INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2004; **60**:1049–1074 (DOI: 10.1002/nme.992)

Generation and adaptation of computational surface meshes from discrete anatomical data

Pascal J. Frey*,†

GAMMA project, INRIA Domaine de Voluceau-Rocquencourt, BP 105, 78153 Le Chesnay cedex, France

SUMMARY

Fast and accurate scanning devices are nowadays widely used in many engineering and biomedical fields. The resulting discrete data is usually directly converted into polygonal surface meshes, using 'brute-force' algorithms, often resulting in meshes that may contain several millions of polygons. Simplification is therefore required in order to make storage, computation and display possible if not efficient. In this paper, we present a general scheme for mesh simplification and optimization that allows to control the geometric approximation as well as the element shape and size quality (required for numerical simulations). Several examples ranging from academic to complex biomedical geometries (organs) are presented to illustrate the efficiency and the utility of the proposed approach. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: mesh generation; mesh adaptation; surface mesh; simplification; decimation; metric; *a posteriori* error estimate; biomedical data; reverse engineering

1. INTRODUCTION

With the advent of fast and accurate sensing and scanning devices, it is now possible to collect very large datasets of points (or voxels) of many physical objects and human organs in almost real time. The need for increasing accuracy is related to the desire of capturing the geometry of the domain of interest as precisely as scanner technology would permit. Among the various engineering and biomedical applications concerned by this technique, the Digital Michelangelo project (Stanford University) [30, 40] and the Visual Human Project (National Library of Medicine) are both representative of this evolution, each in its own field of application. In such applications, the significant challenge faced by the participants is related to the size of the datasets, typically millions or dozens of millions, if not billions, of points. Range images (or point clouds) are usually not suitable for archiving and processing data

*Correspondence to: P. J. Frey, INRIA, GAMMA project, Domaine de Voluceau-Rocquencourt, BP 105, 78153 Le Chesnay cedex, France.
†E-mail: pascal.frey@inria.fr

Contract/grant sponsor: INRIA

Received 13 March 2003
Revised 4 August 2003
Accepted 1 September 2003

and also for representing 3D geometry, polygonal models are preferred. In the last few years, several reconstruction algorithms have been proposed to extract piecewise linear approximations (i.e. triangulations) directly from point clouds or range images. Usually, these (mostly surface) meshes have the same degree of complexity as the original discrete data. However, such large triangulations cannot be handled in computer memory nor rendered by the current hardware, even with graphic accelerators. Unfortunately, as stated by Luebke [1], 'the number of polygons we want always exceeds the budget polygon we can afford'. Therefore, in order to be easily manageable (this involving storage, transmission, rendering, computation, etc.), the complexity of the polygonal models needs to be reduced dramatically. This process involves notably surface decimation or surface simplification techniques. The purpose of the underlying numerical methods is to reduce, as much as possible, the number of polygons of the model, while preserving the geometric accuracy of the model. In addition, specific requirements can be imposed on the resulting meshes, for instance regarding the element shapes and sizes, in the context of numerical simulations based on finite element or finite volume methods.

1.1. Motivations

As noticed, polygonal meshes are now widely used in computer graphics applications and in numerical simulations, largely because they are able to virtually represent any kind of surface geometry and topology of arbitrary complexity. However, the requirements laid on these meshes differ largely from one domain of application to another. For instance, in computer graphics, the aim is generally to render a complex scene composed of polygonal objects as quickly as possible, based on the hardware technology, without losing too much accuracy. In such a case, the actual frame rate dictates the number of primitives (i.e. triangles) needed to represent the model. Here, polygonal simplification algorithms are tuned to generate levels of detail (LOD) of the objects in the scene, where distant models are represented with lower levels of details than nearby objects. In numerical simulations, the emphasis is laid on the geometric accuracy as well as on the element shape quality control. A compromise is sought between the number of vertices (i.e. the number of degrees of freedom) and the quality of the geometric approximation of the surface. Polygonal meshes are also considered as a convenient data exchange format (more flexible than any CAD file). In this context, the size of the model can be reduced to comply with the bandwidth requirement of a network. In each application field, the final use of the polygonal meshes leads to tailor the simplification algorithms to the needs and requirements of the application.

The main motivation for mesh simplification is to create a somewhat *optimal mesh*, that represents an accurate approximation of the surface geometry with a minimal number of regular (well-shaped) elements. Element shape and size quality and vertex reduction are the two conflicting requirements behind mesh simplification. Hence, in order to preserve the surface geometry, the approximation error must be equi-distributed over the mesh elements, this results in distributing the vertices evenly over the surface. In the following sections, we will see that this can be achieved by using an *a posteriori* geometric error estimate based on the Hessian of the surface [2] and defining a discrete anisotropic metric map at the vertices of the original surface triangulation. This metric map is then used to govern mesh modification algorithms.

1.2. Related work

Since the early papers of Schroeder *et al.* [3] and Turk [4], mesh simplification has really received a lot of attention from numerous researchers and engineers [33, 36–39, 41]. For example, more than 1000 papers can be found on the Internet databases when searching for 'mesh simplification'. This attests of the importance of this topic in many application fields, including computer graphics, virtual reality, biomedical and numerical simulations, etc. Recently, comprehensive surveys of mesh simplification have been compelled by various authors, notably by Luebke [1], Heckbert and Garland [5], and Krus [6] most of them focussing mainly on graphical issues.

In the context of numerical computations, given that the solvers have reached an acceptable level of maturity, it is now possible to envisage complex biomedical simulations, like for instance computational hemodynamics [7]. As efficient surface and tetrahedral mesh generation packages are available, the bottleneck of finite element/finite volumes simulations still relates to the gap between discrete medical images and the finite element model [31].

For discrete data obtained using scanning/sensing devices, various algorithms have been proposed to construct a polygonal mesh from voxels [8] or point clouds [9]. Despite being efficient and robust, these approaches generally suffer several major drawbacks:

(i) discrete data are supposed to be reliable, but scanned data may be very noisy (i.e. points are off the surface);
(ii) the accuracy of scanning and sensing devices leads to unnecessary dense datasets (the density is not directly related to the local geometric complexity) and consequently to very large polygonal models;
(iii) 'brute-force' reconstruction algorithms (like 'Marching-cubes' [8]) often introduce artefacts in the polygonal approximation (for instance, 'staircases' effects);
(iv) in general, no (or very few) attention is paid to the element shape quality, although this is a major requirement in finite element/volume simulations [10].

As the two first problems are closely related to the device technology used in the data acquisition stage, one has to cope at best with these constraints and, currently, no solution has emerged that is able to preserve the accuracy of the data and lead to a minimal set of information. Instead of focussing on optimizing the algorithms connected to data acquisition or data reconstruction (that is certainly also a promising field of research), we focus here on mesh optimization methods that work directly on the polygonal model, with no assumption on the geometry or the topology of the domain. This being stated, the approach we suggest naturally involves two successive stages, a 'filtering' pass (to remove the noise) followed by a remeshing procedure controlled by a metric specification based on the intrinsic properties of the surface.

1.3. General scheme

Formally speaking, the problem we face can be described in the following manner. Let consider a bounded domain Ω in R^3 described by its boundary Σ. Given an initial surface triangulation $\mathscr{M}_{\mathrm{ref}}(\Sigma)$ of Σ, possibly combined with a (discrete) metric map $\mathscr{H}_{\mathrm{ref}}(\Sigma)$ prescribing the desired element sizes at the mesh vertices, the aim is to construct a computational mesh $\mathscr{M}(\Sigma)$. To this end, a geometric surface mesh $\mathscr{M}_{\mathrm{g}}(\Sigma)$ is first generated from the triangulation $\mathscr{M}_{\mathrm{ref}}(\Sigma)$.

In a second stage, the mesh $\mathscr{M}_{\rm g}(\Sigma)$ is optimized with respect to the element shapes and sizes so as to produce a suitable computational surface mesh $\mathscr{M}(\Sigma)$.

Schematically, the strategy we advocate for the construction of the mesh $\mathscr{M}(\Sigma)$ involves the following steps:

- the initial reference mesh $\mathscr{M}_{\rm ref}(\Sigma)$ is simplified and optimized within an 'envelope' (based on the Hausdorff distance) corresponding to a user-specified tolerance, leading to a geometric reference mesh $\mathscr{M}_{\rm ref,\,g}(\Sigma)$;
- a piecewise C^1 continuous geometric support is defined on the mesh $\mathscr{M}_{\rm ref,\,g}(\Sigma)$, in order to define a smooth geometric representation of the boundary Σ;
- the discrete metric map $\mathscr{H}_{\rm ref,\,g}(\Sigma)$ supplied with the mesh $\mathscr{M}_{\rm ref,\,g}(\Sigma)$ is then modified in order to account for the surface geometry and the desired mesh gradation;
- the mesh $\mathscr{M}_{\rm ref,\,g}(\Sigma)$ is then adapted (regarding the element sizes and shapes) to the modified metric map $\mathscr{H}_{\rm ref,\,g}(\Sigma)$ yielding to the computational mesh $\mathscr{M}(\Sigma)$.

In some applications, the proposed approach can be slightly modified to account for the specific nature of the data. This is, for instance, the case when dealing with terrains (or Cartesian surfaces) [11].

1.4. Paper outline

In this paper, we present a global approach for constructing *geometric* as well as *computational* meshes from surface triangulations. In Section 2, we briefly present the three main classes of surface reconstruction procedures suitable for discrete data. In Section 3, we introduce a global approach for building a *geometric* surface mesh from a given surface triangulation. In Section 4, we detail the strategy adopted to construct a *computational* mesh, by taking into account requirements about the element shapes and sizes. To this end, we indicate how to build a proper metric tensor (provided by a 'geometric error estimate') to govern the mesh generation stage. In Section 5, we show how the proposed approach easily extends to the generation of adapted meshes for numerical simulations. Finally, in Section 6, we give several examples of *geometric* and *computational* surface meshes for biomedical applications to demonstrate the efficiency of this approach.

2. SURFACE RECONSTRUCTION

Nowadays, modern scanning devices can provide almost instantaneously accurate and often very large datasets of (non-invasive) discrete information about any biomedical organ. However, such a large volume of sampled data may jeopardize any further real-time processing, storage or transmission of information, because of current hardware limitations. For instance, if we consider a uniform sampling on the surface of a unit cube at the accuracy of 10^{-5}, the related point cloud contains some 60 billions of points (corresponding to roughly 300 billions of tetrahedra)! Moreover, for computational purposes, a surface mesh is usually required as discretization of the domain boundaries. Therefore, surface reconstruction algorithms are used to convert the discrete volumetric data (voxel sets or point clouds) to a piecewise linear approximation that interpolates the underlying surface geometry. Then, this surface triangulation remains the sole representation of the domain geometry, having replaced the initial data. Depending on the technology used, scanning and imaging devices commonly produce either:

- a series of parallel slices,
- voxel datasets (3D images),
- point clouds.

