Improved Techniques for the Generation of 3D Finite Element Meshes of Human Anatomical Structures

Claudio Lobos

THÈSE de DOCTORAT

presentée par

Claudio LOBOS YÁÑEZ

pour obtenir le grade de

DOCTEUR de l'UNIVERSITE JOSEPH FOURIER

Spécialité: Modèles, Méthodes et Algorithmes en Biologie, Santé et Environnement.

Amélioration des Techniques de Génération de maillages 3D des structures anatomiques humaines pour la Méthode des Éléments Finis

---

Soutenue le 5 mars 2009, devant le jury composé de :

J.	TROCCAZ	Directeur de Recherche CNRS	*Présidente*
P.	FREY	Professeur des Universités CNRS	*Rapporteur*
S.	COTIN	Directeur de Recherche INRIA	*Rapporteur*
Y.	PAYAN	Chargé de Recherche CNRS	*Co-directeur de thèse*
N.	HITSCHFELD	Professeur à l'Université du Chili	*Co-directeur de thèse*
F.	JAILLET	Maître des Conférences des Universités	*Examinateur*

Thèse préparée au sein du laboratoire TIMC–IMAG

Équipe: Gestes Médico–Chirurgicaux Assistés par Ordinateur

Amélioration des Techniques de Génération de maillages 3D des structures anatomiques humaines pour la méthode des Éléments Finis

Résumé

La Méthode des Éléments Finis (MEF) est probablement la technique la plus utilisée pour la modélisation du comportement mécanique des solides. Elle s'appuie pour cela sur une discrétisations du domaine modélisé en éléments géométrique simples. Cette partition porte le nom de maillage. La solution numérique calculée par la MEF dépend directement du maillage utilisé.

Dans le domaine médical, les solides modélisés sont de géométrie complexe. De ce fait, nous privilégions une génération de maillage par recalage élastique. Cette méthode permet d'adapter un maillage prédéfini (atlas) aux données du patient afin de représenter le domaine à modéliser. Le recalage élastique applique un déplacement aux sommets de l'atlas sans en changer sa topologie. Les méthodes de recalage élastique ne prennent cependant pas en considération les éléments, par conséquent il est possible de produire des éléments invalides et de mauvaise qualité. Cette thèse présente une méthode de réparation des éléments après application d'un recalage élastique.

Les méthodes de recalage élastique peuvent être limitées lorsque, pour une région spécifique du domaine modélisé, une discrétisation plus fine est requise alors qu'elle ne figure pas dans le maillage atlas. Par exemple dans le domaine de la neurochirurgie, un maillage d'une densité plus importante peut être nécessaire dans la région de la voie d'abord, entre la craniotomie et la tumeur car dans cette région d'intérêt une précision accrue de la simulation est requise. Nous proposons dans cette thèse une méthode de génération de maillage comportant un raffinement local. Cette méthode est appliquée à la neurochirurgie.

Mots clés Éléments Finis, génération des maillages 3D, réparation des maillages 3D, modélisation biomechanique des structures anatomiques humaines.

Improved Techniques for the Generation of 3D Finite Element Meshes of Human Anatomical Structures

Abstract

The Finite Element Method (FEM) is probably the most used strategy to simulate physical phenomena in a domain. The method needs a subdivision of the domain into simpler geometrical structures. This subdivision is known as a mesh. The numerical solution computed by the FEM directly depends on the employed mesh.

In the medical field, the domains to simulate are complex geometries. Due to this complexity, it is preferred to use Registration Methods (RMs) to produce the mesh of the domain to be simulated. The RMs are a family of strategies that "adapt" a predefined mesh (the atlas) to patient data in order to represent the target domain. A RM reallocates the nodes of the atlas without changing the topology of it. Unfortunately, the RMs don't consider element information; therefore it is possible to produce invalid and poor quality elements. This thesis proposes reparation methods to achieve validity and improve the quality of the elements in the mesh after registration. Results are presented for femur and face model.

The most important limitation of RMs is that sometimes the focus of the simulation is given on a particular region. An example of this is the brain tumor resection surgery. In this case, a mesh with higher density of nodes is needed on the region between the opening skull point and the location of the tumor. It is on this region where the simulation must be more precise. Unfortunately an "atlas" mesh cannot be produced for each single case. Therefore a mesh generation technique with region refinement is also proposed on this thesis. Results are shown over neurosurgery.

Keywords Finite Element Model, 3D mesh generation, 3D mesh reparation, human anatomical structures modelling.

Acknowledgments

I wish to thank my thesis directors: Yohan Payan and Nancy Hitschfeld for all the support during all the time this thesis took. Their willing to discuss and teach at any time. In other words, for always been there when help was needed.

Marek Bucki also helped me a lot. From work to French lessons, from ideas to reality. Always a person willing to teach, explain and discuss. Certainly an excellent person to work with.

I would like to thank Fabrice Chassat for being the first person that contacted Nancy at *Universidad de Chile*. Is due to this contact I finished doing a PhD in France.

Also, I would like to thank Fabrice Jaillet and Francisco Galdames for the help in the brain surface model used in this thesis.

Thank to all the people that helped me in my French-adaptation, specially to Fabien Robineau, Olivier Chenu, Sebastien Martin and Nicolas Glade.

All the people at the timc-imag laboratory; specially to the *gmcao* team directed by Jocelyn Troccaz.

I would like to thank Jacques Demongeot for all the support during the *alfa–* crisis and the best willing to find solutions.

Nikolaï Hungr for all the English corrections, but most of all, for being a friend and to all my friends at Grenoble. France is a second home to me thanks to them. *Gracias et toutes me remerciements à vous*.

This thesis has been financially supported by FONDECYT 1061227, ALFA IPECA project, FONDEF D04-I-1237 and ECOS-Sud C06E04.

Contents

Contexte général de cette thèse

Motivation

Si un modèle est une simplification d'un objet réel, une simulation est la simplification des interactions réelles entre un objet et son environnement. Les simulations sont utiles pour prédire le comportement d'un objet suivant les lois qui définissent le système dans lequel l'objet se trouve.

Dans notre cas, l'objectif est de simuler des structures anatomiques humaines dans un contexte de simulation médicale et de chirurgie assistée par ordinateur. Dans ce cadre, les objets sont régis par des lois mécaniques. Ces lois peuvent se traduire par un système d'Équations aux Dérivées Partielles (EDP) qui permettent de prédire le comportement de l'objet.

Les lois mécaniques (décrites par les EDP) agissent sur la structure à simuler. Cette structure est un domaine continuum $\Omega \subset \mathbb{R}^3$. Malheureusement il n'est pas possible de trouver une solution analytique pour les EDP sauf dans le cas où le domaine à simuler est un corps géométrique simple. Comme ceci n'est pas le cas des structures anatomiques humaines, une discrétisation de Ω en formes géométriques simples doit être effectuée. Ces formes géométriques sont appelées éléments qui peuvent être, dans $\mathbb{R}^3$, des tétraèdres, des pyramides, des prismes et des hexaèdres.

Ensuite, une solution au système d'EDP va être calculée pour chaque sommet des éléments qui décrivent Ω. La solution pour n'importe quel point de Ω va alors être une interpolation des solutions trouvées aux nœuds des éléments. Cette méthode d'approximation s'appelle Méthode des Éléments Finis (MEF).

La représentation d'un domaine continu Ω à partir d'une simplification géométrique est connue sous le nom de **maillage**. En ce qui concerne le premier paragraphe de cette introduction, les lois de notre réalité simplifiée sont calculées à l'aide de la MEF et notre objet simplifié (sur lequel est calculée la MEF) correspond au maillage.

Cette thèse est donc focalisée sur la production des maillages des structures anatomiques humaines. Comme le domaine de la génération de maillages est vaste, il est nécessaire de placer les choses dans un contexte plus spécifique.

Le recalage de maillages correspond à une famille de techniques de génération de maillages dans le domaine médical. Le concept général peut être considéré comme le processus visant à "adapter" un maillage générique existant, l'*atlas*, à un autre domaine cible Ω (voir la section 2.3). Dans un processus de recalage, les sommets du maillage source sont déplacés dans le but de représenter le domaine cible Ω. Cependant, la topologie[1] du maillage reste la même. L'objectif du recalage est de représenter le plus fidèlement possible Ω. Toutefois en faisant cette "adaptation" élastique de l'*atlas*, les éléments peuvent devenir invalides (voir la sous-section 1.3.2) ou présenter une mauvaise qualité (voir la sous-section 3.2.1 et 3.2.2). Certaines solutions ont été étudiées afin de garantir la validité du maillage [44, 27], mais à notre connaissance, aucun travail n'a été effectué en se basant sur les mesures de qualité citées précédemment.

Notre conviction est que les techniques de recalage peuvent trouver une solution adéquate pour plusieurs problèmes de modélisation dans le domaine médical. Malheureusement, des maillages de patients qui prennent en compte des données spécifiques au problème à résoudre, sont parfois nécessaires. Prenons par exemple le cas de la résection chirurgicale de la tumeur d'un organe. Dans ces cas, l'emplacement de la tumeur change d'un patient à l'autre et donc le maillage générique utilisé dans la technique de recalage ne peut plus correspondre à l'objectif du modèle, c'est-à-dire la simulation de la résection de la tumeur. Enfin, pour des cas comme la modélisation de la résection per–opératoire d'une tumeur, le maillage à utiliser doit prendre en compte les contraintes de temps de calcul (c'est-à-dire le temps dont la MEF a besoin pour produire une solution numérique), ainsi que d'autres caractéristiques qui seront expliquées dans la section 3.3.

Travail développé

Deux méthodes de réparation de maillages sont proposées dans cette thèse. La première permet de produire un maillage valide et la seconde vise à produire un maillage de bonne qualité. Le paramètre le plus pertinent pour contrôler la validité et la qualité des éléments est donné par la *matrice Jacobienne*, representative du niveau de dégradation d'un élément. Un deuxième paramètre de qualité est également utilisé. Ce paramètre mesure le niveau de co-planarité des sommets dans une face. Notre technique est un processus itératif et suit une approche numérique effectuée en deux étapes. Tout d'abord, elle réalise un état de validité du maillage (à partir de ce point, la MEF est capable de calculer une simulation numérique) et

[1] La relation entre la quantité et la connectivité des sommets, des arêtes, des faces et des éléments d'un maillage.

en second lieu, elle améliore la qualité du maillage, rendu valide à l'étape précédente. Cette dernière étape augmente la précision de la solution trouvée par la MEF. En effet, grâce à l'amélioration de la qualité des éléments, l'erreur réalisée lors de l'intégration sur les éléments par la MEF est réduite.

Le deuxième type de problème étudié dans cette thèse est la génération de maillages pathologie–spécifiques pour les procédures per-opératoires. Dans ce cas, un nouveau maillage doit être généré afin de prendre en compte le cas particulier de chaque patient. La situation per-opératoire se traduit par une contrainte sur la durée du temps de calcul nécessaire à la mise en œuvre de la MEF. Le temps nécessaire, requit par la MEF, pour produire une solution est directement lié à la quantité de sommets (c'est-à-dire des degrés de liberté) du maillage. Par conséquent, l'objectif est de produire un maillage qui est raffiné dans une **région d'intérêt** et peu dense ailleurs. Ainsi, la MEF est en mesure de calculer une solution sans perdre de précision dans la région d'intérêt. De plus, la réduction du nombre de sommets en dehors de la région d'intérêt permet à la MEF de calculer plus rapidement une solution que dans les cas où le maillage a un même niveau de raffinement à l'intérieur du volume modélisé. Pour ce faire, une approche fondée sur les octree est aussi proposée dans cette thèse.

Organisation de la thèse

Le chapitre 1 donne une introduction à la modélisation par la MEF. Le but de ce chapitre est de comprendre la pertinence des maillages et l'influence de leur qualité sur les résultats de la simulation. Le concept de "maillage" est expliqué ainsi que les propriétés de validité et de qualité. L'utilisation de différents types d'éléments est également analysée, et enfin la notion de "précision" du maillage est également expliquée.

L'état de l'art est discuté dans le chapitre 2 où deux idées principales sont étudiées: le *recalage de maillages* et les techniques de *génération de maillages*, illustrées dans le contexte des applications au domaine médical. Une classification des travaux présentés se trouve à la fin du chapitre.

Les techniques proposées pour l'amélioration du maillage après le recalage et la génération de maillages pathologie–spécifique pour les simulations per–opératoires, sont présentées dans le chapitre 3.

Les techniques développées peuvent être appliquées à plusieurs problèmes de simulation et le chapitre 4 montre les résultats obtenus pour les deux problèmes analysés. La technique de Mesh–Matching, qui est une méthode de recalage, est utilisée pour produire des maillages de fémurs ainsi que des modèles du visage. Les méthodes proposées pour produire un maillage valide et de bonne qualité sont ensuite appliquées pour améliorer l'état du maillage. En ce qui concerne la

méthode de génération de maillages, le problème clinique connu sous le nom de “brain-shift” (au cours d’une intervention neuro–chirurgie) est étudié. Le but de cette simulation est de prédire les déformations du cerveau pendant la résection chirurgicale de tumeurs cérébrales.

Global context of this thesis

Motivation

While a model is a simplification of a real object, a simulation is the simplification of real interactions between the object and its environment. Simulations are useful to predict the behavior of an object following some laws that define the system.

In our case, the goal is to simulate human anatomical structures in the framework of medical simulation and computer assisted surgery. In this context, the objects are governed by mechanical laws. These laws can be translated into a system of Partial Differential Equations (PDEs) that allow to predict the behavior of the object.

The mechanical laws (represented by a system of PDEs) take effect over the target structure to simulate. This structure is a continuum domain $\Omega \in \mathbb{R}^3$. Unfortunately it is not possible to find an analytical solution to the system of PDEs unless the domain to simulate be a simple geometrical body. As no human anatomical structure corresponds to a simple geometrical body, a discretization of the continuous domain Ω, into simpler geometrical forms, must be performed. These simpler geometrical forms are called elements and in $\mathbb{R}^3$ they correspond to tetrahedra, pyramids, prisms (or wedges) and hexahedra.

A soultion to the system of PDEs can be computed at each node of the elements representing Ω. The solution to any point in Ω is then obtained as an interpolation of the solutions found for each node of the elements. This approximated solution (to the PDEs) is named the Finite Element Method (FEM).

The representation of a continuum domain Ω through simpler geometrical bodies is known as a **mesh**. Regarding the first paragraph of this introduction, the laws of our simplified reality are computed using the FEM and our simplified object (over which the FEM is computed) corresponds to the mesh.

This thesis is focused on producing meshes for the anatomical structures to be simulated. As the meshing techniques domain is large, it is necessary to put this in context.

Registration corresponds to one family of meshing techniques in the medical field. The general concept can be seen as the process to "adapt" a generic existing

mesh (*atlas*) to another target domain Ω (see section 2.3). In a registration process, the nodes of the source mesh are reallocated in order to represent the target domain Ω. However the topology[2] of the mesh remains the same. The goal of the registration is to represent as close as possible Ω. However, by doing this elastic "adaptation" of the *atlas*, elements might become invalid (see subsection 1.3.2) or present poor quality (see subsection 3.2.1 and 3.2.2). Some solutions have been given regarding the validity of a mesh [44, 27], but to our knowledge, no work has been done regarding the quality measures cited before.

Is to our belief that registration techniques achieve a proper solution for several modeling problems in the medical field. Unfortunately, sometimes meshes that include patient-specific data are also needed. Take for example the case of tumor resection surgery from a particular organ. In those cases the location of the tumor changes from one patient to another and therefore the generic mesh used in the registration technique does not longer properly represent the goal of the model, i.e. the simulation of the tumor resection. Finally, for cases such as tumor resection simulation that need an "interactive" use of the FEM for intra-operative assistance, the mesh must efficiently take into account the constraints of time-computation (i.e. the time the FEM needs to produce a solution for the simulation).

Developed work

Two reparation methods are proposed in this thesis. The first is to produce a valid mesh and the second to achieve good quality. The most relevant parameter to control validity and quality is given by the **Jacobian matrix** which allows to measure the level of the degradation of an element. A second quality parameter is also used. This parameter measures the level of co-planarity of the nodes in a face. Our proposed technique is an iterative numerical approach and is carried out in two independent stages. First, it achieves a valid state of the mesh (from this point the FEM is able to compute a result for the simulation) and second, it improves the quality of the (already valid) mesh. This last step increases the accuracy of the solution given by the FEM, i.e., by improving the quality of the elements, the integration over the elements by the FEM reduces its error.

The other type of problem studied in this thesis corresponds to the generation of pathology-specific meshes for intra-operative procedures. In this type of cases, a new mesh must be generated regarding the particular case of each patient. The intra-operative condition is traduced in a constraint of finding a fast simulation solution *via* the FEM. The time needed by the FEM to produce a solution is directly related to the quantity of nodes (i.e. degrees of freedom) the mesh has. Therefore

[2]The quantity and connectivity relationships between nodes, edges, faces and elements in a mesh.

the goal is to produce a mesh that is refined in a particular **region of interest** and coarse elsewhere. By this, the FEM is able to compute a solution without loosing precision in the region where the focus of the modeling is. Moreover, the reduction of nodes in coarse regions allow the FEM to perform a faster solution than using a mesh with the same precision (quantity of nodes and elements) over the entire domain Ω. In order to do so, an octree-based approach is proposed.

Organization of this thesis

Chapter 1 gives an introduction to modeling using the FEM. The goal of this chapter is to understand the relevance of the meshes in the simulation field and how a "good" mesh can produce an important difference in the simulation results. The concept of "mesh" is explained as well as validity and quality mesh properties. The use of different types of elements is also analyzed and finally the concept of "precision" of the mesh is also explained.

The state of the art is discussed on chapter 2. Here two main threads are studied: the mesh registration and the mesh generation techniques, both with applications to the medical field. A classification of the reviewed works is presented at the end of the chapter.

The proposed techniques for both, the improvement of the mesh after registration and the generation of pathology-specific meshes for intra-operative simulations, are presented in chapter 3.

The developed techniques can be applied to several simulation problems and chapter 4 shows the results obtained for the two analyzed problems. The Mesh-Matching technique, which is a registration method, is used to produce meshes of femur and face models. The reparation and quality improvement technique is then applied to improve the state of the mesh. Regarding the proposed mesh generation technique, the problem known as the "brain-shift" during neuro-surgery is studied. The goal of this simulation is to consider the deformations of the brain during tumor resection surgery.

Chapter 1

Introduction to meshing in medical applications

Chapitre 1: Introduction au maillages dans le domaine médical

Abstract

This chapter introduces the reader to modelling problems in the context of biomechanical simulations. The key idea is to explain the global context of this thesis by introducing the reader into the field of medical applications.

It starts with the general context of this thesis. Then there is a global introduction to the Finite Element Modelling and finally there is a brief discussion in terms of well defined and good quality Finite Element meshes.

1.1 Continuum mechanics

Continuum mechanics (CM) is a branch of mechanics that deals with the analysis of the kinematics[1] and mechanical behavior of materials modeled as a continuum, e.g., solids and fluids (i.e., liquids and gases). In a nutshell, CM assumes that matter is continuous (ignoring the fact that matter is actually made of atoms). This assumption allows the approximation of physical quantities over the materials, such as energy and momentum, at the infinitesimal limit. Differential equations can thus be employed in solving problems in CM.

Let Ω be a volumetric domain defined in $\mathbb{R}^3$ and P_i a set of points inside Ω. A deformation occurs in Ω as external forces F_{ext} (surfacic or volumetric) are applied to some $P_i \in \Omega$. If Ω is considered as an elastic body, the deformation is characterized by:

[1] A branch of dynamics that describes the motion of objects without consideration of the circumstances leading to the motion

- a displacement vector field $\vec{u}$ caused by F_{ext} applied over one or more P_i.
- an internal state of deformation, mathematically defined by a "strain tensor" for each $P_i \in \Omega$.
- the "force reaction" of the body to the external forces; this reaction is mathematically defined by a "stress tensor" for each $P_i \in \Omega$.

The Local Equilibrium of the Medium (LEM) is an expression defined for each $P_i \in \Omega$ that links the displacement, the strain and the stress tensor through Partial Differential Equations (PDE). In order to do so, an analytical relationship is assumed between the strain and the stress tensors, known as the **Constitutive Law** of the material.

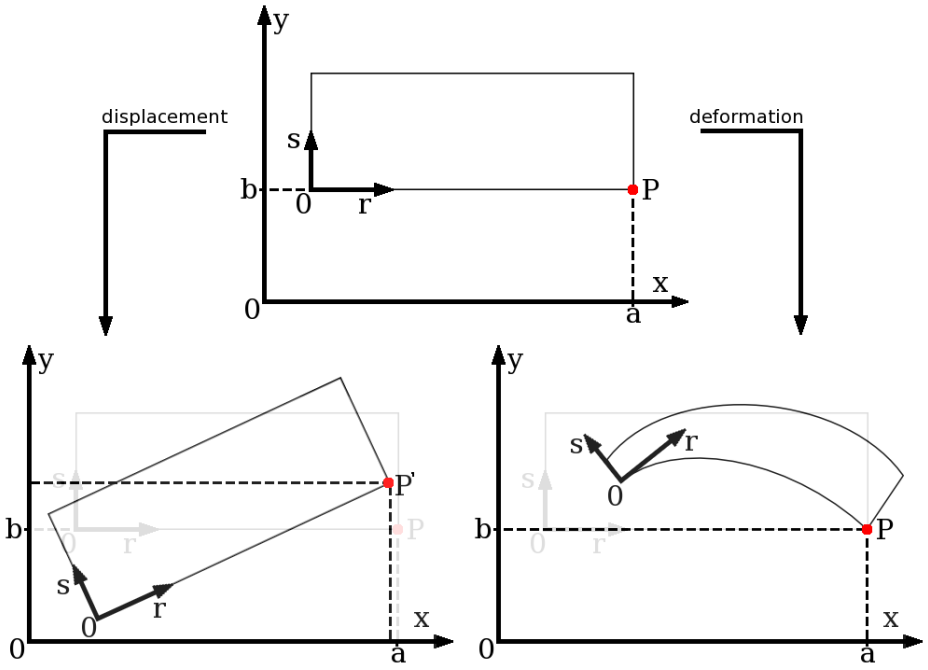

Figure 1.1: Displacement *vs* deformation. Top: Initial system. Bottom left: the displacement (a difference regarding the x and y axes). Bottom right: the deformation (a difference regarding the r and s axes).

The difference between displacement and deformation is made in function of the Reference System (RS) used to measure each one of them. The first one is made regarding the overall RS. The second one, uses a relative RS associated to the domain. Figure 1.1 illustrates the initial state of the system at the top; at the bottom left panel a displacement is produced as the coordinates of point P change to P' regarding the x and y axes, however no deformation is presented in the r and s axes. At the bottom right panel the inverse situation is produced as no

displacements of P is presented in the x and y axes. In the other hand, point P has a severe deformation regarding the r and s axes.

The strain level measures the domain deformation regarding the relative position of $P_i \in \Omega$. Let $\vec{a}$ be the initial position of P_i and $\vec{b}$ be the position of P_i after the deformation in the relative axes: r, s. The strain level is then defined as:

$$\frac{|\|\vec{a}\| - \|\vec{b}\||}{\|\vec{a}\|}$$

The importance of measuring the strain level is that when it is inferior to 10% the "small strain" hypothesis can be made, which assumes a linear geometrical resolution of the PDEs. Otherwise, a "large strain" framework is necessary, which means a much more complex resolution of the system.

If the **Constitutive Law** can be assumed to be linear another simplification can be made: an elasticity tensor is computed to link the stress and strain tensors. On the contrary, if a linear law does not account properly for the mechanical behavior of the tissues, a non-linear law can be chosen to account for the hyperelastic behavior of soft tissues (see [8] for more details).

Most of the modeling works proposed in the literature assumed both, the small strain and the linearity of the **constitutive law** hypotheses. The term "linear elasticity" is then commonly used, i.e. assuming linearity of the geometry and the mechanics. In that case, it can be shown that the constitutive behavior of the material can be characterized by only two parameters: the **Young's modulus** that depends on the stiffness of the material, and the **Poisson's ratio** that is related to the compressibility of the material.

Whatever the modeling assumptions are, the PDEs that govern CM cannot be analytically solved over the full domain. They are therefore numerically solved over a piecewise discretization of the domain, usually by means of the Finite Element Method (FEM), the Finite Difference Method and the Finite Volume Method. This thesis focuses on the use of the FEM since most of the modeling works in the biomechanical field use this method.

The name FEM is due to the principle that the continuous space is subdivided in a finite number of smaller and simpler geometries, namely the **elements**. This subdivision of the continuous space is called the **mesh**. This thesis is about analyzing, producing and repairing meshes for the FEM.

1.2 Constructing the solution of the FEM

The finite element method requires the discretization of the spatial domain Ω with "finite elements" interconnected at points called "nodes". The PDE of the CM can then be solved inside each element which geometry is quite simple and regular.

For two dimensional problems, triangles and rectangles are commonly used while in three dimensions tetrahedral and hexahedral elements are very popular. As compared to finite difference methods, the finite element mesh may be entirely unstructured, which makes the modeling of complicated and irregular geometries more convenient.

Altogether, the elements should cover the entire domain as accurately as possible. For a given set of boundary conditions (i.e. forces and constraints applied on nodes) and assuming an analytical constitutive law, the unknowns of the systems, i.e. node's displacements, are solved for each element. Then, thanks to interpolation "shape functions" (usually linear or quadratic), displacements, strains and stresses can be computed at each point inside the elements. Because of element interconnections and continuity of the shape functions from one element to another one, the FEM provides solutions of the system inside the full domain.

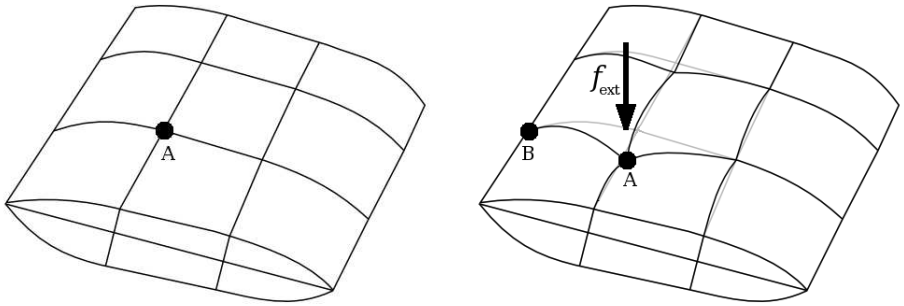

Figure 1.2: An example of a deformed domain simulated with the FEM. Left: the initial system. Right: the domain is deformed by the influence of an external force.

Figure 1.2 shows an example of FEM over an arbitrary domain. In this example, an external force f_{ext} is applied over node A. Node B is considered as fixed (this is an example of constraint applied over a node). The position of the other nodes in the mesh is obtained in function of f_{ext} and all the modeling parameters in each node (strain, stress and constraints). Finally the position of each point in the domain is obtained by the interpolation function over each element.

1.3 Meshing

1.3.1 What is a mesh?

Several definitions can be found to this question. In this thesis a mesh shall be seen as a partition of an arbitrary domain into simpler geometrical objects (or

elements). Those elements are compound of nodes, edges, faces and the relations between them.

Probably the most common definition is "a mesh is a tessellation of the space", which is equivalent to our definition since a tessellation is a collection of non-overlapping elements that fill a domain.

Achieving this subdivision of the space is not an easy task, therefore many ways to produce this tessellation have been proposed. Chapter 2 will give a reference over the most popular and useful ones to the medical domain.

1.3.2 What is a valid mesh?

The FEM can be seen as an integration problem (a system of PDE). The computation of this integration is produced over the elements that describe the domain. An element is invalid when the area or volume that it describes is malformed. Two types of malformations can be presented in the mesh:

- Edge inversion: this involves several elements and it is produced when the intersection of neighbor elements has a positive volume.
- Concave element: this is a local problem and it is produced when one or more faces of the element are concave (this problem cannot be produced in triangle faces).

In both cases, the area or volume described by the elements is artificially increased and do not represents the current state of the domain causing an integration over a "phantom" sub-domain. The case of "edge inversion" is presented in 2D in figure 1.3. In this case, the domain is represented by two triangles t_1 and t_2. Node A, that belongs only to t_1 is dragged into the region defined by t_2. In this new scenario the domain corresponds to the shaded area. If the mesh defined by t_1 and t_2 is used to represent this new state of the domain, the integration should be over $t_2 - t_1$. However, it occurs over $t_2 + t_1$.

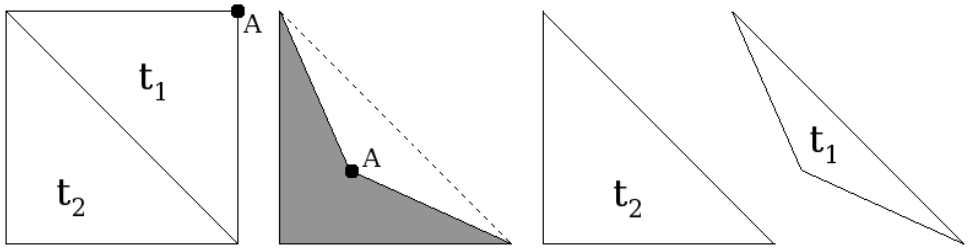

Figure 1.3: At left a valid 2D mesh. At the right an invalid mesh where shaded triangles have a negative area.

The same problem can be expanded to 3D with hexahedra. Figure 1.4 shows an example of a concave hexahedron. A valid hexahedron (left) is transformed by the displacement of point A (right) in such a way that it turns into an invalid configuration. The rendering of the hexahedron is shown as a prism (wedge) defined by the triangle B,C and D and its projection through the z axis. The final numerical integration occurs over the domain described by this prism (while the actual domain is the concave hexahedron).

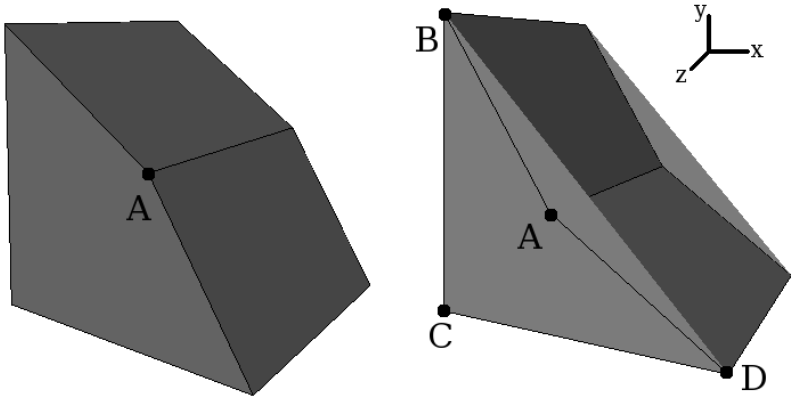

Figure 1.4: Left: a valid hexahedron. Right: a concave hexahedron considered as an invalid configuration.

The Jacobian matrix of an element can be used to determine if an element is valid or not (subsection 2.3.4 will explain in details this problem).

Note that in the literature, a valid mesh can be also named as a regular mesh [45].

1.3.3 What is a good mesh?

Two aspects should be considered to determine if a mesh is good or not. The first has to deal with the representation level of the domain (RLD). This variable is measured as the difference between the areas or volumes of the actual domain and the final mesh.

The second aspect is the quality. The *perfect* element can be described regarding some relations between the angles, edge's length, distance between specific element points, circum-circle, etc. Unfortunately, no "magic" quality measurement exists for an element, because it depends on the numerical method been used and on the problem being solved.

In the case of tetrahedra, the work of Shewchuk in [61] is probably the most relevant article concerning mesh quality. Note that in his web page[2] an unpublished article of 66 pages (just about tetrahedrons quality) can be found. The idea of putting a lot of emphasis in quality is because bad quality elements can lead to several computational errors in the simulation.

In order to illustrate some quality measures, let's describe briefly some classical quality criteria, namely the *aspect ratio*, the *warping factor* and the *dihedral angle*.

To obtain the **aspect ratio** (AR) of an element, the distances between element's faces must be computed. The AR for the element is then defined as the ratio between the maximal and the minimal distance. Therefore, an AR = 1 corresponds to an ideal element and as the AR reaches high values, the element becomes increasingly distorted. See [40] for more details.

The **warping factor** (WF) is a quality measure over the element's faces. For each face, the distances of the face's nodes to an average plane are computed. If all the nodes are co-planar, then the WF is 0 and the face is said to be "perfect". As the WF increases, the quality of the face (thus the element) decreases. Note that WF is out of context for triangle faces as 3 nodes are always co-planar.

The **Dihedral Angle** (DA) is the angle formed between two planes. Therefore several DA values are computed for each element (all the angles between connected faces). The DA of the element is the "worst" value among all the calculated DA. Note that for each type of element the optimal DA is different: 60° for the tetrahedron and 90° for the hexahedron. Therefore, the quality of the element decreases when the "worst" DA of it moves further from the optimal value.

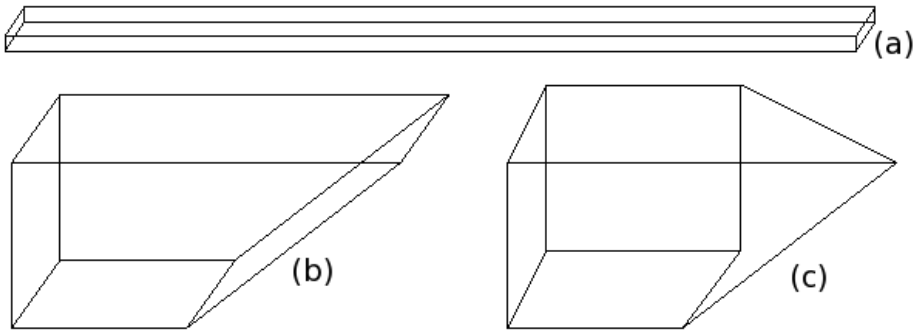

Figure 1.5: Bad elements regarding (a) *aspect ratio*, (b) *dihedral angle* and (c) *warping factor*.

Note that improving some of these quality measures doesn't necessarily produce a "good" element. For example, if only WF and DA are considered as quality

[2] `http://www.cs.berkeley.edu/~jrs/`

measures then the element in figure 1.5 (a) would be considered as *perfect*. However in terms of AR, this element has a very poor quality. Following the same principle, figure 1.5 (b) has a poor DA but a perfect WF and a good AR, and (c) has a poor WF and acceptable DA and AR.

1.4 Choosing the type of element

In this section the advantages and disadvantages of using the different types of elements in a mesh are presented. Three categories will be analyzed: tetrahedral, hexahedral and mixed-element meshes. As it will be shown in chapter 2, most of the meshing techniques proposed in the medical field produce tetrahedral meshes. Some works are done with hexahedra and just a few with mixed-elements.

As this thesis is focused in medical simulations, many works in this field consider deformable domains. A very important work over model deformations has been done by Benzley *et al* in [6] from the mechanical point of view. This work showed the difference obtained by using tetrahedral or hexahedral meshes in terms of incompressibility and plasticity abilities of each type of element. In this study a simple bar, fixed at one end, with a rectangular cross-section was used to compare the performance of linear and quadratic displacement assumption over tetrahedra and hexahedra meshes. The linear or quadratic property of an element has to deal with the manner to represent their edges and in consequence, the volume the element covers. As figure 1.6 shows, in a linear element the edges are represented as an interpolation of the two border points. In the case of quadratic elements, the edges are quadratic functions and therefore an additional middle point is necessary to describe the function. If linear tetrahedra have 4 points, its quadratic counterpart has the same 4 point plus 6 others located on each edge of it.

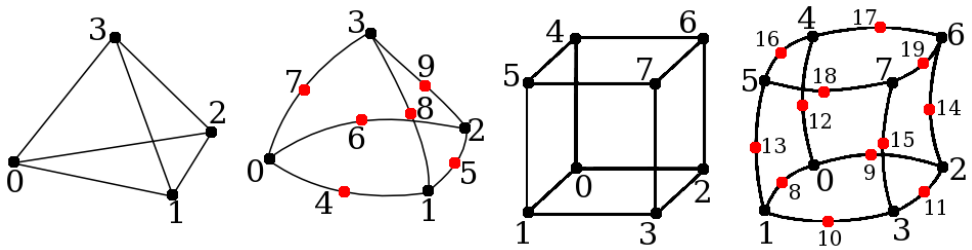

Figure 1.6: From left to right: Linear Tetrahedron, Quadratic Tetrahedron, Linear Hexahedron and Quadratic Hexahedron.

The work from Benzley *et al* in [6] has several important conclusions:

- Linear hexahedra (LH) can generally deform in a lower strain energy state (i.e. eigenvalues), thus making them more accurate than linear tetrahedrons (LT) in numerous situations.
- The comparison of linear static bending[3] situation indicated that LT models produced errors between 10% and 70% in both displacement and stress calculations. Such errors are obviously unacceptable for stress analysis work. On the contrary LH, quadratic hexahedra (QH), and quadratic tetrahedrons (QT) models all provided acceptable results, even with relatively coarse meshes.
- The linear static torsion problem[4] again showed that the LT element produced errors of an unacceptable magnitude. This problem also demonstrated that, because selective integration is only effective on the bending problem, the LH element, without a significant number of degrees of freedom, produces poor results. Here, as in the previous problem, the QH element is superior.
- Significant information is conspicuous in the nonlinear elasto-plastic calculations. Here, as before, all but LT models are adequate for bending calculations. However, not only LT, but QT models seemed to underperform both LH and QH elements.

