XX(178686.1)

TK165.G8
.G88
2009

Centro de Investigación y de Estudios Avanzados del I.P.N.

Unidad Guadalajara

Cortando Tejido Suave usando el XFEM: una Perspectiva de Cirugía Virtual

Tesis que presenta:
Luis Fernando Gutiérrez Preciado

para obtener el grado de:
Maestro en Ciencias

en especialidad de:
Ingeniería Eléctrica

Director de Tesis:
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, Agosto 2009.

Centro de Investigación y de Estudios Avanzados del I.P.N.

Unidad Guadalajara

Cutting Soft Tissue using the XFEM: a Virtual Surgery Perspective

A thesis presented by:
Luis Fernando Gutiérrez Preciado

to obtain the degree of:
Master in Science

in the subject of:
Electrical Engineering

Thesis Advisors:
Dr. Félix Francisco Ramos Corchado

Guadalajara, Jalisco, August 2009.

Cortando Tejido Suave usando el XFEM: una Perspectiva de Cirugía Virtual

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Luis Fernando Gutiérrez Preciado
Ingeniero en Computación
Universidad Autónoma de Aguascalientes 2002-2007

Becario de CONACYT, expediente no. 213030

Director de Tesis:
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, Agosto, 2009.

Cutting Soft Tissue using the XFEM: a Virtual Surgery Perspective

Master of Science Thesis
In Electrical Engineering

By:
Luis Fernando Gutiérrez Preciado
Engineer in Computer Science
Universidad Autónoma de Aguascalientes 2002-2007

Scholarship granted by CONACYT, No. 213030

Thesis Advisors:
Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, August, 2009.

Resumen

Actualmente, una diversidad de simulaciones y aplicaciones médicas han sido desarrolladas en computación gráfica, como son: el diagnostico médico, prácticas de procedimientos, planeación de cirugías, etc., todos, simulan objetos deformables empleando modelos físicos para obtener resultados confiables, sin embargo, una simulación médica debe ser interactiva y en tiempo real, requiriendo algoritmos precisos y rápidos.

Un procedimiento común en cirugía es cortar tejidos con bisturí, tijeras, entre otras herramientas. Cortar es un trabajo complejo para representarlo mediante una simulación; porque requiere de la modificación de la topología y del remallado del modelo. En la actualidad existen muchos enfoques para cambios en la topología, algunos de estos son animaciones no interactivas, inestables o no lo suficientemente precisos para aplicaciones médicas; comparando los enfoques existentes, se ha encontrado que el XFEM es estable, preciso, con excelente desempeño y adecuado para simular una cirugía virtual en tiempo real; por otro lado, para mantener las ventajas provistas, se requiere de la selección y creación de un conjunto de métodos que concuerden con los requerimientos del XFEM.

Los algoritmos necesarios para crear una simulación de cirugía alrededor del XFEM son: La integración tiempo, técnicas de mayado, detección de colisiones, métodos de mapeo, etc.; por lo tanto, es importante buscar un desempeño excelente en cada uno de estos algoritmos y en la interacción entre ellos, todo esto con miras a una simulación interactiva y en tiempo real. Se ha propuesto un método de mapeo embebido en el control de la topología física, el cual permite interrelacionar los elementos del XFEM con el mayado visual y de colisión, agilizando de esta forma la interacción con el usuario. El XFEM, utilizado como núcleo de los métodos de computación gráfica, ayudó a simular de forma más eficiente modelos delgados como la piel, haciendo a la vez posible la interacción en tiempo real, lo que permitirá realizar simulaciones más complejas y de mayor impacto en el área médica.

Abstract

Nowadays, a diversity of medical simulations and applications have been developed in computer graphics (CG) such as medical diagnosis, procedures training, surgery planning, etc.; all these, simulate deformable objects employing physical models in order to get reliable results; nevertheless, a medical simulation must be interactive and in real time, requiring accurate and fast algorithms.

A common procedure in a surgery is to cut tissues with scalpel, scissors, among other instruments. Cutting is a challenging work to simulate, because it requires the modification of the topology and the remesh of the model. Currently there are in CG many approaches for topological changes, some of these are non-interactive animations, unstable or not precise enough to medical applications; comparing with the existing approaches, it has been found that the XFEM is stable, accurate, with excellent performance and suitable for virtual surgery in real time; on the other hand, to maintain the provided advantages it is required the selection and creation of a set of methods that fulfill the requirements of the XFEM.