As the nature of the data largely differs from one technique to another, reconstruction methods have been developed for each of these types. In this section, we briefly describe the principles behind these three main classes of surface reconstruction techniques.

2.1. Data reconstruction problem

As stated above, the vast majority of the reconstruction algorithms aims at extracting a triangulation (a surface mesh) from the sampled data. In this context, a typical process to construct finite element meshes involves three successive stages: image processing to extract the region (domain) of interest, then surface triangulation and finally volume meshing. The first stage is usually semi-automatic (though it requires a minimal user interaction) whereas the last two stages are fully automatic.

Several image segmentation, image filtering and image enhancement procedures can be combined to extract the region of interest from the discrete data. This procedure has been an active area of research for many years and has led to powerful algorithms (see Reference [12] for a survey of image processing techniques applied to discrete images) that will not be described here. However, we briefly present the main classes of surface triangulation algorithms based on discrete data.

2.2. From slices to surface triangulation

The data consists here in a series of m equally spaced slices (planes), each containing $n \times n$ pixels. A scalar value is attached to each pixel corresponding to a grey level and representing the density of the material tissue at this pixel location.

The main principle behind this class of algorithms is rather simple, running along contours, surface triangles are constructed between two adjacent slices. The desired surface triangulation is then obtained by merging altogether the sets of triangles resulting from pair treatment. Eventually, various criteria have been suggested to control the resulting surface triangulation: volume maximization, surface minimization, edge length or angle minimization, etc. [13]. The main problem that arises with such a method is related to the fact that the number of closed contours may change from one slice to another, thus leading to a 'branching problem'. This can be solved in various ways. Authors have suggested approaches based on heuristic assumptions favourizing for instance vertical connections, others have focussed on more global technique to generate directly a polyhedral representation (using a Delaunay-based reconstruction algorithm).

Figure 1 illustrates a surface triangulation obtained using a Delaunay-based reconstruction algorithm. In this technique, two cross sections are connected using 2D Delaunay triangulations. Each polygonal contour is the piecewise linear approximation of the surface. Once the triangulation has been constructed, a colouring scheme can be used to identify external from internal triangles depending on whether they are outside or inside the contours [14].

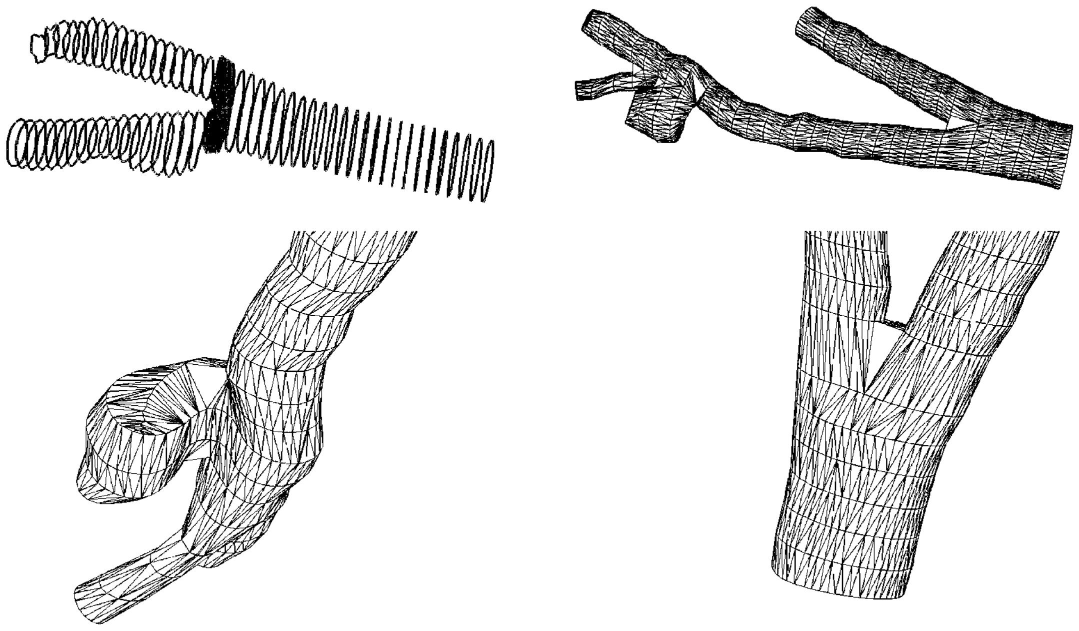
Figure 1. Delaunay-based surface reconstruction algorithm used for connecting series of parallel slices of an aneurysm. Bottom: local enlargements showing the surface elements near the singularities of the surface (mesh courtesy of E. Saltel, Gamma project, INRIA-Rocquencourt).

2.3. *Triangulation of point clouds*

In this technique, there is no need for the image processing stage, as the points are assumed to lie on the surface. The data consists in a (sparse) set of unorganized points possibly supplied with normal directions (Figure 2, left-hand side). Reconstructing a surface triangulation from a point cloud is a common problem in various fields of applications, like reverse engineering, image processing, computer graphics, etc. Data may come from a large range of devices, including laser range scanners or implicit surfaces, for instance. This topic has received much attention in the last few years. The main issues are related to the need to deal with arbitrary topology, non-uniform sampling and to produce geometric meshes with strong requirements: accurate piecewise linear approximation, smooth manifolds, etc.

One of the most robust method to address this problem is based on a two-stage method: at first a 3D Delaunay triangulation (made up of tetrahedra) is constructed over the set of points, then the method extracts a (smooth) polygonal surface by selecting appropriate faces (triangles) from the 3D triangulation based on geometric and topological considerations. Boissonnat and Cazals [9] advocate using the natural neighbour interpolation to define the reconstructed surface. This approach involves constructing a Delaunay triangulation then using the dual Voronoi segments to eventually refine the initial triangulation locally to match a sampling condition. Alternate, more heuristic techniques have also been proposed to remove faces from the Delaunay triangulation.

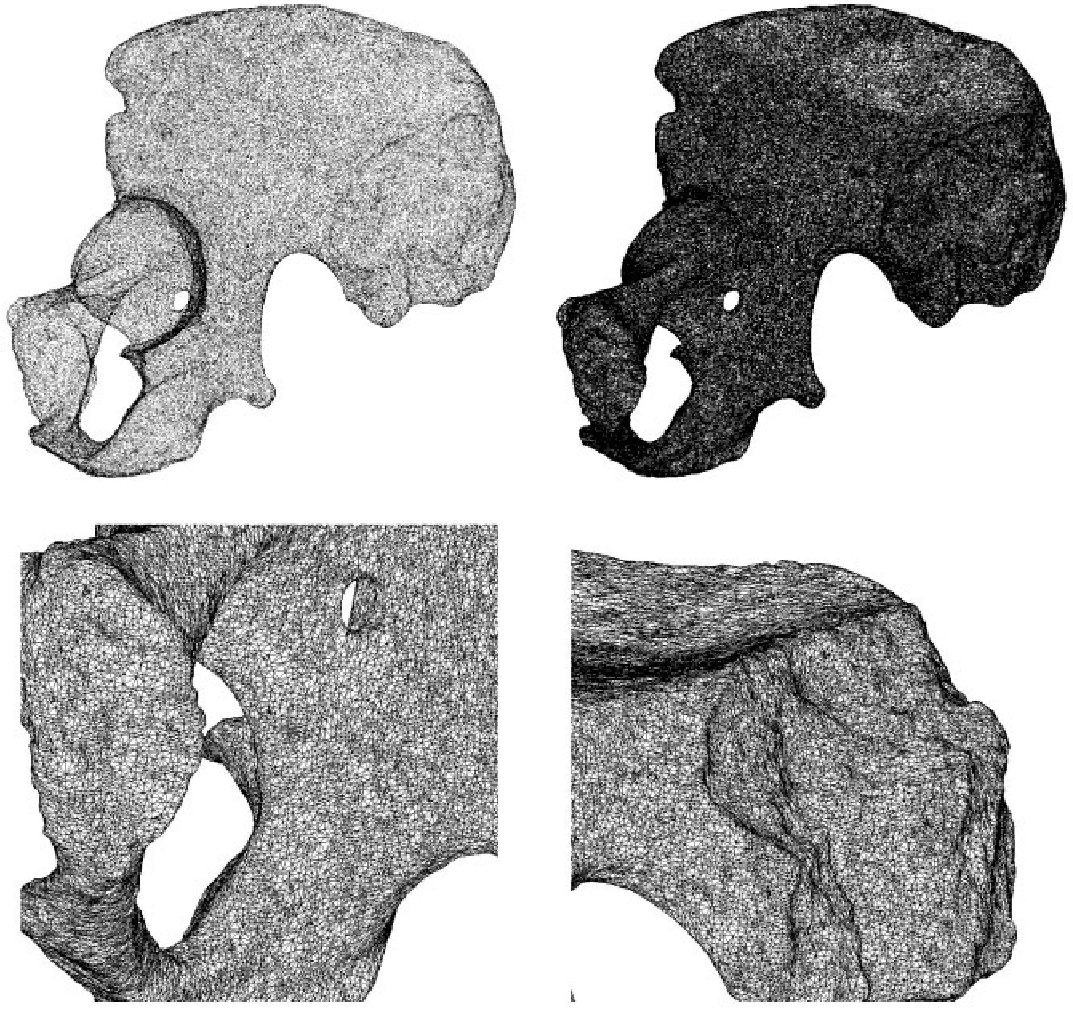

Figure 2. Top: initial point cloud and reconstructed surface triangulation based on a Delaunay tetrahedralization. Bottom: local enlargements showing the elements of the surface triangulation (mesh courtesy of P.L. George, Gamma project, INRIA-Rocquencourt).

Figure 2 (right-hand side) presents the result of this type of algorithm in the case of a relatively dense and uniformly sampled point cloud.