As chapter 2 will show, most of the meshes used in medical simulations only consider linear elements. There are just a few works that use quadratic elements, even taking into account that some commercial Finite Element (FE) solvers like ANSYS®are prepared to manage those kinds of elements. In other words, there exist FEM solvers that manage quadratic elements, however and even when this may lead to better simulation results, there are few mesh generators that produce quadratic elements.

Other type of variables can be taken into consideration to choose the elements in the mesh. For example, regarding linear elements, only triangular faces will always be planar. Once again, commercial FE solvers like ANSYS® have specific and well defined thresholds to accept those kinds of elements. Even though making a non-triangle face "more" planar has a solution (this will be shown later in section 3.2), this is an extra problem to consider while using other type of element than the tetrahedron (or a dual element taken from it).

Finally, there are some other works that have studied the use of different types of elements [40, 20]. The most general definitions consider tetrahedra, pyramid, prism (wedge) and hexahedra. Some motives to use mixed-element meshes are:

[3]Evaluate linear elements in a single new state after bending the bar.

[4]Evaluate linear elements in a single new state after some torsion is applied over the bar

- A better approximation of the domain in a given region of it. This is the case for meshes that are governed by one type of element, but that prefer in some very specific configurations, to replace bad quality elements with another type of elements.
- Transitions between different levels of refinement. Even though this can be achieved by using just one type of element, some works prefer to include different types of elements during transitions between coarse and refined regions.

The "locking effect" could be another drawback to the use of tetrahedra for quasi-incompressible materials (refer to Hughes in [33] for more details).

1.5 Important remarks over meshing

Validity (or mesh regularity) should be clearly differentiated from the mesh quality. The element's quality optimization[5] involves node displacement. This last procedure might cause neighbor element invalidation. Therefore quality optimization should be always made considering not affecting the validity of the rest of the elements. For instance figure 1.7 shows how the displacement of point A could be considered as an action to improve the top left triangle's quality. Unfortunately this causes the invalidation of the two right triangles and makes the entire mesh useless (it cannot be used to produce a simulation with the FEM).

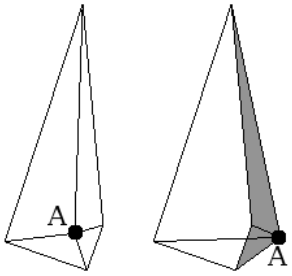

Figure 1.7: Right: a valid 2D mesh. Left: an invalid mesh where shaded triangles covers an area that is two times considered in the numerical integration.

As mentioned in 1.3.3 the representation level of the domain (RLD) is the difference in area or volume between the domain and the final mesh. Another remark can be made relating the RLD and the mesh quality. Figure 1.8 (a) shows an arbitrary domain. In (b) a tessellation of the domain is presented and the RLD is

[5] In the meshing field, optimization refers to mesh quality improvement without inserting new nodes.

shown by the shaded regions. In (c) some triangles of (b) were split by the addition of new points which makes the RLD much more precise.

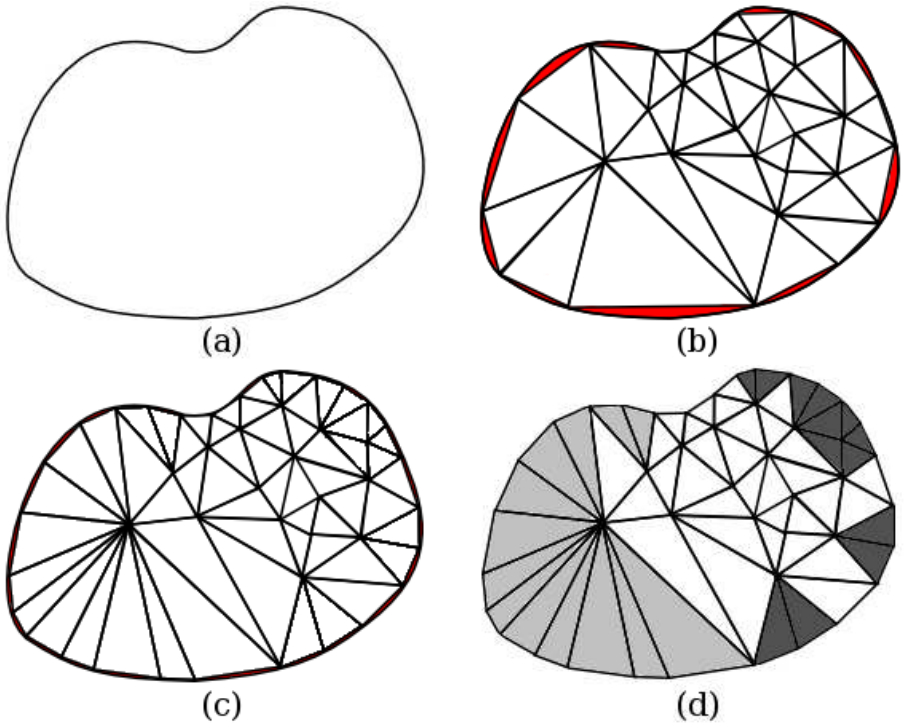

Figure 1.8: (a) input domain; (b) a mesh of the input domain where shaded regions represents the RLD; (c) some elements of (b) were split which causes an improvement of the RLD; (d) the mesh from (c) where the split triangles from (b) are shaded. The minimal angle quality measure is applied over shaded triangles and light ones are "bad" while dark ones are "good".

In figure 1.8(d) shaded elements correspond to the triangles of (b) that were split resulting in the tessellation presented in (c). As mentioned in subsection 1.3.3 one of the quality measures is the Dihedral Angle (DA) for 3D and just the angle for 2D. If this quality measure is applied to the final mesh in figure 1.8(d) light shaded triangles would be considered as "bad triangles" (because the minimal angle is less than a threshold value) and dark shaded as "good". Following the "angle" quality parameter, the overall mesh quality in (b) would be better than the one presented in (c). Therefore one important remark is that improving the RLD does not necessarily increase the quality of the elements and vice–versa.

As shown through the chapter, several factors must be considered before building a mesh. In particular the following questions should be answered:

- should we use one type of element or several? which ones?

- how quality will be measured?
- how quality improvement will be achieved when necessary?
- how invalid elements will be repaired?

Once answers are provided for those questions, the next step concerns the way the mesh is going to be built. Next chapter addresses this problem with the description of the main meshing techniques provided in the biomedical literature.

Chapter 2

Meshing anatomical structures – state of the art

Chapitre 2: Maillages des structures anatomiques – état de l'art

Abstract

This chapter describes the main techniques that are employed in the literature to generate a 3D Finite Element (FE) mesh of a human organ from medical imaging data.

Section 2.1 gives a brief introduction and establish the context of the chapter by analyzing the process "from medical image to simulation".

As many meshing techniques use "generic meshing concepts and basic algorithms" section 2.2 provides an overview of Delaunay, grid, voxel, octree, marching cubes and advancing front techniques.

Following the general meshing classification proposed in section 2.1, sections 2.3 and 2.4 make the point over the state of the art. The works are sub-classified according to their focus on mesh quality and/or fast simulation constraints.

Finally section 2.5 summarizes all the reviewed techniques and examples grouping them as a function of the meshing classification proposed in section 2.1 but also according to the basic meshing concepts and algorithms they stand on, and the type of elements they produce.

2.1 Introduction

There are several manners to produce a mesh. There are not only different techniques but different types of elements and they all have their advantages and disadvantages. However, each technique starts with some input information on the geometry or the domain to discretize. Input data is described in subsection 2.1.1.

According to the available input information, several types of meshing techniques have been published in the literature. A global classification for these techniques is proposed in subsection 2.1.2.

Finally, it appears that FE simulations can be constrained by the need for obtaining rapid results (for example in the case of an intra–operative surgical use) or to get precise results. This point is discussed in subsection 2.1.3

2.1.1 Inputs for FE meshes in the medical field

In medical applications, data of the domain to mesh comes from one or more of the following inputs:

- Computed Tomography (CT) images.
- Magnetic Resonance Images (MRI).
- Segmented images.
- A surface mesh.
- A cloud of points.
- A pre-generated volumetric FE mesh that describes a generic organ.

While CT is more adapted to bone visualization, MRI is better to differentiate soft tissue from an image. Both techniques can be the starting point for the elaboration of a 3D organ geometry. As a person lies in a scanner, several slices of a body region are captured in images. Figure 2.1 shows a surface mesh built from a set of piecewise 2D images. Pages *et al* describe one alternative to do the whole process: from image to surface and volume meshes, in [52].

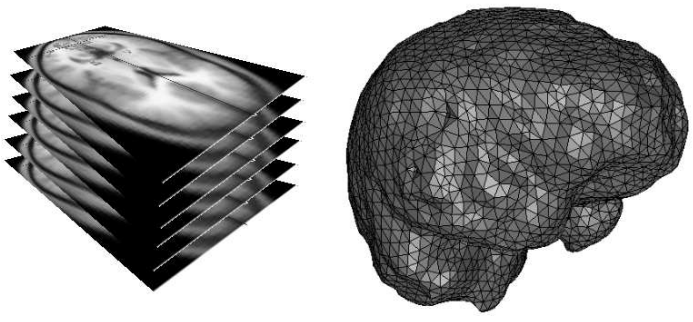

Figure 2.1: Left: an example of 2D scan images of the brain. Right a surface triangle mesh.

A more elaborated input can be the same set of images but with already segmented organs. This process produces a partition of a digital image into multiple

regions with objects and boundaries (points, lines, curves, etc) in the images. Several general-purpose algorithms and techniques have been developed for image segmentation. Since there is no general solution to this problem, these techniques often are combined with domain knowledge, i.e. semi-automatic process leaded by the user to identify the target structures.

Subsequently, from the segmented images, a cloud of points can be extracted or a surface model can be constructed. In order to produce a surface mesh, algorithms like the Marching Cubes can be applied (see subsection 2.2.3).

Most of the meshing techniques that will be explained in further sections start from a surface model of the organ. However one family of meshing techniques does not: a cloud of points is enough as they have additional information on the target organ to mesh (see section 2.3 for more details).

Figure 2.2 shows the different paths to produce a volumetric FE mesh from a set of images and further steps. Note that from a cloud of points, it is also possible to build a triangulation and therefore a surface mesh.

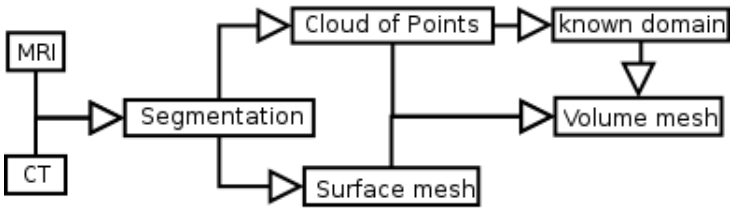

Figure 2.2: The different alternatives to go from imaging to volume mesh generation.

2.1.2 Global classification of meshing techniques

When the geometry to mesh is known *a priori* there is a substantial advantage because there is more information that can be used. For example, if the target geometry to mesh is a femur, a pre-calculated mesh (ATLAS) that describes a generic femur can be build. Now, to mesh any other femur, only the external surface information of this new target is necessary and by causing node displacement over the ATLAS, it is possible to fit the target information. In other words, it is possible to start from a valid pre-defined FE mesh solution and then deform it in order to fit the target's border conditions.

Consider now the case where anatomically malformed femurs have to be modeled. The previous described technique may not work as it is constrained to "small differences" between the ATLAS and the target. If this constraint is not respected, the elements of the resulting mesh can be strongly degenerated or present bad qualities, which will decrease the accuracy of the simulation. Therefore for those

types of problems (unknown domain and big changes from one target to another) it is recommended to use other types of meshing techniques.

In most cases those techniques start from an input surface mesh and create a new target-specific volumetric mesh. The other option is to use the surface information to know if a point is outside or inside the geometry boundaries. In those cases, the resulting mesh will not depend on the properties (density, types of elements) of the input surface mesh. This last family of techniques can also be adapted to use a cloud of points instead of a surface mesh; however the resulting mesh can be not so accurate in those zones where point density is not high enough.

Regarding all this, we can now identify two main streams to generate a volumetric mesh:

- **Mesh adaptation**. Those techniques start with a pre-defined FE mesh ATLAS and then, by moving the ATLAS nodes, achieve the representation of a new target domain.
- **Mesh generation**. Those techniques produce an *ad-hoc* volumetric mesh from the specific input domain.

Mesh adaptation is presented in section 2.3 and mesh generation in section 2.4.

2.1.3 Simulation types

In general, two types of simulations can be distinguished: the ones focused on obtaining fast / interactive results and the ones focused on precise / accurate results. We could also say that some techniques focus on both fast and precise results, but in real practice, most of the times we have to make a choice.

It is very important to mention the differences between fast generation and fast simulation in the meshing field. The first refers to the time needed to produce a mesh. The second refers to the time a numerical framework, like the Finite Element Method (FEM), needs to compute the simulation results.

Regarding the meshes, two aspects must be analyzed. The **quantity of elements** in the mesh is an important variable. The more elements the mesh has, the more the time the FEM needs to produce the simulation results. In the other hand, with more elements it is possible to achieve a better representation of the domain by increasing the quantity of elements in the zone where higher geometry accuracy is required.

The other important variable to achieve a good result is the **quality of the mesh**, however this will increase only the precision of the results. If fast results are required, three options seem possible:

- Few elements [42].
- The use of a cluster of powerful computers with parallel algorithms [16].
- The use of a modified FEM [58, 54, 19, 51].

2.2 Basic tools and properties of mesh generation

2.2.1 Delaunay property

In 2D, a Delaunay triangulation [21] for a set P of points in the plane is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$ (except for the three points of P that conform the triangle). In 3D, the concept remains the same, replacing the circumcircle by the circumsphere. In particular, the 2D Delaunay triangulation is the one that maximizes the minimum angle in comparison with any other triangulation of the same set of nodes.

Delaunay is a mathematical property. In the rest of this thesis, a Delaunay mesh is said to be a mesh (of triangle or tetrahedra) that satisfies the above property.

Several meshing algorithms start from the Delaunay mesh because it gives an initial quality value to the mesh. This quality value can be improved by moving, adding or removing some points, achieving a better representation of the input domain.

If a triangle does not satisfy the Delaunay property, the flipping edges technique [59] can always repair this problem in 2D. Figure 2.3 illustrates this technique. At the left is a triangulation for a set of points that do not satisfy the property. At the right, the same points are plotted but now with a different triangulation that satisfies the property by changing the common edge of the triangles. In 3D, the face swapping [34] strategy allows to perform the analogous operation.

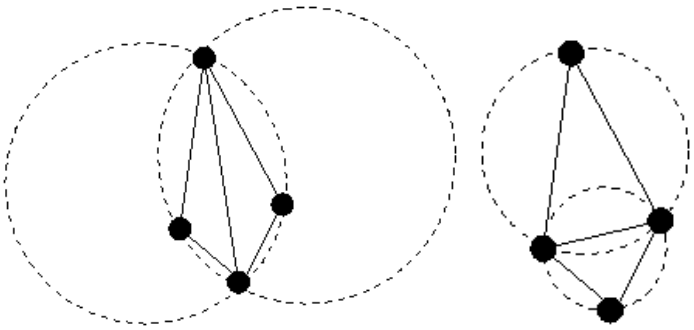

Figure 2.3: The Delaunay property. Left: the property isn't achieved. Right: using the flipping edge technique, the property is satisfied.

As mentioned before, a Delaunay mesh is a reference in the field of tetrahedral meshing techniques. A lot of works stand over the Delaunay property. However it is very important to mention that the Delaunay property is a "good" starting point, but not sufficient to produce quality meshes. Sometimes, a Delaunay mesh can lead to numerical instabilities and that's why it must be refined in order to accomplish other quality constraints. Probably one of the most used refinement

technique to produce high quality meshes was proposed by Ruppert in [56]. An implementation, freely available on the web, was implemented by Shewchuk in his application Triangle [60] with some improvements over the quality of the output mesh. Just to mention one of this improvements, Miller *et al* in [46] continued with the work by improving the triangle's quality in terms of minimal and maximal angle.

To be clear about it, the workflow in most cases is to produce an initial triangle mesh from a cloud of points. The second step is to achieve the Delaunay property for every triangle in the mesh. Finally, the algorithms proposed by Ruppert, Shewchuk, Miller and many others, improve the quality of the Delaunay mesh by moving, adding or removing some points.

In section 2.4 some meshing techniques are presented that not only satisfy but improve the quality for **tetrahedral** meshes (instead of the presented work that is focused on triangles).

2.2.2 Grid, Voxel and Octree meshes

A grid (or structured) mesh is a regular tessellation of a given space. It is usually formed using a unique type of elements (squares, rectangles, triangles, etc) that can be:

- equal in angle: all the elements have the same inner angles but not the same area or volume.
- equal in area or volume: elements can differ in their inner angles.

A voxel is a small volume element, representing a value on a regular grid in the three dimensional space. This is analogous to a pixel, which represents 2D image data. Voxels are frequently used in the visualization and analysis of medical and scientific data. A **voxel mesh** is a particular grid where all the elements are hexahedra of the same size.

Voxel meshes are particular useful when the simulation is performed over a 3D imaging exam and virtually all the voxels can be activated. For instance in the context of the brain activity study (functional MRI for example), works like [4] and [1] (just to mention some) allow to identify the regions of the brain that are activated and to compare between different groups of subjects.

In those studies, voxels in the mesh allow to identify not only if there is activity in certain region but also, with a pre-defined scale, the level of activity as figure 2.4 shows[1]. Here the level of the mesh refinement is crucial to detect with precision the regions that are activated. In the other hand, there is a direct relation between the number of voxels and the computing time to process the information.

[1]Image taken from wikipedia `http://commons.wikimedia.org/wiki/Image:FMRI.jpg` with a Public-domain license. Page last visited: July 21, 2008.

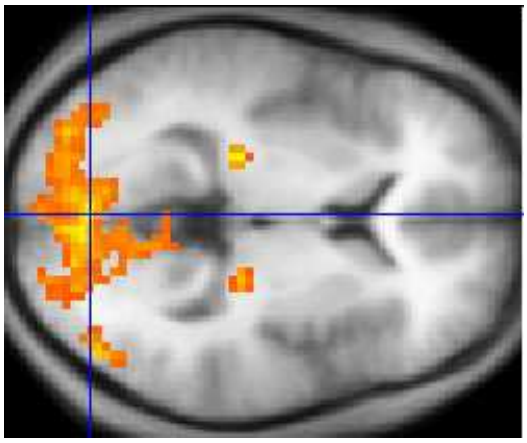

Figure 2.4: An example of a voxel mesh over head MRI. The voxel in this case, measure the level of activity over different zones of the brain.

Therefore, finding the adequate level of element's density (the size of each voxel) is crucial.

An octree is a volumetric mesh and it can be seen as a derivate product of the quadtree in 2D. A quadtree is a tree data structure in which each internal node has up to four children. Quadtrees are most often used to partition a two dimensional space by recursively subdividing it into four quadrants or regions. The regions may be square or rectangular, or may have arbitrary shapes. This data structure was named a quadtree by Raphael Finkel and J.L. Bentley in [26].

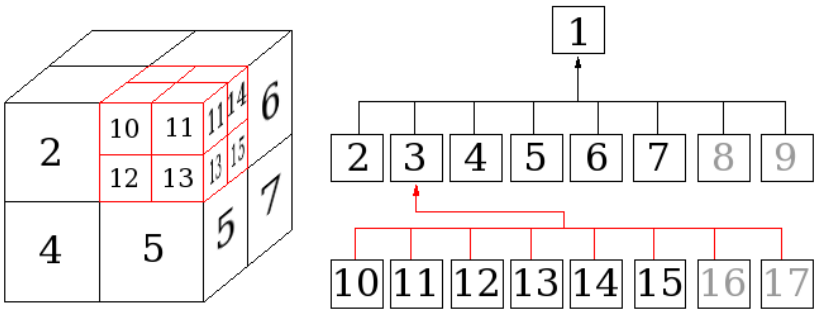

Figure 2.5: At the left an example of mesh generated with the octree algorithm. At the right the corresponding tree data structure of it.

Figure 2.5 shows an example of the tree structure for an octree. Every cube (or octant in this case) is divided, when necessary, in 8 new octants. In the figure, the initial octant numbered 1 was divided in the octants from $2 \rightarrow 9$, then the octant numbered 3, was divided in octants from $10 \rightarrow 17$. Octants 8, 9, 16 and 17 are not visible.

To stop the refinement (the subdivision criterion) on a quadtree or octree, there are two options:

- Geometrical constraint. Like a minimal distance between octree surface points and the actual surface of the domain.
- A level constraint. Like to reach a maximum number of subdivisions, points or elements in the mesh.

Figure 2.6 shows an example of octree mesh from a prostate surface mesh. At the left the input surface mesh of a prostate is shown in solid and transparent elements correspond to the octree. Half of the octree mesh has been deleted to show the interior of it. At the right, the same meshes are plotted but now with the input prostate mesh in transparent. Colors in the octree mesh (at the right) show the different sizes of inner elements that can be found. This difference is explained as "big" inner elements don't intersect the input prostate surface mesh and therefore do not continue the splitting process.

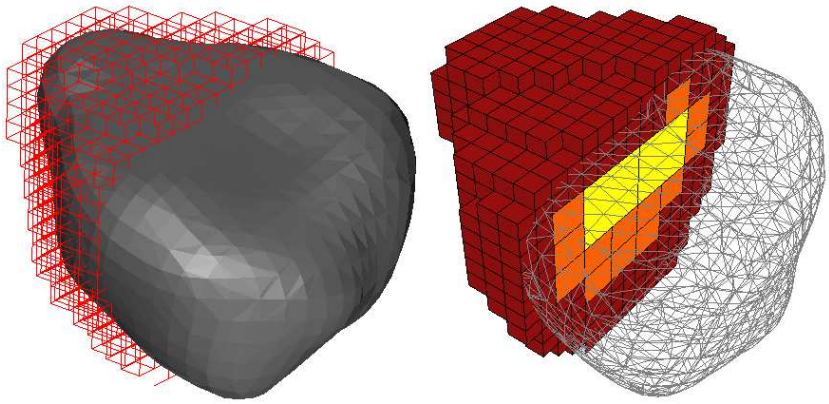

Figure 2.6: An example of octree mesh from a prostate. Left: the octree mesh is shown in transparent while the input prostate surface mesh is shown in transparent in the right panel.

The previous description includes the "pure" octree technique, but in order to make this mesh suitable for FEM it is necessary to manage the transitions between coarse and more refined regions. Here three options are possible:

- manage transitions using only hexahedra [66].
- manage transitions using different types of elements (tetrahedra, pyramid and prism or wedge) [32, 31].
- do not manage transitions and modify the classical FEM [51].

The two first alternatives use templates (patterns) to identify different types of configurations and add elements avoiding the refinement of the entire mesh to the same level. These two alternatives also need a technique in order to achieve

better surface representation, like a projection of all the nodes that reside outside the surface into it. This can lead to distorted elements and has to be handled by a reparation algorithm. The other option to achieve surface representation is the largely used marching cube algorithm that will be presented in the next subsection.

The third alternative doesn't need any further changes from the current state of the mesh. The effort must be made over the FEM in order to accept elements with points inserted in their edges. The PhD thesis of Nesme [51] describes how this can be done.

2.2.3 The Marching Cubes Technique

This is a technique published by Lorensen and Cline in [43] for extracting a polygonal mesh of an isosurface from a three-dimensional scalar field (sometimes called voxels).

The algorithm proceeds through the scalar field, taking eight neighbor locations at a time (thus forming an imaginary cube), then determining the polygons needed to represent the part of the isosurface that passes through this cube. Since there are 8 points in each cube and every point can be in two states, inside or outside the isosurface, there are $2^8 = 256$ different combinations. However, most of the combinations are topologically equivalent and by applying permutations, rotations and switching states of inside/outside, these combinations can be reduced to 15 patterns as shown in figure 2.7.

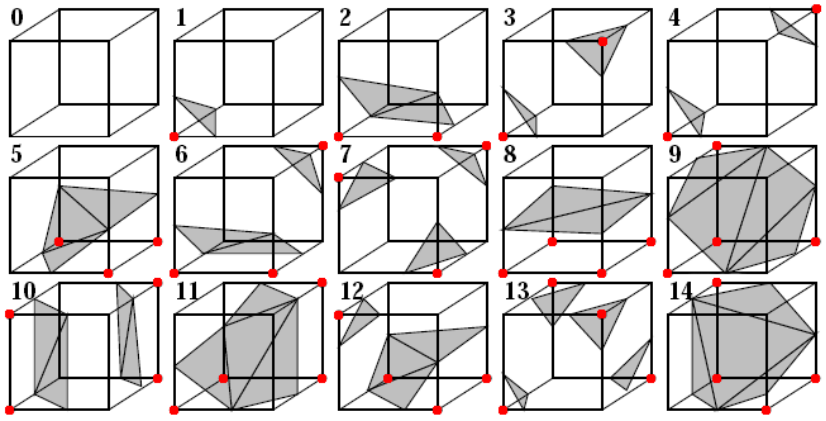

Figure 2.7: The 15 marching cubes patterns. Red points can be inside or outside the target isosurface to represent. The resulting faces will represent the isosurface.

For the switching states, if there are 7 points inside or just one of the entire cube, the same pattern (number 1) will apply to find the intersection face. Even though this procedure is robust, it doesn't work all the time as it doesn't consider

neighboring information. The Marching Cubes technique can produce holes as described and solved by Chernyaev in [15]. These holes are the result of combining two different configurations for neighbors as figure 2.8 shows.

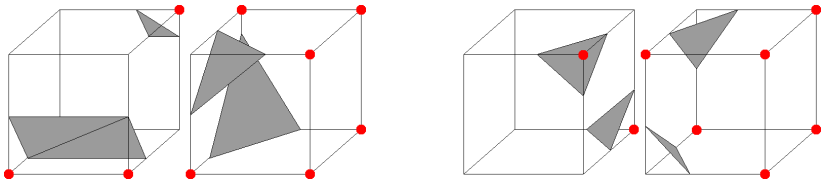

Figure 2.8: Two case of erroneous surface representation, by applying the patterns between two neighbors: a hole is produced in the element's shared face. Red points, are outside points. At the left pattern 6 and the complement of pattern 3. At the right pattern 3 and its complement

The other option to solve the above problem is to produce a tessellation of the cell into tetrahedra and build the isosurface applying the marching tetrahedra algorithm [53, 25]. This method is similar to marching cubes but based on a tetrahedron instead of a cube. The advantage is that in this case, no hole can be produced. Unfortunately the topology produced by this new process isn't necessarily the same as the one obtained without tetrahedra tessellation (See Velasco *et al* in [63]).

The Marching Cubes and their improvements are not only used with the voxel meshes, but can be combined with the octree and grid meshes. This is a fast and well studied technique to improve the surface representation of an input domain.

2.2.4 Advancing Front Technique

The Advancing front is a 2D and 3D meshing technique that stand from a surface (or in 2D a boundary) representation of the domain. Each face in the surface mesh is considered as a front. The idea is to expand all the fronts into the inner part of the volume until the entire domain is meshed.

Once a front is expanded, it is no longer considered as a front. Finally all shared faces will no more be fronts as they cannot be expanded. The selection of points to create the new faces (thus elements) encourages the use of existing points. Figure 2.9 illustrates how the advancing front works.

Note that this technique can be adapted to produce any type of elements (tetrahedra, hexahedra and mixed-elements [42] in general). Also note that there is no constraint over the technique to insert a new point. There are several advancing front - based techniques and the difference between them is the manner to chose where and when inserting a new point. This last step tries to maximize the quality

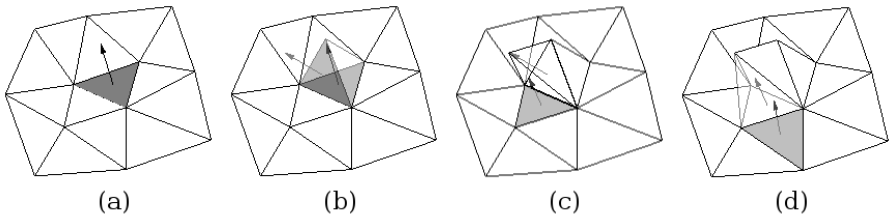

Figure 2.9: The advancing front technique: (a) A portion of a surface mesh with one front to expand, (b) the tetrahedron is created and the new faces can be treated as fronts, (c) another expansion using a recently inserted front and (d) another expansion using already inserted points.

of the elements in the mesh. As there are several options to measure the quality, this technique has several flavors.

In the next two sections more complex meshing strategies are presented. Several among them, use the techniques described above as a starting point or as part of the meshing process.

2.3 Mesh Adaptation

2.3.1 The mesh-matching algorithm

Couteau *et al* in [20] proposed the so called **Mesh-Matching** (M–M) algorithm. The statement of this work is that building a "suitable" mesh for their purposes is a complex task: count with just a few elements and achieve a "good" domain representation is always a difficult compromise. Therefore in many cases, meshes are built by hand. As this task is time consuming and cannot be done in practice for every patient, the idea is to use a pre-defined FE mesh of the target domain in order to achieve its representation.

The pre-defined mesh is called the ATLAS. This mesh is, in most cases, built by hand in order to preserve element quality, orientation and density in the desired regions, in terms of what is important for the simulation.

To build a new mesh, target information must be represented with a cloud of points at the surface of the new domain to mesh. Then a registration process *matches* surface nodes of the ATLAS with the target surface points. The goal of this process is to find a 3D transform T that is the combination of a rigid-body transform RT, a global warping W and local displacement function S as follows:

$$T_p = RT \circ W \circ S$$

Where p is a vector gathering the parameters of RT, W and S. Let $M = \{M_i, i = 1, \ldots, N_1\}$ and $P = \{P_i, i = 1, \ldots, N_1\}$ be respectively the ATLAS

surface nodes and the target surface points (for example patient data obtained by the segmentation algorithm). The elastic registration algorithm minimizes a least-squares criterion $E(p)$ given by

$$E(p) = \sum_{i=1}^{N_1} \frac{1}{\sigma_i^2} [dist(P, T_p(M_i))]^2 + R_p$$

where R defines a regularization term that is applied to S to obtain a smooth displacement function, σ_i^2 is the noise variance of the measurement i and $dist$ is the distance between P and a point M_i' (transformed by T).

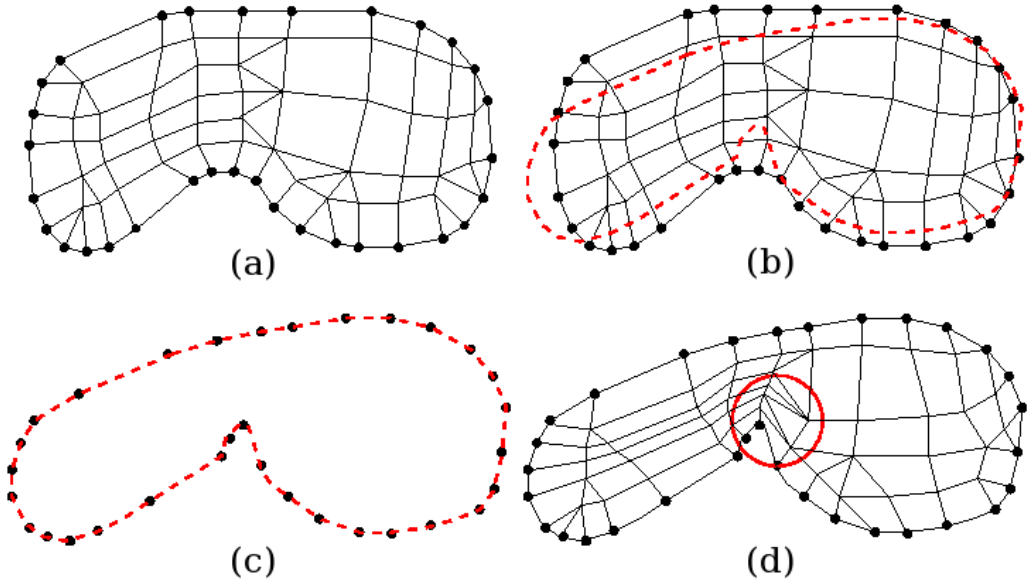

Figure 2.10: The mesh-matching algorithm: (a) The ATLAS mesh, (b) the ATLAS and the target surface domain, (c) the matching of ATLAS surface points into the target domain and (d) the final target mesh obtained by applying the transformation function T into the entire ATLAS mesh.

As T is a 3D transformation function, it can be applied to any point in the initial model. When T is applied to the internal nodes of the ATLAS, a final 3D target mesh is achieved. The matching of a domain can be seen in figure 2.10 where the different steps of the algorithm are shown. Also note that there is a circle in picture (d) of the figure to reflect bad quality elements. This is to reflect that morphing an ATLAS into another domain is not an easy task. The ATLAS must be prepared to confront those types of problems. If in most cases, the target meshes differ a lot from one to another in a specific region, the ATLAS can prevent this type of problems by adding points in those zones.

Another strategy to solve this problem is to use triangles (or tetrahedra in 3D) on those zones. The benefit of this is that contrary to quad (or hexahedra) it is

much more difficult to put a triangle or a tetrahedra in an invalid state (please see subsection 2.3.4 for more details). Therefore a probably good atlas would be the one shown in figure 2.11. Note that knowing the zones that deform the most, allows adjusting the number of elements to put in each zone. Therefore, this final mesh represents better the target domain with fewer element than the resulting one from figure 2.10 (as fewer elements are presented in the zones where the deformation is less important).

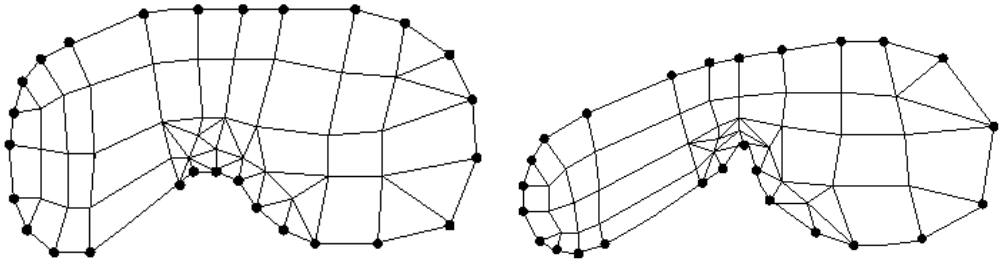

Figure 2.11: An improvement to the ATLAS: using more triangles in complicated zones (left) can improve element quality in the resulting mesh (right).

The M–M algorithm can be used with meshes of any type of elements. In the particular case of Couteau *et al* in [20], the goal was to obtain femora head meshes and the tested ATLAS was composed of hexahedra and wedges (prisms). In the article, a test over 10 proximal femora is presented. The level of element distortion was satisfactory in comparison to the initial ATLAS.

2.3.2 Meshing 4D Domains

The work of Montagnat and Delingette in [49] is focused on building **surface models** of 3D domains over time (4D). Despite the surface models (instead of volumetric meshes), this work is important to this *state of the art* as it considers deformable domains through time. But before explaining the 4D technique, lets see the 3D deformable meshes from the same authors in [48].

The ATLAS mesh (initial model), is a simplex mesh. This mesh can be seen as the dual mesh of a triangulation. Each point of the simplex corresponds to a triangle's centroids[2] in the triangulation as shown in figure 2.12 where the simplex mesh is represented by dashed lines. The simplex mesh has some interesting properties like constant connectivity (each point is connected to three others in the discretization), highly deformable, avoiding surface parametrization problems and other properties as mentioned by Delingette in [22].

[2]Note that this is different to the *Voronoi diagram*. If not familiar with this last concept, see: `http://en.wikipedia.org/wiki/Voronoi`

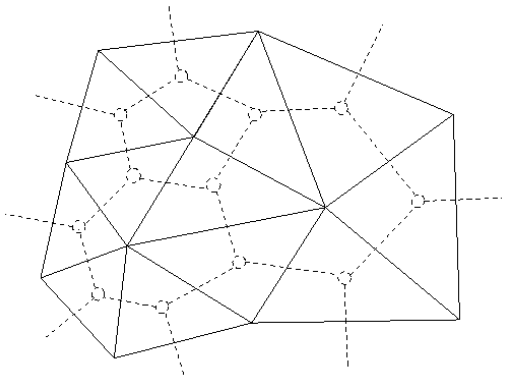

Figure 2.12: A triangle mesh and its corresponding simplex mesh (in dashed lines) by the junction of all the triangle's centers.

Back to the work [48], an analysis of registration techniques *vs* Free Form Deformations (FFD) is made. It mentions the positive and negative aspects of both techniques and finally reports a workflow to merge them. It analyzes the different results obtained by each technique.

There are several types of registration. The basic one is rigid registration, that consider 6 Degrees Of Freedom (DOF). Then, as the number of DOF increases, the registration can achieve a better representation. The authors work shows results with 6,7 and 12 DOF. It also analyzes the result with one registration governed by cubic B-spline.

To couple registration and FFDs, the authors introduce a λ factor that allows the interaction of both registration and FFD. When $\lambda = 0$ the point's displacement has a minimum of DOFs (Registration). As λ approaches to the value 1, the number of DOFs increases. Let V_i^{t+1} be the position of vertex i in iteration $t+1$, then the position of any vertex in the mesh can be obtained as:

$$V_i^{t+1} = F(V_i^t, V_i^{t-1}) + \lambda f_i^{FFD} + (1-\lambda) f_i^{registration}$$

where F is a function of the previous position of the vertex i, $f_i^{registration}$ are the forces that control the regular registration and f_i^{FFD} are the forces that control FFD.