The algorithms necessary to create a surgery simulation around the XFEM are: the integration time, meshing techniques, collision detection, mapping methods, etc. Therefore, it is important to look for an excellent performance in each one of these algorithms and in the interaction between them, all of this in sight of an interactive and real time simulation. We propose an embedded mapping method inside the control of the physical topology that enables the relation of the XFEM elements with the visual and collision mesh, making the user interaction more dynamic. The XFEM, used as a core of the CG methods, helps to simulate in an efficient manner thin models as the skin and, as a consequence, making possible the interaction in real time, which is going to allow the creation of more complex simulations with mayor impact in the medical area.

Acknowledgments

To CINVESTAV for giving me the opportunity to grow intellectually and personally.
To CONACYT for the scholarship which made all this possible.
To Dr. Félix Ramos for his interest in this project, the time and the help to overcome the problems which came up during the development of this project.
To SOFA Team.

Table of Symbols

Symbol	Description	Where defined
σ	stress	2.3.1
ϵ	strain	2.3.1
E	Young's modulus	2.3.1
v	Poisson ratio	2.3.1
$\mathbf{u}$	displacement vector	2.3.1
$\mathbf{c}$	matrix of material constants	2.3.1
$\mathbf{L}$	differential operator matrix	2.3.1
Φ	shape function	2.3.2
ρ	mass density	2.3.2
$\mathbf{B}$	strain matrix	2.3.2
$\mathbf{K}$	stiffness matrix	2.3.2
$\mathbf{M}$	mass matrix	2.3.2
$\mathbf{f}$	forces	2.3.2
$\mathbf{C}$	damping matrix	2.3.2
n	number of element nodes	2.3.2
$\mathbf{R}$	rotation matrix	2.3.4
$\mathbf{p}$	position of the element (Corotational FEM)	2.3.4
$\psi(x)$	enrichment function	2.4
$\mathbf{a}_j$	added DOFs	2.4
$H(x)$	Heaviside function	2.4
$\mathbb{M}$	lumped mass matrix	2.4.3

Contents

Chapter 1

Introduction

1.1 Computer Graphics in Medicine

The computer sciences collaborate of interdisciplinary form, and many of its areas have been developing new methods and algorithms in the medical contexts; areas such as artificial intelligence, robotics, human computer interaction, and computer graphics. From the perspective of computer graphics in medicine, the computer can be an useful tool that allows to the surgeons or medical students to analyze patients before of performing the surgery, or even more, simulating the physical behavior of the body, permitting in this way, the possibility to train the students, thus avoiding the risks caused to humans by the lack of experience of the procedures execution.

A very common procedure in a surgery is the incision of the tissues, for instance, the cut of the skin. This procedure can be performed in minimal invasive surgery or in an open surgery; in the first one, due to its reduced operating area, the interventions are smaller than in the open surgery that can apply many more instruments and large interventions. Moreover, many important and complex open surgeries start with incisions, for example, some tumor extractions, cardiac interventions and cerebral surgeries.

1.2 Surgery simulation

In order to simulate the incisions in a realistic manner, are required the physical based methods, which give the adequate accuracy; furthermore, it is desirable an interactive simulations that react in real time; consequently, it is necessary to look for efficiency in the methods and, due to our medical perspective, the stability in the simulation is also essential. Despite, many approaches deal with topological changes, these do not fulfill with the necessary characteristics previously described.

It has been found that the XFEM, as a physical based method (based on the FEM), allows the creation of robust simulations. The use of the XFEM in the medical context was introduced by [48] implementing 2D simulations. Further, [21] establish the factors to ensure stable simulations suitable in surgical simulations in 3D.

1.3 The Aim and scope

In spite of the fact that we identify the XFEM as the main method that ensures a robust simulation, there is no framework to exploit its benefits to make it applicable to complex simulations as surgery simulation. Therefore, it is necessary to design a framework completely focused on the XFEM in order to obtain efficient responses and, in the same time, to indicate how to implement it obtaining efficient results.

The main goal of this thesis is to give a specific set of practical methods that allow the implementation of the XFEM in an efficient simulation, achieving with the characteristics that a virtual surgery demands. For this purpose, it is necessary to design a method that works as mediator between the physical original mesh and the others meshes (visual and collision), also the creation of a specific cutting algorithm that quickly updates the associations of the meshes and finally, describing practical ideas to implement efficiently this method.