2.4. Isosurface triangulation

In this approach, the data consists in using a signed distance function and to compute its zero set. In other words, the surface is seen as a level surface of an implicit surface (i.e. an iso-surface) defined over the entire embedding space. With 3D grey level images (such

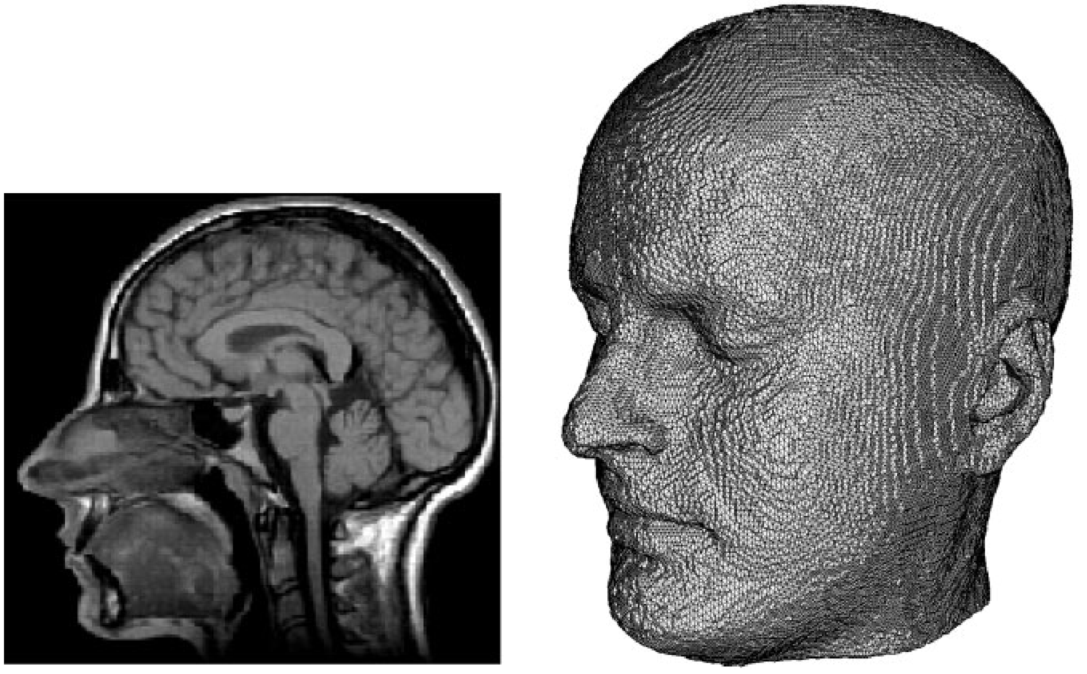

Figure 3. From 3D image to surface triangulation: slice through a volumetric MRI image (left-hand side) and result of a 'Marching-Cubes' surface reconstruction algorithm (right-hand side) (data courtesy of M. Seppa, Low Temp. Lab., Helsinki Univ. Tech., Finland).

as those provided by MR imaging systems), the density function can be seen as a discrete implicit function whose value is known only at sampled points (the voxels). Such methods have been successfully applied to surface triangulations by Lorensen and Cline [8], that designed the well-known 'Marching-Cubes' algorithm, and then updated or modified by various contributors [32, 35].

The nicest feature of this algorithm is related to its, almost biblical, simplicity. Given a voxel defined as a set of 8 pixels forming a hexahedral cell, the method consists in analysing this cell so as to determine whether it is intersected by the implicit surface. This check is simply based on the evaluation of the sign of the function at the pixels: a sign change along an edge cell indicating that the surface cross the edge between the two pixels. The location of the intersection point can be determined using interpolation schemes. This results in a series of possible configurations ($2^8 = 256$) that can be reduced to a minimal set of pre-defined patterns (16) after rotation and symmetry. Although several patterns correspond to ambiguous configurations they can be removed by introducing more information (for instance using additional sampling points inside a voxel).

Figure 3 illustrates a surface triangulation obtained using a 'Marching-Cubes' like algorithm. Notice that the surface tends to be jagged at a scale of a voxel because of the axis aligned voxel faces. This artefact can be removed using smoothing techniques described in the following section.

Having constructed a surface triangulation using any of the methods described above, we now turn to the problem of extracting a computational mesh from such a triangulation. This involves first generating a 'reference mesh', i.e. a mesh representing an accurate piecewise linear approximation of the surface, having an optimal (e.g. minimal) number of elements that can be referred to as the model for further treatments.

3. CONSTRUCTION OF A GEOMETRIC MESH

Once the surface triangulation has been generated as described in the previous section, the geometry of the domain is almost captured. However, each of the reconstruction methods described in the previous section usually suffers at least one of the following major drawbacks:

(i) noisy data (points are 'off' the surface),
(ii) unnecessary large number of polygons (triangles),
(iii) rather poor element shape quality.

Therefore, a post-processing stage is highly required to construct a geometric mesh from the input surface tessellation. In particular, a non-shrinking mesh smoothing procedure is introduced to remove the artefacts of 'Marching-Cubes' algorithms. Then, a mesh optimization procedure is carried out to remove redundant nodes or elements and to insert missing pieces of information. In this section, we show how these steps are indeed part of a global meshing scheme to produce a geometric mesh. Before discussing a general method to extract a geometric from a given triangulation, let us define more precisely the concept of a geometric mesh.

3.1. Definition of a geometric mesh

Let consider a bounded domain Ω in R^3 described by its boundary Σ. A *regular mesh* of Ω is a mesh for which all elements are regular (equilateral when speaking about triangles). As the existence of such a mesh is not proved, we will consider as regular, the 'best' mesh that one can construct. From a practical point of view, two types of surface meshes can be considered:

- *uniform meshes*, in which the element size is constant throughout the whole domain and
- *geometric meshes*, in which the element size is locally adapted to the surface curvatures.

Notice that having a uniform mesh does not guarantee anything about the quality of the surface approximation, except being infinitely refined. A geometric mesh though, is a mesh adapted to the geometry of the surface, the element size being closely related to the surface curvatures. In the context of surface meshing, having a geometric continuity of level 1 is usually considered sufficient [2].

Definition 3.1
A surface mesh is said to be 0-*order geometric* if each triangle is close to the surface (approximation property $P0$). A surface mesh is said to be 1-*order geometric* if the plane supporting each triangle is close to the tangent plane at each point of Σ close to the triangle (smoothness property $P1$).

Actually, the approximation property $P0$ relates the measure of the gap G_1 between the points of the triangle and the corresponding points on the portion of the surface discretized by the triangle. On the other hand, the smoothness property $P1$ is related to the measure of the gap G_2 between the normal of the triangle and the normal at any point of the portion of the surface discretized by the triangle. Usually, two requirements are added to these properties, asking for a minimal number of triangles in the triangulation and controlling the element shape quality, the latter being especially important in the context of numerical simulations.

The analysis of the surface curvatures allows to minimize the deviation between the tangent planes of the piecewise linear interpolation and that of the true surface [34]. The two gaps G_1 and G_2 can be bounded by a given tolerance provided the triangle size at each vertex $h(P)$ is proportional to the two radii of curvatures. A so-called discrete geometric metric $\mathscr{H}_{\mathrm{ref,g}}$ can be defined in the tangent plane of each vertex P of the geometric reference mesh $\mathscr{M}_{\mathrm{ref,g}}$ as:

$$\mathscr{H}_{\mathrm{ref,g}} = (h_1 \;\; h_2)\begin{pmatrix} \dfrac{1}{(\alpha\rho_1(P))^2} & 0 \\ 0 & \dfrac{1}{(\eta(\alpha,\rho_1(P),\rho_2(P))\rho_2(P))^2} \end{pmatrix}\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = 1$$

where α is a coefficient related to the given tolerance ε and $\eta(\alpha,\rho_1(P),\rho_2(P))$ is a function depending on α, $\rho_1(P)$ and $\rho_2(P)$ (the two radii of principal curvature) to guarantee a deviation identical in both principal directions.

3.2. Mesh smoothing procedure

As pointed out, most surface triangulations constructed with an iso-surface reconstruction algorithm are usually faceted. It is thus necessary to smooth the surface before proceeding to the generation of a geometric mesh or the simplification procedure described below may fail to remove small artefacts. Most smoothing procedures suffer several problems. One of the most annoying one is related to the *shrinkage* of the initial surface. As the smoothing techniques are usually carried out a large number of times, a triangle may degenerate into a single point. For instance, a continuous Gaussian filter consists in convolving a vector function parametrizing the surface with a Gaussian kernel. This technique can be extended to the case of discrete surfaces [15]. In this case, the complexity of the algorithm is linear in the number of vertices. The new position of a vertex is computed as a weighted average of the positions of its first-order neighbours (i.e. the vertices connected to the vertex by an edge) and the process is repeated a number of times. To prevent shrinkage, two consecutive Gaussian smoothing steps are applied. After a first step with a positive scale factor λ, a second step is carried out through all vertices, but with a negative scale factor μ such that $0 < \lambda < -\mu$. In [15], this procedure has been proved producing a low-pass filter effect where surface curvature takes the place of frequency. Special care must be taken in the algorithm to preserve feature lines (ridges, borders, etc.). Figure 4 shows the result of the Gaussian smoothing procedure on a reference mesh generated by a 'Marching-Cubes' algorithm.

3.3. Geometric mesh simplification

As indicated before, the initial reference mesh $\mathscr{M}_{\mathrm{ref}}(\Sigma)$ is, in general, not suitable for describing the surface geometry accurately, for at least two reasons. Indeed, it usually contains (i) a prohibitive number of (redundant) elements as well as (ii) geometric artefacts (small features) that need to be removed. To overcome these problems, the approach we advocate here consists

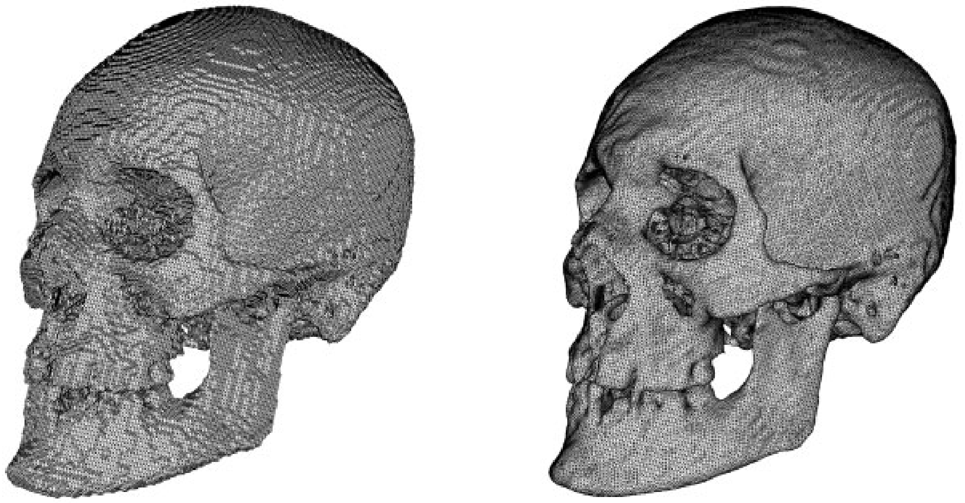

Figure 4. Illustration of the mesh smoothing procedure (data courtesy Epidaure, INRIA-Sophia).

in simplifying the surface triangulation using local mesh modifications. To this end, we introduce a global tolerance envelope around the surface (at a given Hausdorff distance) to control the simplification operations, similar to that proposed by Cohen *et al.* [16, 17] and Borouchaki [18]. The aim is to remove redundant elements (that do not really contribute to the geometric approximation of the surface) as well as small features, while preserving the underlying surface geometry.