With this equation, it is possible to iteratively increase the DOFs. The first "rigid" registration will stop when a threshold of the level of displacement is achieved. Then the DOFs are incremented by the use of λ and the registration continues until the final registration becomes a FFD with cubic B-splines.

The proposed registration hierarchy allows this work to efficiently match one atlas against a cloud of points as some other optimizations are proposed in the article. But going one step further, some organs in the body are important to

analyze in motion. Therefore a substantial application of the presented work is to model 3D domains that change with time (4D), as for example: the heart.

The work [49], first study a set of images that represent the states of the target organ through time. With the ATLAS of the organ, registration matches the cloud of points of the domain in the different states. As the domain changes, the ATLAS changes too and through all the different states, it simulates the motion of the organ.

2.3.3 Mesh Warping

As mentioned by Wolberg in [64], there are several algorithms to morph an image into another. One of them is the mesh warping algorithm that was first introduced for a special effect sequence in the movie "Willow" in 1988 by Smythe in [62]. It allows producing a smooth transition between two images by using the same mesh in both images and identifies some control points that will lead the transformation.

The work of Castellano-Smith *et al* in [13] extends the 2D warping to 3D. In order to do so, the developed technique by Castellano-Smith is strongly based on the 2D image registration proposed by Schnabel *et al* [57] that works with grids (see section 2.2.2).

Following the same principle as the one in the previous section, the schema of Schnabel is divided in two steps. First, the global motion is corrected using a rigid (6 DOF) or affine (12 DOF) transformation. The global motion then becomes the starting estimate for the second stage, where the local motion is further modeled using Free Form Deformations (FFD) based on B-splines. The combined motion model can be written as:

$$\mathrm{T}(x, y, z) = \mathrm{T}_{global}(x, y, z) + \mathrm{T}_{local}(x, y, z) \tag{2.1}$$

with the local motion at each point given by the 3D tensor product of the familiar 1D cubic B-splines. The optimal transformation T is determined by minimizing a registration cost function:

$$\mathcal{C} = -\mathcal{C}_{similarity}(\mathcal{I}_A, \mathrm{T}(\mathcal{I}_B)) + \lambda \mathcal{C}_{deformation}(\mathrm{T}) \tag{2.2}$$

The similarity term maximizes the voxel similarity between the ATLAS image ($\mathcal{I}_A$), and its counterpart in the target image ($\mathcal{I}_B$). The deformation cost term is defined as the 3D equivalent of a thin-plate bending energy in order to maximize the smoothness of the transformation, weighted by a factor λ.

As mentioned in the article, single-resolution FFDs are limited by low mesh resolutions, and may develop folding at high resolutions. Therefore a multi-resolution mesh representation is proposed. Each image is registered several times by using grids of different level of refinements. Note that control points can differ

from one grid to another one. After registration, the local deformation of each point in the image volume domain is given by the sum of the local deformations across levels:

$$\mathrm{T}_{local}(x, y, z) = \sum_{h=1}^{H} \mathrm{T}^{h}_{local}(x, y, z) \tag{2.3}$$

where each T^h_{local} is computed with respect to the B-spline of level h. The use of multi-resolution FFDs avoids the folding problem. Therefore the deformation cost term regularizing the smoothness of the deformation in equation 2.2 is no longer crucial (λ can be set to 0).

In practice, this technique is applied by Castellano-Smith *et al* in [13]. In that study, a volumetric ATLAS of the brain is built from a set of MRI and the control points are selected. To build a new brain mesh, it is necessary to count with MRIs of the target (equivalents to the ones from the ATLAS) and to identify the position of the control points. The registration process then matches the ATLAS mesh to the target mesh. Note that the registration provided in the paper is made from a set of 2D images. As their points are correlated to a 3D mesh, it is possible to obtain a 3D mesh as a result.

The big difference between this method and the Mesh Matching (M–M) technique is that it allows identifying inner points to lead the registration. This is very useful to consider the matching of inner structures like in the case of the brain. However, Berar *et al* proposed in [7] a combined approach using the M–M and a multi-resolution grid for matching high resolution ATLAS into low density target data. This was applied to mandible simulation. Another important remark over this last work is that for the definition of control points, they didn't use geometrical points but anatomical ones like the teeth, even though the teeth were not presented in the target model.

Even though the ATLAS can be built with different meshing techniques, it is important to mention that in the work of Castellano-Smith *et al*, the ATLAS was built from a set of segmented images. Then with a based Marching-Cubes algorithm they produced a surface model of the brain and inner structures. Their paper doesn't give details on the volumetric mesh generation; however it mentions that the final mesh is a tetrahedral mesh of high quality. The last important thing to mention is that they have used quadratic tetrahedra, i.e. 10 nodes as explained in section 1.4.

Finally the differences between this registration process and the one proposed by Montagnat and Delingette in [49] are:

- The type of mesh they use (grid *vs* simplex).
- The first approach uses an iterative strategy where the level of the mesh refinement and the registration are coupled in each iteration.

- The entire process is leaded by some specific control points that are added in each iteration.

2.3.4 Untangling and improving quality after mesh adaptation

The accuracy and efficiency of the solution to numerical systems depend on the quality of the mesh. Moreover, in the case of registration techniques, it is possible to compress/stress an element to the point where it becomes folded (i.e. degenerated and therefore invalid for the FEM) as figure 2.13 shows. This can occur since the registration techniques do not consider the geometry of the elements in the mesh (the registration is made regarding the points and not the elements).

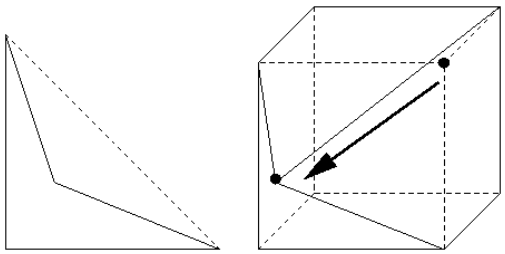

Figure 2.13: A folded face (left) where the dotted line is a diagonal and a folded hexahedron (right) where the dotted lines represent a perfect hexahedron.

The workflow of element reparation after registration should be:

- make the mesh valid (repair tangle and folded elements) and
- improve element's quality when possible.

In the work of Schnabel *et al* [57] presented in subsection 2.3.3, this was done in 2D for each image as the idea of hierarchically increase the refinement level of the mesh[3], avoids the presence of "folded" squares. When passing to the 3D volumetric mesh, this problem was not relevant as the final mesh was tetrahedra-only type. Note that folded faces cannot be produced with triangle faces.

One solution to solve tangled and folded elements was proposed by Luboz *et al* in [45]. This work continues the Mesh Matching (M–M) technique presented in subsection 2.3.1, as it repairs the invalid elements that are sometimes produced after the registration.

The first step consists in detecting the tangle elements. To do so, the Jacobian matrix is used regarding the transformation between the reference and the actual space. This transformation can be computed for any type of element. In particular

[3]Count with several meshes of different levels of detail. For example a grid with 4, 16 and 64 quads.

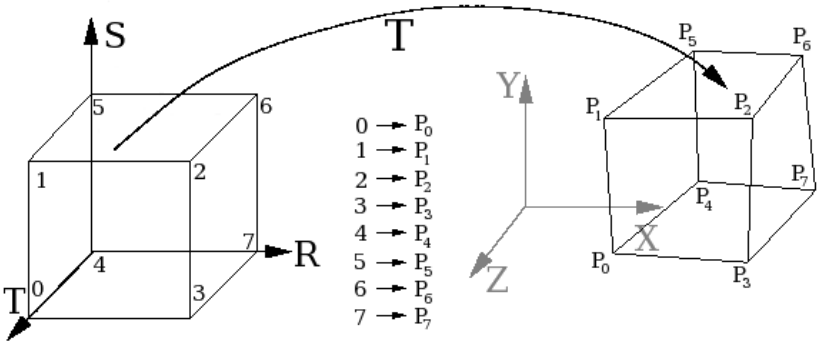

Figure 2.14: The reference and actual space from which the Jacobian matrix is calculated. Left: the reference space and right: the actual space.

figure 2.14 shows an example for a hexahedron: at the left the reference space $\mathcal{R}_{rst}$ and at the right the actual space $\mathcal{R}_{xyz}$. The Jacobian matrix for this transformation can be written as:

$$\mathrm{J} = \begin{bmatrix} \mathrm{J}_{xx} & \mathrm{J}_{xy} & \mathrm{J}_{xz} \\ \mathrm{J}_{yx} & \mathrm{J}_{yy} & \mathrm{J}_{yz} \\ \mathrm{J}_{zx} & \mathrm{J}_{zy} & \mathrm{J}_{zz} \end{bmatrix} = \begin{bmatrix} \sum_i \frac{\partial N_i}{\partial r} x_i & \sum_i \frac{\partial N_i}{\partial r} y_i & \sum_i \frac{\partial N_i}{\partial r} z_i \\ \sum_i \frac{\partial N_i}{\partial s} x_i & \sum_i \frac{\partial N_i}{\partial s} y_i & \sum_i \frac{\partial N_i}{\partial s} z_i \\ \sum_i \frac{\partial N_i}{\partial t} x_i & \sum_i \frac{\partial N_i}{\partial t} y_i & \sum_i \frac{\partial N_i}{\partial t} z_i \end{bmatrix}$$

Where N_i are the interpolation shape functions, x_i, y_i and z_i are the actual coordinates and r, s, t the coordinates in the reference space.

It is possible to apply the Finite Element Method (FEM) if this transformation matrix exists for any element in the mesh. This is equivalent to say that simulation is possible when this matrix is not singular. It is possible to detect singularities by the analysis of the sign of the determinant of **J** (**detJ**). By evaluating **detJ** in each vertex of the element, a singularity is presented when not all these values have the same sign (i.e. when a null value is presented somewhere inside the element).

In order to repair those elements, the gradient of the **detJ** is computed. For a distorted element k, a displacement $Disp_{k,j}$ for the vertex j is proposed as follows:

$$Disp_{k,j} = \sum_i \nabla_j \mathbf{detJ} = \begin{bmatrix} \sum_i \frac{\partial \mathbf{detJ}_i}{\partial x_j} \\ \sum_i \frac{\partial \mathbf{detJ}_i}{\partial y_j} \\ \sum_i \frac{\partial \mathbf{detJ}_i}{\partial z_j} \end{bmatrix}$$

where i are the vertex index of element k. The total displacement $Disp_j$ for a given vertex j is computed by adding each $Disp_{k,j}$ obtained for every invalid element k where the vertex j is presented. This displacement vector is then normalized and the final position P'_j for a given point j with current position is P_j can be obtained as follows:

$$P'_j = P_j + disp_j \times W_j$$

where W_j is a weight factor chosen to constrain the displacement.

Another approach to solve this problem was proposed by Li and Freitag in [38]. This work intends an optimization approach to untangle (equivalent to solve the folding in 2D) the mesh. It searches to maximize the minimal area or volume of the elements. The first step is to detect tangled elements. This is done by computing the Jacobian sign at each element's vertex. A very important point must be noticed here: the work mentions that an element with one vertex of negative Jacobian can be a valid element. Note that this refers to tangling, i.e., an element can be not tangled having one negative Jacobian vertex. However, this has nothing to deal with the FEM. A negative Jacobian vertex does not allow to compute the FEM, therefore the statement should be: a mesh can be untangled but count with negative Jacobian vertex, however this mesh will not be valid from a FEM point of view as it is not possible to compute a simulation with.

The **laplacian** smoothing technique (see [27]) consists in positioning each vertex v of the mesh in the center of the polygon formed by all the vertex connected to it. This method operates heuristically and does not assure quality improvement over the elements of the mesh; for example it can produce slivers[4]. Therefore, optimization techniques based on some quality measure (dihedral angle, aspect-ratio, etc.) are preferred. Unfortunately optimization approaches have a computational cost much higher than the laplacian smoothing. Therefore a combined strategy is proposed in [27] by Freitag. But before going into details it is necessary to introduce two concepts.

The laplacian (this is a smoothing process) by its own does not consider quality measures. Therefore the first modification is the introduction of a "smart lapla-

[4]Tetrahedra where all the points tend to be co-planar.

cian". This is the typical laplacian but it will be applied only if the new position for v improves the quality of the elements.

Even though it is not presented like this, for understanding purposes lets say that the active value $actVal_v$ corresponds to the worst element quality associated to vertex v, and $actVal_{mesh}$ the worst element quality in the mesh. Now to improve element quality, a user-defined $threshold$ value is used and the following iterative algorithms are proposed (each algorithm is independent):

1. If $threshold < actVal_{mesh}$ the *smart laplacian* is performed. Otherwise the optimization is used
2. Apply *smart laplacian* and compute the new $actVal_v$. If $threshold < actVal_v$ repair it using optimization.
3. If $threshold < actVal_{mesh}$ the process is finished. Otherwise apply *smart laplacian* and compute the new $actVal_{mesh}$. If $threshold < actVal_{mesh}$ apply optimization.
4. Compute $threshold = actVal_{mesh} + C$, where C is an arbitrary constant value. Then proceed as the second algorithm (from this list).

The author concludes that all the above algorithms but the third were superior to the *smart laplacian* and optimization approach. However, the third combined approach was the fastest and it also improved the mesh (only with a worst quality than the others but still, better than the input).

2.4 Mesh generation

2.4.1 Red Green tetrahedral meshing technique

Molino *et al* [47] have introduced the **red green** subdivision algorithm. This meshing technique is said to be for modelling highly deformable objects.

The implemented approach starts with a regular octree mesh, i.e. all the octants in the mesh are refined at the same level. Therefore it can be seen as a voxel-based approach but conserving the tree structure of an octree.

Once the desired level of refinement is achieved, each cube is divided in 6 pyramids and each of them in 2 tetrahedrons. As this is done for each hexahedron in the mesh, the global tree structure can be conserved as shown in figure 2.5. The difference is that for each leaf in the tree, there will be a reference to the 12 tetrahedrons that replace the original cube.

In order to achieve good surface representation some tetrahedrons must be split. Therefore a pattern that enables tetrahedron subdivision is applied over the regions where more elements are needed. This pattern can be applied recursively over tetrahedra. Those elements are said to be *red* (they can still be refined). However,

between refined and not refined regions there must be a consistent transition. In order to avoid propagation of the refinement level, new *green* patterns (see figure 2.15) are applied. Those green tetrahedrons cannot be refined anymore. If for some reason a green tetrahedron must be refined later, it is replaced including his *brothers* with the hierarchical red father (using the tree data structure that enables this process to be fast and robust). Now, with this red tetrahedron the subdivision is applied until the desired element density is achieved.

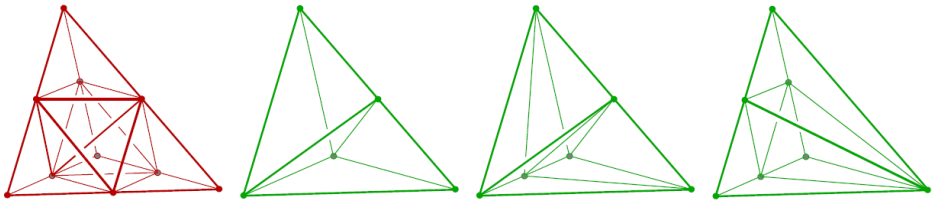

Figure 2.15: The **red green** patterns to split tetrahedrons. The standard red refinement (left) and the three green patterns for transitions between regions of different refinement levels (right).

The red green meshes are desirable because, except for the green tetrahedrons in the transitions, they produce equilateral tetrahedrons (in difference with a pure *Delaunay* approach), avoiding sliver tetrahedra (which increases the level of computation errors in the simulation). It is for this reason that they are said to be optimized for objects that will be highly deformed.

In the context of neurosurgery and brain modeling, Fedorov *et al* in [23] proposed to use the FEM to project the output mesh of the **red green** algorithm in order to achieve the correct external surface representation. They defined the displacements of the boundary nodes toward the object surface using a distance map, and solved the system with boundary conditions defined by the displacements of the mesh vertices.

2.4.2 Marching Cubes with Delaunay-based meshes towards several bodies simulation

The work of Audette *et al* in [5] might be considered as an accurate simulation because it builds detailed meshes for several structures of the brain. However their intention is to produce a realistic simulation of the pitatoria surgery for training purposes. Through virtual reality, this simulation is produced in real-time, therefore it can be considered as part of the real-time simulations.

To simulate the endoscopic pituitary surgery, they produce several non connected triangle surface meshes. Each mesh represents a volumetric or a surface

mesh of the head structures like the pituitary gland and the arteries, the cranial nerves, the dura mater, the brain and other pathological structures.

In order to produce those surface meshes they first generate a dense surface mesh of high accuracy and topological fidelity, resulting from the Marching Cubes algorithm. The second step corresponds to a decimation process in order to obtain a mesh of the desired density.

The third step consists in building the simplex mesh of the previous calculated mesh (to see a description of the simplex mesh, please see subsection 2.3.2). An optimization process leaded by a balloon force can act over the points of the simplex mesh to cause it to expand until some image-based force halts this expansion. In consequence each dual triangle is reallocated in order to count with edges of equal or locally consistent length. In other words, as the relation between a triangle mesh and the simplex is one-to-one, a displacement over a point in the simplex mesh causes a displacement over the three corresponding points in the triangle. Therefore the optimization over all the simplex points improves the quality of the triangles (as an optimization runs over each triangle's center).

Once all the high quality meshes are obtained, Audette *et al* in [5] propose to add inner points inside the surfaces that will be modeled as a volumetric object. To do so they use a criterion of "near-equal length of the tetrahedron". Once they have enough inner points the authors proceed to build a Delaunay mesh of it.

Finally, they improve the quality of the mesh by making an optimization over the dihedral angle[5]. They achieve an excellent representation of the inner structures of the brain, allowing via Virtual Reality, the training on pitatoria surgery. This strategy is under the category of **Volume generation techniques** because even though the authors include the input surface mesh to produce the volume mesh, in a previous step, they regenerate the surface mesh with the Marching Cubes technique. Then they build a new dense surface mesh that soon after is decimated and goes through a quality improvement process.

A previous work by Pages *et al* in [52], developed a general purpose technique similar to the one above. The goal was to develop an application to go from MRI and CT images to volume meshes. The first step was to achieve image segmentation. As this process is even more complicated when the simulation must consider several sub-structures of the image, they used a user-guided segmentation algorithm. Once the segmented images are produced, the only user interaction is to decide the level of precision (see 1.3.3) the mesh must have. With the segmented images, the marching cubes algorithm is applied to produce an initial surface mesh. Then a bilaplacian algorithm produced a smooth surface (this step wasn't applied by Audette *et al* in [5]) to finally proceed with a decimation algorithm to decrease the number of triangles. Now with a "good" quality surface

[5]The angle formed between two planes.

mesh, the inner points are inserted. These new points are inserted regarding the Delaunay property to produce an initial volume mesh. In this work, the quality of a tetrahedron, is defined by $Q = \alpha\rho/h$, where ρ is the inradius, h the largest edge length and α a normalization coefficient (to get a quality of 1 for a regular element). Following this quality function, an optimization process improves the element's quality by node relocation and face flipping operations.

2.4.3 Variational tetrahedral meshing

This work was developed by Alliez *et al* in [2]. It is a general purpose meshing technique starting from a Delaunay mesh. Then the meshes are improved in quality by optimal point insertion in terms of tetrahedral quality.

Concerning the mesh generation, the important constraints to this work are:

- Count with a fair shape quality measure. They make a reference to the work of Shewchuk in [60] and conclude that: "The **radius ratio**, which takes the quotient of inscribed and circumscribed sphere radii (times three for normalization purposes), is a good measure for any kind of degeneracy". This specially avoids the presence of sliver elements.
- Sizing requirement. Citing directly from [2]: "Accuracy and efficiency of numerical solvers depend on the local size of tetrahedra. Consequently, a sizing field, prescribing the ideal local edge length as a function of space, must be added". In consequence, a smooth transition between elements of different sizes is necessary to avoid bad dihedral angles.
- Boundary requirements. Two options are possible to represent the external surface of the domain to mesh: directly include the input surface mesh in the final output mesh or re-mesh the input domain in order to control its quality and density. The first, of course, achieves perfect surface representation as shown in subsection 2.2.4, however as Alliez *et al* intend to produce a meshing technique for general purposes, the second option was chosen. In this manner, there is no dependency on the density or quality of the input domain. Therefore this technique can be classified as a **Volume re-generation technique**.

Regarding all the above points, Alliez and colleagues present a Delaunay-based optimization technique, called "Variational Tetrahedral Meshing", to efficiently mesh a bounded 3D domain of arbitrary topology or number of connected components. The base of the algorithm considers optimal surface approximation by the work of Cohen-Steiner *et al* in [17] and Optimal Delaunay Triangulation by the work of Chen and Xu in [14]. Regarding those techniques the authors propose a simple minimization procedure that alternates global 3D Delaunay triangulation and local vertex relocation to consistently and efficiently minimize a global en-

ergy over the domain. It results in a robust meshing technique that generates high quality isotropic meshes in terms of radius ratios, as well as angles. A notable feature of the method is that it removes slivers (the principal problem of the Delaunay technique as explained in subsection 2.2.1) inside the domain. To provide a flexible meshing tool, they also introduce an automatic sizing field construction that guarantees an arbitrary smooth gradation of the mesh together with faithful approximation of the domain boundary.

The authors provided several comparisons with other quality tetrahedral meshing techniques. In all the presented cases their technique gave the highest mesh quality in terms of tetrahedra radius ratios (the one that avoid sliver tetrahedrons and also considers the dihedral angle).

2.4.4 A brief remark concerning surfaces

As reviewed in the previous works, to get a good representation of the input domain surface is always a difficult task. Many approaches go through high refinement levels to achieve such a good representation and then, by decimation or point reallocation, decrease the number of faces and try to improve their quality.

Another option to improve surface quality is to change the topology of the mesh. This is not exactly decimation as the work of Bunin in [12] shows. Even though it is focused in 2D, it can be adapted to improve the quality of surface meshes in 3D. The quality improvement over the surface is achieved by removing "bad quality" or "undesirable" sub-domains of the mesh and by re-meshing those zones. Contrary to decimation, this doesn't only change the topology but can also improve the orientation of the elements. In particular this approach improves quadrilateral meshes.

When the focus is triangle surfaces, one remarkable work is the one by Coll *et al* in [18]. The algorithm combines different meshing improvements techniques that incrementally modify an initial triangle mesh by local changes. The element insertion or removal is done in such a way that the mesh quality is maintained during the process. The local operations are: deletion, insertion and addition of Steiner[6] points. The overall process goes through a definition of quality zones. Therefore vertices of skinny triangles are forced to those zones. If a vertex must be removed, new vertices are placed in quality zones to improve the overall quality.

[6]If in a given Delaunay tetrahedralization or triangulation of an arbitrary domain, a new point is inserted in such a way that the entire mesh still satisfies the Delaunay property, this new point is said to be a **Steiner** point.

2.4.5 Hexahedral meshes

Unstructured quadrilateral/hexahedral meshes are also an important alternative to simulate applications with the Finite Element Method (FEM). However, it still remains a challenging and open problem to generate adaptive and quality quad/hex meshes directly from volumetric data, such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI).

The work of Zhang and Bajaj in [66] goes in this direction. The overall strategy consists of:

- Select a starting level of octree refinement for uniform mesh generation with correct topology, regarding the volumetric data.
- Produce a representative quad/hex mesh of the input domain.
- Improve the quality of the final elements.

Once an initial octree level of refinement is achieved (see subsection 2.2.2) five patterns enable transitions between refined and coarse regions. Those patterns are applied in function of the vertex signs. This is based on the "dual contouring method" proposed by Ju *et al* in [35]. This last method consists in an approximation of the surface by assigning directions to the vertices in all the sections where domain intersection is produced.

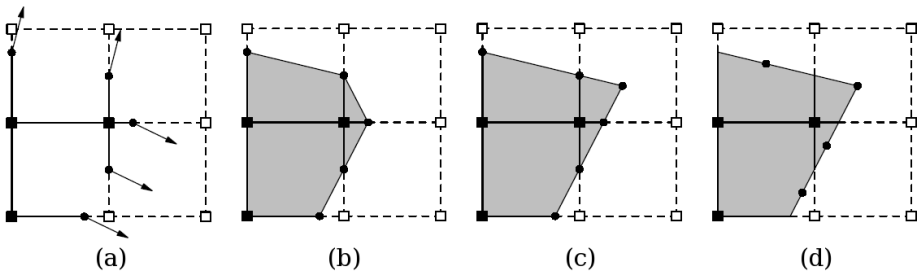

Figure 2.16: Extended Marching Cubes and dual contouring: (a) Octree mesh with domain cutting intersections and the normal of those cutting points, (b) the marching cubes output, (c) the extended marching cubes contour and (d) the dual contour mesh.

Figure 2.16 shows the contour detection and assigns a sign to each octree vertex regarding the normals of the intersection points. This can be seen in (a) as the black points rest inside the domain. Image (b) represents the output from the marching cubes technique. Image (c) shows the "extended marching cube" (EMC) technique proposed by Kobbelt *et al* in [37] and (d) the points of the dual contour regarding the EMC. As in subsection 2.3.2 the simplex mesh was the dual of a triangle mesh, here a dual mesh is expanded to a more global concept as it is not restricted to triangles (see [35] for more details).

With all these techniques, the work in [66] allows to produce quad/hexa meshes that have regions with different levels of refinement and achieve a "good" surface representation. Finally in [65] Zhang and Zhao continues the previous work by adding two new patterns to drastically decrease the number of transition elements between refined and coarse regions.

2.5 Summary over presented techniques

In the two previous sections, several meshing techniques were presented. In the following tables a summary of them is proposed, involving all the relevant aspects of meshing.

meshing type	reviewed works
registration	Couteau [20], Montagnat [49], Castellano-Smith [13], Berar [7]
generation	Alliez [2], Audette [5], Molino [47], Abell [1], Ashburner [4], Velasco [63], Pages [52], Nesme [51], Lorensen [43], Payne [53], Ferrant [25], Montagnat [49] Frey [28], Zhang [66]

Table 2.1: The presented techniques grouped by the type of technique to produce the mesh.

There are more generation techniques than registration methods as table 2.1 shows. Note that this doesn't mean that in medical applications, the generation methods are more used than registration techniques. As generation methods tends to solve general problems there is an important global effort to improve them. In the other hand, registration methods are mostly used in the medical field as the difference from one target to another one makes possible to re-use already generated meshes (ATLAS). Registration methods are also largely used in the medical field due to the complexity of the anatomical structures. The goal isn't just to achieve surface representation but also to conserve some element orientation (alienation) regarding sub-structures (like different types of materials, behavior, physical properties, etc).

Also note that several tetrahedral generators improve the quality of the input surface mesh and then continue with some point insertion strategies (using the modified input surface mesh) in order to produce the final volume mesh. In other words, techniques that use the input surface mesh, as the Advancing Front, seem to be "not so used" in the medical field.

Table 2.2 presents the reviewed techniques regarding the basic and most common meshing techniques and concepts explained in section 2.2. Note how the Delaunay property is used by all the techniques that produce tetrahedral meshes. It is

meshing technique	Delaunay	octree-alike	march. cubes
Ruppert [56]	ok	–	–
Shewchuk [60]	ok	–	–
Miller [46]	ok	–	–
Abell [1]	–	ok	–
Ashburner [4]	–	ok	–
Finkel [26]	–	ok	–
Nesme [51]	–	ok	–
Lorensen [43]	–	ok	–
Chernyaev [15]	–	ok	ok
Payne [53]	ok	ok	–
Ferrant [25]	ok	ok	–
Velasco [63]	ok	ok	ok
Couteau [20]	–	–	–
Castellano-Smith [13]	–	ok	ok
Berar [7]	–	–	–
Molino [47]	ok	ok	–
Audette [5]	ok	ok	ok
Pages [52]	ok	ok	ok
Alliez [2]	ok	–	–
Chen [14]	ok	–	–
Frey [28]	ok	–	–
Zhang [66]	–	ok	ok

Table 2.2: Meshing techniques using the basic techniques: (1) Delaunay property, (2) octree, grid or voxel and (3) marching cubes

important to mention that not all the techniques produce volumetric meshes. Some of them are used only to achieve a good surface representation. Another important remark is that several techniques producing tetrahedra, also use the octree-alike methods to produce a primary tessellation.

Table 2.3 focuses on the type of element produced by the volumetric mesh generation techniques. Note how mixed–element meshes are by far the less used.

The goal of this chapter was to show the different possible techniques to produce a suitable mesh for the FEM. Therefore, several related works were omitted as the meshing techniques they used were already explained.

Regarding all the mesh generation techniques presented in this chapter, a workflow for general mesh generation is presented in figure 2.17. This is based on the work of Frey in [28]. In this picture an example of "isosurface reconstruction" is the Marching Cubes algorithm. The "mesh optimization" with the "geometric

meshing technique	tetrahedra	hexahedra	mixed-elements
Abell [1]	–	ok	–
Ashburner [4]	–	ok	–
Nesme [51]	–	ok	–
Lorensen [43]	ok	–	–
Payne [53]	ok	–	–
Ferrant [25]	ok	–	–
Velasco [63]	ok	–	–
Couteau [20]	–	–	ok
Castellano-Smith [13]	ok	–	–
Berar [7]	–	–	ok
Molino [47]	ok	–	–
Audette [5]	ok	–	–
Pages [52]	ok	–	–
Alliez [2]	ok	–	–
Frey [28]	ok	–	–
Zhang [66]	–	ok	–

Table 2.3: The final type of elements that the analyzed meshing techniques produce.

constraints" refers to the element deletion or insertion per region in function of the simulation requirements.

Finally, regarding all the adaptation techniques, figure 2.18 shows the equivalent workflow that we propose for a general mesh registration technique. The "rigid registration" can be of different DOF (6,7 or 12) as mentioned in subsection 2.3.2. The "elastic registration" is equivalent to "Free Form Deformation" or "Non Rigid Registration" concepts. The "Element Reparation" corresponds to a re-allocation of the negative Jacobian points and the "quality improvement" needs a measure of quality and then performs an optimization over the elements regarding the selected criterion. Note that a "Smoothing" process is also proposed in order to produce the final volume mesh of the target domain.

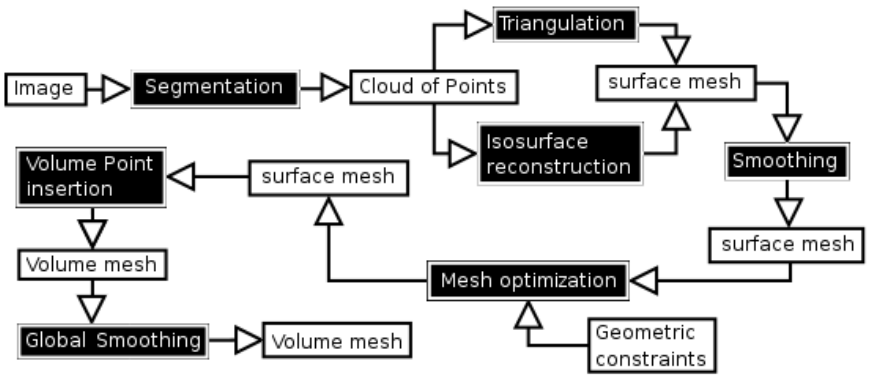

Figure 2.17: The mesh generation general flow. From Image to volume mesh generation based on Frey [28], where a white box represents input/output data and a black box is a process.

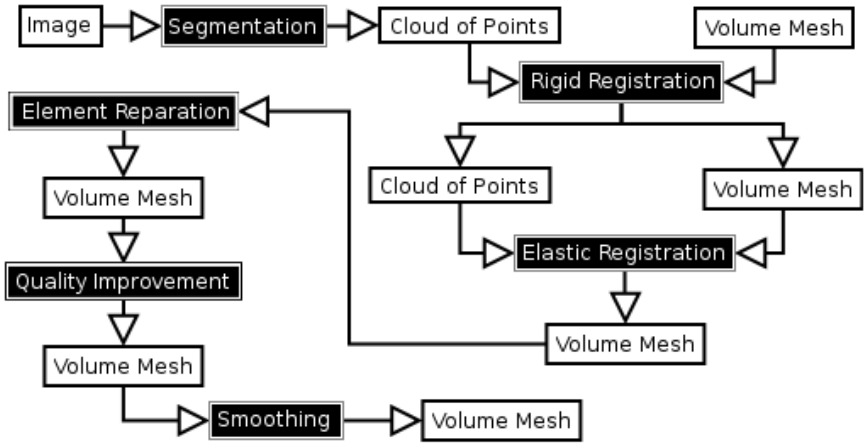

Figure 2.18: The mesh adaptation general flow. From Image to volume mesh generation, where a white box represents input/output data and a black box is a process.

Chapter 3

Simulation in the medical field

Chapitre 3: Simulation dans le domaine médical

Abstract

In this chapter the proposed solutions are explained in detail. First, section 3.1 gives a global introduction and presents the motivation for the two analyzed meshing techniques shown in this thesis. The first technique is presented on section 3.2 and corresponds to the reparation of meshes after registration. The second strategy, presented on section 3.3, is the generation of meshes where the focus is given on a particular region of the domain and it should be mostly applied in surgery where real-time simulations will be needed.

3.1 Meshing anatomical structures

In order to simulate any relevant phenomenon in the medical field like fracture, deformation, resection and many others, a model must be produced. As mentioned in the previous chapters, the Finite Element Method (FEM) is probably the most used technique to achieve those types of simulations. The FEM needs a subdivision of the domain, into simpler geometrical elements. The junction of all the elements (plus the nodes, the connectivity between them, the faces and the edges) receives the name of *mesh*.

This thesis is focused on two strategies to build a proper mesh and probably a good example to introduce both is the simulation of liver tumor resection. The liver as itself is a very important organ and also, from a mechanical point of view, involves a complex anatomical architecture. Seven different regions (the "lobes") with specific material behavior can be identified in the organ (figure 3.1 [1]). More-

[1]Image taken from wikipedia `http://commons.wikimedia.org/wiki/Image:Gray1087-liver.png` with a Public-domain license.

over, several sub-structures can be distinguished, like the capsule of glisson, vascular branches or the biliary tree, which adds mechanical non-linearity complexity to its simulation.

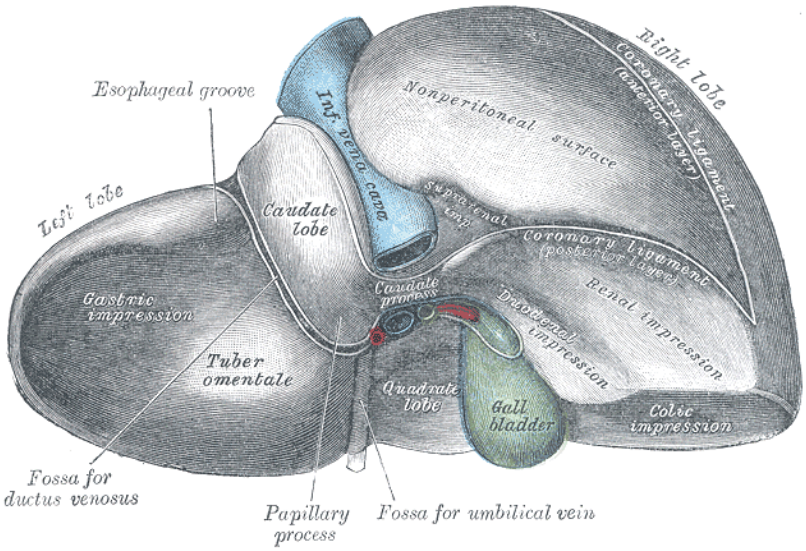

Figure 3.1: Anatomical view of the liver from Gray in [30].

The liver medical scan images might show anomalies in the organ. In order to confirm the existence of a tumor, a tissue sample is withdrawn and sent to the laboratory for testing. This surgical gesture (the biopsy) will finally indicates if the sample corresponds to a tumor.

Simulating this biopsy gesture is challenging from a modeling point of view (patient-specific geometry, anisotropy and non-homogeneity of the substructures, displacement due to breathing, between others).

At least to take into account patient-specific geometry and sub-structures organization (i.e. a mesh that identifies these sub-structures), the mesh registration techniques are probably the most adapted ones.

These techniques enable the use of a predefined mesh (the *atlas*) in order to adapt it to another domain (patient data). They are very important to the medical field as the *atlas* can be defined regarding all the organ internal structures, considering particular element types, orientation, density, quality and all the parameters relevant to the simulation. The construction of such an *atlas* might be a complex and hard task; however this manual work should be made only once. When the desired *atlas* is built, the registration techniques adapt the *atlas* mesh to the patient data thus generating a patient-specific mesh with the same topological configuration as the one from the *atlas*.

Section 3.2 will show how registration methods can be applied to simulations where anatomical information of the entire organ is crucial. In particular the Mesh-Matching (M-M) [20] algorithm is used as example and improved. It was shown in [44] that after registration, the M-M sometimes produced invalid and poor quality elements in certain regions. A strategy to repair and improve the quality of those meshes is one of the products of this thesis.

However, these registration techniques may be insufficient when a specific surgical treatment (for example tissue resection) has to be fully modeled for an intra-operative case. In that case, the focus of the simulation is on the region where the tumor is located. Even though the rest of the liver regions remains important to the simulation, it is on the region of the tumor where higher precision is needed and therefore an important optimization, regarding the mesh, must be made. In that case, the goal of the mesh generation process is to produce a mesh that is refined on the region of the tumor and coarse elsewhere. This optimization allows to reduce the number of nodes in the mesh and in consequence, enables a faster computation of the FEM. Note that the rest of the regions are still considered, but with less precision than the region of the tumor. Section 3.3 presents our developed algorithm to produce mixed-element meshes with localized region refinement and high control over the mesh quantity of nodes.