With this approach we aim to select fast methods of computer graphics such as collision detection and time integration; considering also, the design of a specific mapping method for the XFEM that allows an adequate interaction between different topologies. Moreover, it is considered the implementation of the XFEM in 2D, this can help to test the performance comparing with other methods; afterwards, the approach is implemented and tested in an interactive open surgery simulation employing 3D models.

As remark, in this work are not considered the use of haptic tools and 3D sensors to simulate the virtual instrument; and also we don't pretend to modify the equations of the XFEM.

1.4 Outline

This thesis is organized as follows:

In the chapter 2, it is described the state of the art, showing many works related with medical applications in virtual reality, topological changes and virtual surgery simulations. Furthermore, it is included the basic concepts of the finite element method and the extended finite element method required to the complete understood of the thesis.

The chapter 3 focuses on the explanation of the proposed framework, where are indicated the adaptations to the selected algorithms and, later, the alternative mapping method is detailed. Further, the cutting algorithm is showed and described. Finally, we point out some practical considerations that improve the realism of the simulation.

In the chapter 4, the cases of study are described, following with the description of SOFA framework and the explanation of how to implement the approach in it; the implementation is described in 2D and 3D. Also, in this chapter are illustrated the tests of the approach finalizing with the implementation of the study case.

Lastly, the chapter 5 indicates the most important contributions of the approach; moreover, we establish some questions that are left for future research.

Chapter 2

Physical-based models for virtual surgery

2.1 Introduction

Nowadays the Virtual Reality (VR) has been applied to lot areas such as entertainment, education, military, industrial, architecture, robotics, medicine and others. The most applications look for an immersive interaction with the user in the virtual environments, for this reason has been designed a diversity of devices that we can classify as *interaction devices*: gloves, head mounted displays, movement sensors, haptics[10], etc.; *acquisition data devices*: specialized cameras and scanners, the most commons in medicine are a Computerized Tomography (CT) and Magnetic Resonance (MR).

In this chapter will be described the related work of CG and virtual reality in medicine, first will be mention the background of the medical applications, followed by the deformable objects modeling methods considering the cutting approaches; further, the basic concepts of the Finite Element Method and eXtended Finite Element Method are described and finally, we analyze and discuss the related work indicating some required improvements.

2.2 Virtual reality applied to medicine

The importance of VR in medicine is growing, the reason is that it gives the possibility to create an environment in which represent the organs in 3D and letting the user to interact with this virtual models avoiding the risk of working with real patients. Nowadays the VR research has been focused on the realism of the simulations implying to consider the physical reactions of the objects, it means to base the result in mathematical models in order to generate accurate simulations.

Computer graphics (CG) was introduced to the medical research in mid of 70's with a three-dimensional visualization of CT. Today, a variety of medical applications has been developed in different areas that include medical diagnosis, procedures training, pre-operative planning, telemedicine and many more; some medical applications that can be found in the related work are: the *3D organs representation* of the human anatomy for purposes like education as the Visible Human Project [1], surgery planning [39] or for diagnosis and surgery simulation [3]; *virtual endoscopy* simulates a real endoscopy where the medic inserts an endoscope (camera in a fiber-optic tube) into the body, it can be simulated interactively by letting the user to control the virtual endoscope while he observes the 3D models of organs [24, 25]; *suture simulation* where the students can be evaluated during the suture process, they interact with devices to simulate the needle and other instrument to hold the needle, the suture can be external as a laceration of a hand [7] or internal like a vascular surgery [19]. For more medical applications in CG we refer the survey in [47] where described important techniques and algorithms for the visualization and interaction with the medical data. The creation of a surgery simulation is a complex work that involves many methods of CG such as: soft tissue deformation modeling, meshing techniques, collision methods, mapping methods and haptic devices (cp. Fig. 2.1). In the next section we describe the soft tissue deformation methods to model physical and graphically the deformation of an object.