3.3.1. Discrete Hausdorff distance. The simplification algorithm described hereafter is based on the evaluation of the Hausdorff distance between the initial triangulation $\mathscr{M}_{\mathrm{ref}}$ and the geometric reference mesh $\mathscr{M}_{\mathrm{ref,\,g}}$. Let recall that the distance from a point X in R^3 to a bounded domain F is given by

$$d(X, F) = \inf_{Y \in F} d(X, Y)$$

where $d(.,.)$ denotes the usual Euclidean distance. Let F_1 and F_2 be two bounded domains in R^3, if $\rho(F_1, F_2)$ measures the quantity $\sup_{X \in F_1} d(X, F_2)$, thus the Hausdorff distance $d_{\mathrm{H}}(F_1, F_2)$ between F_1 and F_2 is defined as

$$d_{\mathrm{H}}(F_1, F_2) = \sup(\rho(F_1, F_2), \rho(F_2, F_1))$$

This continuous measure is rather expensive to compute (as it must be checked for all points lying in the plane of the triangles), we introduce a discrete approximation of the Hausdorff measure, following the ideas suggested by Borouchaki [18].

To ensure a proximity property, a global tolerance envelope around the surface (at a given Hausdorff distance δ) is introduced on both sides of the reference surface mesh $\mathscr{M}_{\mathrm{ref}}$. Then, it is sufficient to check that each triangle resulting from the mesh optimization procedure remains contained within this envelope. Moreover, to ensure the smoothness property P1, another constraint is added yielding to satisfy both the containment and the smoothness requirements:

$$d_{\mathrm{H}}(K, \mathscr{M}_{\mathrm{ref}}) \leqslant \delta \quad \text{and} \quad \langle v_k(K), v(K) \rangle \geqslant \cos(\theta)$$

where $v_k(K)$ is the unit normal to the surface at the vertex k of triangle K and $v(K)$ is the unit normal to K, ($\langle , \rangle$ denoting the standard scalar product in R^3).

3.3.2. Simplification procedure. Formally speaking, the proposed method consists in removing iteratively the vertices from the reference mesh. To this end, two mesh modification operations are carried out: the edge removal and the edge flipping, provided the two previous requirements are not violated.

The edge flipping operation consists in replacing the two triangles sharing an edge by the alternate configuration (the two other triangles constructed on the same set of four points). This operation is applied if the two triangles are strictly coplanar to preserve the surface curvature sign.

The edge collapsing operation consists in removing a vertex P from the mesh, thus creating a hole, and to reconstruct the mesh of the *ball* of P (the first order neigbours) by retriangulating the cavity. As computing the true Hausdorff distance is a rather tedious and expensive task, it is possible to bound this measure by keeping track of the deformations hierarchically. With each triangle K^i of the current mesh $\mathscr{M}^i$ can be associated a majoration of its Hausdorff distance with respect to $\mathscr{M}_{\mathrm{ref}}$:

$$h(K^{i+1}(Q)) = d_{\mathrm{H}}(K^{i+1}(Q), \mathscr{B}^i(P)) + \max_i h(K^i(P)).$$

Hence, the triangle $K_i^{i+1}(Q)$ does not degrade the piecewise linear approximation of the surface geometry if it is such that $h(K_i^{i+1}(Q)) \leqslant \delta$ (when the vertex P is collapsed on the vertex Q). To conform to the two geometric properties P0 and P1, the edge collapsing operation can be carried out if

$$d_{\mathrm{H}}(K^{i+1}, \mathscr{M}_{\mathrm{ref}}) \leqslant \delta \quad \text{and} \quad \langle v_k(K^{i+1}(Q)), v(K_i^{i+1}(Q)) \rangle \geqslant \cos(\theta)$$

where θ is a prescribed tolerance angle (defining a local cone of tolerance at each vertex).

As no attention is explicitly paid on the triangle shape quality, degenerated (flat) triangles may be created. To avoid this problem, a control of the shape quality is introduced by bounding the quality degradation by a tolerance value β. The mesh quality will then be preserved from mesh $\mathscr{M}^i$ to mesh $\mathscr{M}^{i+1}$ if, for each new triangle K^{i+1}:

$$q(K^{i+1}(Q)) \geqslant \beta \min_i q(k^i(P))$$

$q(.)$ denoting any monotone measure of the shape quality of a triangle [14].

Finally, in order to optimize the element shape quality of the resulting geometric mesh, a smoothing procedure is applied on all mesh vertices. The node displacements vectors are usually computed using a discrete Laplacian. Points are moved step by step toward an optimal position (i.e. a position corresponding to a configuration of triangles of optimal shapes). Figure 5 shows examples of geometric reference meshes obtained using this simplification procedure.

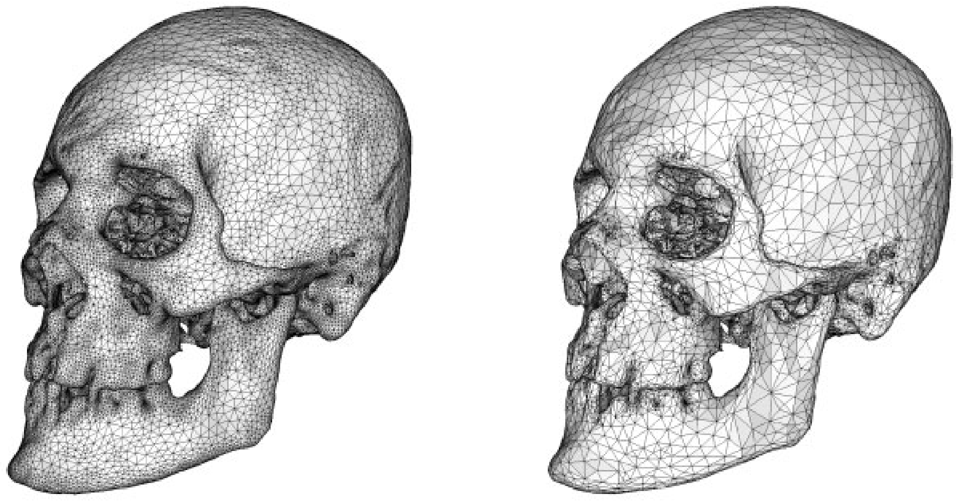

Figure 5. Two instances of geometric meshes corresponding to the Hausdorff simplification procedure for tolerance values of $\delta = 0.1\%$ and $\delta = 0.5\%$ of the bounding box size.

4. GENERATION OF A COMPUTATIONAL MESH

The geometric reference mesh $\mathscr{M}_{\mathrm{ref,\,g}}$ constructed according to the scheme described in the previous section is generally not directly suitable for computational purposes. Indeed, this mesh represents an accurate piecewise linear approximation of the surface geometry (conforming a prescribed error tolerance). However, numerical simulations require an additional control of the mesh density (the concentration of vertices in each region of the mesh) and the mesh gradation (the size variation between neighbouring triangles). As mentioned previously, the mesh density in a geometric mesh is only related to the local surface curvatures. But, depending on the sampling of the data, the local density of the geometric mesh may be not sufficient, in other words, the mesh is locally lacking vertices to fully capture the behaviour of the surface geometry. It is possible to measure the adequacy of the mesh with respect to the geometric metric, using the notion of average edge length. As the geometric metric prescribes the triangle sizes, the aim is to generate a mesh in which each edge has an average length close to the desired size. Such a mesh is called a *unit mesh*. We will show in this section that the desired computational mesh is indeed a unit mesh.

4.1. Metric issues

The metric $\mathscr{H}$ is defined at point P as a 3×3 symmetric definite positive matrix. For such a metric, the scalar product of two vectors in the Euclidean space R^3 can be defined as

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathscr{H}} = {}^{\mathrm{t}}\mathbf{u}\mathscr{H}\mathbf{v}$$

as well as the Euclidean norm as

$$\|\mathbf{u}\|_{\mathscr{H}} = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle_{\mathscr{H}}} = \sqrt{{}^{t}\mathbf{u}\mathscr{H}\mathbf{u}}$$

Hence, the distance between two points A and B reads as

$$d_{\mathscr{H}}(A, B) = l_{\mathscr{H}}(\overrightarrow{AB}) = \|\overrightarrow{AB}\|_{\mathscr{H}}$$

Considering that the metric tensor $\mathscr{H}$ usually changes from one point to another, leads to introduce the notion of the *average length* of a mesh edge as

$$l_{\mathscr{H}}(\overrightarrow{AB}) = \int_0^1 \sqrt{{}^t\overrightarrow{AB}\,\mathscr{H}(A + t\overrightarrow{AB})\overrightarrow{AB}}\,\mathrm{d}t$$

This notion of average length with respect to a metric tensor is closely related to the notion of the length of a parametrized arc if a Riemannian structure is induced over the domain by the metric $\mathscr{H}$. Practically, we can define a discrete metric tensor $\mathscr{H}$ at each mesh vertex such that the desired size of any vector in any direction around the vertex is equal to one: $\|\mathbf{u}\|_{\mathscr{H}} = 1$. Geometrically, this relation means that the geometric locus of the point P_i equidistant (at a distance 1) of point P describes an ellipsoid. As constructing a *unit mesh* with respect to a given metric $\mathscr{H}$ is not always possible, we consider that a mesh is conforming a metric specification $\mathscr{H}$ if for any edge AB we have

$$\frac{1}{\sqrt{2}} \leqslant l_{\mathscr{H}}(AB) \leqslant \sqrt{2}$$

The size variation in the mesh can be defined as the ratio between the edge lengths for each pair of adjacent edges [19]. Along an edge AB, the size variation writes

$$c(AB) = \max\left(\frac{h(A)}{h(B)}, \frac{h(B)}{h(A)}\right)^{l_{\mathscr{H}}(PQ)^{-1}}$$

where $h(A)$ represents the desired size at A and h is a monotone function such that $h(0) = h(A)$ and $h(1) = h(B)$. Usually, to obtain well-shaped triangles on to control the mesh density, the size variation must be bounded by some tolerance value β for the mesh edges. The problem turns out to find a proper coefficient η to reduce the largest size along the edge in order to have $c(AB) = \beta$. The solution consists in setting

$$\eta < \left(\frac{\beta}{c(AB)}\right)^{l_{\mathscr{H}}(AB)}$$

With this procedure, the geometric metric map $\mathscr{H}_{\mathrm{ref,\,g}}$ can be modified to account for the desired mesh gradation.

It is often desirable to find an interpolated metric along a segment AB, knowing the metrics $\mathscr{H}_1$ and $\mathscr{H}_2$ at the two edge endpoints. To this end, we can use a linear interpolation scheme, considering the parametrization $\gamma(t) = A + t\overrightarrow{AB}$ of the segment AB:

$$\mathscr{H}(t) = ((1-t)\mathscr{H}_1^{-1/2} + t\mathscr{H}_2^{-1/2})^{-2}, \quad 0 \leqslant t \leqslant 1$$

which corresponds to a linear variation of the sizes along the segment AB [20]. This scheme will be useful, for instance, when splitting a mesh edge into two sub-edges and inserting a new mesh vertex, to determine the metric at this new vertex.