3.2 Mesh reparation and quality improvement after registration

In section 2.3 some registration techniques were introduced. In the medical field, this option is sometimes preferred since the *atlas* (in many cases, built by hand) considers specific element type, orientation, quality, size and many other characteristics important to the specific simulation.

As the nodes of the elements in the *atlas* suffer a reallocation in order to represent the target domain Ω, two types of problems can arise:

- the quality of the elements can slightly decrease (globally speaking).
- elements might become invalid for the FEM.

One of the objectives of our work was to provide reliable methods to avoid such kinds of problems.

Two measures were considered in the mesh reparation algorithm: the determinant of the Jacobian matrix (as a measure of validity and quality) and the warping factor (as a measure of quality). Both are explained in detail in the following subsections and finally both are employed in the two implemented strategies to repair the meshes.

3.2.1 The Jacobian determinant

The Jacobian determinant det**J** explained in subsection 2.3.4 is used to determine the degeneration level of an element. The value of det**J** indicates if an element is invalid, presents bad quality or if it is good enough to continue with the simulation.

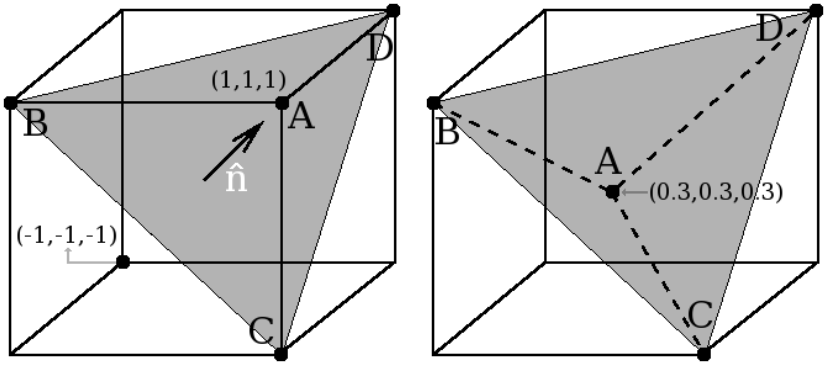

Figure 3.2: Left: a valid element where node A has a positive distance to plane BCD. Right: an invalid element where node A has a negative distance to plane BCD.

In geometrical terms, the validity and quality of an element can be described regarding the position of its nodes. Let A be a node of an element and BCD a plane formed by the other nodes of the element that are connected to A. In a valid element, node A has a positive distance to plane BCD. If node A is moved till the point where its distance to BCD is negative then det**J**(A) is negative and the element is said to be invalid. The valid case can be seen at the left panel of figure 3.2 for an hexahedron. At the right, an invalid configuration is shown as the distance of node A to plane BCD is negative.

After the adaptation (or registration) of the *atlas*, element quality decreases and invalidity often arises. Therefore, before explaining the mesh reparation algorithm that we propose, it is important to understand the behavior of the det**J** function:

- The value of det**J**(A) only depends on (and affects) the nodes connected to A. The values of det**J** for nodes non-connected to A in the element do not change by a reallocation of A.
- The value of det**J**(A) depends on the size of the element. When the element is scaled (increase the element size by keeping the relative distances between nodes) the value of det**J**(A) changes.
- A "perfect" element is achieved when all the nodes in the element have the same value for det**J**.

- DetJ is a continuous increasing function.

In order to normalize the detJ function, a ratio known as the Jacobian ratio is used. This ratio is computed for each node regarding the value of the detJ of each element node: $Jac_i^{ratio} = detJ^{max}/detJ_i$, where $detJ^{max}$ is the maximum detJ value of the element. In order to illustrate the detJ and the Jac^{ratio} functions a perfect cube of length side 2 centered at 0 is defined. Then node $A(1,1,1)$ is replaced with node (a,a,a), where $a \in [-1.9, 1.9]$. Figure 3.3 shows how the function varies as the node is reallocated.

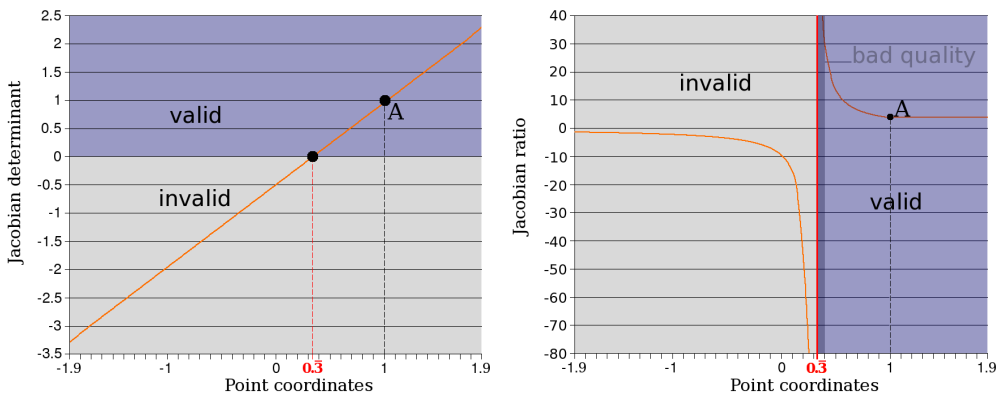

Figure 3.3: Left: the Jacobian determinant values as the position of an hexahedron node is moved along its coordinates (the value of x,y and z for the node are the same). Right: the same conditions are used to compute the Jacobian ratio value.

The value of Jac_i^{ratio} gives a reference of the element degradation at node i regarding all the locations of the element nodes. Note that when detJ$_i$ function has negative value, it indicates that the element is inverted (thus invalid) at node i, but there is no measure of the degradation level of the element. In the other hand, when the det J$_i$ is positive it is not possible to determine if node i is close to an invalid position; in figure 3.2 this corresponds to place node A with a positive distance to BCD but at the same time, close to it. The Jac^{ratio} can establish the element "degradation level" at each node, because when the node i is close to an invalid position it has a positive high value and as it approaches to the optimal position (regarding all the element nodes) its value tends to 1 (as the right panel of figure 3.3 shows). Note that when the value of Jac_i^{ratio} is 1, increasing detJ$_i$ will continue to result in a value of 1 for Jac_i^{ratio}, therefore increasing detJ$_i$ will only decrease the value of detJ for the rest of the element nodes (decreasing the element quality).

As mentioned before, the detJ value can be used to detect invalid and bad quality elements. An invalid element is the one that has negative or nil detJ for one

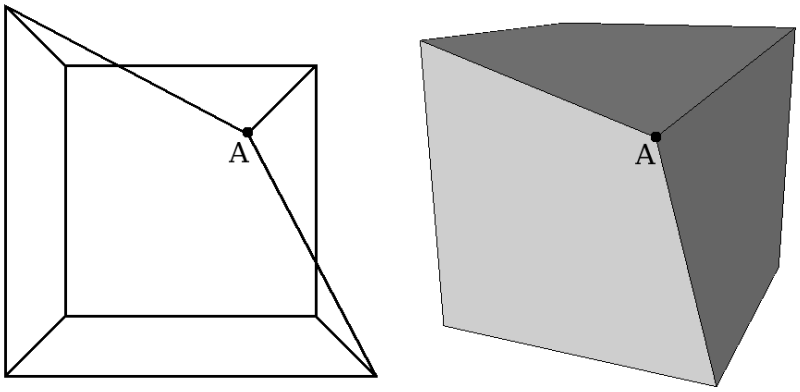

Figure 3.4: Left: transparent drawing of an hexahedron of $Jac^{ratio} = 29.9401$. Right: a solid drawing of the same hexahedron

or more of its nodes. When all the nodes have positive det**J** values the element can be seen as "valid". However following the standards of ANSYS®, an element is unacceptable for the FEM, when its Jac^{ratio} is superior to 30. Figure 3.4 shows an hexahedron where the "worst" $Jac^{ratio} = 29.9401$ (thus acceptable for ANSYS®). Note that a very small displacement over node A could easily increase its Jac^{ratio} value as **this function is exponential** (see the right plot of figure 3.3).

3.2.2 The warping factor

The warping factor WF is a quality parameter that measures the level of co-planarity of the nodes in a face. Note that this measure is meaningless for triangles[2]. The plane used to measure the node co-planarity is obtained as an average of the nodes and the implementation only considers quadrilateral faces. If n_i correspond to the nodes that define the face, a point over this plane is computed as:

$$avg = \frac{1}{4}\sum_{i=0}^{3} n_i$$

The normal $\hat{n}$ of the plane is computed as:

$$\hat{n} = (n_2 - n_0) \times (n_3 - n_1)$$

The distance of each node n_i to the plane is then:

$$dis(n_i) = |(n_i - avg) \cdot \hat{n}|$$

[2]The author refers to linear elements. This is not true for quadratic elements as explained in subsection 1.4.

If only the sum of the distances is used to measure the WF it would not consider the scale of the face. For this reason the WF of the face is computed as:

$$WF = \frac{1}{\sqrt{area}} \sum_{i=0}^{3} dis(n_i)$$

where $area$ corresponds to the area of the face formed by the nodes projected into the average plane. By doing this, whatever the distance of a node to the plane is, it will not produce the same result for a small and large face.

A coplanar face as a $WF = 0$. As the value of the WF increases the face decreases its quality. The WF cannot be negative as the absolute value of the distances is used and the area is always a positive number.

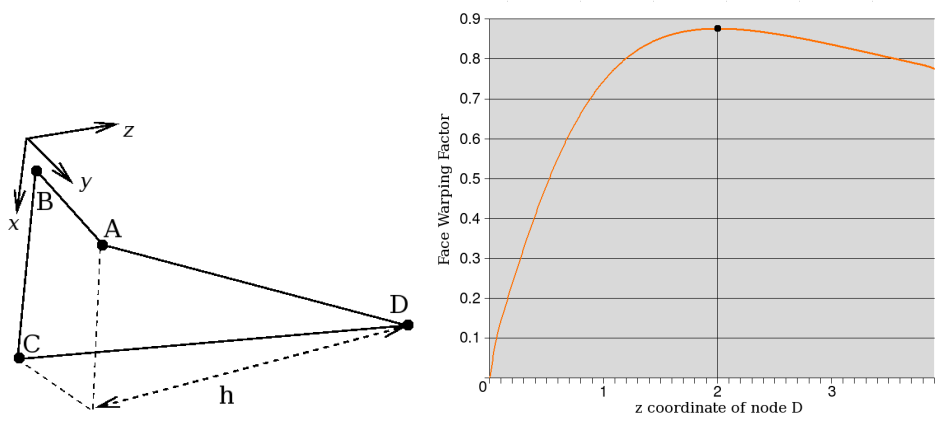

Figure 3.5: The warping factor behavior as one face node increases its z coordinate.

Figure 3.5 shows an example of how the use of the face area affects the WF. In the example a square face of length side $= 1$ was defined over xy plane. The coordinate z of node D was incrementally increased to determine the behavior of the WF function. The result can be seen at the right panel of figure 3.5 where the maximum value of the WF (the highest degradation of face) was 0.877383 at $D_z = 2$. As D_z continues to increase, the WF of the face decreases due to the area of the face that makes less relevant the displacement of node D (in a large scale the nodes tend to be coplanar).

The WF can be bigger than 1 when several nodes are displaced. For instance consider the case of a quadrilateral planar face $ABCD$ of length side $= 2$ and centered at the origin. If nodes A and C are moved to $y = 5.5$ and nodes B and D to $y = -5.5$ then the WF of the face is 9 as figure 3.6 shows. Note that in this case, the WF will continue to increase its value indefinitely as the nodes move far away from the average plane.

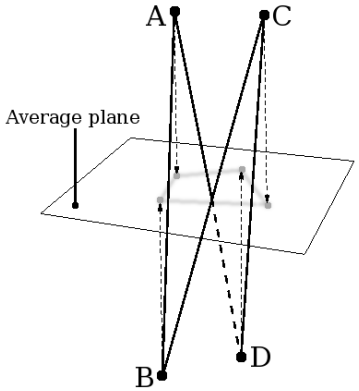

Figure 3.6: The warping factor can continue to increase when more than one node is displaced.

3.2.3 First approach: Jacobian gradient

A first approach to produce valid and quality meshes was implemented following the work of Luboz *et al* in [44]. This approach detects the invalid nodes by computing for each element the det**J** value. The direction of displacement for invalid node i is then computed as:

$$\nabla(\mathbf{detJ}_i) = \begin{bmatrix} \frac{\partial \mathbf{detJ}_i}{\partial x_i} \\ \frac{\partial \mathbf{detJ}_i}{\partial y_i} \\ \frac{\partial \mathbf{detJ}_i}{\partial z_i} \end{bmatrix}$$

The resulting vector is a direction dir_e computed regarding all the locations of the element nodes (as det$\mathbf{J}_i$ is computed involving all the element nodes). Then dir_e is normalized and the displacement distance dis_e to move the invalid node i is computed using an average of the element edge lengths.

The process is repeated iteratively for all the elements in the mesh, therefore if a given node is invalid regarding more than one element e, the final direction and displacement distance for i is computed as an average of dir_e and dis_e respectively.

$$displacement_i = \frac{1}{B} \sum_{e=1}^{m} \alpha_e \cdot dir_e \cdot dis_e$$

where m is the total amount of elements that share node i, B is the number of elements invalid at i and $\alpha_e = 1$ if the det$\mathbf{J}_{i,e} \leq 0$ or $\alpha_e = 0$ if det$\mathbf{J}_{i,e} > 0$.

All the above work was introduced by Luboz *et al* in [44]. Unfortunately this work didn't consider quality measures like the constraint of $Jac^{ratio} < 30$ or the WF.

Our developed strategy considers all the above process plus a quality improvement where a $Jac^{ratio} \in [1, 30[$ is searched for each element. The quality process follows the same principles as the validity process: detection of bad quality nodes and improvement using the $\nabla \det \mathbf{J}$.

This strategy works fine with simple cases. However some drawbacks were identified:

- This strategy is very sensible to the dis_e employed for both validity and quality process. If the value of dis_e is small, the process might need several iterations to achieve the desired results. If dis_e is large, it might cause neighbor elements invalidity or quality degradation. Even when all the elements attached to the node that has been repaired remain with good qualities, the displacement might be important. This is not good for the surface node after registration since these nodes must remain as close as possible to the surface of the target domain to represent.

- The developed application detects all the "bad" nodes and then the validity and quality process improve the Jac^{ratio} one by one. This approach presents problems with regions involving several "bad" nodes.

- It is not easy to include the warping factor WF parameter. If WF reparation process runs after the Jacobian quality process, it might cause another degradation over the Jacobian (quality and validity).

In order to understand the problems with systems of "bad" nodes, consider a perfect grid made of 9 well connected hexahedra as figure 3.7a shows. The nodes are then artificially moved to the positions showed at b and c of the same figure. Our first implemented strategy would search a local solution for node 1 first, without taking into account the nodes 2 and 3. The solution for node 1 is $1'$ where the Jac^{ratio} of this node is < 30 for all the elements that share it (figure 3.7d).

The drawback of this solution is that it involves an important displacement of node 1 and the algorithm, that searches with small displacements the optimal location for the node, will be very slow to achieve this state.

The optimal solution for this system should consider the improvement of the elements by a displacement of all invalid and bad quality nodes and not based on local improvement for each node. The next subsection explains how this can be achieved (including the WF).

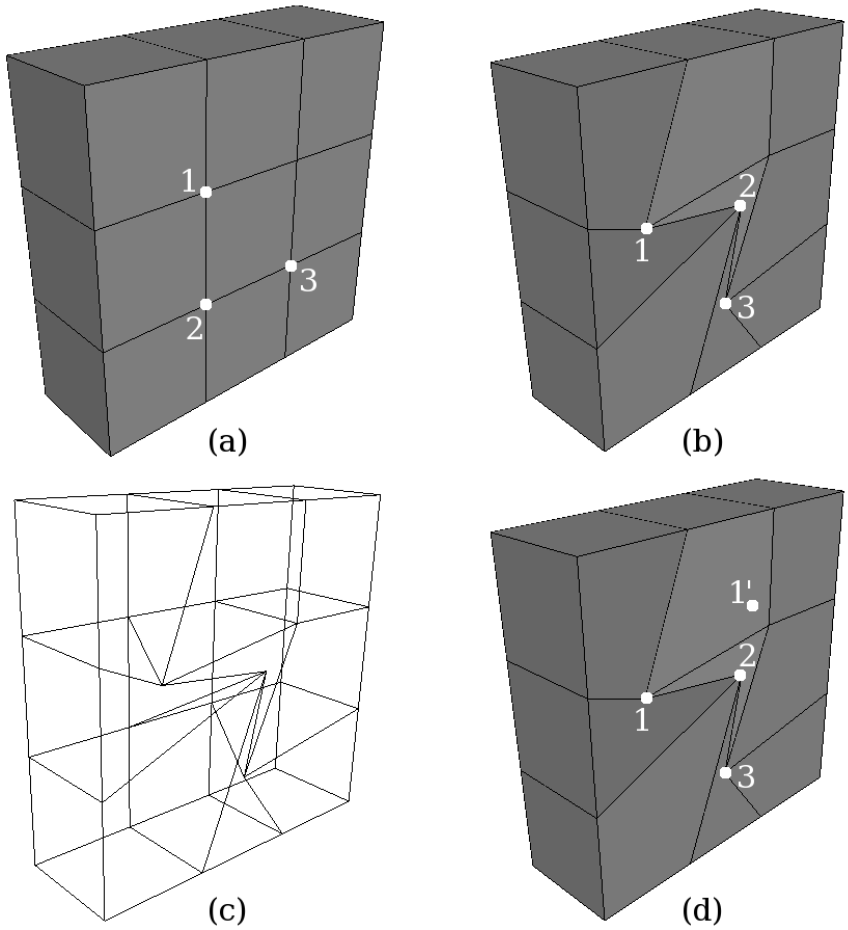

Figure 3.7: (a) a perfect grid 3×3 of hexahedra, (b) and (c) a deformed state of the same grid and (d) a possible solution for node $1 \rightarrow 1'$.

3.2.4 Second approach: iterative Jacobian and warping reparation

The big difference between the previous and the current approach is merely conceptual. The Jacobian determinant (det**J**) and the ratio of the Jacobian determinants (Jac^{ratio}) can be seen as functions that depend on the location of the nodes. The same happens with the warping factor WF.

A first step is to achieve a valid mesh (det**J** > 0). To achieve this task all the invalid elements, in particular the nodes that make the element invalid, are detected. For each node i a function can be established as:

$$F = \mathbf{detJ}_i - WF_i = \sum_{e=1}^{m}\sum_{j=1}^{n} \mathbf{detJ}_{j,e} - \sum_{f=1}^{q} WF_i$$

where $e = 1 \ldots m$ are the elements, $j = 1 \ldots n$ are the nodes of element e (where one of the nodes is i) and $f = 1 \ldots q$ are the faces that share node i.

Note that in this approach, all the elements that share node i are used to determine the new location of i. In the previous approach, only the elements that where invalid at node i where used to compute ∇ det$\mathbf{J}_i$. The current approach considers all the elements since a reallocation of invalid nodes might cause "good" neighbor elements to become invalid. In this manner, the new location of i considers the non degradation of valid neighbors.

Algorithm 1 Grouping connected invalid nodes

```
Require: mesh
  for each element e in the mesh do
    for each node i of e do
      if detJ_i ≤ 0 then
        let elements_i be the elements associated to i.
        boolean found = false.
        for each region r in regions do
          if r has an element of elements_i then
            join elements_i and i to r
            found = true
            break
          end if
        end for
        if !found then
          create new region r_i with elements_i and i
          attach r_i to regions
        end if
      end if
    end for
  end for
```

The det$\mathbf{J}_i$ is negative at least for one element and the WF might be > 0 for some faces. The idea is to search a new position for node i where F increases its value, i.e. the goal is to maximize F by increasing the negative det$\mathbf{J}_i$ value and the co-planarity factor of faces attached to i.

The $\nabla_i F$ gives the direction of displacement and once again the distance of displacement is an average of the edge lengths attached to i. Each time a new

displacement is done over i det$\mathbf{J}_i$ is checked for all the elements e. If $\forall e, \mathbf{detJ}_i > 0$ the validity routine stops.

Algorithm 1 shows how the invalid nodes are detected and grouped following element connectivity relationships. A $region$ is a list of invalid nodes and all the elements associated to them. Grouping enables a subsystem solution instead of local solution as the one obtained in the first approach (see figure 3.7d). Moreover, the iterative approach finds a solution for each region locally, as the invalid nodes outside the region are not taking into account. The main problem with the iterative approach is that it takes time to find a solution. The more the nodes to reallocate, the more the time to produce the solution. Therefore the idea is to isolate the non-connected regions to speed-up the process. The local solution is optimal regarding all the invalid nodes on the region.

Algorithm 2 Reparation of invalid nodes

Require: mesh
  group invalid nodes in $regions$
  for each $r \in regions$ **do**
    boolean $allpositive$
    repeat
      $allpositive = true$
      for each node $i \in r$ **do**
        Compute the direction dir_i
        Compute the distance dis_i
        Compute new position for node $i : i' = i + dir_i \cdot dis_i$
      end for
      for each node $i \in r$ **do**
        move $i \rightarrow i'$
      end for
      for each element $\in r$ **do**
        for each node $i \in$ the element **do**
          if det$\mathbf{J}_i < 0$ **then**
            $allpositive = false$
            break (continue with next "repeat-until" iteration)
          end if
        end for
      end for
    until $allpositive$
  end for

Algorithm 2 shows how a valid state of the mesh is achieved. The direction of displacement is computed for each coordinate of node i at a time, regarding

all the elements attached to i. First, Q_0 is computed as $F(x_i, y_i, z_i)$, then $Q_1 = F(x_{i+\epsilon}, y_i, z_i)$. If $Q_1 > Q_0$ then the direction of displacement dir_i of coordinate x of node i is $\nabla_x F_i = (Q_1 - Q_0)/\epsilon$. If $Q_1 \leq Q_0$ then dir_i is $-\nabla_x F_i$. This operation is repeated for each coordinate of each node in the $region$. The distance of displacement dis_i is computed as the 10% of the average edge length (between the edges associated to node i).

Once the reparation of invalid elements is achieved, another process repairs the elements of poor quality. The algorithm is the same but function F is different and instead of searching a positive jacobian, the condition to stop is that every element must have a $Jac^{ratio} < 30$. Function F is constructed as fallows:

$$F = -Jac_i^{ratio} - WF_i = -\sum_{e=1}^{m}\sum_{j=1}^{n} Jac_{j,e}^{ratio} - \sum_{f=1}^{q} WF_i$$

The Jac^{ratio} is employed in this function as the goal is to obtain an optimal state for the elements that presented bad quality. The Jac^{ratio} function presents a discontinuity when passing from negative to positive values (see the right plot of figure 3.3). Because of this discontinuity the global process was divided in two steps: first, achieving a valid mesh and second, improving the quality of it. If the Jac^{ratio} were used to produce a valid mesh, the iterative method should search for the most negative value of that function, in consequence it would never achieve validity. This is why the det**J** was chosen for the valid mesh process. When the quality process starts, all the nodes have a det**J**> 0, thus a $Jac^{ratio} > 1$ and the quality improvement process can proceed.

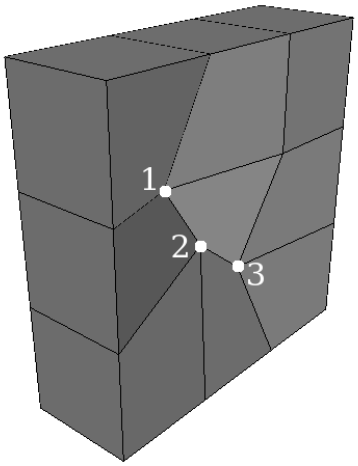

Figure 3.8: The solution for the invalid system described in figure 3.7b.

The initial deformed system presented in figure 3.7b, had 3 invalid elements (the worst had $Jac^{ratio} = -3.06$). The solution given by our algorithm is pre-

sented in figure 3.8. The solution has no invalid elements and the worst element has a $Jac^{ratio} = 10.4037$. Note that this value is better than the searched one ($Jac^{ratio} \leq 30$). As the implemented solution is iterative, the first reallocation of the nodes in which the Jac^{ratio} value that satisfies the quality constraint is chosen as the solution. Remind that the Jac^{ratio} function is exponential, therefore a small node displacement causes an important difference in the value of the function. Regarding the WF, the initial system had a worst $WF = 1.88563$. The final solution presents a worst $WF = 0.677209$ (where a $WF = 0$ is the best value).

3.2.5 Remarks over the implemented solution

The implemented solution presented in the second approach can go much further with the element quality improvement. The maximal quality threshold can be easily changed in order to demand a better quality over the elements in the mesh. Figure 3.9 shows the resulting mesh when the demanded Jac^{ratio} must be inferior to 2 (left) and 1.5 (right). Note how these meshes are much more closer to represent the initial 3×3 hexahedra grid than the resulting mesh of figure 3.8 (where $Jac^{ratio} < 30$ was demanded).

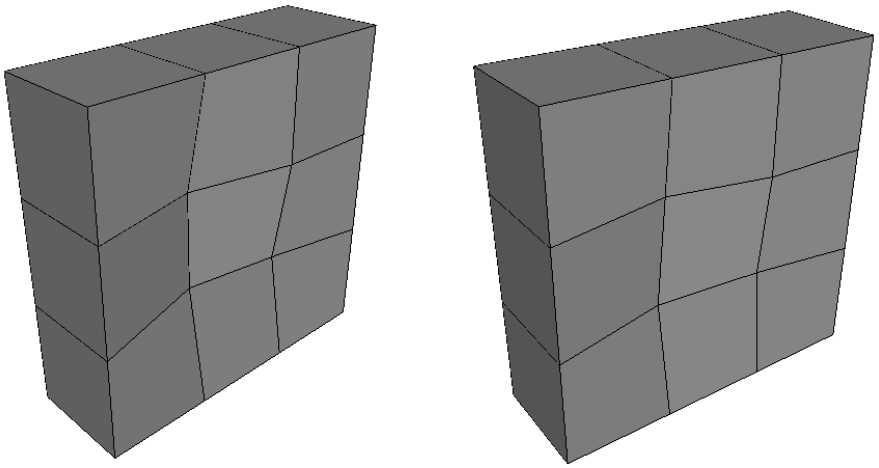

Figure 3.9: Improving elements by changing the quality threshold . Left: elements must have a $Jac^{ratio} < 2$. Right: elements must have a $Jac^{ratio} < 1.5$. For both results the input was the mesh presented in figure 3.7*b*.

The mesh at the left panel of figure 3.9 has a worst Jac^{ratio} value of 1.557 and a worst WF value of 0.24. The mesh of the right panel presents a worst Jac^{ratio} value of 1.376 and a worst WF value of 0.193.

It is important to note that our developed technique searches the minimal node displacement to achieve element validity and quality. The most important qual-

ity measure in our approach is the Jac^{ratio} and its threshold (30) is given by ANSYS®.

The second remark is relative to the differences between the work of Luboz in [44], the first and the second implemented approaches. While Luboz proposed an analytical solution, the both implemented solutions are iterative numerical algorithms. The second important difference is that Luboz didn't consider quality improvements (like the $Jac^{ratio} < 30$ and the WF). Finally, if element e is invalid at node i, the three algorithms propose different manners to find the direction of displacement of i:

- The first implemented strategy searches the direction only in function of element e.
- The algorithm by Luboz first searches all the elements that are invalid at node i, then the direction is computed regarding all those elements.
- The second implemented approach computes a direction of displacement regarding all the elements attached to i.

Our final implemented solution should be preferred over the other presented solutions because it considers quality measures and finds a solution for a given node i regarding all the elements attached to it. Note that in the algorithm of Luboz an element e^{good} attached to i might have $Jac^{ratio} < 30$, unfortunately the reallocation of i (in order to correct the invalid and poor quality elements associated to it) cannot assure that e^{good} will remain with $Jac^{ratio} < 30$. Noticing this drawback, our algorithm searches iteratively the minimal displacement of invalid and "bad quality" nodes in which the new position of the node increases the quality of **all the elements** attached to it.

3.3 Generation of FE meshes focused on a particular region

As mentioned in previous chapters, in a simulation the key problem is to solve, via an approximation method, a system of Partial Differential Equations (PDE). The approximation method in this thesis is the Finite Element Method (FEM). The time to produce an approximated solution of this system of PDEs directly depends on the quantity of mesh nodes. However it is possible to compute fast solutions with dense meshes in an optimized FEM as proposed by Cotin [19], Picinbono [54], Schwartz [58] or Nesme [51] between others.

As the mesh increases the level of detail (refined regions with more nodes and elements) the FEM can produce more "precise" results as the equations are evaluated in more points, allowing to show the state of the target organ in a more

detailed manner. In the medical field, the focus is sometimes given in a particular region. However, constraints are globally defined over the organ to simulate, therefore meshes that describe the entire organ are needed. At the same time, there is no need in achieving a high detailed mesh over the entire domain; only the regions of interest are important for the simulation. In the following subsections a detailed explanation on the developed application to produce 3D mixed-elements meshes with local region refinement is presented.

3.3.1 Application overview

An application to produce mixed-element meshes with local region refinement of an input domain Ω was developed. The global steps to produce a volume mesh from the input surface mesh of the domain (Ω_s) go as follows:

- produce an initial octree mesh of Ω.
- achieve a 1-irregular state of the mesh.
- produce a congruent transition between refined and coarse regions.
- use a registration method and achieve surface representation of Ω.
- obtain the simulation result with the FEM.

What is new in this process? In normal cases, the octree technique splits the elements that intersect the domain's surface Ω_s until a certain criterion is achieved. In the case of this thesis, it is the user that defines where more detailed regions are needed. This is the main difference.

The Region of Interest (RoI) becomes the most important concept in this approach. The RoI is defined by a polyhedron that intersects Ω_s. The result of this intersection should be a positive volume. The octree technique is then modified to split only the octants (see subsection 2.2.2 for more details) that are in the intersection of the RoI and Ω.

Several RoIs can be defined. If the result of the intersection between the RoIs and Ω_s is null, a point, a segment or a face, the mesh will not be refined (i.e. the mesh will only be refined when the intersection between RoIs and Ω is another 3D volume). If one RoI intersects the entire domain Ω, then the final mesh will be equally refined all over Ω.

3.3.2 Application inputs

In the octree technique "the first mesh" is a cube that covers the entire domain Ω. In order to produce a final mesh, the technique splits in eight new cubes the elements that intersect the surface of Ω (Ω_s) until a certain condition is reached.

The developed technique is based on the octree technique and the inputs are the following:

- A surface mesh Ω_s that describes the domain Ω to mesh.
- A polyhedron that describes a region of Ω where elements will be more refined (the Region of Interest: RoI).
- An estimated number of nodes the mesh should have (the meshing technique stops the refinement when the number of nodes in the mesh is bigger than this maximum number of nodes).

Figure 3.10 left shows an arbitrary domain defined by Ω_s. The middle panel shows the same surface mesh with the RoI (polyhedron in green) and at the right the same Ω_s and RoI are shown plus the first state of the mesh that corresponds to the Bounding Box[3] (BBox) of Ω_s.

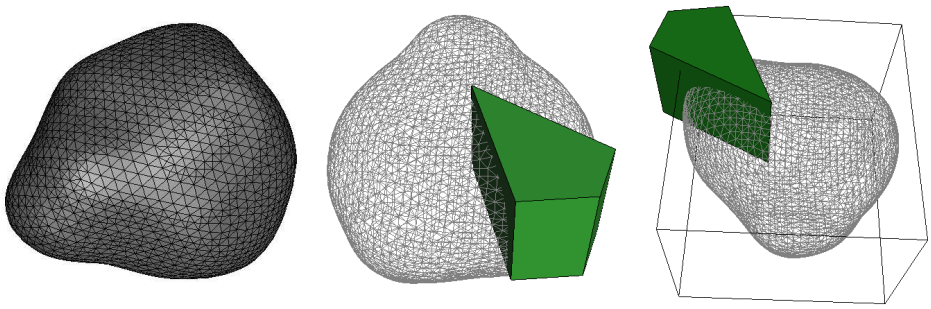

Figure 3.10: Left: an arbitrary domain defined by the input surface mesh Ω_s. Middle: the same Ω_s in transparent and the RoI in solid. Right: the same Ω_s, the RoI and the first level of Octree mesh.

The input surface mesh Ω_s (figure 3.10 left) represents the organ to mesh. Remind that this is a surface mesh and the output of our meshing technique is a volume mesh that has to be more refined in the RoI. It is important to mention that Ω_s should be a closed geometry: every edge should be shared by two and only two faces of Ω_s. The quality of the faces is not relevant as this input mesh is only used to determine if the elements of the final output mesh are inside, outside or intersect the input domain.

[3]From all the nodes in the input domain, the minimal and maximal coordinates per axis are searched. The BBox is an hexahedron constructed from points $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$

3.3.3 Splitting an element

The output mesh changes as the octree technique splits the elements. Let Ω_i^{out} be the output mesh at iteration i of the octree technique. Then Ω_0^{out} corresponds to the mesh defined by the BBox of Ω (figure 3.10 right). In order to produce the next stage of the mesh (Ω_1^{out}) it is necessary to split the BBox in eight new cubes as figure 3.11 shows.

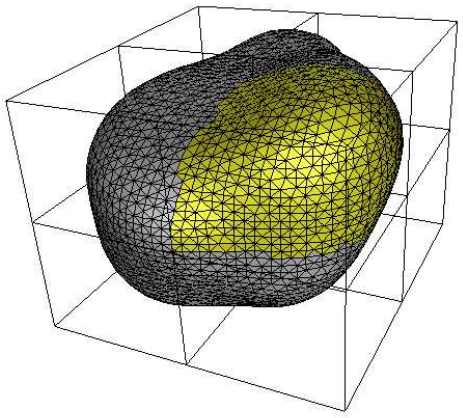

Figure 3.11: A subset of "intersection faces" is assigned to each element. The top-right-front octant has the yellow faces of Ω_s.

Each element resulting from the split process must be checked for intersections with Ω_s and the RoI. If an element of Ω_i^{out} doesn't intersect Ω_s, it can be completely inside or completely outside Ω. When an element is completely outside it is removed from Ω_i^{out}. In the same manner, elements might be inside, outside or intersect the RoI. Table 3.1 summarizes the possible combinations of intersection states and what to do in each case.

	inside Ω_s	intersects Ω_s	outside Ω_s
inside RoI	split	split	remove
intersects RoI	split	split	remove
outside RoI	no split	no split	remove

Table 3.1: Element states regarding the intersections with Ω_s and RoI.

Adding a new hexahedron in Ω_i^{out} is a fast operation as the nodes are efficiently inserted in the mesh (see appendix A). Then the most expensive task is to compute element intersection with Ω_s and the RoI. If for each element, intersections are computed with every face of Ω_s this operation would be extremely expensive with high detailed inputs (meshes with a high number of faces). Therefore each

element keeps a reference to the subset of Ω_s (Ω_s^{el}) that it intersects (see figure 3.11). When the element splits, each son checks intersections only with faces in Ω_s^{el} of the parent element.

A procedure to detect face-edge intersection was implemented. Let $\hat{n}$ be the normal of the plane in which the face is defined and v a point in that plane (a node of the face for example). If the edge is formed by points P_1 and P_2, then the distance of the points to the plane can be defined as $h_1 = \hat{n} \cdot (P_1 - v)$ and $h_2 = \hat{n} \cdot (P_2 - v)$. If h_1 and h_2 have the same sign, then no intersection is produced. Otherwise the intersection of the edge and the plane is the point P that is obtained as:

$$P = P_1 + \frac{h_1}{h_1 - h_2} \times (P_2 - P_1)$$

Now to detect if point P is "inside" the face it is necessary to check the sign of $P_i^{cross} = (P - V_i) \times (V_{i+1} - V_i)$, where V_i is a vertex of the face ($\forall\, V_i$ of the face). If the sign of P_i^{cross} changes from one vertex to another, then P is outside the face, thus no intersection is produced between the edge and the face. Otherwise the intersection is the point P.

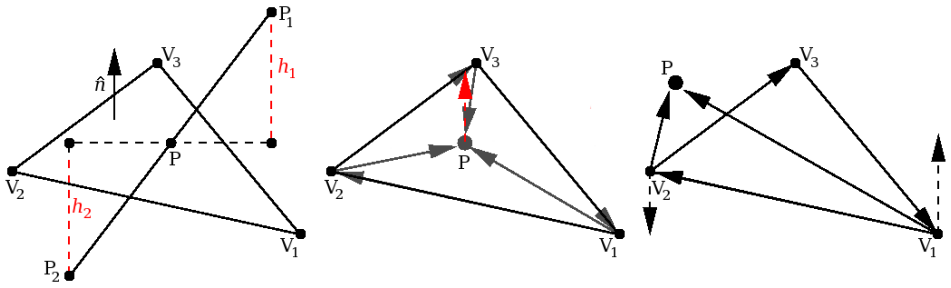

Figure 3.12: Left: performing edge-plane intersection. Middle: case of a point inside the face. Right: case of a point outside the face.

Figure 3.12 shows how the edge-plane intersection point P is found (left); in the middle panel, a case where all P_i^{cross} have the same orientation (red vector); at the right panel, a point outside is detected as $(P - V_1) \times (V_2 - V_1)$ has not the same orientation as $(P - V_2) \times (V_3 - V_2)$; the resulting vectors of the cross products are in V_1 and V_2 as dotted lines.