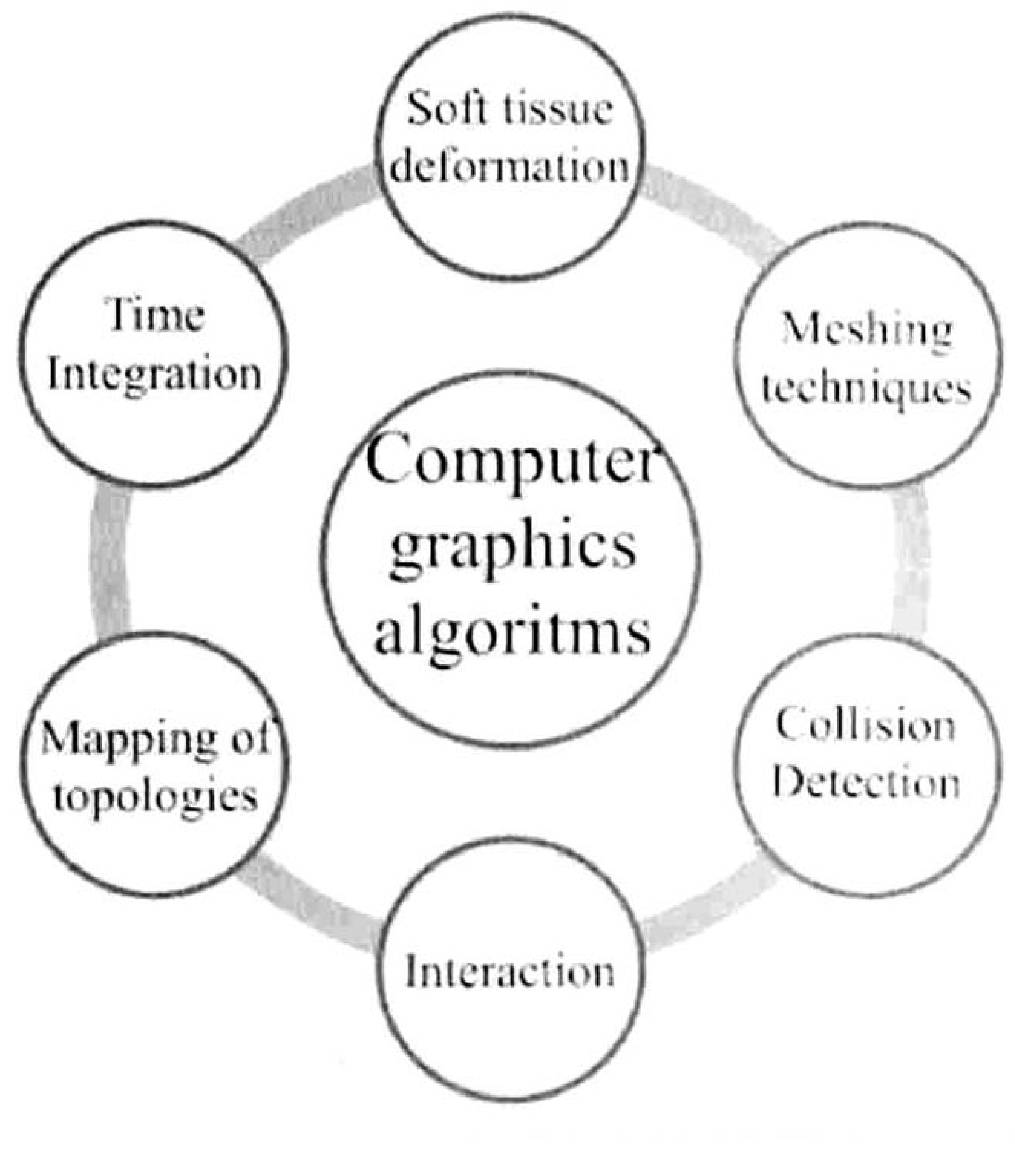

Figure 2.1: Principal methods of computer graphics to consider in the creation of a simulation

2.2.1 Soft tissue deformation

In order to create a realistic simulation is needed to consider the behavior of the tissue. In CG an interesting research area is to simulate soft materials e.g. the hair, cloth, plastic objects and human tissues. The soft-tissue modeling methods can be: geometrically-based and physically-based; a typical example of the first is the free-form deformation (FFD) method; for the second the most used are: Finite Element Method (FEM), Mass-Spring Model (MSM), Finite Difference Method (FDM), among others.

The *MSM* consists of a deformable object that is discretized into a system of mass points connected by elastic links; this method is intuitive and easy to implement, it also can be applied to real-time simulations and, moreover, it prevents of numerical instability and it is parallelizable; however mass-springs connectors by itself has no volume, the topology of the MSM is dependent of the mesh resolution and even worst the MSM is not as accurate as FEM. The MSM can be suitable for animations of cloths but for lack of accuracy is not plausible in surgery simulations. Has been compared an optimized linear FEM and the MSM obtaining the first better results in a similar computation time [17]. The *FEM* is deeply described in section 2.3. An overview of physically based models and applications can be found in [35].