4.2. Definition of the geometric support

As already mentioned, the geometric reference mesh $\mathscr{M}_{\mathrm{ref,g}}$ is intended to remain the sole available representation of the geometry. In order to construct a computational mesh $\mathscr{M}$, adapted to the metric $\mathscr{H}_{\mathrm{ref,g}}$, new vertices must be created and inserted in the geometric mesh, so as to conform to the size specifications of $\mathscr{M}$. Given a non-singular point P, the aim of the geometric support is to return the location of the closest point P_Σ on the surface Σ as well as the normal to the surface at P_Σ [21].

In our approach, the construction of the geometric support requires defining a piecewise planar surface, at least G^1 continuous, based on the geometric mesh $\mathscr{M}_{\mathrm{ref,g}}$ on each triangle of $\mathscr{M}_{\mathrm{ref,g}}$. This approach is partly inspired of the method suggested by Walton and Meek [22], that consists in defining a network of boundary curves to the patches, as well as in defining the transverse tangent planes to these curves using only the normals to the surface at the triangle vertices. Each patch is then defined, from its boundary, independently from the adjacent triangles, while preserving the tangent planes requirement using Gregory's polynomials [23]. A nice feature of this technique is that it is able to define the network of curves from the sole data of the vertex normals, i.e. with a minimum of reliable information. Each boundary curve is represented as a cubic polynomial, the principal normals at the endpoints corresponding to the normals to the surface. The transverse tangent planes to the boundary curves are generated by the tangent vectors to the curve points and using a quadric interpolation scheme of the binormals at the endpoints. Thus, knowing only the two normals at the endpoints of a mesh edge is indeed sufficient to define a boundary curve as well as the transverse tangent plane to this curve. Finally, based on the transverse tangent planes to each patch, a Gregory's patch is defined using a more classical approach. We have also extended this technique to be able to deal with discontinuities (ridges, corners, etc.) present in realistic geometric models.

4.3. Surface mesh optimization

To generate a computational mesh $\mathscr{M}$, the geometric reference mesh $\mathscr{M}_{\mathrm{ref,g}}$ is optimized iteratively using local mesh modifications. By interpolating the discrete metric map $\mathscr{H}_{\mathrm{ref,g}}$ defined at the vertices of $\mathscr{M}_{\mathrm{ref,g}}$, a continuous metric map $\mathscr{H}^{\mathrm{c}}_{\mathrm{ref,g}}$ can be defined, thus leading to define a Riemannian structure in $\Omega \subset \mathrm{R}^3$. The problem becomes then to construct a unit mesh $\mathscr{M}$ with respect to $\mathscr{H}^{\mathrm{c}}_{\mathrm{ref,g}}$. As indicated in the previous section, the mesh $\mathscr{M}$ is a unit mesh if, for any edge AB, the length $l_{\mathscr{H}^{\mathrm{c}}_{\mathrm{ref,g}}}(AB)$ computed in the underlying Riemannian structure, is equal to one. From practical reasons, a mesh $\mathscr{M}$ such that

$$\frac{1}{\sqrt{2}} \leqslant l_{\mathscr{H}^{\mathrm{c}}_{\mathrm{ref,g}}}(AB) \leqslant \sqrt{2} \quad \forall AB \in \mathscr{M}$$

can be considered a unit mesh.

To generate the computational unit mesh $\mathscr{M}$ with respect to $\mathscr{H}_{\mathrm{ref,g}}$, the edges of the geometric reference mesh are analysed: all edges having an average length larger than one are split into unit sub-segments while the small edges (having a length smaller than one) are collapsed by merging their two endpoints together. Similar to the construction of a geometric mesh, the local mesh modification procedures are driven by shape and size quality improvements. This requires measuring the quality of the triangles with respect to the metric $\mathscr{H}^{\mathrm{c}}_{\mathrm{ref,g}}$. In a

Riemannian space, the quality $q(K)$ of a triangle K can be defined, following [24] as

$$q_{\mathscr{H}_{\mathrm{ref,g}}}(K) = \min_{1 \leqslant j \leqslant 3} Q^j_{\mathscr{H}}(K)$$

where $q^j_{\mathscr{H}}(K)$ is the shape quality of K in the Euclidian space defined by the metric $\mathscr{H}_{\mathrm{ref,g}}$ associated with the vertex j of K. To measure the quality $q^j(K)$, one has simply to transform the Euclidean space associated with the metric $\mathscr{H}_{\mathrm{ref,g}}$ specified at the vertex j into the usual Euclidean space, and then to consider the quality of the resulting transformed triangle $K^j(K)$ of K: $q^j_{\mathscr{H}_{\mathrm{ref,g}}}(K) = q_{\mathscr{H}_{\mathrm{ref,g}}}(K^j)$. Hence, it comes

$$q^j_{\mathscr{H}_{\mathrm{ref,g}}}(K) = 4\sqrt{3}\,\frac{\sqrt{\mathrm{Det}(\mathscr{H}^j_{\mathrm{ref,g}})}\,S_K}{\sum_{a_j \in K} l^2_{\mathscr{H}^j_{\mathrm{ref,g}}}(a_j)}$$

where $l_{\mathscr{H}^j_{\mathrm{ref,g}}}(a_j)$ is the Euclidean length of the edge a_j with respect to $\mathscr{H}^j_{\mathrm{ref,g}}$ and $\mathrm{Det}(.)$ denotes the determinant associated with the matrix of $\mathscr{H}^j_{\mathrm{ref,g}}$.

At the end of the mesh optimization stage, a smoothing procedure is applied on all mesh vertices. In this case, the optimal location of vertex P is computed based on the average edge lengths using a discrete Laplacian.

4.4. Volume mesh generation

At this point, we have been able to construct a unit surface mesh with respect to the discrete metric map $\mathscr{H}_{\mathrm{ref,g}}$, given an arbitrary surface triangulation $\mathscr{M}_{\mathrm{ref}}$. The construction of a volumic computational mesh is based on a classical approach [25]: at first an empty tetrahedral mesh (containing no internal point) is created using the Delaunay kernel, then this mesh is iteratively enriched by adding internal points. This second step is repeated until all edges have an average length equal to one with respect to $\mathscr{H}^{\mathrm{c}}_{\mathrm{ref,g}}$. At the end, a smoothing procedure is applied on the mesh vertices to improve the overall mesh quality. Again, this scheme based on edge length computations, works for both the isotropic and the anisotropic case. Figure 10 shows an example of a volumic mesh.

5. MESH ADAPTATION

After decades of research and development, the numerical simulations based on finite element/volume methods have reached a high level of maturity, allowing to resolve with a great deal of success a large range of engineering problems. In these numerical techniques, the approximation error can be related to the number of elements, the mesh density and the node locations [10]. In this context, the aim of mesh adaptation is to permit a better control of the error by creating meshes more adapted to the physical model (usually based on PDE's) as well as to its numerical resolution. Two issues are then especially relevant: the conception of accurate error estimates that are able to translate the approximation (or interpolation) error into metric maps and the development of mesh generation techniques governed by such metric maps. In this section, we will briefly introduce a geometric error estimate based on the interpolation error and we will explain how the mesh generation techniques described in the previous sections can be easily extended in the context of mesh adaptation.

5.1. A geometric error estimate

Let $\mathscr{S}(\Omega)$ denotes the exact solution of a PDE problem and $\mathscr{S}_i(\Omega)$ be the numerical finite element solution of this problem. The problem consists in computing the gap $e_i(\Omega) = \mathscr{S}(\Omega) - \mathscr{S}_i(\Omega)$ between $\mathscr{S}(\Omega)$ and $\mathscr{S}_i(\Omega)$ representing the error related to the finite element solution computed on a mesh $\mathscr{M}_i(\Omega)$ and then in deducing (usually using a majoration of this gap) a new mesh $\mathscr{M}_{i+1}(\Omega)$ such that the estimated gap between $\mathscr{S}(\Omega)$ and the solution interpolated on mesh $\mathscr{M}_{i+1}(\Omega)$ is bounded by a given tolerance value. The finite element $\mathscr{S}_i(\Omega)$ is not interpolant and moreover, it is not possible to guarantee that the solution $\mathscr{S}_i(\Omega)$ coincide with the exact solution $\mathscr{S}(\Omega)$ in at least one point of each element. Hence, it seems difficult to quantify the gap $e_i(\Omega)$. As described in References [2, 20], we introduce an indirect approach to measure the gap.

Let $\widetilde{\mathscr{S}}_i(\Omega)$ denotes the interpolation of $\mathscr{S}(\Omega)$ using the mesh $\mathscr{M}_i(\Omega)$ and $\widetilde{e}_i(\Omega)$ denotes the gap $(\mathscr{S}(\Omega) - \widetilde{\mathscr{S}}_i(\Omega))$ between $\mathscr{S}(\Omega)$ and $\widetilde{\mathscr{S}}_i(\Omega)$, $\widetilde{e}_i(\Omega)$ is the so-called *interpolation error* on $\mathscr{S}(\Omega)$ with respect to $\mathscr{M}_i(\Omega)$. In order to quantify the gap $e_i(\Omega)$, we assume the following relation:

$$\|e_i(\Omega)\| \leqslant C\|\widetilde{e}_i(\Omega)\|$$

where $\|\,.\,\|$ is a norm of R^3 and C a constant independent of $\mathscr{M}_i(\Omega)$. In other words, we assume that the finite element error is bounded by the interpolation error. This allows us to consider the following problem: given a mesh $\mathscr{M}_i(\Omega)$ and the interpolation $\widetilde{\mathscr{S}}_i(\Omega)$ of $\mathscr{S}(\Omega)$, generate a mesh $\mathscr{M}_{i+1}(\Omega)$ on which the interpolation error is bounded by a given tolerance value. As $\widetilde{\mathscr{S}}_i(\Omega)$ can be seen as a discrete representation of $\mathscr{S}(\Omega)$, the problem turns out to characterize the meshes on which the interpolation error is bounded. It has been proved that the analysis of a 'measure' of the interpolation error leads to define an anisotropic metric map which prescribes element sizes and directions [26]. To measure the interpolation error, we consider the discrete L_∞ norm of the error defined in triangle K as

$$\|\widetilde{e}_i(K)\|_{\infty,K} = \max_{x\in K} |\widetilde{e}_i(x)|$$

where x covers the points of K. In Reference [2], we show that this error is related to the absolute values of the Hessian of the solution $\mathscr{S}_i(\Omega)$ at the mesh vertices and can be written as

$$\|\mathscr{S}_i - \widetilde{\mathscr{S}}_i\|_{\infty,K} \leqslant \tfrac{9}{32} \max_{x\in K} \max_{\mathbf{v}\subset K} |\langle \mathbf{v}, H_{\mathscr{S}_i}(x)\,\mathbf{v}\rangle|$$