With the above method it is possible to check if an element e intersects a face f (of Ω_s or the RoI). The intersection is produced in any of the following cases:

- An edge of e intersects f.
- An edge of f intersects a face of e.

- The face f is contained in e.

When no intersection is produced between the element and the faces, by checking only one point of e it is possible to determine if e is completely inside or outside Ω_s or the RoI. Both geometries are then managed in different ways.

The RoI is defined as a convex polyhedron. The normal of each face of the RoI polyhedron is set to the interior of it. The distances between the polyhedron faces and the point to be checked are computed. If any of these distances is negative, the point is outside the RoI. In any other case the point, thus the element, is completely inside the RoI.

Regarding Ω_s the situation is not that simple as Ω is not necessarily a convex domain. Therefore testing the "side" of the point (the sign of the distance) with every face of Ω_s isn't a reliable test to determine if the element is inside or outside Ω.

The implemented test needs a point outside Ω. Let P_{bbox} be the vertex of the Bounding Box (BBox) of Ω with the highest values in each coordinate $(x_{max}, y_{max}, z_{max})$. An addition of ϵ in every coordinate to this point will certainly result in a point that lies outside Ω_s: $P_{out} = P_{bbox} + \epsilon$. A "virtual edge" can now be created between P_{out} and the point P of the element to be checked. The intersection of this edge and every face of Ω_s is then computed. If an odd number of intersections is produced then P is inside otherwise the point (thus the element) is outside.

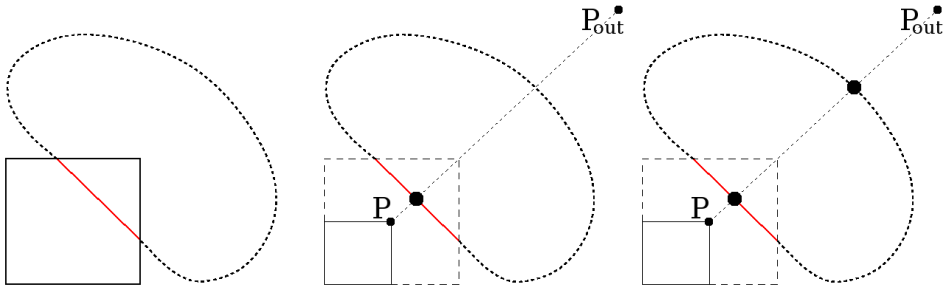

Figure 3.13: The different results of outside/inside element detection using a subset or the entire set of Ω_s faces.

It is necessary to compute the intersection between the "virtual edge" and all the faces of Ω_s because it is possible to find some configurations where testing only with local element intersection faces would fail. An example of this is shown in figure 3.13. The left panel shows one element e to be split, where the straight line represent the subset of faces of Ω_s that e intersects (Ω_s^e). The middle panel shows the only intersection produced between the "virtual" edge and Ω_s^e (this implies that the element is inside). The right panel shows how two intersections are produced when all the faces of Ω_s are checked (thus the element is outside).

3.3.4 Achieving the desired level of refinement

As mentioned in subsection 3.3.1 the desired level of refinement is specified by the user. In order to produce a mesh, the user must specify an estimated number of nodes for the mesh ($nodes_{ref}$). In each iteration of the octree (split the cubes that intersects the RoI in eight new cubes) the current number of nodes ($nodes_{current}$) is checked against the input given by the user. If $nodes_{ref} < nodes_{current}$ the splitting process is finished.

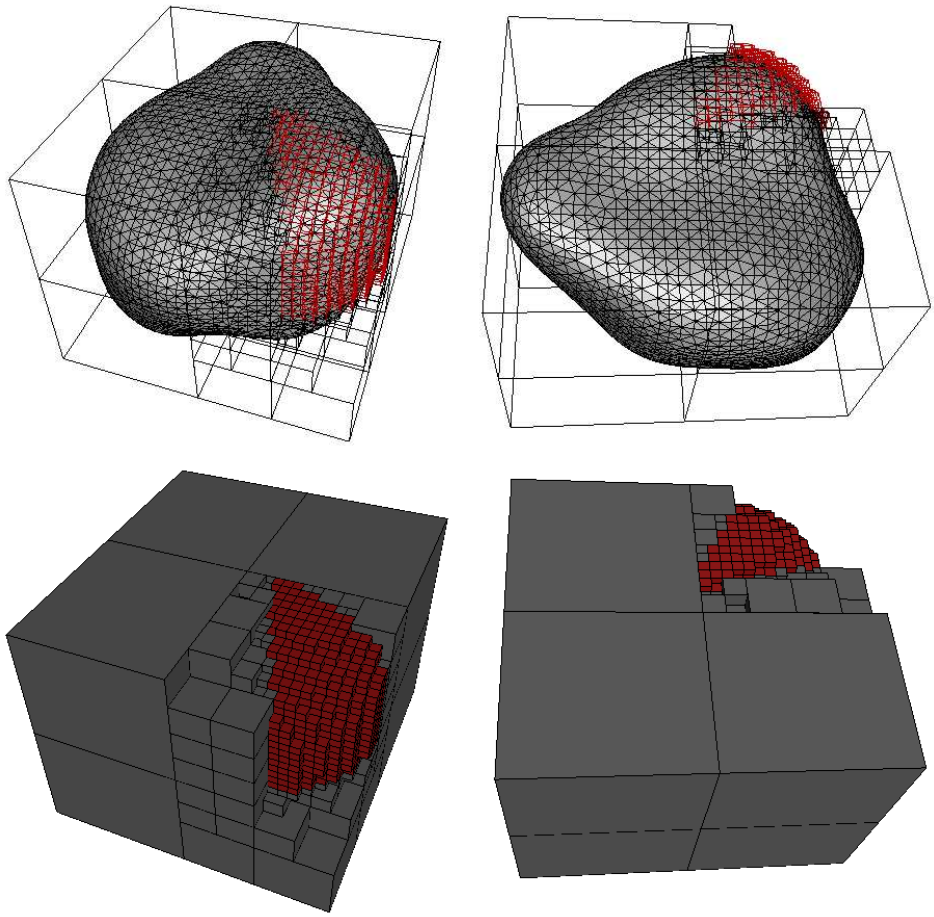

Figure 3.14: Refinement of the mesh in the RoI. At the top, the images show the input domain Ω_s in solid and the output mesh in transparent. Red elements intersect the RoI. At the bottom only the output mesh is presented in solid.

To summarize the process, the user specifies the domain to mesh with a surface mesh Ω_s, the RoI with a simple polyhedron and the estimated number of nodes $nodes_{ref}$. With this information, the process splits the hexahedra that intersect the RoI in 8 new cubes and removes the ones that lie outside Ω_s. Therefore an output

to Ω_s with the RoI presented in subsection 3.3.2 would be figure 3.14, where the mesh counts with 2656 nodes and 1955 hexahedra. It is important to mention that 1477 hexahedra intersect the RoI, which represents 75% of the total number of hexahedra. This mesh took 16 seconds to be produced and such a time can be explained by the number of intersection tests to be performed. Note that Ω_s counts with 6434 triangles, therefore an hexahedron that doesn't intersect Ω_s implies the computation of 6434 intersection tests.

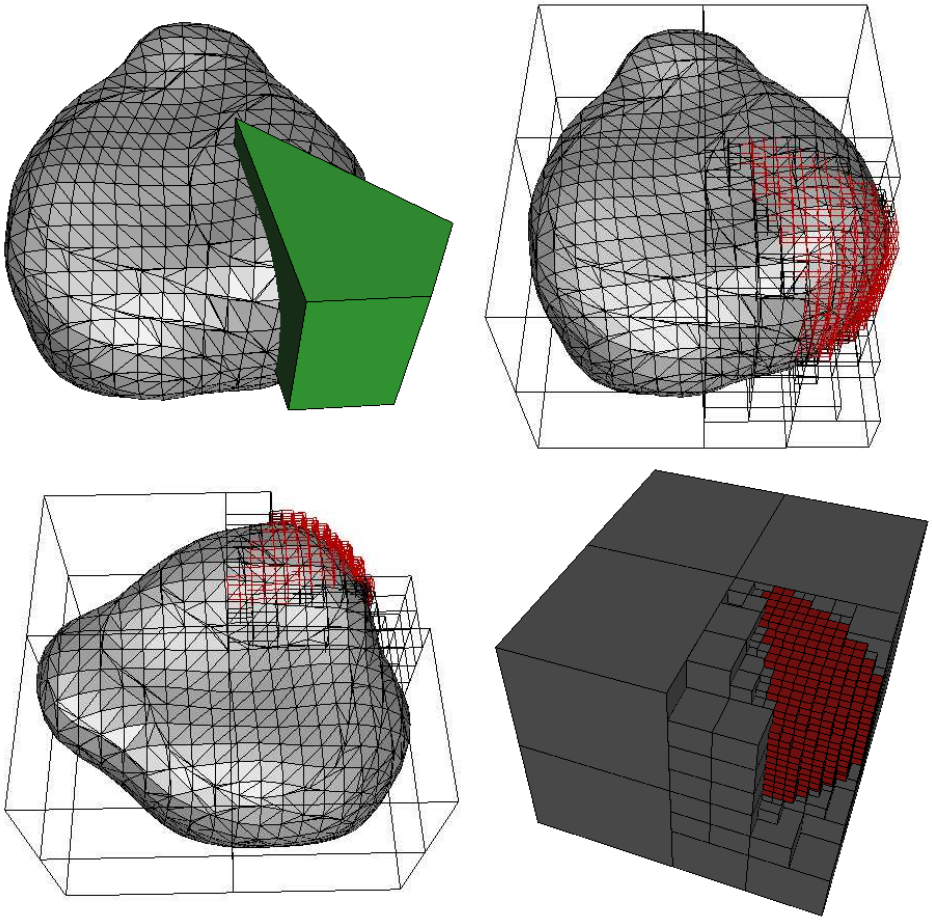

Figure 3.15: The same as figure 3.14 but with a coarse representation of the input domain.

The reference number of nodes given by the user in this case was 2000. The final quantity of nodes in the output mesh is given by the level of refinement of the octree, therefore the amount of nodes in the output mesh depends directly on the amount of elements that intersect the RoI at the time of the splitting process. For instance in the output mesh of figure 3.14, five octree subdivisions were per-

formed. In the case of only 4 octree subdivisions, using the same Ω_s and RoI, the output mesh would count with only 702 nodes and 396 elements (with 63% hexahedra population inside the RoI). Note that a reference number of nodes between 703 and 2656 would produce the same output mesh (the one presented in figure 3.14).

Another input mesh of the same domain Ω is shown in figure 3.15. This new Ω'_s counts with 1214 nodes and 2424 triangle faces. The output mesh for this new case is produced in only 6 seconds, however the number of nodes is 2641 and the number of hexahedra is 1942. This little difference can be explained as Ω'_s looses some precision of the domain representation in comparison with the previous input mesh Ω_s, therefore some elements that intersected the domain before, now don't.

The important lesson to keep is that regarding the time to produce the mesh, the developed application is faster when coarse surface meshes are employed. In other words, the developed technique is sensitive to the number of faces of the input surface mesh.

3.3.5 Managing transitions

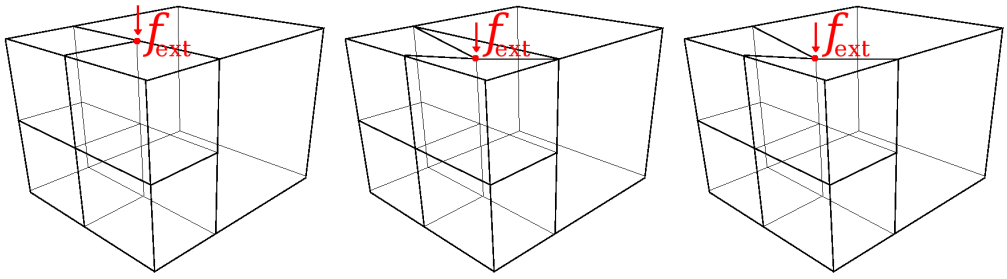

Figure 3.16: Elements with nodes inserted in their edges cause problems to the FEM as Ω is not continuously represented. An external force might not be well represented in the entire domain as neighbor elements don't congruently connect.

The current mesh has refined and coarse regions. In some cases the elements count with several nodes inserted in their edges. This is not allowed in a FE simulation as the discretization is not continuous through Ω. Figure 3.16 shows an example where an external force is applied over a node that present "bad connectivity" between neighbor elements. In this case a smooth transition between the refined and the coarse elements is not achieved (middle panel). The right panel shows how the big hexahedron should deform regarding the external force. However this is impossible as the node where the force is applied doesn't belong to this hexahedron. If this node was associated to the "big hexahedron", the el-

ement would no longer be an hexahedron and the FEM should be modified to consider such an element (hexahedron with 1 node in the middle of one of its edges). Moreover, the FEM should be modified to consider all the cases where hexahedra presents nodes in their edges and faces.

Instead of modifying the FEM to consider complex configurations of the hexahedra, the proposed solution allows the use of other type of elements, thus a mixed-element mesh is produced. By the use of several templates, it is possible to determine the different configurations of the hexahedron with nodes inserted on its edges or face center points. In function of the configuration, tetrahedra, pyramids and prisms (wedges) are inserted without adding new nodes (only new edges).

The templates consider permutations. For instance, a template with only one bisected edge[4] is described on figure 3.17. As twelve configurations are possible, the permutation allows to go from the given solution of the template to the actual state of the hexahedron.

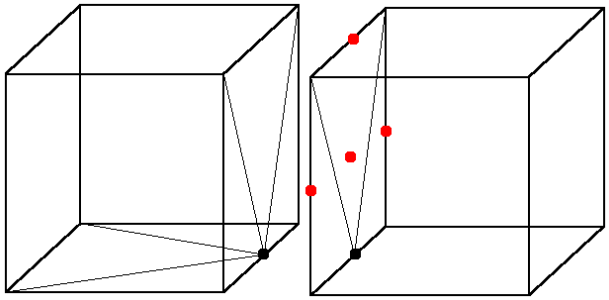

Figure 3.17: The hexahedron at the left only has one bisected edge. Its neighbor has the same information over the shared face. Note that red nodes cannot be possible as the left hexahedron would also consider this information to produce its subdivision.

As a result of the subdivision of the hexahedra new edges that describe the new elements are inserted inside the element and on its faces. Note that neighboring elements are always congruently split as the templates have only one manner to split the element without inserting new nodes. This can be seen in figure 3.17 where the red nodes of the right hexahedron describe an impossible configuration (as this is a shared face, the configuration is the same for both elements). In other words regarding the shared face, the right hexahedron could have any other node inserted in their edges or faces but the ones in red.

Once the new edges are inserted, the hexahedron is replaced with the new mixed-elements. For instance, the example shown in figure 3.18 at the top left

[4] An edge with one middle node.

panel has two bisected edges and only the edges over the faces are shown. At the top right panel, all the edges are inserted and it can be seen as no new node is inserted in the element. Finally at the two bottom panels the final elements are shown.

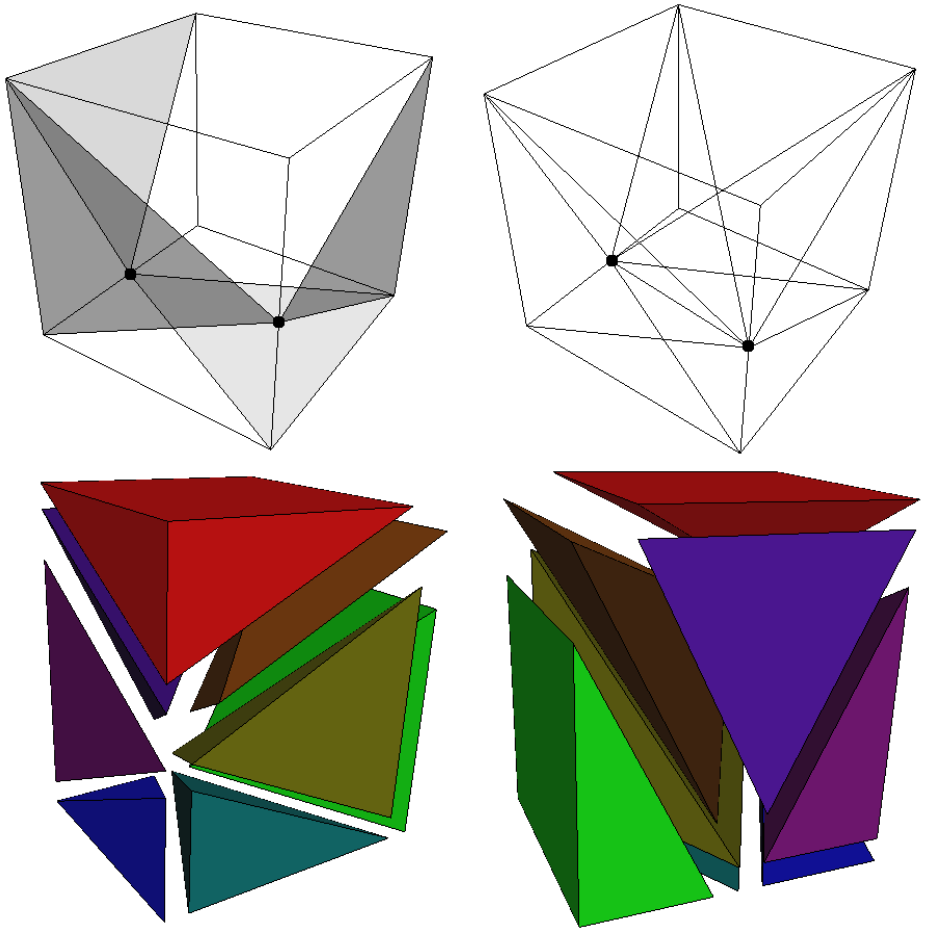

Figure 3.18: Splitting an hexahedron with two bisected edges into mixed-elements. Top left: the insertion of the edges over the faces. Top right: the insertion of internal edges. Bottom left and right: two views of the six tetrahedrons and two pyramids (red and green) in which the hexahedron was split.

There exist 26 implemented templates going from one to nine nodes inserted in edges or face center. Those templates correspond to the most common cases. If for a given hexahedron configuration, the template doesn't exist, then the procedure is to split the hexahedron in eight new cubes (as in the octree refinement). In this manner, the process assures a valid tessellation of Ω (without elements with nodes inserted on their edges or faces).

The property of having all the edges bisected at most once is called one-irregular. The implemented templates work only on one-irregular meshes; therefore applying the templates directly over the meshes of figures 3.14 and 3.15 wouldn't work properly.

The node management explained in appendix A allows to easily determine if an hexahedron achieves the one-irregular property. As a consequence all the hexahedra that don't achieve this property are split in eight hexahedra until the entire mesh is one-irregular. After this step, the templates can be applied and a congruent mesh of Ω is produced.

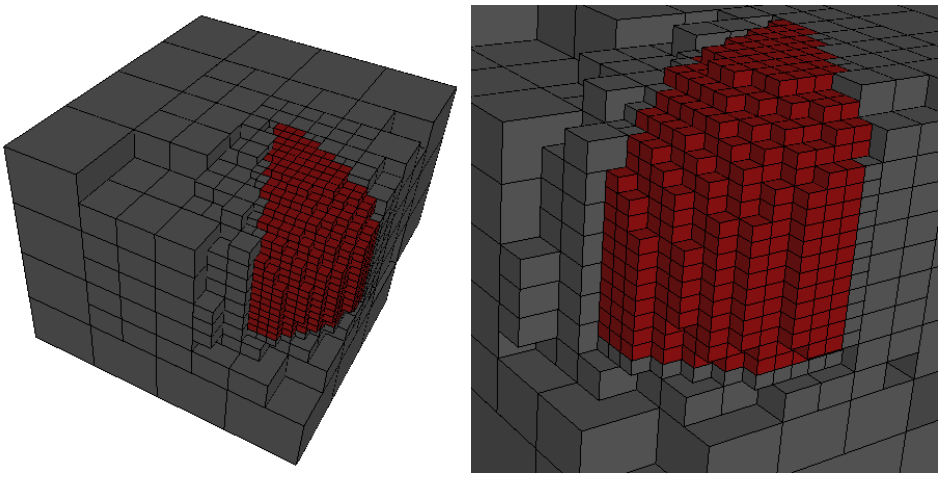

Figure 3.19: One-irregular mesh of Ω showed in figure 3.14.

Figure 3.19 shows the one-irregular version of the example shown in figure 3.14. The one-irregular mesh has 3111 nodes, this means that 455 nodes have been added by the process (this corresponds to 17.1% of the initial nodes). The number of hexahedra is 2302 (an increase of 17.7% regarding the initial number of elements).

Figure 3.20 shows the result once the templates are applied over the one-irregular mesh. This mesh has 3046 nodes and 4333 elements. The elements are: 1978 hexahedra, 97 prisms, 1446 pyramids and 812 tetrahedrons.

There are 1477 hexahedra (34% of the elements) and 2035 nodes (66.8% of the total nodes) in the RoI. Note how the total number of nodes decreases in the mixed-element mesh regarding the one-irregular mesh($3111 \rightarrow 3046$). This is due to the fact that some hexahedra had some portion outside Ω; therefore as they are split into simpler geometries, some of these new elements are completely outside Ω and in consequence they are removed from the mesh.

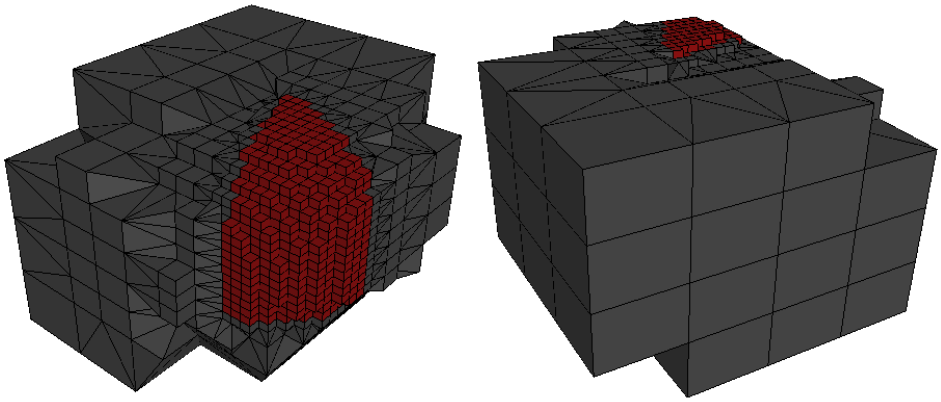

Figure 3.20: The templates are applied to the one-irregular mesh in order to produce a mixed-element mesh without node inserted in the middle of edges or faces.

3.3.6 Quality measures

As mentioned in previous chapters, an extraordinary number of quality measures has been proposed to measure the quality of the elements, ranging from bounds on solid angles to more complex geometric ratios (See [39, 61, 66] for more details). As the developed technique produces a mixed-element mesh, a quality criteria must be presented for each type of element.

The tetrahedra are measured with an Aspect Ratio Coefficient (ARC) that includes the volume of the element to avoid sliver elements. This measure is obtained as follows (see [24] for more details):

$$\text{ARC} = \frac{(\frac{1}{6}\sum_{i=1}^{6}(l_i^2))^{3/2}}{8.47867V^{el}}$$

Where $l_i (i = 1, \ldots, 6)$ are its edge lengths and V^{el} is the volume of the tetrahedra. The value 1 corresponds in this case to an equilateral tetrahedra. In general an ARC = 1 corresponds to an ideal element and as the ARC value increases the element is said to be more distorted.

For the other elements, several "lengths" are computed. The ARC is then the ratio between the longest and the shortest of all the computed lengths. In the case of the hexahedron the lengths between "opposite face center points" are obtained (see figure 3.21 left panel). In the case of the pyramid three lengths are also used: The two firsts are constructed using the middle points of opposite edges in the base rectangular face. The third length is the one that represents the height of the pyramid (see figure 3.21 middle panel). For the prism, any of the rectangular faces is selected. Two lengths are obtained as in the case of the pyramid rectangular

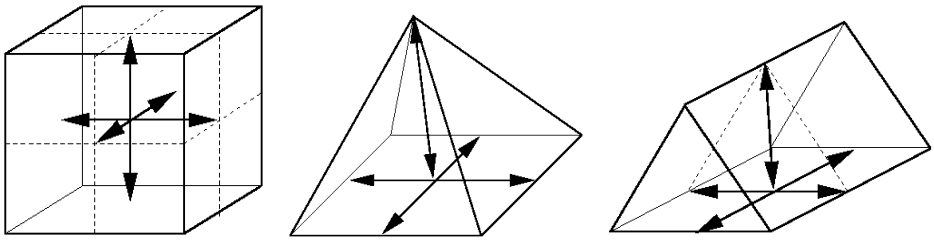

Figure 3.21: The segments that are used to determine the Aspect Ratio Coefficient (ARC) for an hexahedron, pyramid and prism. The ARC is the ratio between the longest and shortest of the three segments.

square face. The third length is obtained using the height of the triangle face that corresponds to the average of the two original triangle faces of the prism (see figure 3.21 right panel).

3.3.7 Achieving surface representation

At this stage of the process, several elements have some portions of their volume outside Ω. In order to achieve a better representation of the surface, the points that reside outside Ω must be projected into Ω_s.

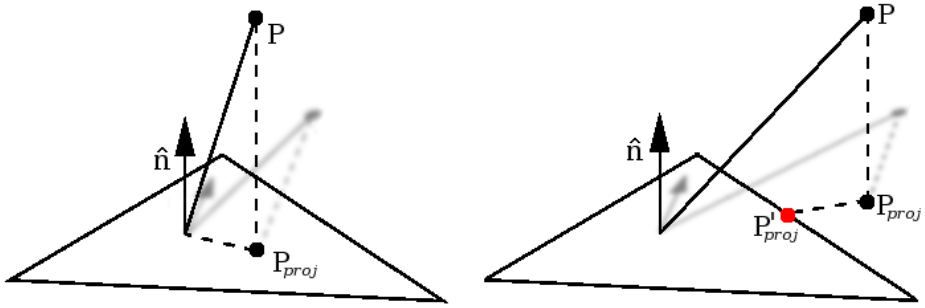

Figure 3.22: The closest projection of a node to a face. Left: the projection of the node is inside the face. Right: the projection is outside the face and the closest point of the face to this node is the final result (P'_{proj}).

The nodes of the elements that intersect Ω_s are checked to label the ones that reside outside the domain. For each one of these nodes, the closest position is computed regarding the "intersection faces" of the element. Using the intersection method explained in subsection 3.3.3, the projection of the node into the plane where the face resides is obtained. If the projected point P_{proj} is inside the face, then this point corresponds to the better projection (regarding only this face). If

P_{proj} is outside the face then the closest projection of the node into the face corresponds to the closest point of the face to P_{proj} as shown in figure 3.22.

The element intersects one or more faces. For each "outside" node the projection is computed with all the faces the element intersects and the closest one is chosen. If the outside nodes are moved directly to the projected position it might cause edge crossing and poor quality elements (like slivers) as only the elements that intersect the surface are "pushed" to achieve surface representation. In order to avoid this, another strategy that uses the FEM is proposed where not only the elements that intersect Ω_s are deformed but all, since a mechanical "pressure" exerted on the outside nodes is going to pressurize and therefore reallocate the internal nodes.

The key idea of such surface representation algorithm is to use a mechanical simulation to constrain the deformation between the octree mixed-element mesh (the source) and the input surface mesh (the target). The outside nodes of the source mesh are driven (following the closest projection point) with a step-by-step displacement towards the destination mesh, leaving a mechanical model perform the inner nodes relaxation throughout the deformation. It is also important to mention that it was chosen to implement a compressible material behavior (Poisson $\nu = 0.3$), in order to allow a mechanical compression of the octree mesh.

The above approach, unlike direct projection of surface nodes that disregards inner nodes position, is guaranteed to not produce invalid elements[5]. To preserve the mesh overall quality the elements quality is checked at each step of the deformation. If necessary, the mechanical resistance or "stiffness" is artificially increased for those elements that suffer the greatest quality loss before proceeding to the next deformation step (for example by increasing the $Young$ modulus value by a factor of two). This can result in oscillations between neighboring elements. Therefore in order to guarantee the algorithm termination, a constraint was used: if for a given element the $Young$'s modulus value reaches a predefined threshold, the algorithm stops increasing the stiffness for that element. Note that a realistic deformation of the mesh is not the goal of this process. The virtual mechanical medium is merely used to compute the inner relaxation of the nodes at each deformation step: the arbitrary increase of the $Young$'s modulus prevents the most exposed elements from being excessively deformed and possibly degraded.

The entire process can be seen in algorithm 3. Note that in the Do-While loop, if some elements need to be stiffened, the deformation for a given step is redone using the same initial node positions S and I along with the updated elements E. Another very important remark is that this method does not insert new points to the input mesh. The final result for the example domain Ω shown through the

[5] As explained in subsection 2.3.4 an element is said to be invalid when one or more nodes of it have a negative Jacobian value.

Algorithm 3 Surface representation

Require: source mesh and destination mesh.

Let E be the set of elements in the source mesh, defining the mechanical model.
Let S be the set of surface nodes in the source mesh.
Let I be the set of inner nodes in the source mesh.
Let D be the destination surface mesh.
for all surface nodes P in S **do**
  compute the projection vector of P on D: $U(P)$.
end for
Let $Step$=0.
repeat
  for all surface nodes P in S **do**
    Compute displaced node P': $P' = P + U(P)/$MAX_STEP.
  end for
  Let S' be the set of resulting surface nodes positions
  Compute the inner nodes deformations using the mechanical model E, constrained by the new surface nodes positions S'
  Let I' be the resulting inner nodes positions
  for all elements in E **do**
    Let Q be the quality of E given the new nodes positions S' and I'
    Let Y be the stiffness of the current element
    if Q is not acceptable and $Y <$ Y_MAX **then**
      increase Y in E: $Y = 2 * Y$
    end if
  end for
  if no change in element stiffness have occurred in E **then**
    accept the deformation and proceed to next step: $S = S'$, $I = I'$ and $Step = Step + 1$
  end if
until $Step =$ MAX_STEP

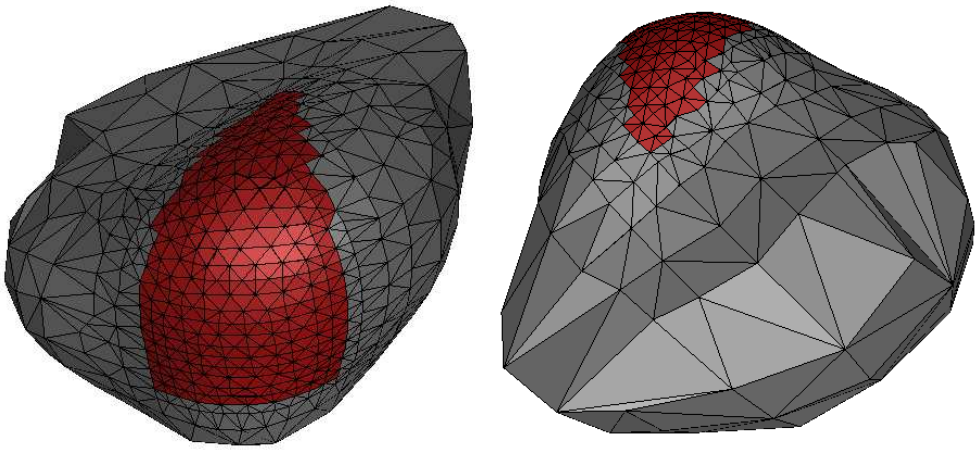

Figure 3.23: The output mesh for the given domain Ω.

chapter can be seen in figure 3.23. This mesh counts with 3206 nodes and 8477 elements (1234 hexahedra, 59 prisms, 2163 pyramids and 4991 tetrahedra).

Finally, algorithm 4 shows the overall process to produce mixed-element meshes from an octree-based approach and local region refinement, where Ω_s corresponds to the input surface mesh of Ω, RoI is the Region of Interest (where the mesh needs to be more refined) and $nodes_{ref}$ is an estimated number of nodes the mesh should have (given by the user).

Algorithm 4 Mesh generation with region refinement

Require: Ω_s, RoI and $nodes_{ref}$.

Let $nodes_{current}$ be the current number of nodes in the mesh.
repeat
  for all elements that intersect the RoI **do**
    Split in 8 new cubes.
    Remove elements outside Ω_s.
  end for
until $node_{current} > node_{ref}$
repeat
  for each element in the mesh **do**
    if the element isn't one-irregular **then**
      Split the element in 8 cubes
      Remove the new cubes if they reside outside Ω_s
    end if
  end for
until the entire mesh is one-irregular
for each element in the mesh **do**
  if the element is one-irregular **then**
    Split in mixed-elements
    Remove elements outside Ω_s
  end if
end for
Achieve surface representation following algorithm 3

Chapter 4

Applications of the developed meshing techniques

Chapitre 4: Application des techniques de maillages développés

Abstract

In this chapter the developed meshing techniques reviewed on chapter 3 are applied to medical simulations. The reparation method after registration of section 3.2 is applied to femur and maxillo-facial simulations and the algorithm to generate meshes with localized region refinement of section 3.3 is applied to the brain-shift problem.

4.1 Meshing a femur

Section 2.3 made an introduction to mesh registration (adaptation). The key idea of this process is that an existing mesh named the "atlas" Ω^{atlas} is adapted to represent another domain Ω. In this process the nodes of the Ω^{atlas} are moved in order to fit, as close as possible Ω. Sometimes the displacement of the nodes produces a malformation in some elements that become invalid or of poor quality. To measure the malformation of element e, the determinant of Jacobian matrix (det**J**) was computed for each node i of e. If one or more det$\mathbf{J}_i < 0$ were detected, e was declared invalid. When det$\mathbf{J}_i > 0 \forall i \in e$, the quality of the element is estimated through the "Jacobian ratio" (Jac_i^{ratio}) that was computed as $Jac_i^{ratio} = detJ_e^{max}/detJ_i$, where $detJ_e^{max}$ is the maximum value of det**J** between all the nodes in e. The behavior of the functions det**J** and Jac^{ratio} were presented in subsection 3.2.1. Table 4.1 summarizes the values in order to classify the elements in the categories of: "invalid", "bad quality" and "good" elements.

The warping factor WF measures the level of node co-planarity in a face. A $WF = 0$ means a co-planar face. When WF increases its value, the face is said

element category	det**J**	Jac^{ratio}
invalid	negative	negative
bad quality	positive	≥ 30
good	positive	between $]0, 30[$

Table 4.1: Classification of invalid, bad quality and good elements for elements after registration.

to be "less co-planar". The developed reparation strategy for malformed elements in a mesh consider the WF as another measure of quality in order to reallocate "invalid" and "bad quality" nodes.

Note that det**J**, Jac^{ratio} and WF are functions measured at each node. The two firsts are computed regarding all the nodes in an element and the third is made regarding all nodes in a face. It is for this reason that an element is said to be "invalid" or presents "bad quality" at some node.

4.1.1 Femur mesh atlas

A femur mesh atlas constructed by hand was introduced by Couteau *et al* in [20] in order to test their developed technique, the Mesh-Matching (M-M). This atlas mesh (Ω^{atlas}) counts with 4052 nodes and 3018 elements (2960 hexahedra and 58 prisms). The first approach of Couteau did not consider element reparation in terms of the mesh validity and quality. The work of Luboz *et al* [45] was the first to consider mesh reparation using the Jacobian matrix after the mesh-matching process. The work of Luboz is focused on achieving a valid configuration (det**J** > 0) in the entire mesh but it does not consider quality notions for the elements.

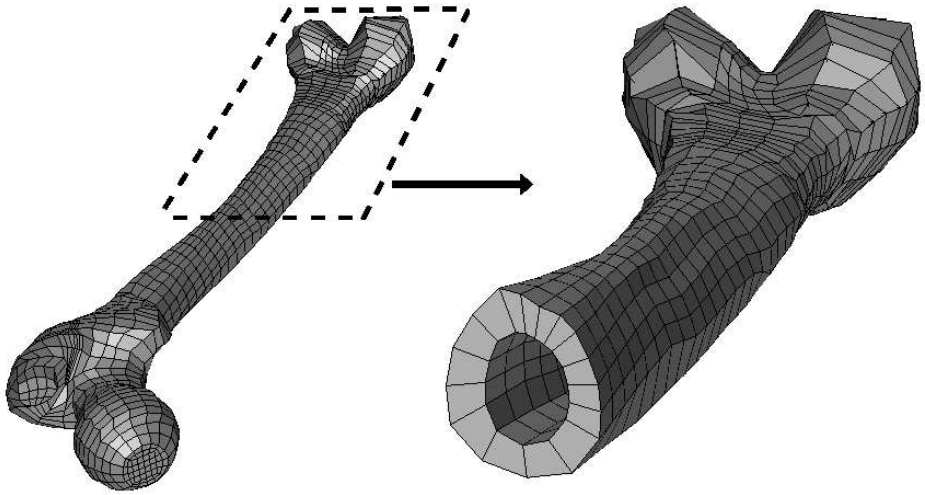

Figure 4.1: The atlas mesh of the femur.

Figure 4.1 shows Ω^{atlas} employed in the matching (registration) process. This mesh has the following real dimensions: $448mm$ height, $117mm$ width and $74mm$ depth. Note that no invalid elements were detected in this mesh, however it counts with 6 bad quality elements (det**J** ratio $\in]30, 81.4875]$ and a worst WF of 1.2728).

