh of the meccano. The main idea of the new mesh generator is to combine an automatic parametrization of surface triangulations [6], a local refinement algorithm for 3-D nested triangulations [17] and a simultaneous untangling and smoothing procedure [4].

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 by M. Floater in [6, 7, 8, 9]. 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 [25, 19, 27]. 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 [18] as one of the first tentative to solve the problem. In addition, Floater et al [10] give a simple counterexample to show that convex combination mappings over tetrahedral meshes are not necessarily one-to-one.

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 which illustrate the efficiency of this strategy. Finally, the conclusions and future research are presented in Section 5.

2 Meccano Technique Algorithm

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 [21, 2, 22]. 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 meccano and solid boundaries.
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 plane faces to the true boundary surface and by relocating the inner nodes at a *reasonable* position. After those two steps, the resulting mesh is generally tangled, but it has an admissible topology. Finally, a simultaneous untangling and smoothing procedure is applied and a valid adaptive tetrahedral mesh of the object is obtained.

3 Meccano Technique for a Complex Genus-Zero Solid

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 as datum a triangulation of the solid surface. 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 than cube faces, so that each patch is mapped to a cube face.

We note that even being poor the quality of this initial triangulation, the meccano method can reach a high quality surface and volume triangulation.

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.

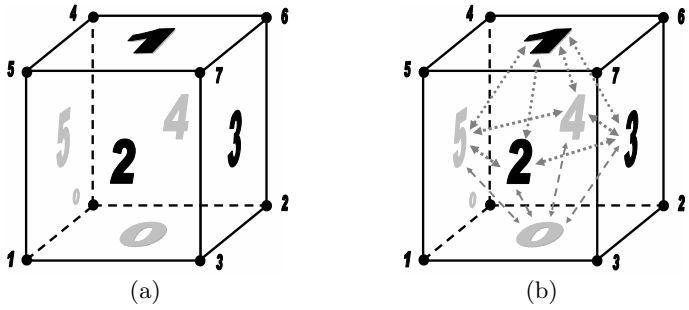


Fig. 1. Meccano formed by one cube: (a) notation of nodes and faces of the cube and (b) connectivities of faces

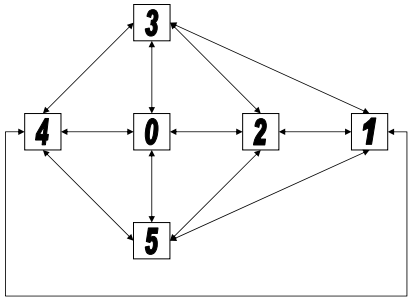


Fig. 2. Planar graph $\mathcal{G}_{\mathcal{C}}$ associated to the cube

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. For a genus-zero solid, if the center of the cube is placed inside the solid, the existence of an admissible mapping is ensured.

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, 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 $\Pi : \Sigma_{\mathcal{C}} \rightarrow \Sigma_{\mathcal{S}}$ be a piecewise function, such that $\Pi_{|\Sigma_{\mathcal{C}}^i} = \Pi^i$ where $\Pi^i : \Sigma_{\mathcal{C}}^i \rightarrow \Pi^i(\Sigma_{\mathcal{C}}^i) \subset \Sigma_{\mathcal{S}}$. 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 $\Sigma_{\mathcal{S}}$.

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 in Figure 2, then we parametrize each patch to a cube face.

Partition of the Solid Surface Triangulation

In the following we call *connected subtriangulation* to 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.

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 $\mathcal{G}_\mathcal{C}$. Otherwise, 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 number of 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 $\mathcal{G}_\mathcal{S}$ is comprised only six head vertices, but the compatibility of $\mathcal{G}_\mathcal{S}$ with $\mathcal{G}_\mathcal{C}$ 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 what follows we denote $\Sigma_\mathcal{S}^i$ the solid surface patch defined by the triangles of $\mathcal{T}_\mathcal{S}^i$.