The Hessian matrix being symmetric, it can be decomposed as follows: $H = \mathscr{R}\Lambda\mathscr{R}^{-1}$, where $\mathscr{R}$ (resp. Λ) is the eigenvectors (resp. eigenvalues) matrix of the Hessian matrix. To define a positive symmetric definite matrix (to define a tensor metric), we pose

$$|H| = \mathscr{R}|\Lambda|\mathscr{R}^{-1} \quad \text{with} \quad \Lambda = \begin{pmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{pmatrix}$$

We can now consider the majoration

$$\|\mathscr{S}_i - \widetilde{\mathscr{S}}_i\|_{\infty,K} \leqslant c \max_{x\in K} \max_{\mathbf{v}\subset K} \langle \mathbf{v}, |H_{\mathscr{S}_i}(x)|\mathbf{v}\rangle$$

In the numerical simulations, we are willing to equi-distribute the interpolation error in all directions over the mesh elements. Let ε be the maximal error accepted, we must have

$$\max_{x\in K} \max_{\mathbf{v}\subset K} \langle \mathbf{v}, |H_{\mathscr{S}_i}(x)|\mathbf{v}\rangle \leqslant \frac{\varepsilon}{c} \quad \text{or} \quad \max_{x\in K} \max_{\mathbf{v}\subset K} \left\langle \mathbf{v}, \frac{c}{\varepsilon}\, |H_{\mathscr{S}_i}(x)|\mathbf{v}\right\rangle \leqslant 1$$

For practical reasons, the minimal $h_{\min}$ (resp. maximal $h_{\max}$) sizes needs to be bounded, hence we define the following metric tensor:

$$\mathscr{H} = \mathscr{R}\tilde{\Lambda}\mathscr{R}^{-1} \quad \text{with} \quad \tilde{\Lambda} = \begin{pmatrix} \tilde{\lambda}_1 & 0 & 0 \\ 0 & \tilde{\lambda}_2 & 0 \\ 0 & 0 & \tilde{\lambda}_3 \end{pmatrix}, \quad \tilde{\lambda}_i = \min\left(\max\left(\frac{c|\lambda_i|}{\varepsilon}, \frac{1}{h_{\max}^2}\right), \frac{1}{h_{\min}^2}\right)$$

Notice that this metric tensor provides an anisotropic metric, the principal directions are given by the eigenvectors and the corresponding sizes by the eigenvalues. Actually, this metric tensor prescribes a unit average length in all directions around the mesh vertices. Moreover, this error estimate can be legitimately called a geometric estimate. Indeed, the solution can be seen as a (hyper)surface and the desired unit mesh is then a geometric mesh of this surface (the element size being related to the local second order derivatives of the surface, the principal curvatures).

5.2. General scheme

At each iteration stage, the adaptation scheme consists mainly in creating the mesh $\mathscr{M}_{i+1}(\Omega)$ from the mesh $\mathscr{M}_i(\Omega)$ and a physical solution $\mathscr{S}_i(\Omega)$ associated. Using the interpolation error estimate introduced above, a discrete metric map $\mathscr{H}_i(\Omega)$ is defined over the mesh vertices, prescribing the desired element sizes and directions in an optimal mesh for the solution $\mathscr{S}_i(\Omega)$. In our context, the approach retained is based on

- the construction of the unit surface mesh $\mathscr{M}_{i+1}(\Gamma)$ (with respect to $\mathscr{H}_i(\Gamma)$) using a mesh optimization procedure,
- the generation of the unit volume $\mathscr{M}_{i+1}(\Omega)$ using a Delaunay-based approach.

The pair (mesh,metric) forms the control space that is used to control the generation of adapted meshes. Conceptually, this scheme is not different from the general scheme used to create a computational mesh. The main difference lies in the metric definition: the metric $\mathscr{H}_i$ is the combination (the intersection, mathematically) of the geometric metric introduced in Section 3.1 and the computational metric supplied by the error estimate. The adaptation process stops when a stable solution is achieved for the couple (mesh,solution).

6. APPLICATION EXAMPLES

In this section, we present several examples of geometric and computational meshes generated using the approach described in the previous sections. We describe here three series of examples:

Table I. Statistics about the geometric surface meshes.

Meshes	np	ne	$\mathrm{np_g}$	$\mathrm{ne_g}$	δ	t (s)
$\mathrm{trolloc}_1$	22 677	45 357	8606	17 215	0.1	2.9
$\mathrm{trolloc}_2$	22 677	45 357	3492	6987	0.5	2.3
head_1	67 108	134 212	6742	13 480	0.1	6.8
head_2	67 108	134 212	2254	4504	0.5	4.9
$\mathrm{hip25}_1$	132 538	265 084	9803	19 615	0.1	16.5
$\mathrm{hip25}_2$	132 538	265 084	4415	8239	0.5	15.5

the first series is related to the generation of geometric reference meshes, the second series to computational meshes and the third series present several adapted meshes. All surface meshes have been generated using `Yams`, a surface remeshing software developed at INRIA-Rocquencourt by the author [27]. All volume meshes have been generated using `GHS3D`, a Delaunay-based tetrahedral mesh generation software [28].

6.1. Geometric meshes

In computer graphics, efficient visualization or geometry processing relies on the generation of geometric meshes. The aim of geometric meshing is to create a reference instance (a reference mesh) of the geometry of the model, an *optimal* mesh that suits better the visualization process. For a fixed number of points np, an optimal geometric mesh $\mathscr{M}_{\mathrm{ref,g}}$ approaches the (smooth) surface geometry at best using anisotropic (stretched) elements, i.e. following the eigenvectors of the curvature tensor of the surface [29]. Geometric meshes generated using the Hausdorff distance usually present these desired anisotropic features, as illustrated in the figures hereafter.

We present several examples of geometric meshes corresponding to various Hausdorff simplifications. For each example, we will report: the number of vertices np and elements ne of the initial triangulation, the number of vertices $\mathrm{np_g}$ and elements $\mathrm{ne_g}$ of the geometric meshes, the tolerance value δ (expressed in percentage of the bounding box size) used to control the mesh simplification process, the and the cpu time in seconds required to create the mesh (including the I/O on a HP PA-RISC 500 Mhz workstation) (Table I).

Figure 4 illustrates the effect of the smoothing procedure to remove the jagged surface at the level of the voxels. In this case, 150 iterations of a Gaussian filtering algorithm have been applied, the scale factors used were, respectively: $\lambda = 0.33$ and $\mu = 0.34$. The 'smooth' mesh (right-hand side) contains exactly the same number of points than the initial triangulation (left-hand side). In Figure 5 are represented two geometric meshes, skull_1 and skull_2, corresponding to two levels of simplification (Hausdorff distances $\delta = 0.1\%$ and $\delta = 0.5\%$ of the bounding box size). In this case, the size variation has been truncated by specifying a minimal (resp. maximal) size at vertices, to preserve the overall element shape quality and to bound the anisotropic aspect ratio.

Figures 6 and 7 illustrate two series of mesh simplification based on the Hausdorff distance with no explicit control on the maximal element size (a shape quality control has been applied to avoid the creation of degenerated 'skinny' triangles). The resulting geometric meshes (for different tolerance values) exhibit anisotropic features corresponding to the intrinsic properties of the underlying surface.

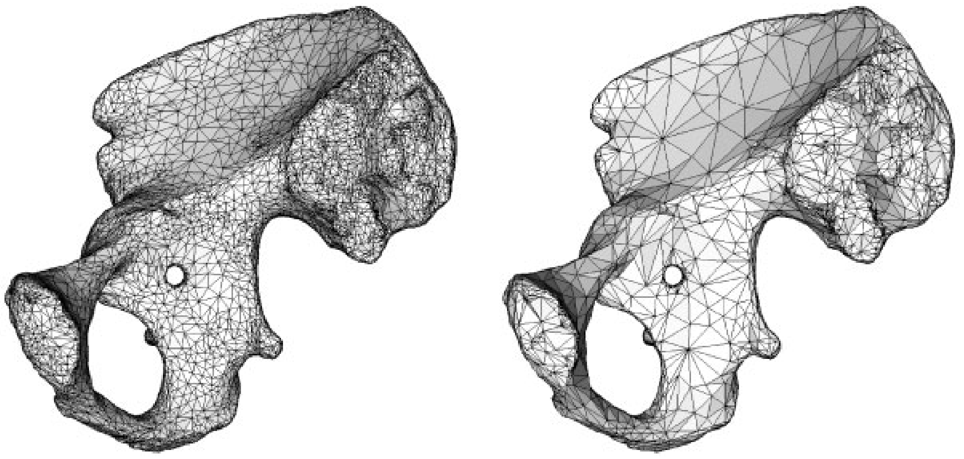

Figure 6. Initial surface triangulation (left) and geometric mesh simplification based on the Hausdorff distance for the tolerances $\delta = 0.1\%$ (left) and $\delta = 0.5\%$ (right) of the bounding box size.

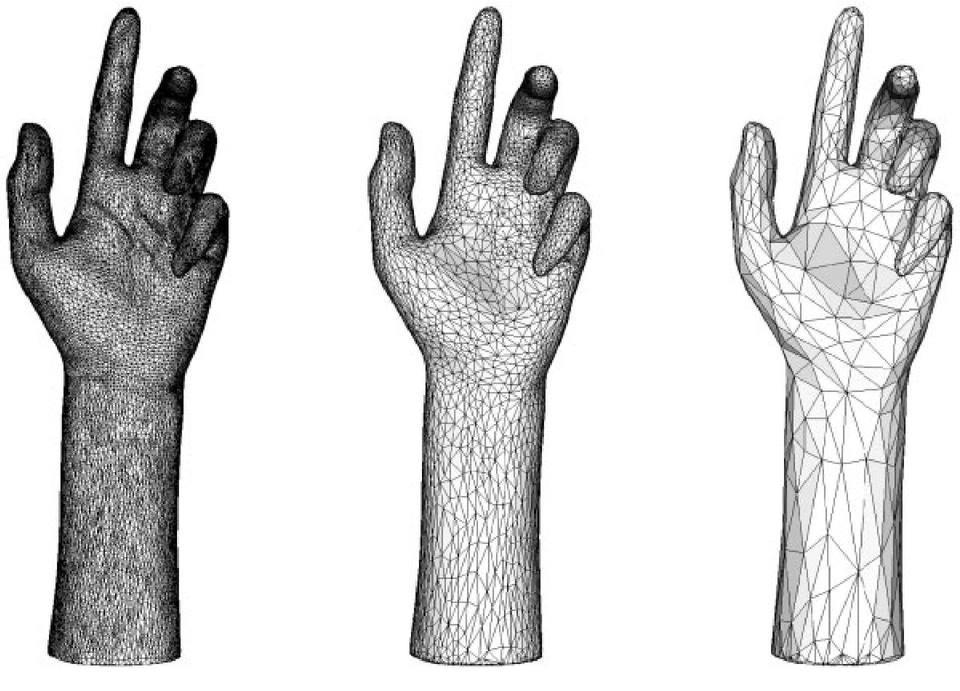

Figure 7. Initial surface triangulation (left) and geometric mesh simplification based on the Hausdorff distance for the tolerances $\delta = 0.1\%$ (middle) and $\delta = 1.5\%$ (right) of the bounding box size.