4.1.2 Improving the quality of the atlas mesh

As mentioned in section 3.2 the implemented solution considers a first stage of achieving validity and a second of improving the quality of the mesh. Therefore when the implemented solution is applied over the "bad quality" Ω^{atlas} the Validity Reparation Process (VRP) finished without changing the node positions. In the other hand, the Quality Reparation Process (QRP) improved the mesh: results are shown in table 4.2

	Ω^{atlas}	Ω^{atlas}_{rep}
elements with $Jac^{ratio} < 0$	0	0
elements with $Jac^{ratio} > 30$	6	0
$min\ Jac^{ratio}$	1	1
$max\ Jac^{ratio}$	81.4875	29.7313
worst WF	1.2728	0.2729

Table 4.2: Comparison between femur atlas before and after reparation.

As the scale of Ω^{atlas} is known and the position of each node i is also known before and after reparation, the displacement distance for i can be measured as:

$$dis_i = \|(x_i^0, y_i^0, z_i^0) - (x_i^1, y_i^1, z_i^1)\|$$

where 0 is before reparation and 1 is after reparation. The distance dis_i is then also measured in mm.

The **maximum** displacement over the entire mesh was $0.068mm$ and only 6 nodes where reallocated (all the rest of the nodes remained at the same position regarding the Ω^{atlas} mesh). As mentioned in subsection 3.2.4, the implemented solution isolates the systems of nodes to repair. In this particular case, 4 isolated systems were repaired involving 15 elements.

4.1.3 Adapting the femur mesh atlas to patient geometries

The algorithm was tested against 5 femur examples taking from real patient data. The M-M algorithm produced the match of the *atlas* to the cloud of points that represented the external surface of patient femurs. The atlas plus the 5 final meshes after reparation, can be seen in figure 4.2.

	A	A_{rep}
elements with $Jac^{ratio} < 0$	2	0
elements with $Jac^{ratio} > 30$	3	0
$min\ Jac^{ratio}$	-385.619	1.02156
$max\ Jac^{ratio}$	751.634	29.9809
worst WF	1.768	1.768

Table 4.3: Comparison between femur A before and after reparation.

Table 4.3 shows the results for the reparation of the first femur. In this case the largest displacement of a node was $1.42mm$. It is important to insist in the point that any $Jac^{ratio} < 0$ is considered worst than a $Jac^{ratio} \gg 30$, as a negative value makes the mesh invalid (thus useless for the FEM to perform a simulation).

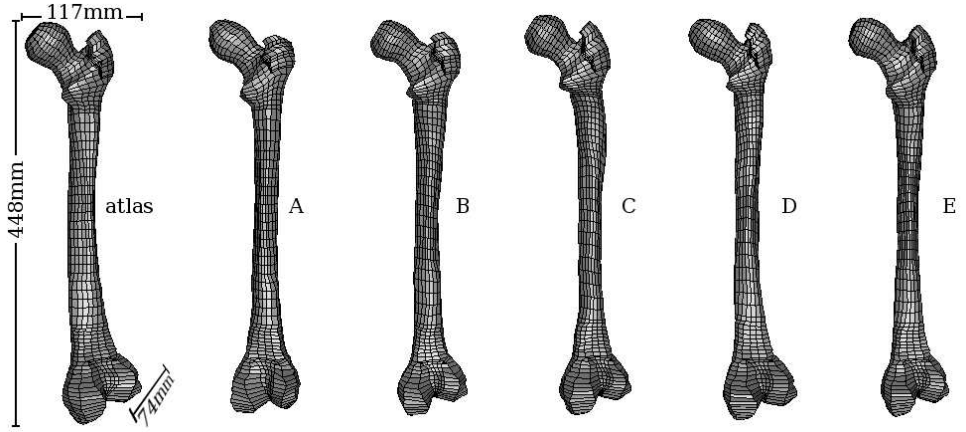

Figure 4.2: The atlas and 5 repaired femurs

Table 4.4 shows the results for femur B where the largest node displacement is $0.0735mm$.

	B	B_{rep}
elements with $Jac^{ratio} < 0$	0	0
elements with $Jac^{ratio} > 30$	10	0
$min\ Jac^{ratio}$	1.0339	1.03396
$max\ Jac^{ratio}$	551.239	29.7313
worst WF	1.888	1.888

Table 4.4: Comparison between femur B before and after reparation.

For femur C the results are in table 4.5, with a largest node displacement of $0.0439mm$.

	C	C_{rep}
elements with $Jac^{ratio} < 0$	0	0
elements with $Jac^{ratio} > 30$	4	0
$min\ Jac^{ratio}$	1.06254	1.06254
$max\ Jac^{ratio}$	114.015	29.3822
worst WF	1.594	1.594

Table 4.5: Comparison between femur C before and after reparation.

In the case of femur D the largest node displacement is $0.0536mm$ (see table 4.6). At this point it can be seen how the reparation of poor quality elements involves a small distance of the nodes. This is due to the fact that function Jac^{ratio} is exponential as it was shown in subsection 3.2.1. The node displacements, in order to produce a $Jac^{ratio} : 1000 \rightarrow 30$, is very small. However a $Jac^{ratio} : 30 \rightarrow 2$, needs very large node displacements.

	D	D_{rep}
elements with $Jac^{ratio} < 0$	0	0
elements with $Jac^{ratio} > 30$	4	0
$min\ Jac^{ratio}$	1.01939	1.01939
$max\ Jac^{ratio}$	190.129	29.9261
worst WF	1.594	1.594

Table 4.6: Comparison between femur D before and after reparation.

Finally, table 4.7 shows the results for femur E, where the largest node displacement was 0.1532.

	E	E_{rep}
elements with $Jac^{ratio} < 0$	1	0
elements with $Jac^{ratio} > 30$	4	0
$min\ Jac^{ratio}$	-77.1519	1.02682
$max\ Jac^{ratio}$	53.7628	29.8869
worst WF	2.1388	2.1388

Table 4.7: Comparison between femur E before and after reparation.

The presented tables (from 4.3 to 4.7) show that all the tested meshes achieved validity and quality after reparation. It was possible to find a reallocation of the nodes in order to achieve a $Jac^{ratio} < 30$, which is acceptable for ANSYS®. Regarding the WF no improvements were made. This is due to fact that very small displacements were produced. Moreover, as explained in subsection 3.2.4, for each node i the direction of displacement is computed by $\nabla F(Jac_i^{ratio}, WF_i)$. In this function F, the Jac^{ratio} function is more important than the WF, therefore the reallocation of the nodes searches to improve more the Jac^{ratio} than the WF.

4.2 Meshing a face for maxillo-facial surgery

The goal of this simulation is to determine possible outcomes before maxillo-facial surgery. To this purpose a face atlas mesh was produced and, as for the case of the femur, the Mesh-Matching (M-M) algorithm is used to adapt the *atlas* mesh to patient data.

The goal of showing this example in this thesis is to test the reparation algorithm with a complex model (due to the face anatomy) that has a tendency to produce invalid and poor quality elements (in particular, in the regions of the eyes and the lips). The face model counts with 8746 nodes, 6030 hexahedra and 314 prisms. Figure 4.3 shows the employed *atlas* model developed in [50].

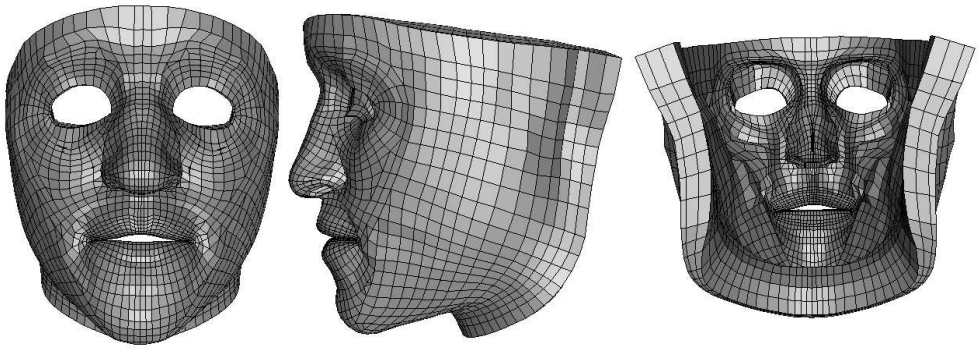

Figure 4.3: The atlas face model

The mesh is registered to patient surface model extracted from data made available thanks MAP5[1] Laboratory (Yves Rozenholc, Université Paris 5). Actually, only the nodes of the mesh shown at the left panel of figure 4.4, are used. After registration, the mesh counts with 12 invalid elements ($Jac^{ratio} < 0$) and 5 bad quality elements ($Jac^{ratio} > 30$). Our developed algorithm is applied to repair the mesh and the final result can be seen at the right panel of figure 4.4 and 4.5.

The 12 invalid elements are repaired in the first stage of our algorithm, leaving the mesh with 13 bad quality elements. Note that the process of repairing an invalid element, might use a displacement distance of the invalid nodes that, due to the iterative behavior of the reparation algorithm, produces not only a valid but also an element of acceptable quality. This explains the 13 elements to repair in the quality process (instead of 17: the initial 5 of bad quality plus the 12 that were invalid and then repaired by our algorithm).

The quality improvement process then is executed and finally solves all the problems except for two elements that remain with a poor quality. The Jac^{ratio}

[1] UMR CNRS 8145.

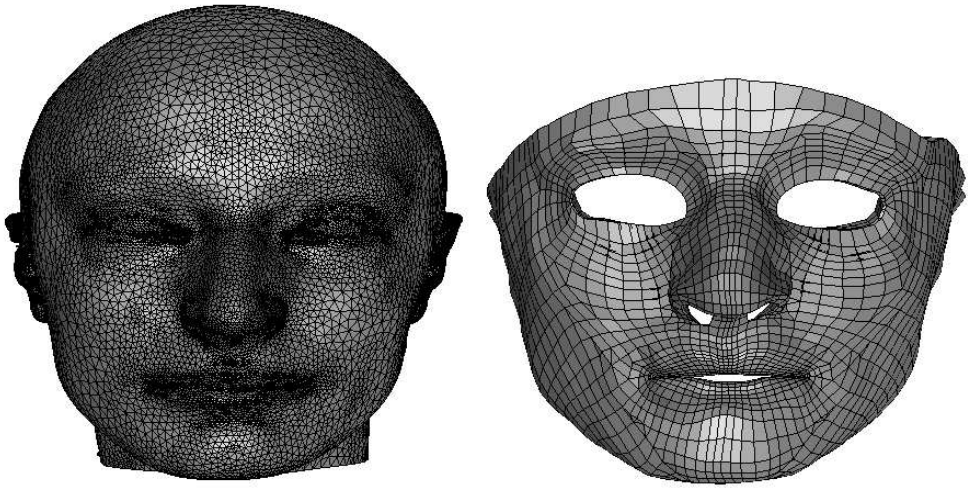

Figure 4.4: The atlas face model after registration and reparation: front view

value for those elements are: 83.8 and 746. The other 6342 elements of the mesh finish the process with a $Jac^{ratio} < 30$.

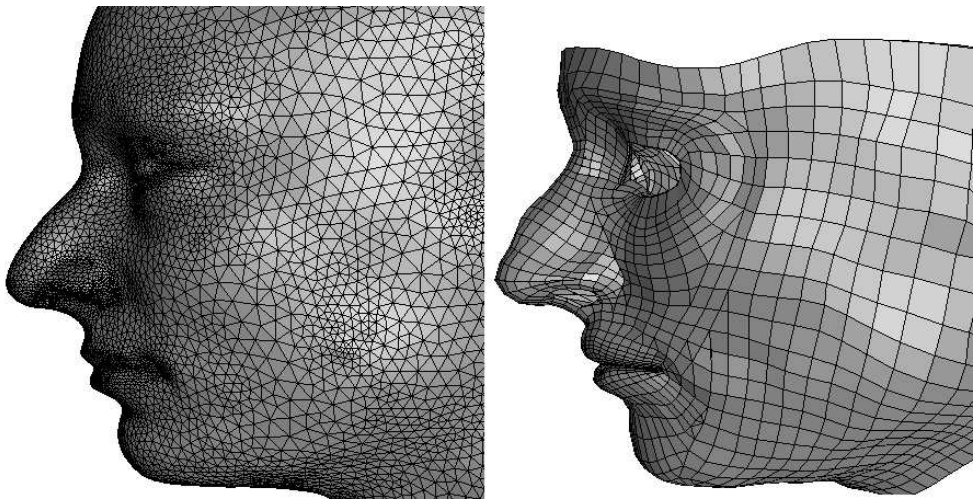

Figure 4.5: The atlas face model after registration and reparation: profile view

As explained in subsection 3.2.1, the determinant of the Jacobian matrix (det**J**) is computed at each node regarding all the nodes in the element. Then the Jac^{ratio} is computed also for each node as a function of all the det**J** of the element nodes. Our developed algorithm then searches to repair the elements by reallocating **only** the nodes where the Jac^{ratio} value is not acceptable. For this reason it is not possible to ensure that the algorithm will always find a solution for the "bad"

elements. In some cases, it would also be necessary to reallocate some other nodes that have an acceptable Jac^{ratio} value, in order to provide to the "bad nodes" more space to find a solution (for which the element becomes acceptable). This will be discussed in subsection 5.2.1.

Even though the quality threshold of $Jac^{ratio} < 30$ is not achieved for every element in the mesh, the resulting mesh can be used by ANSYS®. The only difference is that results of the simulation might be inaccurate at the zone where the poor quality elements are. Note that without the reparation, the mesh could not be used for FE analysis.

4.3 The brain shift

Accurate localization of the target is essential to reduce morbidity during a brain tumor removal intervention. Image guided neurosurgery nowadays faces an important issue for large skull openings, with intra-operative changes that remain largely unsolved.

Once the skull is opened a deformation of the brain naturally occurs. This phenomena is known as "the brain-shift". The causes of this deformation can be grouped by:

- physical changes (dura opening, gravity, loss of cerebrospinal fluid, actions of the neurosurgeon, etc) and
- physiological phenomena (swelling due to osmotic drugs, anesthetics, etc).

As a consequence of this intra-operative brain-shift, pre-operative images no longer correspond to reality. Therefore the neuro-navigation system based on those images doesn't necessarily represent the current situation.

In order to face this problem, various teams [16] have proposed to incorporate into existing image-guided neurosurgical systems, a biomechanical modeling to compensate the brain deformations by updating the pre-operative images and planning according to intra-operative brain shape changes. For this, such measured changes (for example the changes of the external shape of the brain tissues in the opening skull region) are given as new boundary conditions to the biomechanical model of the brain tissues that infers the new position of the tumor. Such intra-operative use of a biomechanical model implies that a strong modeling effort must be carried out. Three steps are followed to design the brain model:

- The segmentation of pre-operative images (MRI) to locate the tumor and to build the external surface mesh of the brain.
- The generation of a volume mesh optimized for real-time simulation.

- The creation of a model of the brain-shift with Finite Elements (FE).

4.3.1 Brain-shift simulation constraints

As time is crucial in surgery, the goal of the simulation is to compute a solution for the FEM of the brain as fast as possible. The speed of the FEM directly depends on the number of degrees of freedom the system has, thus an optimal mesh in terms of quantity of nodes must be provided.

Chrisochoides *et al* in [16] mentioned two possible ways to simulate the brain-shift:

- Somehow develop equally accurate, but less computationally demanding models (this could be seen as an optimization to the FEM).

- Use scalable and more efficient implementations of the methods we have.

The second option was explored in [16] by doing a parallelization of the algorithms using an equally and highly refined mesh. The problem with this solution is that it is expensive. In practice, it is difficult to implement a solution of this type in the hospitals, as the budget required for a cluster of computers specially focused on this problem is, in most cases, not feasible.

If instead of focusing on a parallelization of the FEM, an optimization is searched from the meshing point of view, a good representation of the tumor as well as the Skull Opening Point (SOP) and the path between them is mandatory because here is where a greater deformation is expected [55]. This is clearly a Region of Interest (RoI) where elements must be highly refined regarding the rest of the mesh.

A mesh without quality elements can lead to errors in the computation of the FEM thus quality is also an important issue in this problem. Therefore the constraints to model the brain-shift in a real-time application are:

1. The final mesh must be refined enough in the RoI and coarse elsewhere.

2. Achieve surface representation for the input FE mesh.

3. Guarantee element quality throughout the entire mesh.

For all of these reasons, the developed technique explained in subsection 3.3 has been applied to produce a patient specific model to simulate the brain-shift.

4.3.2 Meshing the brain for a tumor resection surgery

A surface model of the brain was constructed from a set of Magnetic Resonance Images (MRI) scanned from a patient. This work was produced by Araya *et al* in [3]. After image segmentation, the algorithm produces a surface mesh of the brain. In this case, the surface mesh has 3152 nodes and 6300 triangles (figure 4.6).

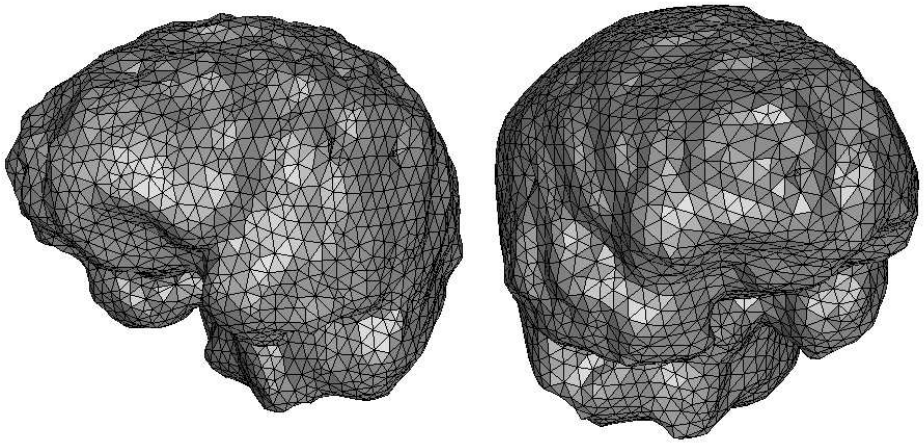

Figure 4.6: Two views of an input surface mesh of the brain.

The information concerning the tumor should be included as part of the surface mesh or as an inner set of connected triangles. However in this model, this information was voluntary omitted in order to use the same surface model to test the mesh generation with several different definitions of the RoI. Note that if the tumor information was presented on the mesh, this would present no problem to produce a mesh with a RoI that doesn't include the tumor, but it would be weird. In this surgical case, the RoI is the path from the skull opening to the location of tumor. Figure 4.7 shows the brain model in transparent and the RoI in a solid green polyhedron.

As mentioned in subsection 3.3.2, one last input is required: the estimated node number (nn) of the volume mesh. Just to recall, the goal of nn is to determine the level of refinement of the mesh. The implemented technique splits the elements until the number of nodes in the mesh is superior to nn. In the example $nn = 500$. Once the algorithm that splits only the elements that intersect the RoI is applied, the total amount of nodes in the mesh is 1211, from which 705 resides inside the RoI. The total amount of elements (only hexahedra at this level of the algorithm) is 822. The resulting output can be seen in figure 4.8.

As explained in subsection 3.3.3 an element can be inside, outside or intersect Ω, where Ω is the domain to mesh (in this case the brain). Note that at this level of

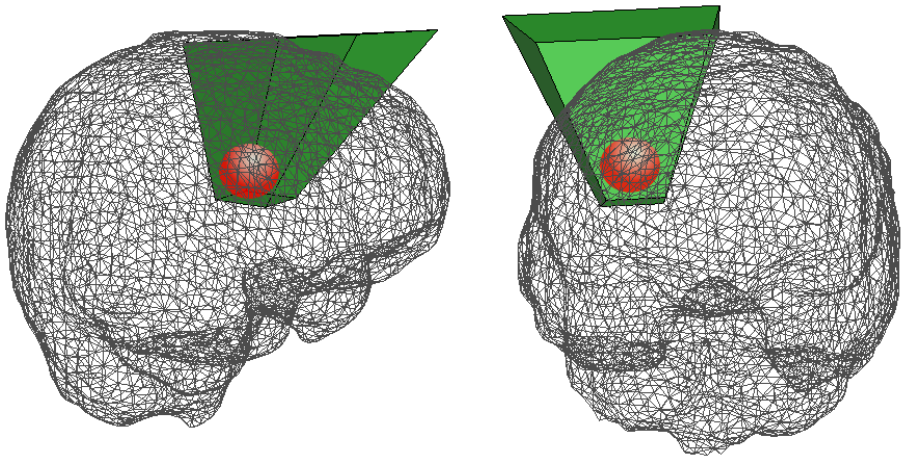

Figure 4.7: Two views of the brain (in transparent), the RoI in a solid green polyhedron and the tumor represented by an sphere.

the algorithm, just a few elements have been removed (the ones that reside outside Ω). If no element were removed, the total amount of nodes would be 1344.

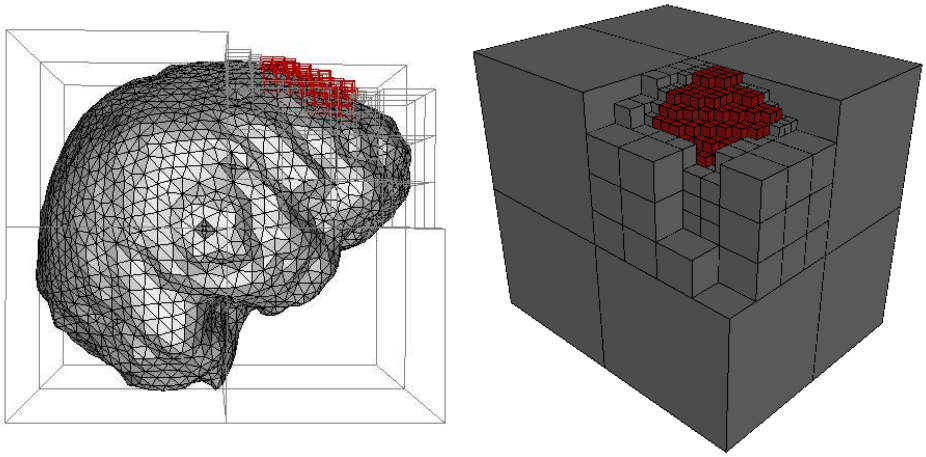

Figure 4.8: Left: the brain shown in solid and volume mesh in transparent. Right: volume mesh in solid. Red elements intersect the RoI.

The next step of the algorithm is to achieve the one-irregular (figure 4.9) state of the mesh. This process inserts more nodes in the mesh, however the subdivision of coarse elements also increase the number of elements that reside outside Ω. At the end of this process, the mesh has 1613 nodes (1774 nodes if no element is removed).

The different templates that manage the transition between coarse and refined mesh regions can now be applied. The resulting mesh is seen in figure 4.10.

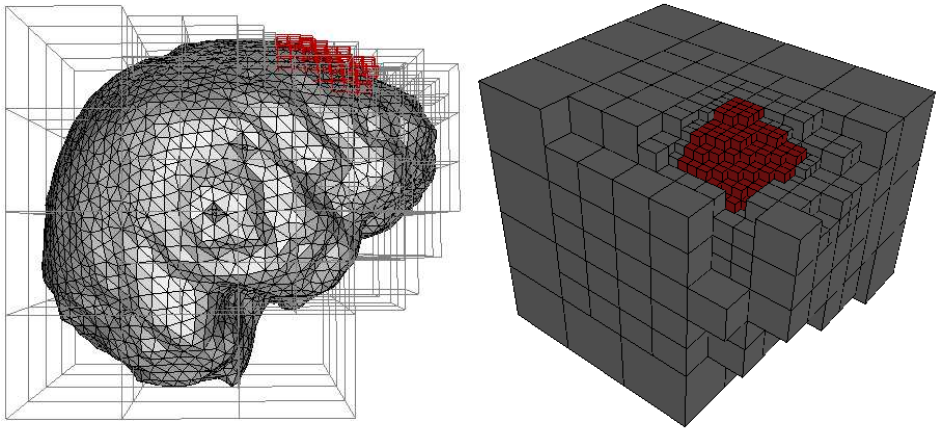

Figure 4.9: Achieving the one-irregular state of the output mesh for the brain shift

The current output mesh is made of mixed-elements. Table 4.8 shows some statistics over the mesh. It is important to note that those statistics are from the entire mesh. Regarding the RoI, the entire region is covered by hexahedra. Another important remark is over the quality of the elements. The perfect element is described by an Aspect Ratio (ar) equal to one. In order to obtain a perfect quality (in the current state of the mesh) it is possible to change the initial octant of the mesh: the Bounding Box ($Bbox$). The current computation of the $Bbox$ searches the minimal and maximal coordinates of the input surface mesh and from those coordinates it constructs an hexahedron that contains Ω. If instead of using the minimal hexahedron, the minimal cube is used, the ar would be one for most of the elements (at least for every hexahedron in the mesh). This is not performed as this strategy has more tendency to produce elements that barely intersect Ω and in consequence, make more difficult the process of surface representation of Ω (the next step in the developed strategy).

	Hexahedra	Prisms	Pyramids	Tetrahedra
Quantity	853	40	1057	629
Volume	37.3942 %	0.688616 %	45.4639 %	16.4532 %
$\overline{ar}$	1.27056	1.28054	1.91196	1.76513
Worst ar	1.27056	1.39044	2.65071	3.0426

Table 4.8: Statistics over the mixed-element mesh, where $\overline{ar}$ is the aspect ratio average.

As table 4.8 shows, the most used and most significant element in the mesh in terms of volume representation is the pyramid. This can be explained as the tem-

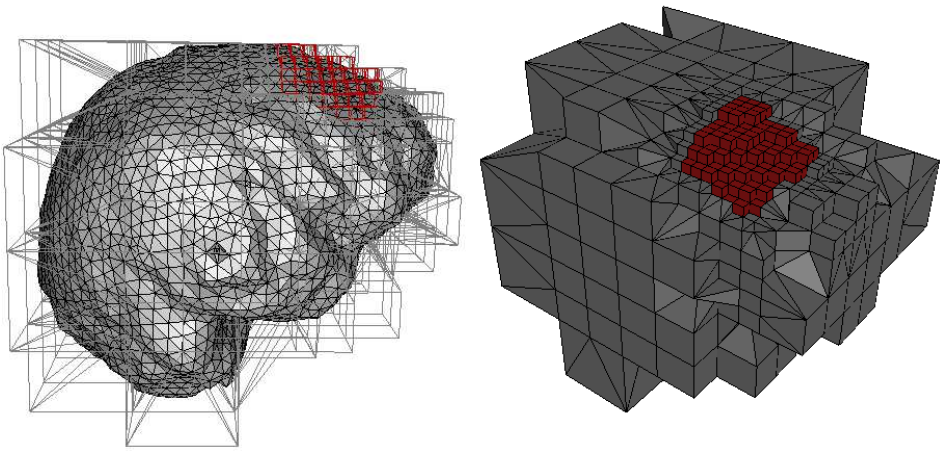

Figure 4.10: Applying the templates to mange transitions for the output brain mesh

plates used to produce a congruent transition between refined and coarse regions insert several pyramids to avoid adding new nodes in the mesh.

The next and final step of the developed application is to achieve surface representation of Ω. As explained in subsection 3.3.7, a registration process will match the outside nodes to the surface following a compression of the entire mesh. The method was developed by Bucki *et al* in [11] and is performed for tetrahedra-only meshes. This causes no problems in our case as a tetrahedralization of the entire mesh can congruently be done (see [40]). This doesn't mean that the final mesh has only tetrahedra, as the relevance of the method is to compute the location of the nodes. Therefore the mesh of mixed-element $\mathcal{M}_{mix}$ is converted into $\mathcal{M}_{tetra}$. The registration occurs over the nodes of $\mathcal{M}_{tetra}$, producing $\mathcal{M}'_{tetra}$ that achieves the surface representation of Ω. In order to produce the final mixed-element mesh, the nodes of $\mathcal{M}_{mix}$ are updated to the locations of $\mathcal{M}'_{tetra}$.

The relation of the nodes between $\mathcal{M}_{mix}$ and $\mathcal{M}_{tetra}$ must be one-to-one. This is true for all the elements in the mesh except for the ones that intersect the surface. Indeed, an hexahedron that is split into tetrahedra might produce an entire tetrahedron outside Ω that could therefore, not be projected into the surface of Ω. To solve this problem it was decided that all the elements that intersect the surface would be "tetrahedralized" in the final state of $\mathcal{M}_{mix}$. The explanation of "tetra-only" at the surface was not included as part of the main algorithm in subsection 3.3.7 because this is not the unique solution. This will be part of the discussions in section 5.2.

Before presenting the results regarding surface representation, table 4.9 summarizes the different states of the mesh at each stage of the algorithm.

	Nodes	Elements	Time (s)
Octree	1211	822	$9s$
One-irregular	1613	1093	$7s$
Transitions	1580	2579	$3s$
Surf. Tetra.	1650	4797	$5s$

Table 4.9: Number of nodes, elements and time at each stage of the brain-shift mesh, where *Surf. Tetra.* corresponds to surface tetrahedralization.

Two options to achieve surface representation are studied. Figure 4.11 shows how surface representation over Ω is achieved using the "direct projection" strategy, which corresponds to project the outside nodes to the closest point on the surface (different from the registration method that will be shown later).

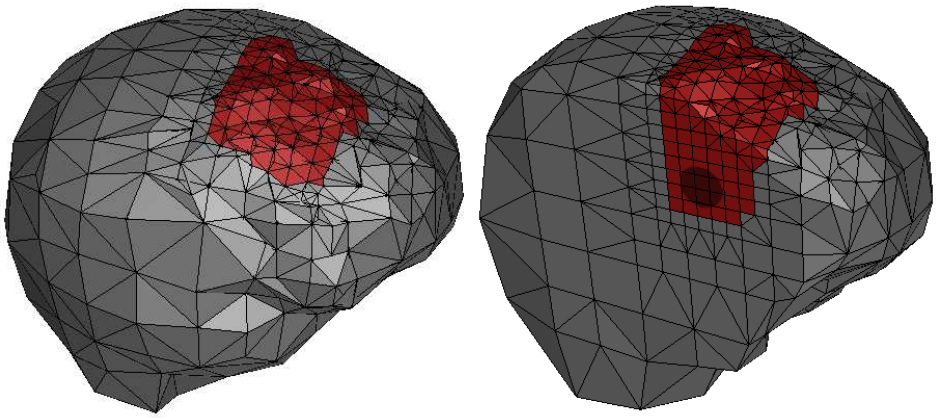

Figure 4.11: Left: the entire mesh. Right: a vertical cut over the same mesh: the circle represents the tumor.

This mesh has perfect orientation at the inner elements (the ones that were not projected into the surface) as, in difference with the registration method, it doesn't reallocate inner nodes. In other words, only elements that intersect the surface "suffer" the deformation to achieve the representation of Ω. The computation of the projections took 6 seconds.

As the elements that intersected the surface were "tetrahedralized" the composition of the mesh, in terms of elements, has changed. Table 4.10 shows the statistics for the current mesh.

The different types of elements, except for the tetrahedra, do not suffer a significant degradation of their quality regarding table 4.8. The quality of the tetrahedra is not acceptable. This degradation can be explained as the elements that intersect

	Hexahedra	Prisms	Pyramids	Tetrahedra
Quantity	479	25	1204	3089
Volume	6.21939 %	0.859618 %	21.1905 %	71.7305 %
$\overline{ar}$	1.27056	1.25675	2.08679	4.8964e+14
Worst ar	1.27056	1.39044	2.65071	1.4584e+18

Table 4.10: Statistics over the projected mesh using "the closest point" over Ω strategy, where $\overline{ar}$ is the aspect ratio average.

the surface are the only ones that suffer a displacement of their nodes in order to achieve the representation of Ω.

Note that the concept of "aspect ratio" (ar) for the tetrahedron is more than just a measure of the length of it edges. The computation involves the volume of the tetrahedron; therefore a sliver element (tetrahedron with almost co-planar nodes) has a very large ar value. Actually, in the presented mesh, there are only two tetrahedra with an unacceptable ar ($> 10^{16}$) and seven "bad" ones (between $[200, 500]$. All the rest of the elements have an $ar \ll 200$.

Note that for the direct projection method, it was also preferred a tetrahedralization of the surface elements. It was shown that tetrahedra suffered an important quality degradation. If hexahedra, prisms and pyramids are directly projected into the surface, not only a quality degradation arises but also it is possible to produce invalid elements (see subsection 1.3.2). The algorithm of section 3.2 could be used to repair those invalid elements, however important node displacements might be needed, loosing surface representation (which is the goal of this stage of the algorithm). Moreover, in several cases it might not exist a solution of the system (see 4.2). This discussion will continue on subsection 5.2.2.

	Hexahedra	Prisms	Pyramids	Tetrahedra
Quantity	479	25	1204	3089
Volume	6.10912 %	1.09946 %	19.0152 %	73.7762 %
$\overline{ar}$	1.63852	1.56582	2.39152	4.58165
Worst ar	5.87285	3.42122	9.29464	3509.89

Table 4.11: Statistics over the projected mesh using the registration method, where $\overline{ar}$ is the aspect ratio average.

For the above reasons, the strategy of achieving surface representation *via* the FEM, where all the elements are mechanically compressed, was preferred. It is clear (at the right panel of figure 4.12) how all the elements, after the registration method, suffer a deformation. The most important difference is that now, the worst element of the mesh is a tetrahedron with $ar = 3509.89$. After that, only 5 tetrahedra have an ar between $[200, 650]$ and all the rest of the elements have an $ar \ll 200$. Table 4.11 summarizes the overall quality results.

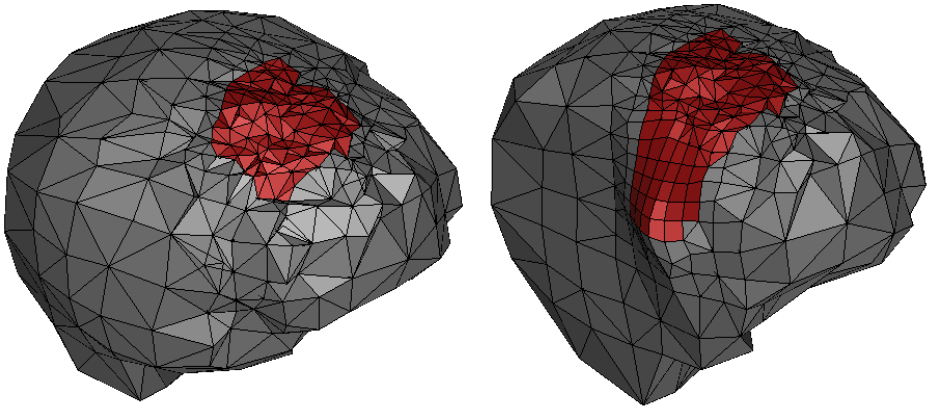

Figure 4.12: Left: the entire mesh. Right: a vertical cut (removing the same elements as in figure 4.11.

Regarding the ar, the results of table 4.11 are much better than the ones of table 4.10. In other words, regarding the quality, the registration method is better than direct projection method. This is quite natural since in the case of the registration method, the FEM compressible brain model did "absorb" the surfacic deformations due to node projections onto the the brain surface. The registration method produces results even close to the unprojected state of the mesh (table 4.8). Of course, this comparison is made over the global statistics. When going into further details, while the worst element of the unprojected mesh has an $ar = 3.0426$, the mesh produced using the registration method has an $ar = 3509.89$ (which is much better than the worst element of the direct projection method, with an $ar = 1.4584e + 18$).

4.3.3 Meshing the brain with a different region of interest

In order to illustrate how easy is to compute the results changing the RoI, figure 4.13 shows the final mesh with a RoI in the top middle section of the brain.

Table 4.12 shows the statistics for the mesh with the direct projection method. Note that the worst element of this mesh has an $ar = \infty$. Once again, only two tetrahedra presented an $ar = \infty$, then 4 tetrahedra with an ar between $[230, 480]$ and all the rest with an $ar \ll 200$.

Table 4.13 shows the statistics for the mesh with the registration method. In this case the worst element has an $ar = 52.6016$.

The mesh using the direct projection method was produced in 32 seconds. The mesh using the registration method, needs the calculation of the projected nodes

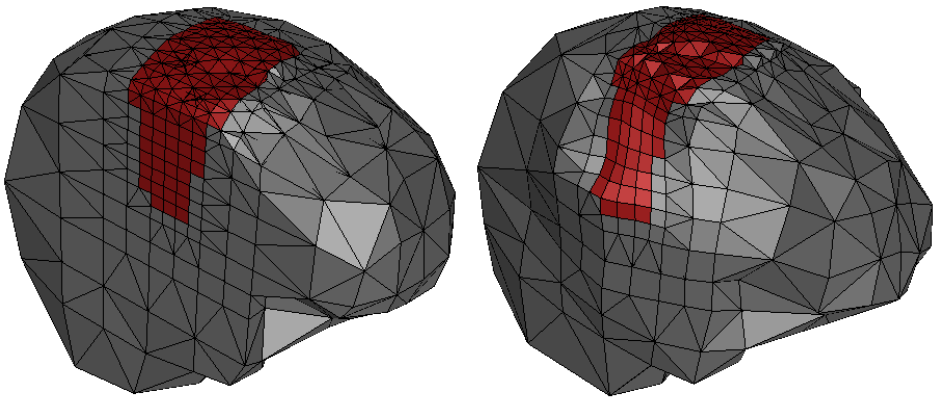

Figure 4.13: Left: the output mesh using the direct projection method. Right: the output mesh for the registration method.