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 the surface triangulations $\mathcal{T}_\mathcal{S}^i$ proposed by M. Floater [6]. So, we define $\left(\Pi^i\right)^{-1} : \Sigma_\mathcal{S}^i \to \Sigma_\mathcal{C}^i$ and we denote $\tau_F^i = \left(\Pi^i\right)^{-1}(\mathcal{T}_\mathcal{S}^i)$ 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 $\mathcal{T}_\mathcal{S}^i$. Fixed the position of boundary nodes $\{Q_{n+1}, \ldots, Q_N\}$ of τ_F^i, 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 the weights of the convex combinations $\{\lambda_{kl}\}_{k=1,\ldots,n}^{l=1,\ldots,N}$ verify

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

In [6] 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 [8]. 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 $\mathcal{T}_{\mathcal{S}}$ 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 [25, 19] verify property (a), but they do not verify property (b), i.e., a triangle belonging to $\mathcal{T}_{\mathcal{S}}$ 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 triangulations τ_K (obtained on $\Sigma_{\mathcal{C}}$ by application of the refinement algorithm of Kossaczky [17]) to the solid boundary. Several problems can appear in the application of this transformation 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 [17], see Figure 3(a). The resulting mesh can be recursively and globally bisected 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.

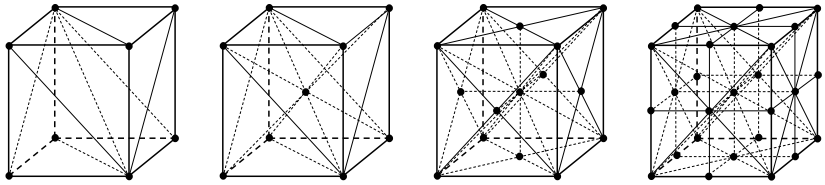


Fig. 3. Refinement of a cube by using Kossaczky's algorithm: (a) cube subdivision into six tetrahedra, (b) bisection of all tetrahedra by inserting a new node in the cube main diagonal, (c) new nodes in diagonals of cube faces and (d) global refinement with new nodes in cube edges

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 [17], of those tetrahedra with a face 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 $\mathcal{T}_\mathcal{S}$ 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.

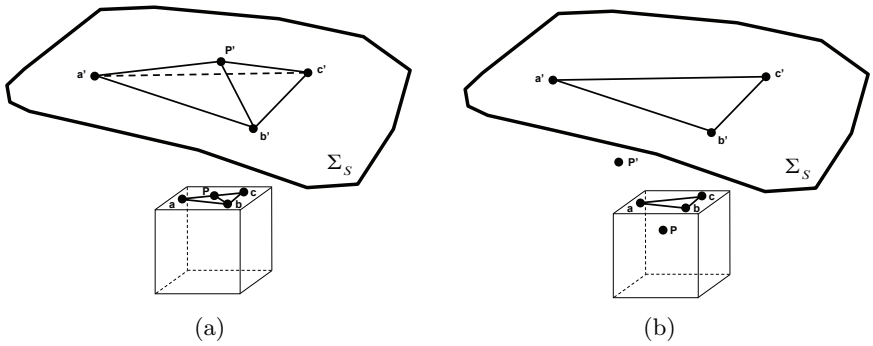


Fig. 4. Node mapping from meccano to real domain: (a) mapping Π from external nodes a, b, c, P to a', b', c', P', and (b) relocation of an inner node P in P'

The threshold of whether to refine the triangle or not is given by a tolerance ε_i fixed by the user. We note that other measures could be introduced in line with the desired approximation type (curvature, points properties, etc.).

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. 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 ε_i.
   b) Refine the mesh.

   While any tetrahedron T is marked.

We denote 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.

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 have checked that this problem only appear when the mesh size of surface approximation $\Pi(\tau_K)$ is the same order than the mesh size of $\mathcal{T}_\mathcal{S}$. So, if a more precise approximation of the solid surface is demanded to the meccano approximation, a simple solution is to refine the given solid surface triangulation $\mathcal{T}_\mathcal{S}$.