Table II. Statistics about the computational meshes.

Meshes	np	ne	$\mathrm{np_c}$	$\mathrm{ne_c}$	β	$Q_{\mathscr{H}}(\mathscr{M})$	$l_{\mathscr{M}}$	t (s)
$\mathrm{mandibula_c}$	9730	20 146	120 091	244 203	1.4	1.5	0.95	35.4
$\mathrm{trolloc_c}$	22 677	45 357	31 132	62 265	1.4	1.3	0.9	20.1
$\mathrm{head_c}$	67 108	134 212	35 812	71 620	1.2	0.95	1.4	27.0
$\mathrm{hip25_c}$	132 538	265 084	56 947	113 903	1.3	1.4	0.9	48.0
$\mathrm{joint_c}$	137 062	274 120	3660	7316	1.3	1.4	0.85	20.0

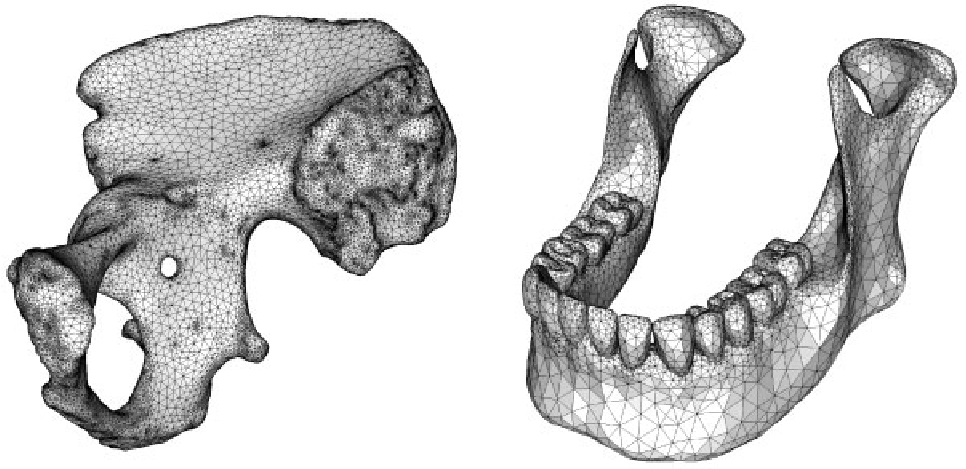

Figure 8. Two examples of computational isotropic surface meshes with control on the element (shape and size) quality as well as on the mesh gradation.

6.2. Computational meshes

As mentioned in Section 4, a computational mesh is a unit mesh with respect to the geometric metric tensor. This Riemannian metric tensor prescribes at each vertex of the surface mesh the orientation and stretching of the locally desired mesh elements. For computational purposes, additional requirements can also be specified, related to the element shape quality, the mesh density and the mesh gradation.

Table II reports statistics about the series of computational meshes, np, ne, $\mathrm{np_c}$, $\mathrm{ne_c}$ represent the number of vertices and elements of the initial triangulation and that of the computational mesh, $Q_{\mathscr{H}}(\mathscr{M})$ is the average mesh quality (measured in the metric $\mathscr{H}$), $l(\mathscr{M})$ denotes the average edge length (w/r to the metric $\mathscr{H}$), β represents the user-specified mesh gradation value, and t is the cpu time in seconds (including the I/O on a HP PA-RISC 550 Mhz workstation). As the aim is to create unit meshes, one can easily notice that the average edge length with respect to the geometric metric tensor is, in some cases, not very close to one. This result can be explained by geometric constraints that may not be compatible with the specified minimal size at vertices prescribed.

Figures 8 and 9 present various examples of computational isotropic and anisotropic meshes. Figure 10 illustrates an example of volumic computational mesh made up of tetrahedral

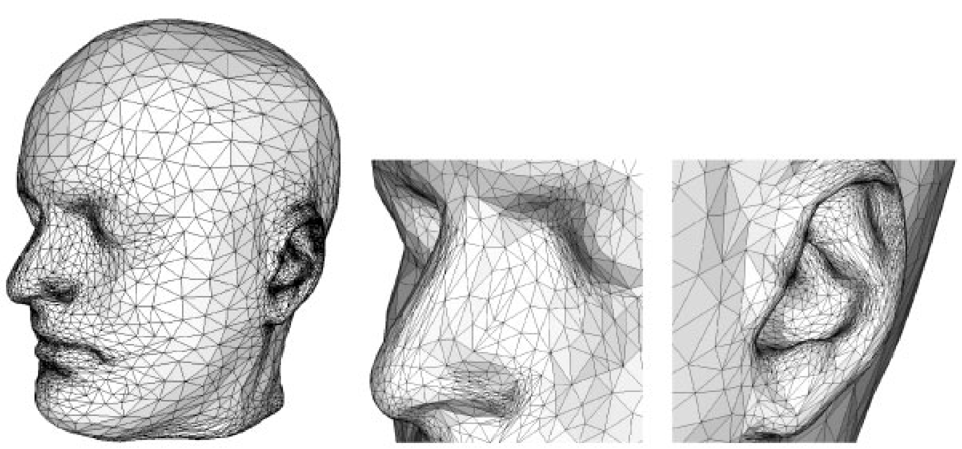
Figure 9. Example of an anisotropic computational surface mesh and local enlargements on the anisotropic features.

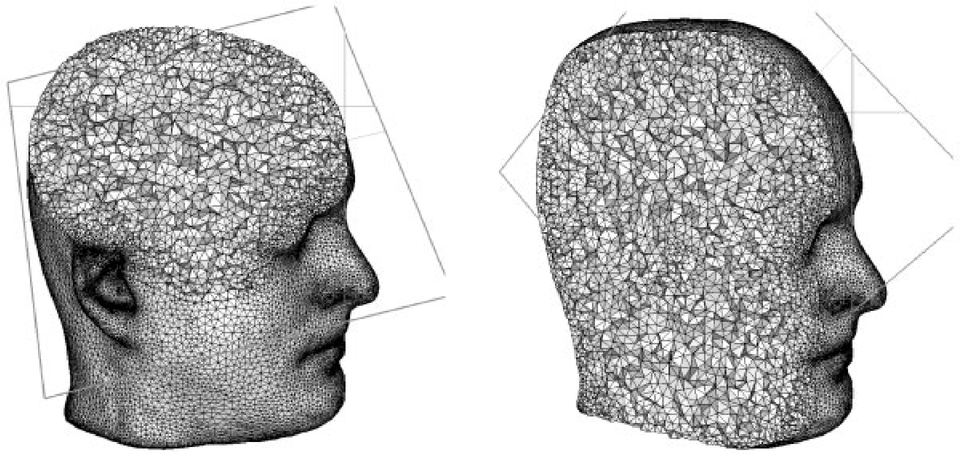
Figure 10. Two cutting planes through a volumic computational mesh corresponding to the head model. The metric tensor in the volume is induced by the geometric metric tensor $\mathscr{H}_{\mathrm{ref,g}}$ defined on the boundary of the domain.

elements.The cutting planes show that the internal element mesh density is related to the surface mesh density. A dilution coefficient has been introduced so as to slightly increase the element size in the volume (mesh gradation control).

The last example concerns a numerical simulation in biomedical CFD. Digital angiographies of diseased vessels provide three-dimensional reconstruction of blood vessels for flow computations using finite element methods. The objective is to study saccular malformations or aneurisms that may result in large artery diseases and are a major cause of death. The vessel wall is locally subjected to a plastic deformation and may become thinner and weaker than

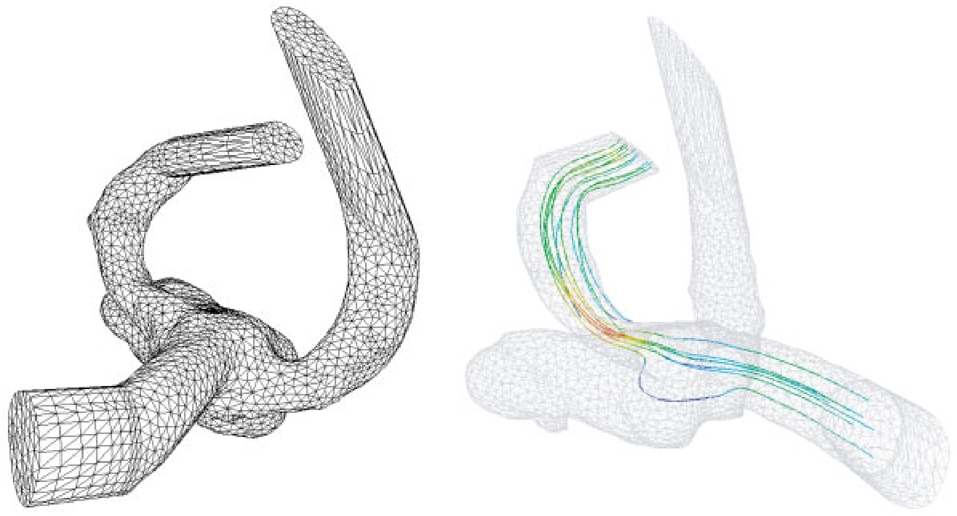

Figure 11. Computational mesh of an aneurism and streamlines in the terminal branch of the vessel (data courtesy of M. Thiriet, INRIA).

the normal (sane) artery wall, a rupture may happen. In this numerical simulation, the vessel wall is assumed to be rigid and the blood is assumed to be incompressible, homogeneous and Newtonian. Figure 11 illustrates the result of the computation (streamlines of the velocity field) on a computational mesh.

6.3. Adapted meshes

In this section, we present an example of mesh adaptation to demonstrate how to capture an analytically defined metric tensor. In this example, five iterations of mesh adaptation have been applied. Figure 12 shows the initial surface mesh and the adapted meshes at iterations 1, 3 and 5 (final). Table III reports the statistics about the adapted meshes. The mesh gradation value has been set to 1.4, the minimal (resp. maximal) size has been set to 0.4 (resp. 10.). At each iteration, the discrete analytical metric field is computed at the mesh vertices and a unit mesh is generated with respect to this field. The analytical function is not explicitly used when inserting or moving a vertex. During the remeshing stage, a linear interpolation scheme is used to find the value of the field at a given vertex location.