There are several medical approaches that apply the physically based deformation methods to simulate in a realistic manner the organs deformation using the FEM [25, 3, 41, 50], MSM [11] and hybrid approaches that uses the FEM and MSM [13]; some authors design its own physical model like the Long Element Method (LEM) where the physical properties are in terms of pressure, density, volume, stress and strain [26]; [51] employ the MM-model to simulate blood vessels they indicate that the MM model is better than traditional methods for noncomplex surface structure.

For virtual surgery the physical deformation of the tissues is necessary, although to create virtual surgeries with higher degree of complexity the simulation must contain the capacity of cut the tissue.

2.2.2 Cutting approaches

To create more complex medical simulations was necessary to include cuts and dissections, this inclusion changes the physical mesh topology; nevertheless, this is a challenging work to simulate in real time with enough accuracy and, the requirement of real-time simulations, forces to create adaptations on the equations and the development of new algorithms that controls the re-meshing of the model.

In the related work there are some approaches that deal with the problem of changing the topology; first of all this problem has to be analyzed from two different considerations, the user interactivity in real-time and accurate simulations. A *real time simulation* has as a principal requirement the speed in the reactions, the responses of the models when the surgical tool penetrates must be at the same time the user interacts and the force feedback must be maintained in high rates; nevertheless this process sacrifices accuracy on the process.

There are simulations that need to be accurate, this is the case of a surgery simulation where all the body reactions must be physically plausible, nonetheless it increments the computational effort creating slower simulations. Looking for a trade-off between medical realism and interactivity in real time, [12] propose to separate the mesh in sub-regions (operation part, non-operation part and the interface between them), adapting the equations to each part obtaining faster responses in the operation region; they test the equations for cutting the skin in a 2D model; on the other hand has to be selected the operation part a priori to be preprocessed leading into a non-fully interactive simulation.

In a surgery simulation, when the virtual surgical tool cuts the tetrahedrons this should show how the incision opens according to the cutting trajectory, to this end, some approaches apply the subdivision, snapping, removal or duplication of elements. A simple manner to cut a tetrahedra is *subdividing* it into more small sub-tetrahedrons. [33] explain how to subdivide tetrahedra in a minimal set of sub-elements, they implement a progressive cut by creating "temporal intersections", [9] create an algorithm to maintain consistent the mesh subdivision of tetrahedral faces by restricting the subdivision to correct junctions with the neighbors, [8] create a state machine for subdividing tetrahedra according to the intersections of the tool blade in a progressive mode considering forward and backward moves and trembling of the tool while the user is cutting, they simulate the skin cutting with a virtual scalpel provided with force feedback; nevertheless subdividing can cause simulation instability because it generates ill-conditioned elements or slivers, also subdivision increments the degrees of freedom impacting directly on the simulation performance.

Another alternative to cut soft tissue is done by *removing* the elements in contact with the blade [14, 15]; [16] first refines the elements closer to the surface where the tool is cutting and removes all elements touched by the blade, implementing with this approach a hepatectomy simulator; despite removing elements has the advantage that it doesn't affects the simulation stability, this method is physically inaccurate and visually uneven; other option, to avoid the previous methods, is a successive *snapping* of nodes to the cutting trajectory [43]; even though, this kind of methods still generating sliver elements; other approaches merge methods like snapping and subdivision [45], but these require to know the first and last point of a cut segment to execute the algorithm, therefore only non-progressive cutting is enabled.

For fractures simulation there are approaches that changes the topology by *duplicating* elements, [32] propose the virtual node algorithm; it consists of replicate cut elements assigning a portion of material to each copy, it maintains the initial FEM mesh conditioning creating a minimal number of elements, the limitation of the method is that a tetrahedra can only be subdivided in no more than four pieces because each sub-element require an original mesh node and also slivers are not completely avoided; the limitations of the virtual node algorithm were resolved by [44], avoiding slivers and allowing arbitrary cutting of tetrahedrons in any number of sub-elements, implementing the algorithm in a simulation where the skin of a tetrahedral face model is cut; nonetheless on both previous approaches has been tested in a offline simulations which is undesirable for a surgery simulation.

[42] create and hybrid approach with Discontinuous Free Form Deformation (DFFD) and FEM, getting accuracy from FEM and speed from DFFD and simulates a progressive cut of a skin of the face; this approach stills subdividing the polygons affecting the performance in each cut element and the results are not enough accurate in large deformations.