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, see Figure 4(b). 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.

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. In practice, a good strategy is: we start meshing the solid by using a high value of ε (a coarse tetrahedral mesh of the solid is obtained) and we continue decreasing it gradually. 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 [16],

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

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 [15].

We have proposed in [4] an alternative to the procedure of [12, 11], 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 and efficient unconstrained optimization method to find the minimum of the modified objective function.

In addition, a smoothing of the boundary surface triangulation could be applied before the movement of inner nodes of the domain by using the procedure presented in [5, 20].

4 Test Examples

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 $\mathcal{T}_{\mathcal{S}}^i$ to a cube face $\Sigma_{\mathcal{C}}^i$ 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 [6, 8].

ALBERTA is an adaptive multilevel finite element toolbox [24] developed in C. This software can be used to solve several types of 1-D, 2-D or 3-D problems. ALBERTA uses the Kossaczky refinement algorithm [17] and requires an initial mesh topology [23]. 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. In addition, the minimum quality of refined meshes is function of the initial mesh quality.

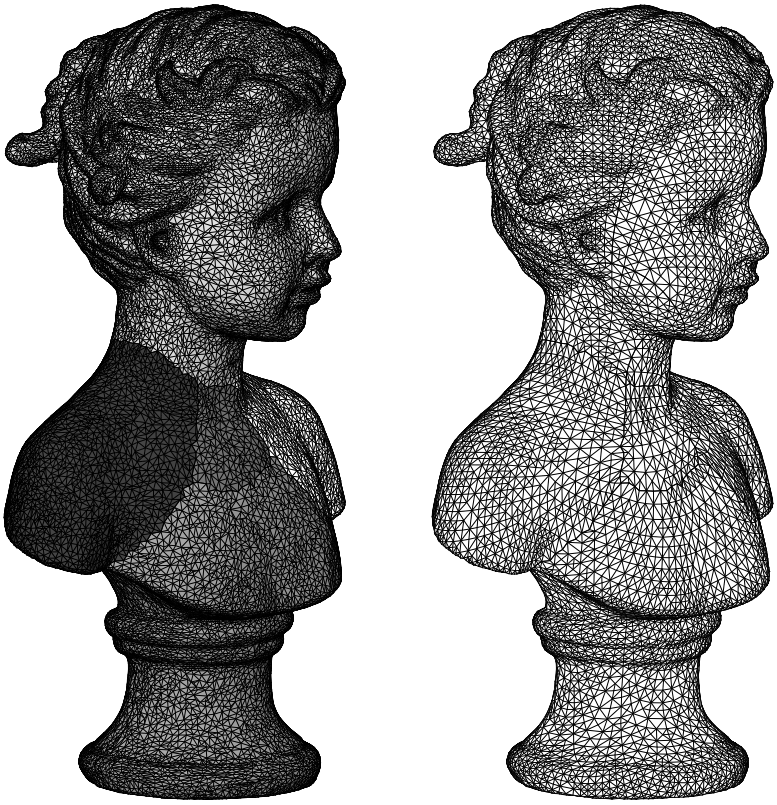
The performance of our novel tetrahedral mesh generator is shown in the following applications. The first corresponds to a Bust and the second to the Stanford Bunny. We have obtained a surface triangulation of these objects from internet. For both examples, the *meccano* is just a cube.

4.1 Example 1: Bust

The original surface triangulation of the Bust has been obtained from the website *http://shapes.aimatshape.net*, i.e. AIM@SHAPE Shape Repository, and it is shown in Figure 5(a). It has 64000 triangles and 32002 nodes. The bounding box of the solid is defined by the points $(x, y, z)_{min} = (-120, -30.5, -44)$ and $(x, y, z)_{max} = (106, 50, 46)$.