	Hexahedra	Prisms	Pyramids	Tetrahedra
Quantity	736	24	1500	3400
Volume	11.3729 %	0.0791253 %	16.1613 %	72.3867 %
$\overline{ar}$	1.27056	1.34322	2.13123	inf
Worst ar	1.27056	1.39044	2.65071	inf

Table 4.12: Statistics with a top middle RoI using the direct projection method, where $\overline{ar}$ is the aspect ratio average.

(from the direct projection) in order to compute the registration. In other words, the displacement direction of direct projection method are used by the registration method as the leading deformation direction. The reallocation of the rest of the nodes is computed in terms of these "leading" deformation directions. The registration method took around 5 seconds, thus the overall time to produce a mesh, with the registration method to achieve surface representation, was 37 seconds.

	Hexahedra	Prisms	Pyramids	Tetrahedra
Quantity	736	24	1500	3400
Volume	11.3729 %	0.0791253 %	16.1613 %	72.3867 %
$\overline{ar}$	1.68174	1.91652	2.49042	2.42475
Worst ar	2.75798	2.79621	9.26597	52.6016

Table 4.13: Statistics with a top middle RoI using the registration method, where $\overline{ar}$ is the aspect ratio average.

4.3.4 Comparison with tetrahedra meshing technique

In the meshing field, the most popular techniques produce only-tetrahedra meshes. One of the most important techniques for tetrahedra was developed by Frey in [29].

The strategy developed by P. Frey produces a mesh from an input surface mesh (Ω_s) that describes a domain Ω. The first step of the algorithm re-meshes Ω_s, thus producing Ω'_s in order to achieve a "good" quality over the triangles of the surface mesh. The quantity of those triangles is also managed in terms of a specific input parameter, namely the "target quantity of nodes" of the mesh. Once Ω'_s is adapted to this desired parameter, it proceeds to build the tetrahedra volume mesh by inserting inner optimal nodes in terms of tetrahedra quality.

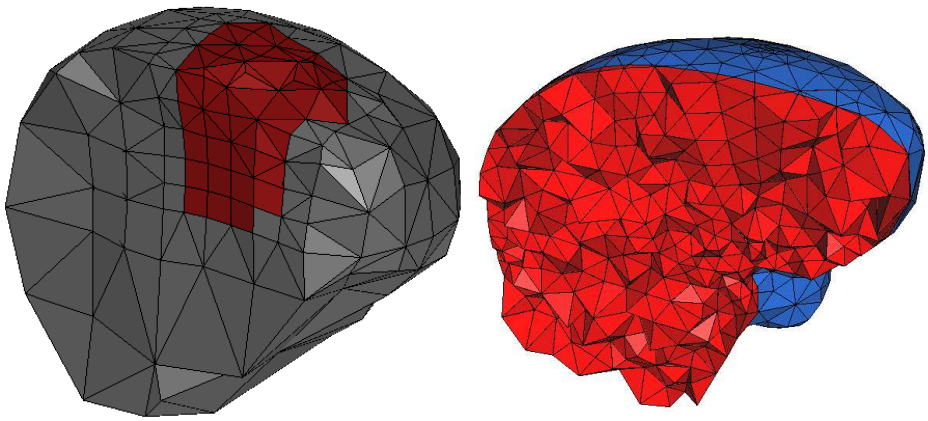

Figure 4.14: Left: an output mesh of the developed technique. Right: a tetrahedral mesh.

Figure 4.14 shows the output mesh of the developed algorithm and the algorithm from P. Frey. The developed technique produces a mesh with 499 nodes from which 59 reside in the RoI. The mesh from P. Frey counts with 2804 nodes from which 57 reside inside the RoI. Table 4.14 shows the statistics for the mesh using the developed technique, where the worst element has an aspect ratio[2], $ar = 279$.

Moreover, figure 4.15 shows an histogram of the quality of the elements for the developed technique. Note that 97% of the elements have an $ar \in [1.3, 7]$.

The tetrahedra strategy (by Frey) can also manage region refinement. However, achieving surface representation and producing quality tetrahedra are more relevant properties to this technique than producing a refined region. Therefore

[2]Remind that the aspect ratio for the tetrahedra is computed including the volume (therefore it detect slivers).

	Hexahedra	Prisms	Pyramids	Tetrahedra
Quantity	72	0	386	1418
Volume	3.90229 %	0 %	12.2747 %	83.823 %
$\overline{ar}$	1.35187	0	1.7168	3.27791
Worst ar	1.62355	0	3.19461	279.005

Table 4.14: Statistics for brain mesh of 598 nodes, where $\overline{ar}$ is the aspect ratio average.

the mesh shown at the right panel of figure 4.14 is the minimal mesh in terms of node quantity regarding the constraints of "surface representation" and quality elements. The quality of this mesh is high; the average aspect ratio ($\overline{ar}$) is 1.31789 and the worst $ar = 3.14675$.

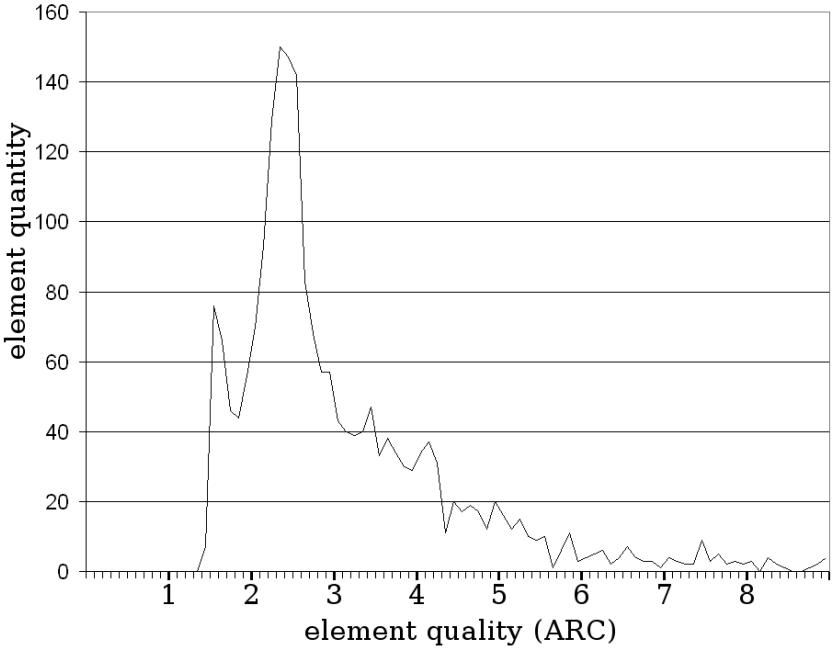

Figure 4.15: Quality histogram regarding the ar for the mesh produced with the developed technique.

When the permitted quantity of nodes is increased, the resulting mesh can show differences between refined and coarse regions. Figure 4.16 shows a brain mesh with 22929 nodes, 5858 triangles and 128393 tetrahedra. It is clear how in the RoI the mesh density is much more important than in the rest of the mesh.

In conclusion, the developed technique seems to be more efficient than the tetrahedra technique when meshes of few nodes and a high concentration of them in

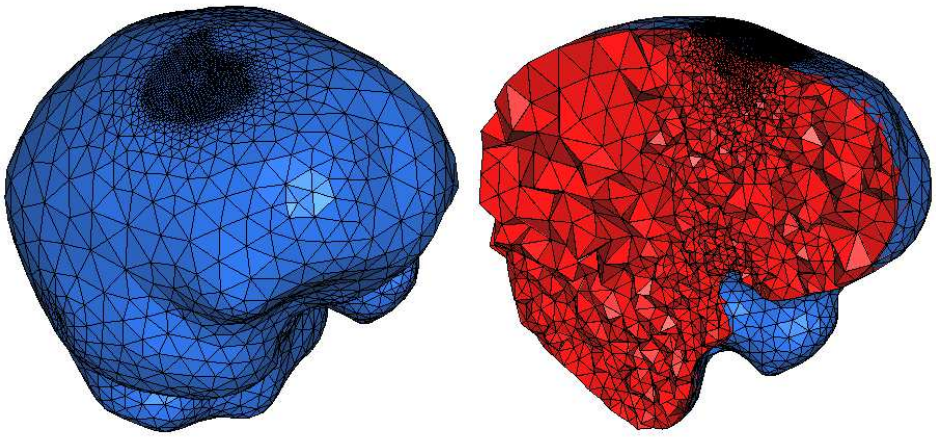

Figure 4.16: Left: Tetrahedra mesh of the brain with RoI. Right: the same mesh with a vertical cut. Mesh developed by P. Frey.

a certain region is needed. Even though good quality is not assured, it remains acceptable. Our developed technique could also be preferred over tetrahedral meshes when mixed-element mesh are desired. For all the rest of the cases in the "brain-shift" simulation problem (specially when dense and high quality meshes are required), the tetrahedra technique of P. Frey should be preferred over our developed technique.

4.3.5 Neuro-navigation system for brain tumor resection

Now that the mesh is constructed, its utility for a neuro-navigation system in the brain tumor resection surgery can be shown. The input surface model of the brain was constructed from a set of MRI of a patient with a tumor. The resulting mesh can be overlapped to those images and shows, during surgery, the simulation of the "brain-shift".

Figure 4.17 plots at the left panel the patient MRI. The middle panel shows the surface model of the brain after segmentation of those images. The right panel plots the constructed mesh with mixed-elements and region refinement regarding the opening skull point and the tumor.

The FEM is used over the produced mesh in order to simulate the brain-shift. One of the most important results of Bucki in [9] is the use of the brain vascular tree as leading points for the simulation of brain deformation during surgery.

The key idea is that before surgery, angiographic MRI scanners of the patient are performed. From this set of images two different segmentation processes enable the construction of:

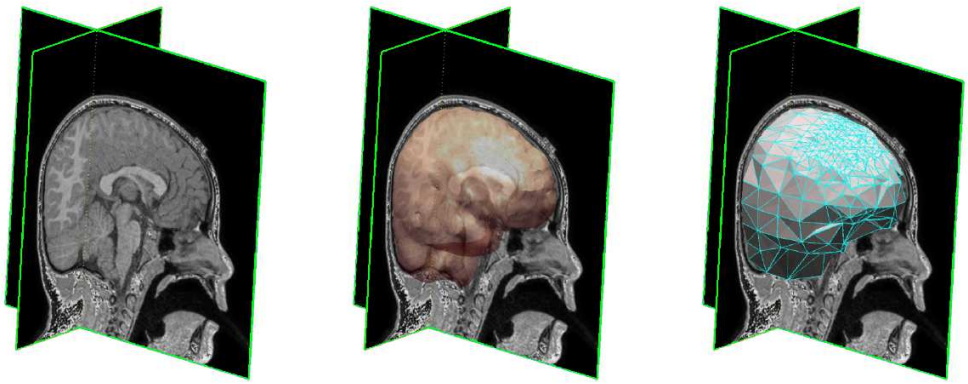

Figure 4.17: Left: patient MRI scanners. Middle: brain segmentation. Right: volumetric model of the brain (from Bucki [9]).

- The surface model of the brain (from which the mesh is constructed).
- The model of the brain vascular tree.

The FEM is pre-computed (before surgery) over the mixed-element mesh that represents the brain. During surgery, the FEM is updated in terms of the position of "tracking points" i.e., the new state of the vascular tree. During surgery, Ultrasound (US) Doppler images are acquired. The relevance of those images is that they detect the liquid flow (for example the blood). Those images are segmented in order to detect the new location of the vascular tree.

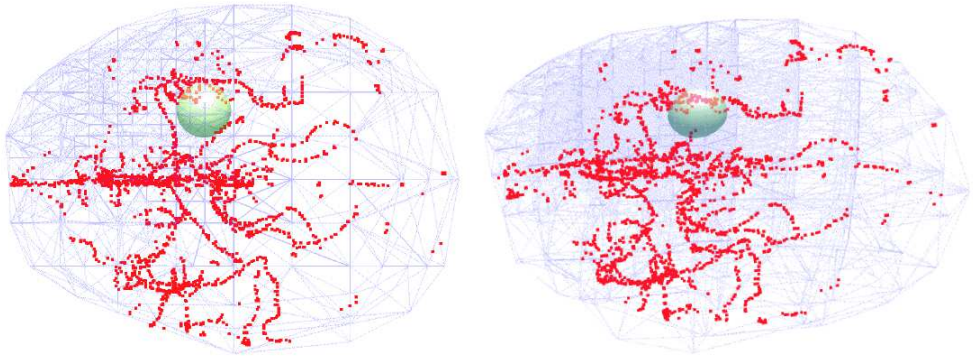

Figure 4.18: The brain mesh and the tumor represented by a sphere. Left: The initial state. Right: a deformated state leaded by the vascular tree (from Bucki [9]).

The initial model of the vascular tree is updated with the current state of it (from the US Doppler images) and this produces a displacement field. This displace-

ment field is applied over the initial model of the brain and the FEM performs the deformation over the entire brain model. The simulation provides additional information over the location of the tumor to the surgeon who might correct or not the trajectory to achieve the tumor resection.

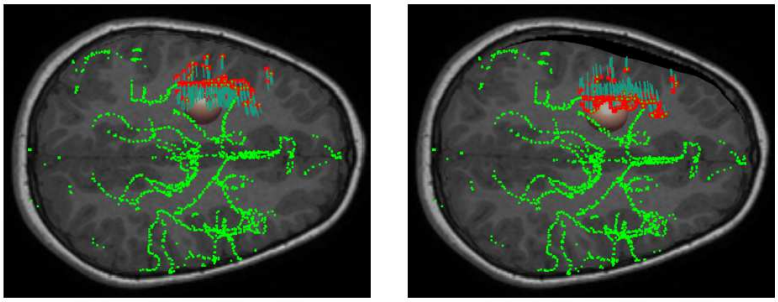

Figure 4.19: The pre-operative images of the brain are deformed using the FEM leaded by the vascular tree. Left: The initial state. Right: a deformated state (from Bucki [9]).

Figure 4.18 shows the deformation of the mesh leaded by the vascular tree. Finally, figure 4.19 shows how the pre-operative images are updated with the computed deformation of the vascular tree. To summarize the process, a set of images $\mathcal{I}^0$ is obtained. Those images are segmentated and two models are created: the mesh model $\mathcal{M}^0$ and the vascular tree model $\mathcal{V}^0$. During surgery, the US Doppler images ($\mathcal{U}^1$) are segmentated and the vascular tree is updated producing $\mathcal{V}^1$. The transformation of $\mathcal{V}^0$ into $\mathcal{V}^1$ produces a deformation field that is applied to $\mathcal{M}^0$ resulting in $\mathcal{M}^1$. The new state of the entire mesh is achieved (including the tumor) *via* the FEM. As $\mathcal{I}^0$ is related to the state of the mesh, $\mathcal{I}^1$ is produced in function of $\mathcal{M}^1$. These new images ($\mathcal{I}^1$) show the deformation continuously over Ω i.e., the brain. Figure 4.20 summarizes the procedure.

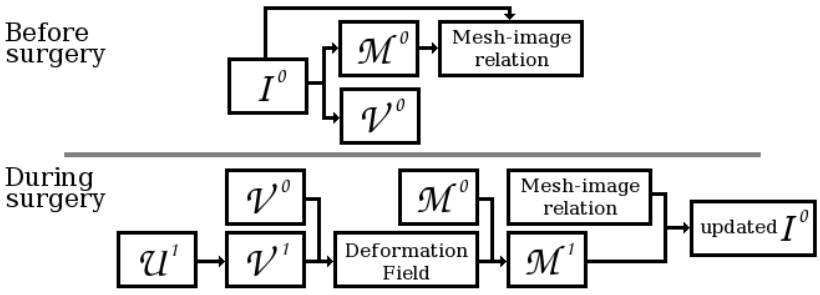

Figure 4.20: Schema to simulate the brain-shift. Where $\mathcal{I}$ represents MRIs, $\mathcal{V}$ represents vascular trees, $\mathcal{M}$ represents meshes and $\mathcal{U}$ represents US Doppler images.

Chapter 5

Conclusions and discussion

5.1 Conclusions

5.1.1 Mesh reparation

A mesh reparation method based on the Jacobian Matrix (subsection 3.2.1) and the Warping Factor (subsection 3.2.2) was implemented in this thesis. The strategy allows to take as independent stages, the reparation to achieve validity and the reparation to achieve quality in the mesh. The reallocation of the nodes to repair the mesh follows a numerical iterative approach, where a function F (defined for each node i), that describes the "quality" of the element in terms of the node current position, is searched to be maximized. This function is constructed regarding the Jacobian matrix determinant (det**J**) and the Warping Factor (WF) for the case of achieving validity in the mesh (subsection 3.2.4). In the case of improving the mesh quality, F is defined in function of the Jacobian ratio (Jac^{ratio}) and the WF (subsection 3.2.4). A valid element for the Finite Element Method (FEM), following the definition of ANSYS®, is the one that counts with det**J** > 0 and a good quality element is the one that has $Jac^{ratio} < 30$. As our solution is an iterative approach, the valid reparation process stops when a first valid stage of the elements is achieved. The same occurs with the quality reparation; the process stops the first time the elements achieve a $Jac^{ratio} < 30$. In the other hand, the WF is a quality measure defined regarding the level of co-planarity of the face. Including the WF in F searches to produce more planar faces in the mesh. Note that no threshold is searched to be reached for the WF as the main goal is to repair the elements regarding the det**J** function. If a strong constraint is used to repair the WF, it might be possible to decrease the chances of finding a solution for the det**J** function.

In the literature, the work of Luboz *et al* in [44] is probably the most similar approach to our work. Luboz proposed the use of the Jacobian matrix to repair

the elements after registration. However no reparation over quality was considered ($Jac^{ratio} < 30$ or the WF). Another difference is that Luboz used the $\nabla \det \mathbf{J}$ to find the direction of displacement and therefore, an analytical solution was found.

The other relevant work, was introduced by Li and Freitag in [38]. This work uses an optimization approach to repair the hexahedra elements in a mesh. The algorithm stops when *the best quality possible to the system* is achieved. This is probably the best solution to the problem of achieving validity and quality in a mesh, however the strategy should be modified to constrain or penalize the displacement of mesh surface nodes.

The examples shown in [38] are basically rectangular and hexahedral grids that are manually distorted and then the algorithm improves the quality (coming back to the initial grid state). If this algorithm is directly applied over a registered mesh, it would probably come back to the initial state: the *atlas* (instead of repairing the registered mesh to be acceptable by the parameters of ANSYS®). Remind that the goal of the reparation algorithm is to achieve a quality mesh by modifying as less as possible the registered mesh. Another modification to the work of Li and Freitag should be to consider the reparation of mixed-element meshes, as in many cases, the *atlas* is not a "hexahedra-only" mesh.

Our implemented solution achieved the reparation of five femurs (section 4.1) and produced a valid mesh for a human face (section 4.2). In this last case, the quality improvement didn't achieve the desired results (all elements in the mesh with $Jac^{ratio} < 30$). The quality improvement was not achieved for two elements. Despite this, the mesh is still usable for FE analysis, as all the elements in the mesh are valid. Moreover, the quality improvement algorithm repaired 11 elements. Note that these eleven elements wouldn't be repaired by the algorithm of Luboz. In the other hand, the algorithm of Li and Freitag might repair the entire mesh, but there is no proof the registered mesh will continue to represent the target domain. For all of these reasons, we think we can say that our implemented approach is probably the most adequate to the problem of repairing meshes after registration.

5.1.2 Meshing domains with a particular region of interest

An algorithm to produce meshes with a particular Region of Interest (RoI) was also proposed in this thesis. It is based on the octree technique. In the case of a normal use of this technique, the octree starts with an hexahedron that covers the entire domain (Ω) to mesh. The process iteratively splits the hexahedron in eight new hexahedra. If an hexahedron is completely outside Ω, it is removed from the mesh. If an hexahedron is completely inside Ω, it is no longer split. In consequence, only the hexahedra that intersect the surface of Ω are split until a certain condition is achieved. Our method proposes to modify the usual octree technique to use as less nodes as possible and also, it searches to count with more

elements in the region where the focus is. Therefore the first modification to the octree technique is that only hexahedra that intersect or are completely inside the RoI are split. The algorithm stops when a certain number of nodes (given by the user) is reached in the mesh. Then, a property called one-irregular is applied over the entire mesh. This property consists in allowing a maximum of one node inserted in an edge or face center. With the one-irregular property over the entire mesh it is possible to use some templates that by the introduction of tetrahedra, pyramids and prisms, allow to achieve a congruent mesh (i.e. elements without nodes inserted in their edges or faces). The last step of the algorithm is to achieve surface representation of Ω. Two options were studied:

- Closest projection of the nodes that reside outside Ω.
- Use the FEM to "compress" the entire mesh into Ω.

The first option produced elements of unacceptable quality. The second strategy produced better elements and therefore, was preferred over the direct projection strategy. The entire process is explained in section 3.3.

In section 4.3, our developed strategy was applied to the problem known as the "brain-shift" in the context of neurosurgery. The idea is to simulate brain deformation during tumor resection surgery. An approach to solve this problem was introduced by Chrisochoides *et al* in [16]. In this approach, they used a regular tetrahedra mesh produced with the red-green meshing technique (Molino *et al* in [47]). The differences between the mesh used by Chrisochoides and our approach are: the use of only tetrahedra and an equally refined mesh over the entire domain. From the simulation point of view, there were also some differences as they used a parallel algorithm to produce the simulation of the brain-shift. Moreover, they proposed the use of intra-operative MRIs to check the brain deformation level and with this information, update the FE model to perform the tracking of the tumor. Even though, from theory, this is a good solution, it is to our consideration inapplicable, as the use of intra-operative MRIs is not considered as an option for the majority of hospitals.

From the meshing point of view, a very good alternative is the type of meshes produced by Frey and George in [29]. They can produce tetrahedral meshes with region refinement, good surface representation and excellent element quality. Probably the most relevant point to differentiate this approach to our developed technique, is the use of mixed-elements. The study performed by Benzley in [6] is regularly cited in the mechanical field as a reference to prefer hexahedra over tetrahedra for the FEM. Our technique tends to produce hexahedra in the RoI, therefore if hexahedra elements are important to the simulation, our implemented solution might be better, as the study of Benzley compared tetrahedra-only and

hexahedra-only meshes, but no mixed-element meshes were studied in the comparison. In the other hand, our technique has shown that some quality problems can arise in the surface elements. Even with the use of FEM to achieve surface representation, a few elements presented poor quality in the final mesh (this is not the case for internal elements, which presented high quality). Probably the only drawback of our current implementation is the problem with quality elements at the surface. Despite this, our technique can produce meshes from any input surface with any local region refinement. Note that it can also produce an equally refined mesh if the RoI covers the entire domain.

5.2 Discussion

5.2.1 Improvements over the mesh reparation

The only one detected limitation to our "reparation after registration" algorithm is that the solution is searched by only the displacement of nodes that make elements invalid or of poor quality. Sometimes after registration, the nodes are so close one from another that the "bad" node has a very small field of displacement to improve the Jac^{ratio} function of the elements as desired. In those cases it is possible to not find a solution for the node in terms of reparation.

A first alternative to solve this problem is by performing a relaxation over the nodes in the region where the problem is detected. In that manner the "bad" node could gain more space to find a new position in which the invalid or poor quality elements achieve the desired level of quality. The problem with this approach is that it could expand the relaxation through an important region or eventually, through the entire mesh. This could result in a lose of representation of the target domain. Probably, for most of the cases, a local relaxation over the nodes would solve the problem; unfortunately it is not possible to ensure a quality representative mesh of the target domain.

A second alternative is to change the strategy of registration and then reparation. When the reparation is performed after registration, the chances of finding a solution are limited. A better strategy would be to not consider the registration and reparation algorithms as different stages, but as a whole. The registration can be performed iteratively and therefore, at each iteration invalid and poor quality elements could be repaired. If at a given iteration, the elements cannot be repaired, it could be possible to go back, increase the stiffness of the unrepairable elements and continue. In the worst case, the elements will finish the registration process with the last acceptable quality found.

5.2.2 Improvements to the meshing technique with a particular region of interest

The main problem with our developed octree-based technique for meshing with a focus over a particular Region of Interest (RoI) is the production of poor quality elements at the surface of the domain. As mentioned before, the direct projection technique produced unacceptable elements at the surface. The use of FEM to achieve the surface representation, improved the quality of those elements to be acceptable, but in some few cases, still of poor quality. The logical solution would be the use of reparation algorithms to improve the quality of those surface tetrahedra of poor quality. Even our proposed solution for the reparation of mixed-element meshes after registration could be used.

Another solution would be to produce a mesh that is closer to the surface of the domain to represent. Figure 5.1 shows at the left panel an example of output mesh (in 2D) from the current stage of the algorithm. At the right panel, some elements that intersect the surface have been changed in order to better represent the surface of the domain.

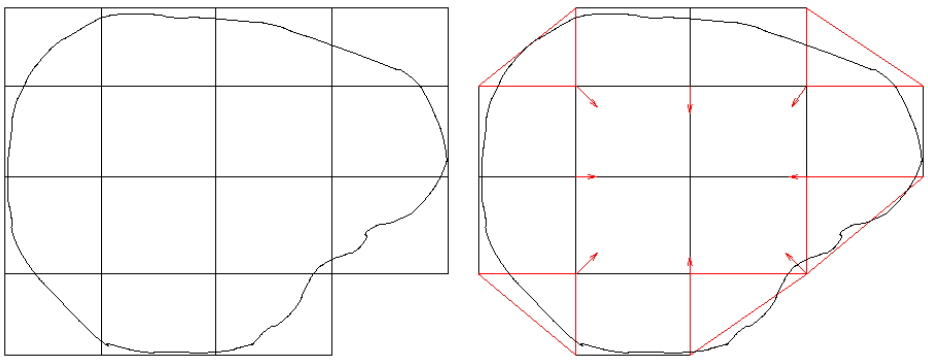

Figure 5.1: Improving the approximation of the surface for our developed octree-based technique.

Note that in 3D this is not a simple problem. Moreover, our developed technique produces mixed-element meshes, therefore it would be necessary to recognize configuration patterns for hexahedra, prisms and pyramids, to change their topology and transform them into other type of elements that achieve better surface representation.

With this “closer to surface target” mesh, both the registration over the surface process and the direct projection approach should be restudied. Note that the direct projection approach, even with this new improved mesh, might still produce sliver elements as only the outside nodes are reallocated over the surface of Ω. In order to avoid this, connected internal nodes could suffer a displacement of the

50% of the outside nodes, following a direction that corresponds the to sum of displacement vectors (right panel of figure 5.1). In this manner, it would be produced an artificial compression of the mesh. In the other hand, the registration approach already deals with this problem by compressing the entire mesh, therefore the motivation to improve the direct projection method is that the registration process sometimes produces important displacements in the internal region. These displacements could be also presented in the RoI and therefore, lose node population in the region where the focus of the simulation is. It is for this reason that a direct projection method, considering the improvements presented here, would probably be a better solution for the problem.

Note that improvements could also be done for the registration alternative: for instance, to constrain the nodes that were produced in the RoI to remain in the RoI after the surface registration process. For this purpose, a double registration should be performed: one to achieve surface representation and another to achieve "internal surface representation" of the RoI. With these improvements, both techniques, registration and direct projection, should be compared in a formal study to determine which one is the best option.

5.3 Publications

The following publications resulted from the work of this thesis:

- **Lobos C.**, Bucki M., Hitschfeld N., Payan Y., **Mixed-element mesh for an intra-operative modeling of the brain tumor extraction**, *Proceedings of the 16th International Meshing Roundtable*, October 2007, published by Springer, Eds.: M. Bewer and D. Marcum, pp: 387–404, [40].

- Bucki M., **Lobos C.**, Payan Y., **Framework for a Low-Cost Intra-Operative Image-Guided Neuronavigator Including Brain Shift Compensation**, *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE EMBC'07)*, 2007, pp: 872–875, [11].

- Bucki M., **Lobos C.** and Payan Y., **Bio-Mechanical Model of the Brain for a Per-Operative Image-Guided Neuronavigator Compensating for "Brain-Shift" Deformations**, *Computer Methods in Biomechanics and Biomedical Egineering*, Issue 1, 2007, pp: 25–26, [10].

- **Lobos C.**, Bucki M., Payan Y. and Hitschfeld N., **Techniques on mesh generation for the brain shift simulation**, *Proceedings of the IV Latin American Congress on Biomedical Engineering*, CLAIB, 2007, IFMBE Proceedings 18, pp: 642–645, [41].

- Nazari M., Payan Y., Perrier Y., Chabanas M. and **Lobos C.**, **A continuous biomechanical model of the face: a study of muscle coordinations for speech lip gestures**. *Proceedings of the eighth International Seminar on Speech Production*, ISSP'08, pp: 321–324 [50].

A book chapter, to be published, resulted from the merge of chapters one and two of this thesis. The reference is:

- Lobos C., Payan Y. and Hitschfeld N., **Techniques for the generation of 3D Finite Element Meshes of human organs**, *Research on Dental Computing and Applications: Advanced Techniques for Clinical Dentistry*.

At least two publications are in the perspectives of the reparation of meshes after registration strategy presented in this thesis. One to explain the algorithm, with a possible comparison between the developed technique and the proposed improvement to perform the matching and reparation at the same time. The other publication corresponds to the results of applying our technique to 45 patient face[1].

Another important publication could be made regarding the improvements presented to the octree-based technique plus a comparison with the direct projection strategy and the registration process.

[1]Extracted from the database of MAP5 Laboratory – Université Paris 5 – UMR CNRS 8145. Thanks to Yves Rozenholc.

Conclusion et discussion en Français

Conclusion

Sur la réparation des maillages

Une méthode de réparation des maillages basée sur la matrice jacobienne (sous–section 3.2.1) et le *Warping Factor* (sous–section 3.2.2) a été développée dans cette thèse. La stratégie considère deux étapes indépendantes; la première produit un maillage valide et la deuxième répare la qualité des éléments. Les deux étapes considèrent un déplacement des sommets. Ces déplacements sont guidés par une méthode numérique itérative, dont une fonction F (définie pour chaque sommet i), qui représente la "qualité" de l'élément, doit être maximisée. Dans le cas de la validité du maillage (sous–section 3.2.4), la fonction est construite en considérant la valeur du déterminant de la matrice jacobienne (det**J**) et le *Warping Factor* (WF). Dans le cas de l'amélioration de la qualité, la fonction F est construite à partir du *Jacobian ratio* (Jac^{ratio}) et du WF (sous–section 3.2.4). En suivant la définition donnée par ANSYS®, un élément est considéré comme valide pour la Méthode des Eléments Finis (MEF), s'il possède un det**J** > 0. Il est considéré de bonne qualité s'il possède un $Jac^{ratio} < 30$. L'algorithme est itératif et par conséquent, il s'arrête dés qu'une solution est trouvée (dont tous les éléments ont un det**J** > 0); pour la qualité, la procédure suit le même principe : l'algorithme s'arrête dès que tous les éléments ont un $Jac^{ratio} < 30$. Le WF est une mesure de qualité définie en fonction de la co–planarité des sommets d'une face. L'inclusion du WF dans la fonction F est à même de produire des faces plus aplaties dans le maillage. Notons qu'aucune valeur du WF n'est cherchée à être optimisée car le but principal est de réparer les éléments en fonction du det**J**. En effet, si le WF avait plus d'importance sur F, il serait alors possible de réduire les chances de trouver une solution pour la fonction det**J**.

Dans la littérature, le travail de Luboz *et al* [44] est probablement la stratégie la plus proche de notre algorithme. Luboz a proposé l'utilisation de la matrice jacobienne pour réparer les éléments après une procédure de recalage. Cependant, Luboz n'a pris en compte aucune considération pour l'amélioration de la qualité

des éléments ($Jac^{ratio} < 30$ ou le WF). Une autre différence avec notre travail est que Luboz a utilisé le $\nabla \det \mathbf{J}$ pour trouver la direction de déplacement des sommets pour les "mauvais" éléments et en conséquence, une procédure analytique est suivie.

Un autre travail pertinent, sur la réparation des éléments a été développé par Li and Freitag [38]. Ce travail utilise une stratégie d'optimisation pour réparer les hexaèdres dans le maillage. L'algorithme s'arrête quand *la meilleure qualité possible pour le système* est trouvée. Ce travail est probablement la meilleure solution pour produire un maillage valide et de bonne qualité. Par contre cette stratégie devrait être modifiée pour pénaliser le déplacement des sommet à la surface dans notre cadre d'application. Sans cette contrainte, le maillage après recalage peut en effet, ne plus représenter correctement le domaine d'étude.

Les exemples montrés dans [38] sont des maillages qui ne considèrent que des hexaèdres. Ceux-ci sont aléatoirement modifiés et en suite, l'algorithme cherche a retrouver l'état initial. Si cet algorithme est appliqué directement sur le maillage recalé, il est même possible de revenir à l'état initial (c'est-à-dire l'atlas). Rappelons que le but de notre algorithme est de réparer le maillage (validité et qualité) sans produire des gros déplacements des sommets. Une autre considération pour l'algorithme de Li et Freitag serait d'étudier des maillages avec des éléments mixtes, parce que, dans la plupart des cas, l'atlas est construit avec plusieurs types d'éléments.

Notre algorithme a réussi à bien réparer cinq fémurs (section 4.1) et a également trouvé un état valide pour un visage humain après recalage (section 4.2). Dans le dernier cas, l'amélioration de la qualité est plus difficile à obtenir (i.e. un maillage dont tous les éléments ont un $Jac^{ratio} < 30$). A la fin de la procédure, deux éléments avaient une mauvaise qualité. Même avec ces deux mauvais éléments le maillage est encore utilisable par la MEF, parce que tous les éléments sont valides. De plus, la qualité de onze éléments a été améliorée. Notons que ces onze éléments n'auraient pas été considérés par l'algorithme de Luboz. D'un autre coté, l'algorithme de Li et Freitag pourrait (éventuellement) tout reparer, par contre rien n'assure que la solution trouvée soit encore représentative du domaine (après recalage). Pour toutes ces raisons, nous pensons que notre solution est la mieux adaptée à la reparation des éléments après une procédure de recalage.

Production des maillages avec une région d'intérêt particulière

Un algorithme pour produire des maillages avec une région d'intérêt (RDI) a également été proposé dans cette thèse. Il est basé sur la technique *octree*. Dans le cas d'une utilisation normale de cette technique, l'octree commence par un hexaèdre qui couvre l'ensemble du domaine (Ω) à mailler. La procédure divise itérativement chaque hexaèdre en huit nouveaux hexaèdres. Si un hexaèdre est complète-

ment en dehors de Ω, il est retiré du maillage. Si un hexaèdre est complètement dans Ω, il n'est plus divisé. En conséquence, seuls les hexaèdres qui intersectent la surface Ω seront scindés jusqu'à ce qu'une certaine condition soit remplie. Notre méthode propose de modifier la technique d'octree pour produire le moins de sommets possibles tout en maximisant le nombre d'éléments dans la RDI du modèle. Par conséquent, la première modification à la technique octree est que seuls les hexaèdres qui intersectent la RDI seront divisés. L'algorithme s'arrête dès que le nombre des sommets voulus est atteint dans le maillage. Ensuite, une propriété appelée *un-irrégulier* est appliquée sur l'ensemble du maillage. Cette propriété permet au maximum un sommet sur une arête ou au centre d'une face. Avec la propriété un-irrégulier il est alors possible d'utiliser certains modèles permettant d'atteindre un maillage régulier (c'est-à-dire, des éléments sans sommets insérés ni dans leurs arêtes ni dans leurs faces) tout en inserant divers tétraèdres, prismes et pyramides. L'objectif de la dernière étape de l'algorithme est de traiter les éléments intersectant la surface Ω. Deux options ont été étudiées:

- La projection directe des sommets situés en dehors de la surface Ω.
- L'utilisation de la MEF pour compresser le maillage entier sur Ω.

La première option a produit des éléments de mauvaise qualité. La deuxième stratégie a produit de meilleurs éléments et, par conséquent, a été préférée. L'ensemble de la procédure est expliqué dans la section 3.3.

Dans la section 4.3, notre stratégie a été appliquée au problème connu sous le nom de "brain-shift" dans le cadre de la neurochirurgie. L'idée est de simuler la déformation du cerveau au cours de la chirurgie de résection tumorale. Une approche pour résoudre ce problème a été présenté par Chrisochoides *et al* [16]. Dans cette approche, ils ont utilisé un maillage des tétraèdres obtenu avec la technique de maillage *red-green* (Molino *et al* [47]). Les différences entre le maillage utilisé par Chrisochoides et notre approche se situent au niveau des éléments utilisés (tétraèdres *vs* maillage mixte) et au niveau du raffinement (uniforme *vs* ciblé). D'autre part, au niveau de la simulation, Chrisochoides utilise un algorithme parallèle ainsi que l'imagerie par résonance magnétique (IRM) intra-opératoire pour vérifier le niveau de déformation du cerveau (avec cette information, ils font la mise à jour du modÃ¨le éléments finis pour effectuer le suivi de la tumeur). Même si d'un point de vue théorique, il s'agît d'une bonne solution, en practique elle semble inapplicable. En effet l'utilisation d'un dispositif IRM intra-opératoire n'est pas considérée comme une option raisonnable pour la plupart des hôpitaux.