7. CONCLUSIONS

In this paper, we have presented a global approach for generating geometric as well as computational meshes from discrete (sampled) data. At first the sampled data is transformed into a valid and conform surface triangulation using various algorithms depending on the data type (series of slices, point cloud, etc.). Then, this surface triangulation is simplified in order to generate a geometric reference mesh which represents an accurate piecewise linear

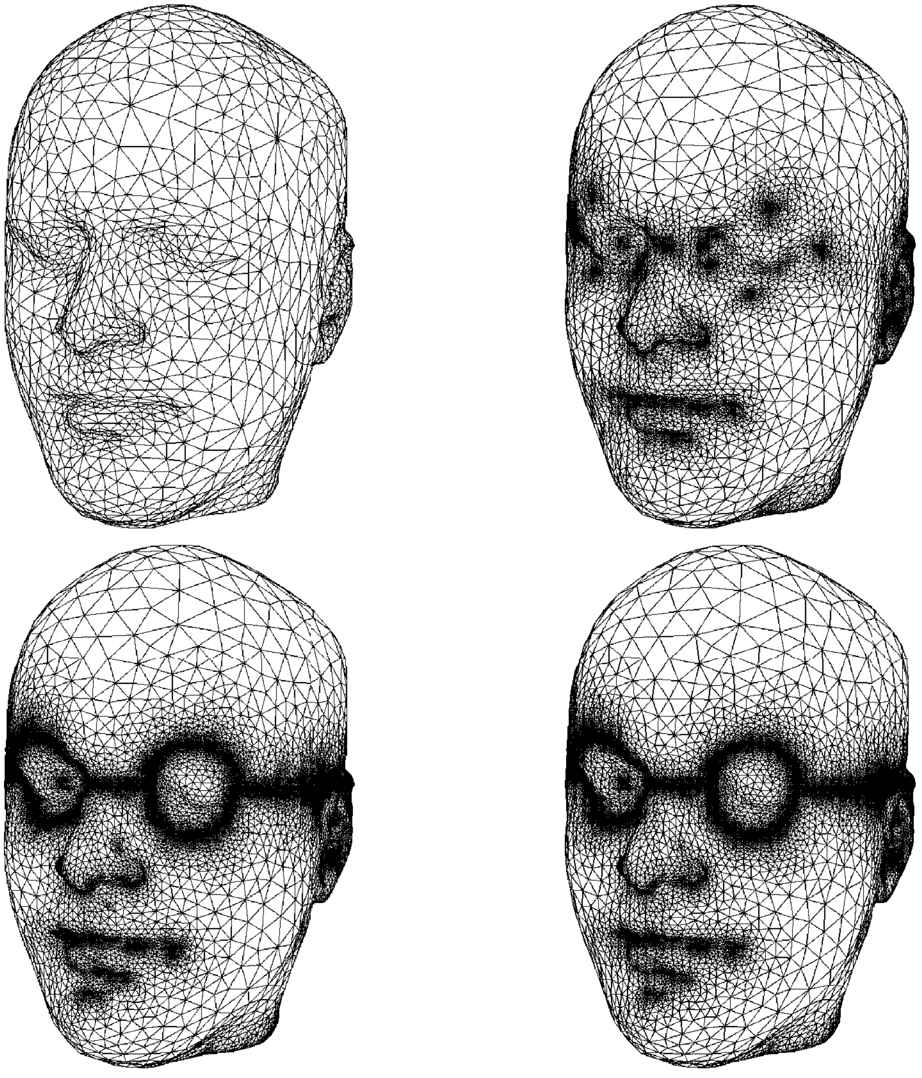
Figure 12. Example of surface mesh adaptation. Adapted meshes at iterations 0 (initial), 1, 3 and 5 (final).

approximation of the underlying surface geometry. Finally, a computational unit mesh is generated with respect to the intrinsic geometric metric map $\mathscr{H}_{\mathrm{ref,g}}$. In addition, we have demonstrated that the same concept can be easily applied for mesh adaptation in the isotropic or anisotropic case.

As future work, we will focus on issues related to the definition of suitable anisotropic metric tensors for numerical simulations, following the ideas described in Reference [20].

Table III. Statistics about the adapted meshes.

Iteration	np	ne	$Q_{\mathcal{H}}(\mathcal{M})$	$l_{\mathcal{M}}$	t (s)
0	3353	6702	1.3	—	—
1	19 150	38 296	1.3	0.84	9
3	32 534	65 064	1.2	0.87	11
4	34 641	69 278	1.2	0.88	11
5	37 758	71 512	1.2	0.89	10

ACKNOWLEDGEMENTS

Initial surface meshes or sampled data presented in this paper are courtesy of the Universidad Santiago de Compostella, the Epidaure project (INRIA-Sophia Antipolis), the Low Temperature Laboratory (Helsinki University), Cyberware (www.cyberware.com). The author wishes to thank Paul Louis George and Frédéric Alauzet (INRIA) for reviewing and Éric Saltel for his help in setting up examples. This work was partly funded by the ARC Vitesv grant (INRIA).

REFERENCES

1. Luebke DP. A developer's survey of polygonal simplification algorithms *IEEE Computer Graphics and Applications* 2001; 24–35.
2. Frey PJ, Borouchaki H. Surface meshing using a geometric error estimate. *International Journal for Numerical Methods in Engineering* 2003; **58**(2):227–245.
3. Schroeder WJ, Zarge JA, Lorensen WE. Decimation of triangle meshes. *Computer Graphics*, *Proceedings of Siggraph'92*, 1992; 65–70.
4. Turk G. Re-tiling polygonal surfaces. *Computer Graphics*, *Proceedings of Siggraph'92*, 1992; 55–64.
5. Heckbert PS, Garland M. Survey of polygonal surface simplification algorithms. *Computer Graphics*, *Proceedings of Siggraph'97*, 1997.
6. Krus M. Maillages Polygonaux et Niveaux de Détails: étude bibliographique. *RR 97-10*, LIMSI/CNRS, Orsay, 1997.
7. Cebral JR, Löhner R. From medical images to CFD meshes. *Proceedings of International Meshing Roundtable*, 1999; 321–331.
8. Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics* 1987; **21**(4):163–169.
9. Boissonnat J-D, Cazals F. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Computational Geometry—Theory and Application* 2002; **22**(1):185–203.
10. Ciarlet PG. *The Finite Element Method for Elliptic Problems*. North-Holland: Amsterdam, 1978.
11. Frey PJ, Borouchaki H. Simplification of terrains by minimization of the local deformation. *Proceedings of Curves and Surfaces Fitting*, St Malo. Nashboro Press: TN, 2002.
12. Nikolaidis N, Pitas I. *3D Image Processing Algorithms*. Wiley: New York, 2000.
13. Boissonnat JD, Geiger B. Three dimensional reconstruction of complex shapes based on the Delaunay triangulation. *RR-INRIA* 1967, April, 1992.
14. Frey PJ, George PL. *Mesh Generation*: *Application to Finite Elements*. Hermès Science Publ.: Paris, 2000; 816.
15. Taubin G. Curve and surface smoothing without shrinkage. *Proceedings of 5th International Conference on Computer Vision*, 1995; 852–857.
16. Cohen J, Varshney A, Manocha D, Turk G, Weber H, Agarwal P, Brooks F, Wright W. Simplification envelopes. *Computer Graphics*, *Proceedings of Siggraph'96*, 1996; **30**:119–128.
17. Cohen J, Olano M, Manocha D. Appearance-preserving simplification. *Computer Graphics*, *Proceedings of Siggraph'98*, 1998; 115–122.
18. Borouchaki H. Simplification de maillage basée sur la distance de Hausdorff. *Comptes Rendus de L'Academie des Sciences* 2000; **329**(1):641–646.

19. Borouchaki H, Hecht F, Frey PJ. Mesh gradation control. *International Journal for Numerical Methods in Engineering* 1998; **43**(6):1143–1165.
20. Alauzet F, Frey PJ. Estimateur d'erreur géométrique et métriques anisotropes pour l'adaptation de maillage. Partie I: aspects théoriques. *Research Report INRIA*, 2003.
21. Frey PJ, Borouchaki H. Geometric surface mesh optimization. *Computing and Visualization in Science* 1998; **1**:113–121.
22. Walton DJ, Meek DS. A triangular G^1 patch from boundary curves. *Computer Aided Design* 1996; **28**(2): 113–123.
23. Gregory JA. Smooth interpolation without twist constraints. In *Computer Aided Geometric Design*, Barnhill RE, Riesenfed RF. (eds). Academic Press: New York, 1974; 71–87.
24. Frey PJ, Borouchaki H. Surface mesh quality evaluation. *International Journal for Numerical Methods in Engineering* 1999; **45**:101–118.
25. George PL, Borouchaki H. Delaunay triangulation and meshing. Application to Finite Elements. Hermès Science: Paris, 1997; 432.
26. Berzins M. Solution-based mesh quality for triangular and tetrahedral meshes. *Proceedings of the 6th International Meshing Roundtable, Sandia Laboratory*, 1997; 427–436.
27. Frey PJ. Yams: A fully Automatic Adaptive Isotropic Surface Remeshing Procedure. *RT-0252*, INRIA Rocquencourt, November 2001.
28. George PL. Premières expériences de maillage automatique par une méthode de Delaunay anisotrope en trois dimensions. *RT-0272*, INRIA-Rocquencourt, November 2002.
29. d'Azevedo EF. Are bilinear quadrilaterals better than linear triangles? *SIAM Journal on Scientific Computing* 2000; **22**(1):198–217.
30. Bernardini F *et al*. Building a digital model of Michelangelo's Florentine Pietà. *IEEE Computer Graphics and Application* 2002; **22**(1):59–67.
31. Boissonnat JD, Chaine R, Frey PJ, Malandain G, Salmon S, Saltel E, Thiriet M. From medical images to computational meshes. *Proceedings of MS4CMS*, Rocquencourt, 2002.
32. Frey PJ, Sarter B, Gautherie M. Fully automatic mesh generation for 3D domains based upon voxel sets. *International Journal for Numerical Methods in Engineering* 1994; **37**:2735–2753.
33. Guéziec A. Surface simplification inside a tolerance volume. *IBM Research Report*, RC-20440, 1996.
34. Hamann B. Curvature approximation for triangulated surfaces. In *Geometric Modelling*, Computing Suppl. 8. Springer: New York, 1993.
35. Hartmann E. A marching method for the triangulation of surfaces. *The Visual Computer* 1998; **14**:95–108.
36. Hoppe H. New quadric metric for simplifying meshes with appearance attributes. *Computer Graphics, Proceedings of Siggraph'00*, 2000.
37. Johnson AE, Hebert M. Control of polygonal mesh resolution for 3D computer vision. *Graphical Models and Image Processing* 1998; **60**:261–285.
38. Kalvin AD, Taylor RH. Superfaces: polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications* 1996; 64–77.
39. Kobbelt L, Campagna S, Seidel HP. A general framework for mesh decimation. *Proceedings of Graphics Interface'98*, 1998; 43–50.
40. Levoy M *et al.* The Digital Michelangelo Project: 3D scanning of large statues. *Proceedings of ACM Siggraph*, 2000.
41. Löhner R. Regridding Surface Triangulations. *Journal of Computational Physics* 1996; **126**:1–10.