We consider a cube, with an edge length equal to 20, as meccano. Its center is placed inside the solid at the point $(5, -3, 4)$. We obtain an initial subdivision of Bust surface in seven maximal connected subtriangulations. In order to get a compatible decomposition of the surface triangulation, we use the proposed iterative procedure to reduce the current seven vertices of the graph $\mathcal{G}_{\mathcal{S}}$ to six. Figure 5(a) shows the resulting compatible partition $\{\mathcal{T}_{\mathcal{S}}^i\}_{i=0}^5$

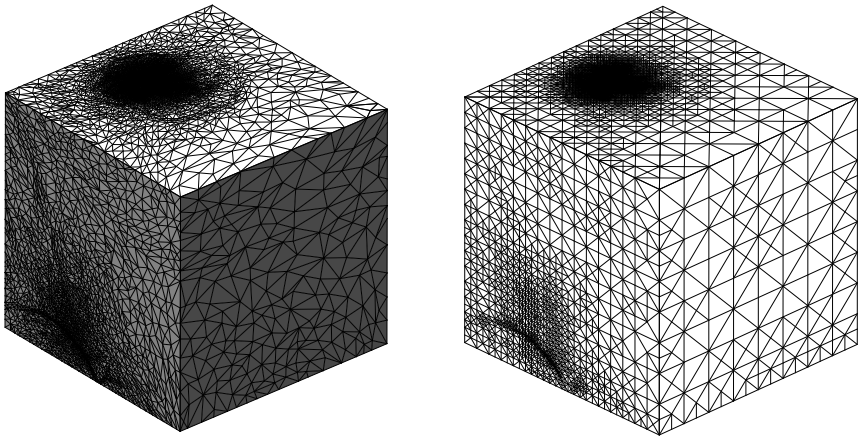
We map each surface patch $\Sigma_{\mathcal{S}}^i$ to the cube face $\Sigma_{\mathcal{C}}^i$ by using the Floater parametrization [6]. Once the global parametrization of the Bust surface triangulation is built, see Figure 6(a), the definition of the one-to-one mapping between the cube and Bust boundaries is straightforward.



(a) (b)

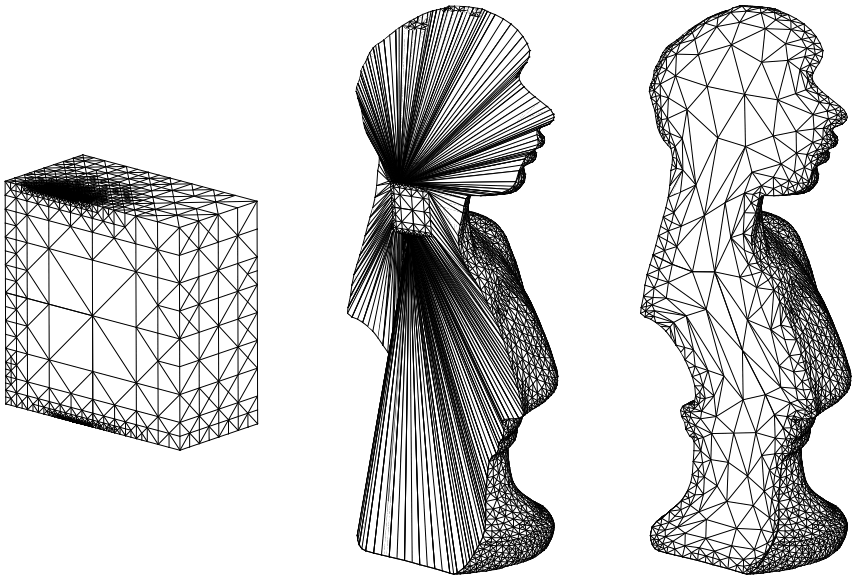
Fig. 5. (a) Original surface triangulation of the Bust with a compatible partition $\{\mathcal{T}_{\mathcal{S}}^i\}_{i=0}^5$ after applying our reduction algorithm and (b) the resulting valid tetrahedral mesh generated by the meccano method