Concernant le maillage, une très bonne alternative est la technique de génération des maillages par Frey et George [29]. Ils peuvent produire des maillages de tétraèdres de haute qualité, tout en ayant des régions plus raffinées que les autres, le tout permettant une bonne représentation de la surface. Sans doute, la différence

la plus pertinente avec notre approche est l'utilisation des éléments mixtes dans notre cas. L'étude réalisée par Benzley [6] est une référence, régulièrement citée dans le domaine de la mécanique. Elle compare les maillages uniquement composés de tétraèdres et ceux composé d'hexaèdres. Par contre les maillages à base d'éléments mixtes n'ont pas été considerés dans la comparaison. Dans sa conclusion Benzley préfère les hexaèdres aux tétraèdres pour la MEF. Notre technique a tendance à produire plus d'hexaèdres dans la RDI ce qui est donc un point positif selon Benzley *et al*.

D'un autre coté, notre technique a montré que certains problèmes de qualité peuvent se poser dans les éléments de surface. Malgré l'utilisation de la MEF, quelques éléments à la surface de Ω sont de mauvaise qualité dans le maillage final (ce n'est pas le cas pour les éléments internes). Il s'agît probablement du seul inconvénient de notre application. Malgré cela, notre technique peut produire des maillages à partir de n'importe quelle surface et de n'importe quelle région d'intérêt. Il convient de noter que notre technique peut produire un maillage uniformément raffiné si la RDI couvre l'ensemble du domaine.

Discussion

Améliorations sur la réparation des maillages

La seule limite identifiée pour notre algorithme de "réparation après recalage" est que la solution n'est trouvée que par le déplacement des sommets qui appartiennent aux éléments invalides ou de mauvaise qualité. Parfois, après recalage, les sommets sont si proches les uns des autres que les " mauvais" sommets ont un très faible champ de déplacement possible pour améliorer la fonction du Jac^{ratio}. Dans ces cas, il se peut qu'aucune solution ne soit trouvée pour ces sommet.

Une première solution pour résoudre ce problème est d'exécuter une "relaxation" sur les sommets dans la région où le problème est détecté. De cette manière le "mauvais" sommet pourrait obtenir plus d'espace pour trouver une nouvelle position dans laquelle l'invalidité ou la mauvaise qualité des éléments soient réparées. Le problème avec cette approche est qu'elle pourrait élargir la relaxation dans une région importante voire éventuellement la totalité du maillage. Dans ce cas, la relaxation pourrait déclencher une mauvaise représentation du domaine cible. Probablement, pour la plupart des cas, une détente locale sur les sommets pourrait résoudre les problèmes; par contre il n'est malheureusement pas possible de garantir un maillage de qualité représentatif de l'ensemble domaine cible.

Une deuxième solution est de changer la stratégie recalage. Lorsque la réparation est effectuée après recalage, les chances de trouver une solution sont limitées. Une meilleure stratégie serait de ne pas envisager le recalage et la réparation

comme des algorithmes séquentiels, mais comme un ensemble. Le recalage peut être effectué itérativement et, par conséquent, à chaque itération un algorithme de réparation des éléments peux être appliqué. Si à certains moments, les éléments ne peuvent pas être réparés, il pourrait être possible de revenir en arrière, augmenter la rigidité des éléments irréparables et reprendre le recalage. Dans le pire des cas, les "mauvais" éléments termineront le processus de recalage avec la dernière qualité acceptable trouvée dans une certaine itération.

Améliorations sur la production des maillages avec une région d'intérêt particulière

Le principal problème de notre technique de production de maillages avec région d'intérêt particulière (RDI) est l'obtention d'éléments de mauvaise qualité à la surface du domaine (Ω). Nous avons proposé une autre technique à partir de l'utilisation de la MEF pour atteindre la représentation de la surface de Ω. Cette technique a montré qu'une meilleure qualité est produite dans l'ensemble du maillage. Par contre une forte qualité ne peux pas être assure partout, surtout, au niveau des élément qu'intersectent la surface de Ω. La solution logique serait l'utilisation des algorithmes de réparation de qualité dans les tétraèdres qui restaient à la surface avec une qualité médiocre. Même notre solution proposée pour la réparation des maillages d'éléments mixtes après recalage pourrait être utilisés.

Une autre solution serait de produire un maillage qui est plus proche de la surface du domaine à représenter. La figure 5.2 montre un exemple en 2D dans lequel il est possible de visualiser (à gauche) l'état actuel du maillage. A droite, certains éléments qui intersectent la surface ont été modifiés afin de mieux représenter la surface du domaine.

Notez qu'en 3D ce n'est pas un problème simple. De plus, notre technique produit des maillages à base d'éléments mixtes; il serait donc nécessaire de reconnaître différents schémas de configuration pour des hexaèdres, prismes et pyramides, pour ensuite modifier leurs topologies et de les transformer en d'autres types d'éléments qui permettent d'atteindre une meilleure représentation de la surface.

Grâce à ce maillage "plus proche de la surface cible", les deux algorithmes (la projection directe et l'utilisation de la MEF pour représenter la surface) devraient être réexaminés. Notons que l'approche de projection directe, même avec une amélioration du maillage, peut encore produire des éléments aplatis à la surface puisque seuls les sommets en dehors de la surface seront poussés vers celle-ci. Afin d'éviter cela, les sommets internes qui forment une arête avec les sommets en dehors de la surface pourraient poursuivre 50% du déplacement reçu par les sommets à l'extérieur. Par la suite, les sommets connectés a ces derniers pourraient

également recevoir 50% du déplacement de ceux-ci derniers, puis etc. (panneau de droite de la figure 5.2). De cette manière, il serait possible de reproduire artificiellement la compression du maillage. En revanche, notons que le recalage a déjà traité ce problème par une compression de l'ensemble du maillage.

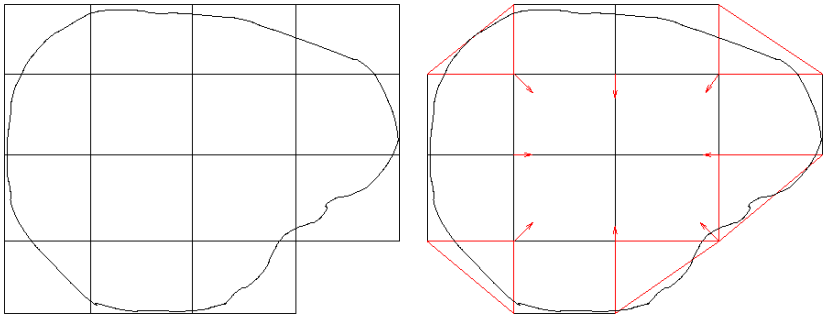

Figure 5.2: Technique d'amélioration de la rerprésentation de la surface pour notre algorithme.

La motivation pour améliorer la méthode de projection directe est que le processus de recalage a parfois produit des déplacements internes importants. Ces déplacements peuvent être également présentés dans la RDI et donc perdre une partie de la population des sommet initialement dans la RDI. C'est pour cette raison que la méthode de projection directe, avec les améliorations présentées ici, serait probablement une meilleure solution pour ce problème.

Notons que des améliorations pourraient aussi être faites pour le recalage : par exemple, une contrainte sur les sommets de la RDI pour y rester après recalage. A cette fin, un double recalage doi être effectué : un pour parvenir à la représentation de surface et un autre pour parvenir à la RDI. Grâce à ces améliorations, les deux techniques (le recalage et la projection directe) devraient être à nouveau comparées dans une étude ultérieure.

Publications

Les publications suivantes ont été produites dans cette thèse:

- **Lobos C.**, Bucki M., Hitschfeld N., Payan Y., **Mixed-element mesh for an intra-operative modeling of the brain tumor extraction**, *Proceedings of the 16th International Meshing Roundtable*, October 2007, published by Springer, Eds.: M. Bewer and D. Marcum, pp: 387–404, [40].

- Bucki M., **Lobos C.**, Payan Y., **Framework for a Low-Cost Intra-Operative Image-Guided Neuronavigator Including Brain Shift Compensation**, *Pro-*

ceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE EMBC'07), 2007, pp: 872–875, [11].

- Bucki M., **Lobos C.** and Payan Y., **Bio-Mechanical Model of the Brain for a Per-Operative Image-Guided Neuronavigator Compensating for "Brain-Shift" Deformations**, *Computer Methods in Biomechanics and Biomedical Egineering*, Issue 1, 2007, pp: 25–26, [10].

- **Lobos C.**, Bucki M., Payan Y. and Hitschfeld N., **Techniques on mesh generation for the brain shift simulation**, *Proceedings of the IV Latin American Congress on Biomedical Engineering*, CLAIB, 2007, IFMBE Proceedings 18, pp: 642–645, [41].

- Nazari M., Payan Y., Perrier Y., Chabanas M. and **Lobos C.**, **A continuous biomechanical model of the face: a study of muscle coordinations for speech lip gestures**. *Proceedings of the eighth International Seminar on Speech Production*, ISSP'08, pp: 321–324 [50].

Un chapitre de livre, doit être publié, résultant des chapitres un et deux de cette thèse. La référence est:

- Lobos C., Payan Y. and Hitschfeld N., **Techniques for the generation of 3D Finite Element Meshes of human organs**, *Research on Dental Computing and Applications: Advanced Techniques for Clinical Dentistry*.

Au moins deux publications sont dans nos perspectives pour la réparation de maillages après recalage. L'un en expliquant l'algorithme, avec une comparaison entre l'application développée dans cette thèse et la technique qui considère nos proposition pour effectuer le recalage et la réparation du maillage en même temps. L'autre publication correspond aux résultats de l'application de notre technique aux modèles des faces de 45 patients[2].

Une autre publication importante pourra être faite en ce qui concerne les améliorations présentées sur la technique basé sur l'octree et une comparaison entre la projection directe et la stratégie de recalage.

[2]Extraites de la base de données de laboratoire MAP5-Université Paris 5 - UMR CNRS 8145. Merci à Yves Rozenholc.

Bibliography

[1] F. Abell, M. Krams, J. Ashburner, R. Passingham, K. Friston, R. Frackowiak, F. Happe, C. Frith, and U. Frith. The neuroanatomy of autism: a voxel-based whole brain analysis of structural scans. *Neuroreport*, 10(8):1647–1651, June 1999.

[2] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics*, 24(3):617–625, 2005.

[3] R. Araya, G. Barrenechea, F. Galdames, F. Jaillet, and R. Rodriguez. Adaptive mesh and finite element analysis of coupled fluid/structure: application to brain deformations. In *Proceedings of the Third International Conference Surgetica*, pages 117–121, September 2007.

[4] J. Ashburner and K. J. Friston. Voxel-based morphometry–the methods. *Neuroimage*, 11(6 Pt 1):805–821, June 2000.

[5] M. A. Audette, H. Delingette, A. Fuchs, O. Burgert, and K. Chinzei. A topologically faithful, tissue-guided, spatially varying meshing strategy for computing patient-specific head models for endoscopic pituitary surgery simulation. *Computer Aided Surgery*, 12(1):43–52, January 2007.

[6] S. E. Benzley, E. Perry, K. Merkley, B. Clark, and G. D. Sjaardema. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings, 4 th International Meshing Roundtable*, pages 179–191. Sandia National Laboratories, Springer-Verlag, 1995.

[7] M. Berar, M. Desvignes, G. Bailly, and Y. Payan. 3d meshes registration : Application to statistical skull model. In *Image Analysis and Recognition*, volume 3212 of *Lecture Notes in Computer Science*, pages 100–107. Springer Berlin / Heidelberg, Septmeber 2004.

[8] J. Bonet and R. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997.

[9] M. Bucki. *Modelisation Biomechanique des Tissus Mous du Cerveau et Developpement d'un Neuronavigateur Permettant la Prise en Compte Per-Operatoire du Brain-Shift*. PhD thesis, Universite Joseph Fourier, 2008.

[10] M. Bucki, C. Lobos, and Y. Payan. Bio-mechanical model of the brain for a per-operative image-guided neuronavigator compensating for "brain-shift" deformations. In *Computer Methods in Biomechanics and Biomedical Egineering*, volume 1, pages 25–26, 2007.

[11] M. Bucki, C. Lobos, and Y. Payan. Framework for a low-cost intra-operative image-guided neuronavigator including brain shift compensation. In *Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE EMBC'07)*, pages 872–875., 2007.

[12] G. Bunin. Non-local topological clean-up. In Philippe P. PÃ©bay, editor, *Proceedings, 15 th International Meshing Roundtable*, pages 3–20. Sandia National Laboratories, Springer-Verlag, September 2006. 2d.

[13] A. Castellano-Smith, T. Hartkens, J. Schnabel, R. Hose, H. Liu, W. Hall, C. Truwit, D. Hawkes, and D. Hill. Constructing patient specific models for correcting intraoperative brain deformation. In *MICCAI*, pages 1091–1098, 2001.

[14] L. Chen and J. XU. Optimal delaunay triangulations. *Journal of Computational Mathematics*, 22(2):299–308, 2004.

[15] E. Chernyaev. Marching cubes 33: Construccion of topologically correct isosurfaces. Technical report, CN/95-17, CERN, 1995.

[16] N. Chrisochoides, A. Fedorov, A. Kot, N. Archip, P. Black, O. Clatz, A. Golby, R. Kikinis, and S. K. Warfield. Toward real-time image guided neurosurgery using distributed and grid computing. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, pages 12–17. IEEE, ACM, November 2006.

[17] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *ACM Transanctions on Graphics (SIGGRAPH)*, pages 905–914, 2004.

[18] N. Coll, M. Guerrieri, and J. A. Sellares. Mesh modification under local domain changes. In Philippe P. PÃ©bay, editor, *Proceedings, 15 th International Meshing Roundtable*, pages 39–56. Sandia National Laboratories, Springer-Verlag, September 2006. 2d.

[19] S. Cotin, H. Delingette, and N. Ayache. Real time volumetric deformable models for surgery simulation. In *VBC*, pages 535–540, 1996.

[20] B. Couteau, Y. Payan, and S. Lavallee. The mesh-matching algorithm: an automatic 3d mesh generator for finite element structures. *Journal of Biomechanics*, 33(8):1005–1009, 2000.

[21] B. Delaunay. Sur la sphere vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

[22] H. Delingette. General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146, 1999.

[23] A. Fedorov, N. Chrisochoides, R. Kikinis, and S. Warfield. Tetrahedral mesh generation for medical imaging. In *Insight Journal - MICCAI Open Source Workshop*, page n/a, August 2005. Electronic publication.

[24] M. Ferrant, S. Warfield, A. Nabavi, F. Jolesz, and R. Kikinis. Registration of 3d intraoperative mr images of the brain using a finite element biomechanical model. In *Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 19–28. Springer-Verlag, 2000.

[25] M. Ferrant, S. K. Warfield, C. Guttmann, R. Mulkern, F. Jolesz, and R. Kikinis. 3d image matching using a finite element based elastic deformation model. In *Medical Image Computing and Computer–Assisted Intervention (MICCAI)*, Lecture Notes in Computer Science, pages 202–209. Springer-Verlag, 1999.

[26] R. Finkel and J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[27] L. Freitag. On combining laplacian and optimization-based mesh smoothing techniques. In *Proc. Symp. Trends in Unstructured Mesh Generation*, pages 37–43. Amer. Soc. Mechanical Engineers, Jun 1997.

[28] P. Frey. Generation and adaptation of computational surface meshes from discrete anatomical data. *International Journal for Numerical Methods in Engineering*, 60:1049–1074, 2004.

[29] P. Frey and P. George. *Mesh Generation: Applications to Finite Elements*. Hermes, Paris, 2000.

[30] H. Gray. *Gray's Anatomy: The Anatomical Basis of Medicine and Surgery*. Churchill-Livingstone, Elsevier, 40 edition, 2008.

[31] N. Hitschfeld. Generation of 3d mixed element meshes using a flexible refinement approach. *Engineering with Computers*, 21:101–114, 2005.

[32] N. Hitschfeld, G. Navarro, and R. Farias. Tessellations of cuboids with steiner points. In *Proceedings, 9 th International Meshing Roundtable*, pages 275–282. Sandia National Laboratories, Springer-Verlag, 2000.

[33] T. Hughes. *The Finite element method : Linear static and dynamic Finite element analysis.*, pages 292–294. Prentice–Hall, 1987.

[34] B. Joe. Construction of three-dimensional improved-quality triangulations using local transformations. *SIAM J. Sci. Comput.*, 16(6):1292–1307, 1995.

[35] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *ACM SIGGRAPH*, pages 339–346. ACM Press, 2002.

[36] D. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*, chapter 6, pages 458–475. Addison-Wesley, third edition, 1997.

[37] L. Kobbelt, M. Botsch, U. Schwanecke, and H. Seidel. Feature–sensitive surface extraction from volume data. In *ACM SIGGRAPH*, pages 57–66, 2001.

[38] X. Li and L. Freitag. Optimization-based quadrilateral and hexahedral mesh untangling and smoothing techniques. Technical report, Argonne National Laboratory, a contract Laboratory of the United States Department of Energy., 1999.

[39] X. Li, J-F. Remacle, N. Chevaugeon, and M. Shephard. Anisotropic mesh gradation control. In *Proceedings, 13 th International Meshing Roundtable*, pages 401–412. Sandia National Laboratories, Springer-Verlag, 2004.

[40] C. Lobos, M. Bucki, N. Hitschfeld, and Y. Payan. Mixed-element mesh for an intra-operative modeling of the brain tumor extraction. In *Proceedings, 16 th International Meshing Roundtable*, pages 387–404. Sandia National Laboratories, Springer-Verlag, 2007.

[41] C. Lobos, M. Bucki, Y. Payan, and N. Hitschfeld. Techniques on mesh generation for the brain shift simulation. In *Proceedings of the IV Latin American Congress on Biomedical Engineering (CLAIB)*, pages 642–645, 2007.

[42] C. Lobos and N. Hitschfeld. 3d noffset mixed-element mesh generator approach. In *The 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 47–52, 2006.

[43] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[44] V. Luboz, M. Chabanas, P. Swider, and Y. Payan. Orbital and maxillofacial computer aided surgery: patient-specific finite element models to predict surgical outcomes. *Computer methods in biomechanics and biomedical engineering*, 8(4):259–265, August 2005.

[45] V. Luboz, Y. Payan, and B. Couteau. Automatic 3d finite element mesh generation: an atlas fitted onto patient data. In *Proceedings of the Fifth International Symposium on Computer Methods in Biomechanics and Biomedical Engineering*, November 2001.

[46] G. Miller, S. Pave, and N. Walkington. When and why ruppert's algorithm works. In *Proceedings, 12 th International Meshing Roundtable*, pages 91–102. Sandia National Laboratories, Springer-Verlag, September 2003.

[47] N.P. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Proceedings, 12 th International Meshing Roundtable*, pages 103–114. Sandia National Laboratories, Springer-Verlag, 2003.

[48] J. Montagnat and H. Delingette. Globally constrained deformable models for 3d object reconstruction. *Signal Processing (SP)*, 71(2):173–186, 1998.

[49] J. Montagnat and H. Delingette. 4d deformable models with temporal constraints: application to 4d cardiac image segmentation. *Medical Image Analysis (MedIA)*, 9(1):87–100, February 2005.

[50] M. Nazari, Y. Payan, Y. Perrier, M. Chabanas, and C. Lobos. A continuous biomechanical model of the face: a study of muscle coordinations for speech lip gestures. In *Proceedings of the eighth International Seminar on Speech Production, ISSP'08*, 2008.

[51] M. Nesme. *Milieu mecanique deformable multiresolution pour la simulation interactive*. PhD thesis, Universite Joseph Fourier, June 2008.

[52] A. Pages, M. Sermesant, and P. Frey. Generation of computational meshes from mri and ct-scan data. *ESAIM: Proceedings*, 14:213–223, September 2005.

[53] B. Payne and A. Toga. Surface mapping brain function on 3d models. *IEEE Computer Graphics & Applications*, 10(5):33–41, 1990.

[54] G. Picinbono, H. Delingette, and N. Ayache. Non-linear anisotropic elasticity for real-time surgery simulation. *Graphical Models*, 65(5):305–321, 2003.

[55] I. Reinertsen, M. Descoteaux, K. Siddiqi, and D. Collins. Validation of vessel-based registration for correction of brain shift. *Medical Image Analysis*, 11:374–388, 2007.

[56] J. Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[57] J. Schnabel, D. Rueckert, M. Quist, J. Blackall, A. Castellano-Smith, T. Hartkens, G. Penney, W. Hall, H. Liu, C. Truwit, F. Gerritsen, D. Hill, and D. Hawkes. A generic framework for non-rigid registration based on non-uniform multi-level free-form deformations. In *MICCAI*, pages 573–581, 2001.

[58] J. Schwartz, E. Langelier, C. Moisan, and D. Laurendeau. Non-linear soft tissue deformations for the simulation of percutaneous surgeries. In *MICCAI*, volume 2208, pages 1271–1272, 2001.

[59] J. Shewchuk. Lecture notes on delaunay mesh generation. Technical report, University of California, 1999.

[60] J. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, May 2002.

[61] J. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *Proceedings of the 11 International Meshing Roundtable*, pages 115–126. Sandia National Laboratories, Springer-Verlag, 2002.

[62] D. Smythe. A two-pass mesh warping alogrithm for object transformation and image interpolation. Technical Report 1030, ILM Computer Graphics Department, Lucasfilm, San Rafael, Calif, 1990.

[63] F. Velasco, J. Torres, A. León, and F. Soler. Adaptive cube tessellation for topologically correct isosurfaces. In *In proceeding of the Computer Graphics Theory and Applications*, pages 220–227, March 2007.

[64] G. Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8–9):360–372, December 1998.

[65] H. Zhang and G. Zhao. Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method. *Finite Element Analysis and Desing*, 43(9):691–704, 2007.

[66] Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering*, 195(9–12):942–960, February 2006.

Appendix A

Efficient node management in the Octree technique

A.1 Motivation

In the octree technique, each leave is an hexahedron and each component of the octree receives the name of octant. The root of the tree, the first octant, is initialized with the hexahedron resulting from the Bounding Box (BBox) of the input surface domain Ω_s to mesh. Is by splitting this hexahedron that the octree tree is generated.

An optimized data structure must be implemented to save the node's information. In difference with the input geometry that is read, here it is necessary to know if a particular node was already inserted or not. If no checking is made over the nodes, it is possible to insert several times the same coordinates with different references by the elements, producing bad connectivity problems.

Figure A.1 shows an example of index inconsistency. Note how at the top panel of the figure the pairs: (8,1), (10,2), (12,5) and (15,6) have the same coordinates. This problem is solved at the bottom panel of the figure by merging those nodes and updating the element's information. This is the type of problems that the developed node container structure tries to avoid while inserting new nodes in the mesh. The goal of this structure is to avoid inserting (saving) two or more times the same coordinates for the same node.

A.2 Unique node identifier

The class *Node* has been implemented. A *vector* object of the standard $C++$ library stores all the nodes (of type $Node$). If each time a new node must be inserted, the entire vector is checked for repetition, the insertion operation tends

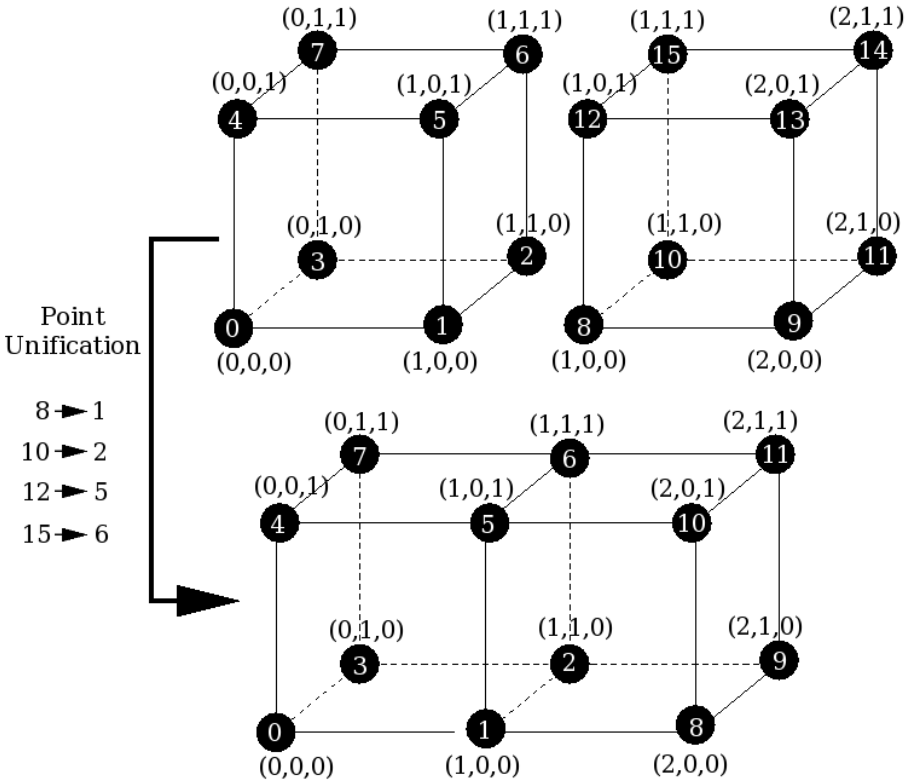

Figure A.1: Top: the right nodes of the first cube and the left ones of the seconds have the same coordinates but different indexes: they are duplicated. Bottom: duplicated nodes have been removed and no data inconsistency is presented.

to be $O(n^2)$, where n is the number of nodes the mesh has. In practice, this means that producing a regular octree mesh (grid) with 9 level of refinements would take more than 30 minutes. With the implemented optimization, the time required to produce this mesh is a couple of seconds.

Node's insertion with the octree technique is always performed at the split process of the octants, therefore this operation will always be performed between two already inserted nodes. The goal of indexing the nodes is to somehow give them a unique identifier.

Let $\bar{AB}$ be one edge of the BBox. Let C be the middle node of $\bar{AB}$. When the octree algorithm splits the BBox in order to produce eight new cubes the edge $\bar{AB}$ is removed and two new edges: $\bar{AC}$ and $\bar{CB}$ replaces it. In terms of coordinates $C = (A + B)/2$ and all the nodes inserted in $\bar{AB}$ will be greater than A and less than B. The key idea is then to give them an identifier that takes advantage of this particular condition in order to easily distinguish the nodes. If a maximum

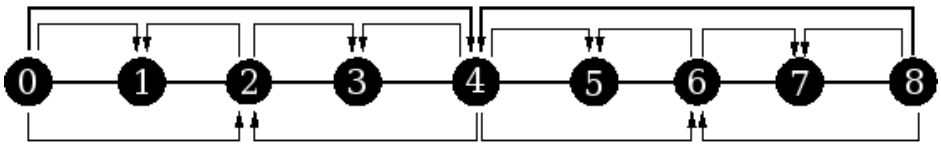

Figure A.2: The nodes in the octree technique are always inserted between two existing nodes. The initial nodes are 0 and 8.

of 9 nodes are allow *per* edge the first node index should be 0 and the last one 8. As the split process produces a node in the middle of the edge, the index for this new node should be 4. Now having 3 nodes and 2 edges, the new nodes should be between 0 and 4 (node 2) and between 4 and 8 (node 6). Following the idea, to obtain a new index it is necessary to add the indexes of the parents and divide by 2 (as to obtain the coordinates of the node). This is shown in figure A.2, where the insertion sequence is the following: 0, 8, 4, 2, 6, 1, 3, 5, 7.

In other words, each node in the octree structure has a predefined index or virtual index v_{idx} regarding the indexes of its parents. The initial indexes are given in the x axis, then the y and finally the z ones. The maximum number of nodes in each edge is set to 513 (from 0 to 512). Therefore the maximum number of nodes in the mesh is $513^3 = 135,005,697$. This can be easily arranged in a global defined vector that contains the amount of nodes allowed *per* axis.

The v_{idx} allows to easily check if a node was inserted or not. For example if the edge (0,64) must be split, the resulting node will have the index 32. If another octant shares the same edge (0,64) and needs to split the edge, it will only be necessary to check if the node with index 32 is already inserted. This avoids the insertion of the same coordinates with different index.

Now that the nodes are indexed, the only problem is to know where to check. As in the split process insertions occurs all the time, this checking should be as fast as possible. If an array containing the status of each v_{idx} is used, it would be a direct operation to check node's previous insertion. Unfortunately, having a boolean array of size $135,005,697$, would be an incredible waste of memory space, specially in cases of small meshes. A structure that evolves regarding the current amount of nodes and with fast access to information would be optimal.

A.3 Saving and sorting the nodes

In this thesis, the only reference to a tree data structure before, is the octree. As showed in figure 2.5, the parent $treenode$ spatially contains his 8 sons. Therefore the only relation between parents and sons is geometrical. Another important use

of trees is to sort data. In particular, a binary search tree (BST) is a structure that have the following properties:

- each $treenode$ (item in the tree) has a value.
- the left subtree of a $treenode$ contains only minor values to $treenode$'s value.
- the right subtree of a $treenode$ contains only values greater than or equal to the $treenode$'s value.

In the best case, insertion and searching information is $O(log\ n)$. In the worst case is $O(n^2)$. The worst case is produced when inserting sorted numbers. In this case the tree is actually a list of numbers.

An AVL tree is a self-balancing binary search tree. A complete description can be found in [36] by Knuth. This type of tree assures that inserting a node takes $O(log\ n)$ and searching a node takes also $O(log\ n)$ all the time. This solution is optimal in terms of memory space and fast enough in terms of insertion and searching.

The height of a tree or a subtree is the number of connection from the root (top $treenode$) to the more distant leaf (bottom $treenode$). The balance factor (b_{factor}) of a $treenode$ is the height of its right subtree minus the height of its left subtree and a $treenode$ with b_{factor} 1, 0, or -1 is considered balanced. A $treenode$ with any other b_{factor} is considered unbalanced and requires rebalancing the tree. The b_{factor} is recalculated when a new node is inserted.

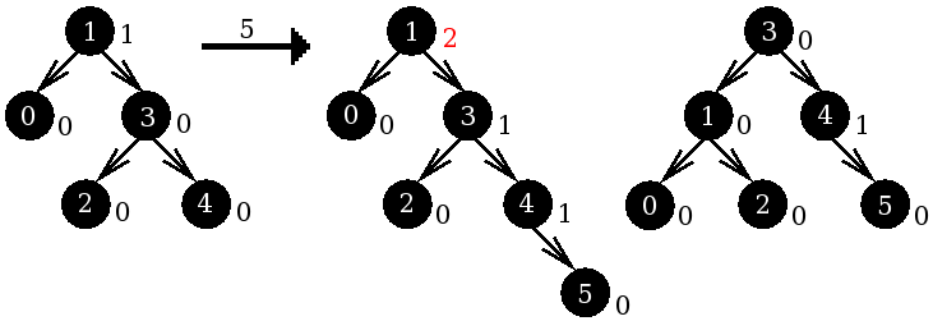

Figure A.3: A balanced tree that becomes unbalanced by the insertion of value 5. At the right, one of the 4 rotation operations rebalance the tree. Next to each $treenode$ is the b_{factor} of it.

Figure A.3 shows one of the rotation operation to regain the balance in the tree. Note that in the middle tree, the b_{factor} of the root becomes invalid as the height of the right subtree is 3 and the left one is 1, therefore its b_{factor} is 2. At the right, the balance is recovered by one of the 4 possible rotations of the nodes.

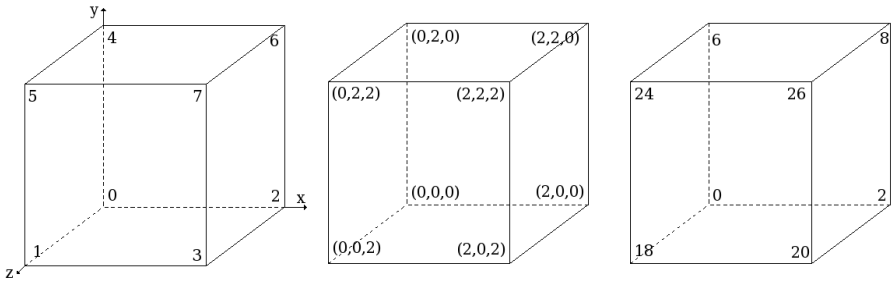

Figure A.4: The different index and coordinates values for the BBox in the case of 3 nodes per axis. Left the index in the node's vector. Middle the actual coordinates of the nodes and right, the v_{idx} associated to each node

A.4 Splitting process in the octree

In order to illustrate the process a domain that has a BBox of (0,0,0) to (2,2,2) is used as an example. Let 3 be the maximum nodes allowed *per* axis. Figure A.4 left, shows the order by which the nodes are stored (the node's index in the node's vector) for the BBox. In the middle, the actual coordinates of the same nodes are shown and at the right, the respective v_{idx} of each node. As those nodes corresponds to the BBox they are directly inserted in the node's vector. The v_{idx} are inserted in an AVL tree that in each $treenode$ has the v_{idx} and the position in the node's vector where the coordinates of the node can be found.

If a node's v_{idx} isn't found in the AVL tree, a new object of type $Node$ containing the coordinates is stored at the end of the node's vector (at position pos). A new $treenode$ is added in the AVL tree regarding the v_{idx} associated to the node. The $treenode$ stores the position pos where the actual node can be found.

Regarding the previous example of figure A.4, the order by which the nodes are inserted in the structures is give by the left figure. Therefore the insertion order of the v_{idx} in the AVL is: 0, 18, 2, 20, 6, 24, 8 and 26 as shown in figure A.4 right.

The evolution of the AVL can be seen in figure A.5. First, the insertion of $treenodes$ 0 and 18 cause no problem as the AVL remains balanced. The first rotation comes with the insertion of value 2. If no rotation is performed, the $treenode$ 2 would be inserted as "the left son" of $treenode$ 18. This last operation would make the tree unbalanced at $treenode$ 0, therefore $treenode$ 2 becomes the root of the tree and the tree regain the balance. After that, the insertion of $treenodes$ 6 and 20 are inserted without rotations. $Treenode$ 24 cause a major rotation of the tree and then, $treenode$ 8 can be inserted without modification to the tree. Finally $treenode$ 26 cause a minimal rotation in the right subtree of the root.

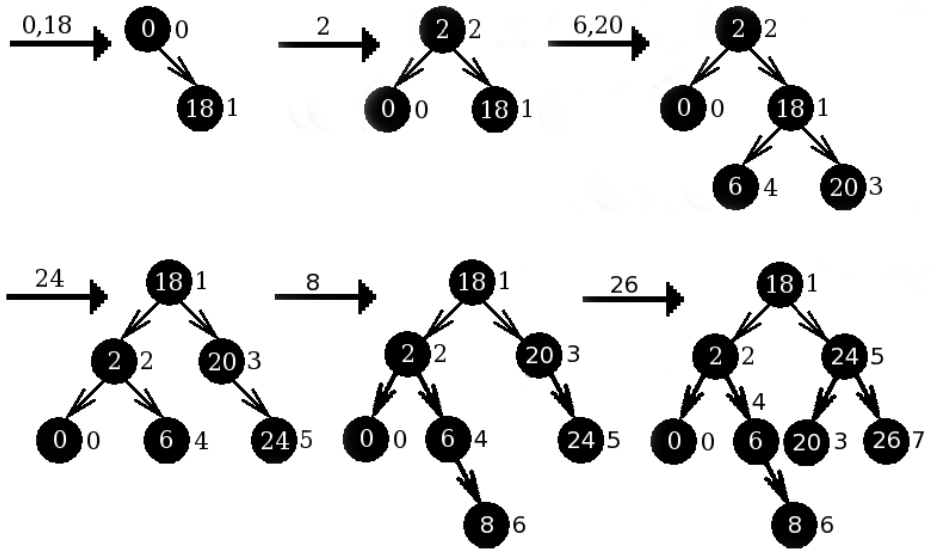

Figure A.5: The evolution of the AVL structure as the nodes of the root octant are inserted (v_{idx}). Next to each node the position reference of each node in the node's vector (real index).

At this point, the tree has all the v_{idx} from the root octant inserted. As all the v_{idx} were directly assigned, it is just now that new v_{idx} will be calculated. For instance, if the edge defined by nodes 2 and 3 (with v_{idx} 2 and 20 respectively) is split, then the new v_{idx} associated to that node is: $(2 + 20)/2 = 11$. The last inserted node was in position 7 of the node container (as eight nodes were inserted) therefore, the coordinates of the new node are inserted in the position 8 of the node container.

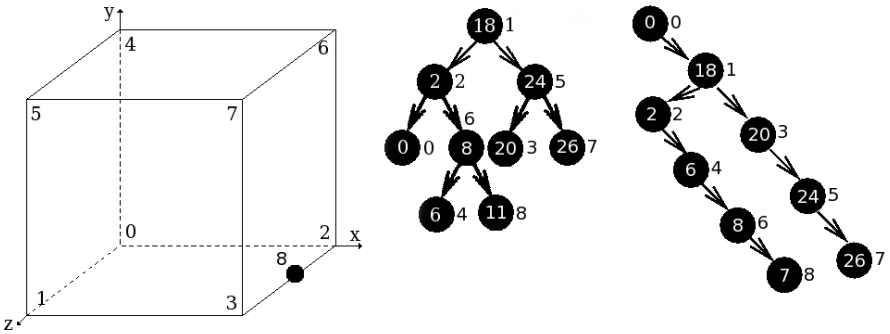

Figure A.6: The comparison between an AVL (middle) and a BST (right) by the insertion of a new node as the cube in the left shows.

Figure A.6 left shows the insertion of a new node between nodes 2 and 3 (real index or position in the node container). The same figure in the middle panel,

shows how the AVL changes with the insertion of the new node ($v_{idx} = 11$ and real index = 8). Finally at the right, the equivalent Binary Search Tree (BST) is presented by inserting the node indexes in the same order as the AVL. As the BST is quite unbalanced, it is easy to see how search and insertion of new $treenodes$ in the tree is not assured to be $O(log\ n)$.