Fixing a tolerance $\varepsilon_2 = 0.1$, the meccano method generates a tetrahedral mesh of the cube with 147352 tetrahedra and 34524 nodes; see Figures 6(b) and 7(a). This mesh has 32254 triangles and 16129 nodes on its boundary and it has been reached after 42 Kossaczky refinements from the initial subdivision of the cube into six tetrahedra. The mapping of the cube external nodes to the Bust surface produces a 3-D tangled mesh with 8947 inverted elements; see Figure 7(b). The relocation of inner nodes by using volume parametrizations reduces the number of inverted tetrahedra to 285. We apply the mesh optimization procedure [4] and the mesh is untangled in 2 iterations. The mesh quality is improved to a minimum value of 0.07 and an average $\overline{q}_\kappa = 0.73$ after 10 smoothing iterations. We note that the meccano technique generates



(a) (b)

Fig. 6. (a) Floater's parametrization of $\{\mathcal{T}_{\mathcal{S}}^i\}_{i=0}^5$ on corresponding cube faces for the bust application, (b) cube tetrahedral mesh obtained by the meccano method



(a) (b) (c)

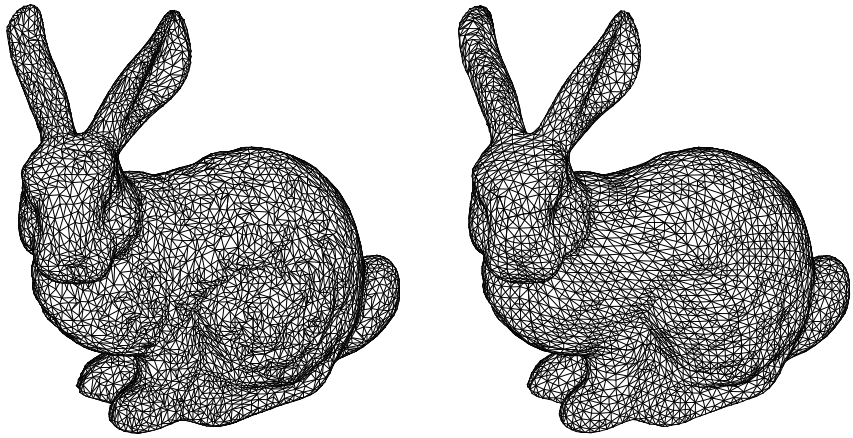
Fig. 7. Cross sections of cube (a) and Bust tetrahedral meshes before (b) and after (c) the application of the mesh optimization procedure

a high quality tetrahedral mesh (see Figure 5(b)): only 1 tetrahedron has a quality less than 0.1, 13 less than 0.2 and 405 lees than 0.3. In Figure 7, we display two cross sections of the cube and Bust meshes before and after the mesh optimization. The location of the cube is shown in Figure 7(b).

The CPU time for constructing the final mesh of the Bust is 93.27 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: 1.83 seconds for the subdivision of the initial surface triangulation into six patches, 3.03 seconds for the Floater parametrization, 44.50 seconds for the Kossaczky recursive bisections, 2.31 seconds for the external node mapping and inner node relocation, and 41.60 seconds for the mesh optimization.

4.2 Example 2: Bunny

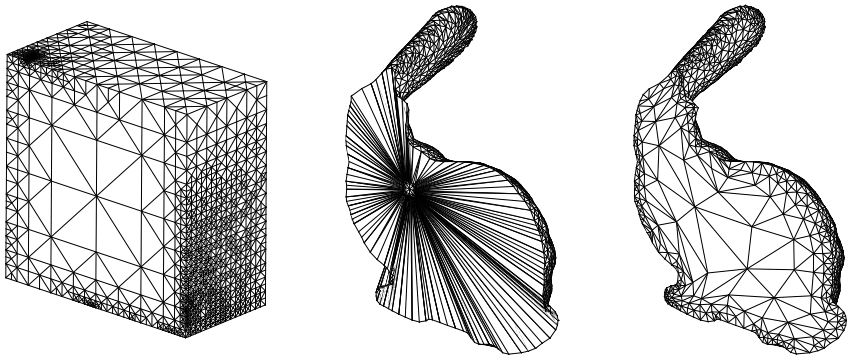
The original surface triangulation of the Stanford Bunny has been obtained from the website *http://graphics.stanford.edu/data/3Dscanrep/* , i.e. the Stanford Computer Graphics Laboratory, and it is shown in Figure 8(a). It has 12654 triangles and 7502 nodes. The bounding box of the solid is defined by the points $(x, y, z)_{min} = (-10, 3.5, -6)$ and $(x, y, z)_{max} = (6, 2, 6)$.



(a) (b)

Fig. 8. (a) Original surface triangulation of the Stanford Bunny and (b) the 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 $(-4.5, 10.5, 0.5)$. We obtain an initial subdivision of the Bunny surface in eight maximal connected subtriangulations using Voronoi diagram. We reduce the surface partition to six patches and we construct the Floater



(a) (b) (c)

Fig. 9. Cross sections of cube (a) and Bunny tetrahedral meshes before (b) and after (c) the application of the mesh optimization process

parametrization from each surface patch $\Sigma_{\mathcal{S}}^i$ to the corresponding cube face $\Sigma_{\mathcal{C}}^i$. Fixing a tolerance $\varepsilon_2 = 0.0005$, the meccano method generates a tetrahedral mesh with 54496 tetrahedra and 13015 nodes. This mesh has 11530 triangles and 6329 nodes on its boundary and has been reached after 44 Kossaczky refinements from the initial subdivision of the cube into six tetrahedra. The mapping of the cube external nodes to the Bunny surface produces a 3-D tangled mesh with 2384 inverted elements, see Figure 9(b). The relocation of inner nodes by using volume parametrizations reduces the number of inverted tetrahedra to 42. We apply 8 iterations of the tetrahedral mesh optimization and only one inverted tetrahedra can not be untangled. To solve this problem, we allow the movement of the external nodes of this inverted tetrahedron and we apply 8 new optimization iterations. The mesh is then untangled and, finally, we apply 8 smoothing iterations fixing the boundary nodes. The mesh quality is improved to a minimum value of 0.08 and an average $\overline{q}_\kappa = 0.68$. We note that the meccano technique generates a high quality tetrahedral mesh: only 1 tetrahedron has a quality below 0.1, 41 below 0.2 and 391 below 0.3. In Figure 9, we display two cross sections of the cube and Bunny meshes before and after the mesh optimization. The location of the cube can be observed in Figure 9(b).

The CPU time for constructing the final mesh of the Bunny is 40.28 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: 0.24 seconds for the subdivision of the initial surface triangulation into six patches, 0.37 seconds for the Floater parametrization, 8.62 seconds for the Kossaczky recursive bisections, 0.70 seconds for the external node mapping and inner node relocation, and 30.35 seconds for the mesh optimization.

5 Conclusions and Future Research

The meccano technique is a very efficient mesh generation method 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. In addition, the quality of the resulting meshes is high.

The definition of an automatic parametrization of a solid surface triangulation to the meccano boundary is a significant advance for the method. To that end, 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.

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 a set of polyhedral pieces with compatible connections.

Acknowledgments

This work has been supported by the *Secretaría de Estado de Universidades e Investigación* of the *Ministerio de Ciencia e Innovación* of the Spanish Government and FEDER, grant contracts: CGL2008-06003-C03.

References

1. Bazilevs Y, Calo VM, Cottrell JA, Evans J, Hughes TJR, Lipton S, Scott MA, Sederberg TW (2008) Isogeometric analysis: Toward unification of computer aided design and finite element analysis. Trends in Engineering Computational Technology 1–16. Saxe-Coburg Publications, Stirling
2. Cascón JM, Montenegro R, Escobar JM, Rodríguez E, Montero G (2007) A new *meccano* technique for adaptive 3-D triangulations. Proc 16th Int Meshing Roundtable 103–120. Springer, Berlin
3. Carey GF (1997) Computational grids: generation, adaptation, and solution strategies. Taylor & Francis, Washington
4. Escobar JM, Rodríguez E, Montenegro R, Montero G, González-Yuste JM (2003) Simultaneous untangling and smoothing of tetrahedral meshes. Comput Meth Appl Mech Eng 192:2775–2787
5. Escobar JM, Montero G, Montenegro R, E. Rodríguez E (2006) An algebraic method for smoothing surface triangulations on a local parametric space. Int J Num Meth Eng 66:740–760
6. Floater MS (1997) Parametrization and smooth approximation of surface triangulations. Comp Aid Geom Design 14:231–250

7. Floater MS (2002) One-to-one Piece Linear Mappings over Triangulations. Mathematics of Computation 72:685–696
8. Floater MS (2003) Mean Value Coordinates. Comp Aid Geom Design 20:19–27
9. Floater MS, Hormann K (2005) Surface parameterization: a tutorial and survey. Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization 157–186. Springer, Berlin
10. Floater MS, Pham-Trong V (2006) Convex Combination Maps over Triangulations, Tilings, and Tetrahedral Meshes. Advances in Computational Mathematics 25:347–356
11. Freitag LA, Knupp PM (2002) Tetrahedral mesh improvement via optimization of the element condition number. Int J Num Meth Eng 53:1377–1391
12. Freitag LA, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Num Meth Eng 49:109–125
13. Frey PJ, George PL (2000) Mesh generation. Hermes Sci Publishing, Oxford
14. George PL, Borouchaki H (1998) Delaunay triangulation and meshing: application to finite elements. Editions Hermes, Paris
15. Knupp PM (2000) 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
16. Knupp PM (2001) Algebraic mesh quality metrics. SIAM J Sci Comp 23:193–218
17. Kossaczky I (1994) A recursive approach to local mesh refinement in two and three dimensions. J Comput Appl Math 55:275–288
18. Li X, Guo X, Wang H, He Y, Gu X, Qin H (2007) Harmonic Volumetric Mapping for Solid Modeling Applications. Proc. of ACM Solid and Physical Modeling Symposium 109–120. Association for Computing Machinery, Inc.
19. Lin J, Jin X, Fan Z, Wang CCL (2008) Automatic PolyCube-Maps. Lecture Notes in Computer Science 4975:3–16
20. Montenegro R, Escobar JM, Montero G, Rodríguez E (2005) Quality improvement of surface triangulations. Proc 14th Int Meshing Roundtable 469–484, Springer, Berlin
21. Montenegro R, Cascón JM, Escobar JM, Rodríguez E, Montero G (2006) Implementation in ALBERTA of an automatic tetrahedral mesh generator. Proc 15th Int Meshing Roundtable 325–338, Springer, Berlin
22. R. Montenegro R, J.M. Cascón JM, J.M. Escobar JM, E. Rodríguez E, Montero G (2009) An Automatic Strategy for Adaptive Tetrahedral Mesh Generation. Appl Num Math 59:2203-2217
23. Schmidt A, Siebert KG (2005) Design of adaptive finite element software: the finite element toolbox ALBERTA. Lecture Notes in Computer Science and Engineering 42, Springer, Berlin
24. Schmidt A, Siebert KG. ALBERTA - An Adaptive Hierarchical Finite Element Toolbox. http://www.alberta-fem.de/
25. Tarini M, Hormann K, Cignoni P, Montani C (2004) Polycube-maps. ACM Trans Graph 23:853–860
26. Thompson JF, Soni B, Weatherill N (1999) Handbook of grid generation, CRC Press, London
27. Wang H, He Y, Li X, Gu X, Qin H (2008) Polycube Splines. Comp Aid Geom Design 40:721–733