[21] employ the extended finite element method (XFEM) to control physically the mesh when a cut appears, while the user is cutting no new elements are created thus the simulation performance is not a greatly impacted and furthermore the XFEM avoids ill-conditioned elements.

The XFEM adds local enrichment functions in sub-regions with discontinuities; this method, proposed in 1999 by [5], exploits the partition of unity property of finite elements [4]. The XFEM is initially utilized for fracture mechanics to simulate crack growth with stiff materials in 2D [34]; this method has been improved and can be applied to different domains such as material interfaces [46, 6], 3D elasto-plastic deformations [23], fluid mechanics, material-nonmaterial interfaces and topology optimization considering void spaces [36]. We explain the XFEM in more detail in section 2.4.

The XFEM was introduced to simulation of surgical cuts by [48, 49], implementing a simulation of dissection in a 2D MRI image of the brain with small deformations. [27] propose a framework based on FEM considering predefined boundaries for cutting and suturing, to cut arbitrary surfaces they employ the XFEM; this approach also allows small deformation as the linear XFEM was used.

[21] establish that the XFEM is stable, accurate, allowing large deformations and suitable for real time surgical simulations; they also mention that a stable and dynamic simulation can be ensure by selecting the appropriate enrichment function and mass lumping technique; they test the method simulating the incision of the skin in an open surgery.

2.3 The Finite Element Method

The FEM is a numerical method that permits to find approximated solutions to physical engineering problems governed by differential equations in partial derivatives. The FEM is employed when the problem domain is complex and the solution is difficult to be obtained analytically; commonly it is applied in problems such as fluid mechanics, heat transfer, solids mechanics, electromagnetic analysis, acoustics, and so on. To simulate the behavior of a phenomenon in a system, we must consider that this will be dependent of the geometry or domain of the system, the properties of the material and boundary and initial and loading conditions; however, in many engineering systems all these factors are very complex hampering an analytical solution, for this reason are used numerical methods that discretize the domain and the most popular of these methods is the FEM.

When the FEM is used generally consists of four steps: the first step is the *modeling the geometry* commonly represented by a set of elements, and the curved surfaces are approximated using straight lines if linear elements are used; for this step there are numerous Computer Aided Design (CAD) software packages that generate the geometry; the second step is the *meshing* process required to discretize the geometry into small *elements*, this step is performed to divide the problem domain. Thereby, the solution of each element is easily approximated and the solution of the whole problem domain is formed by the solution of all elements. The third step is the *property of material* that consists in defining the material properties according to the problem domain; in our case the Young's modulus and Poisson ratio are required (described in section 2.3.1). Finally the fourth step consists into establish the *boundary, initial and loading conditions* where the users can indicate these conditions into the geometrical identities (points, lines and surfaces) or also into the elements.

In order to understand the how the FEM models the soft tissue deformation, it is necessary to know the basic concepts of the linear elastic material model.

2.3.1 Elastic materials

The analysis of elastic materials deals with the behavior of substances with the property of recovering their size and shape when the external forces are removed. In 1676 Robert Hooke formulate the linear elastic behavior, in his law establish that the deformation is proportional to the applied force in the object. Let define F as the applied force that stretch a spring, Δl as the change of the length in the spring (elongation), and k as a force constant of the spring, then the *Hooke's law* for a linear spring is described by

$$F = k\Delta l \tag{2.1}$$

In order to simulate the elastic behavior of more complex systems is necessary to consider some physical fields that are present when a elastic material is deformed such as stress and strain that can be derived from equation (2.1) obtaining

$$Stress = modulus \cdot Strain \tag{2.2}$$

$$\frac{F}{A} = E\frac{\Delta l}{l}, \tag{2.3}$$

where A is the transactional area and E is a property of the nature of the material called Young's modulus. Equation (2.3) indicates that the *stress* is the force per unit area and the *strain* is the change in length per unit length (fractional stretch).

The physical fields (strain, stress, force and displacement) for a 3D elastic object are expressed as matrixes, and it's important to analyze the relations among these.

Stress

In an object the components of stress are indicated on the surface of an infinitely small cubic volume (cp. Fig. 2.2). The sign convention for the subscript is that the stress σ_{ij} is acting on the *i-th* surface with direction *j-th*. Therefore, there are nine stress components and by taking moments of forces on the central axis of the cube at the equilibrium state, it is easy to confirm that

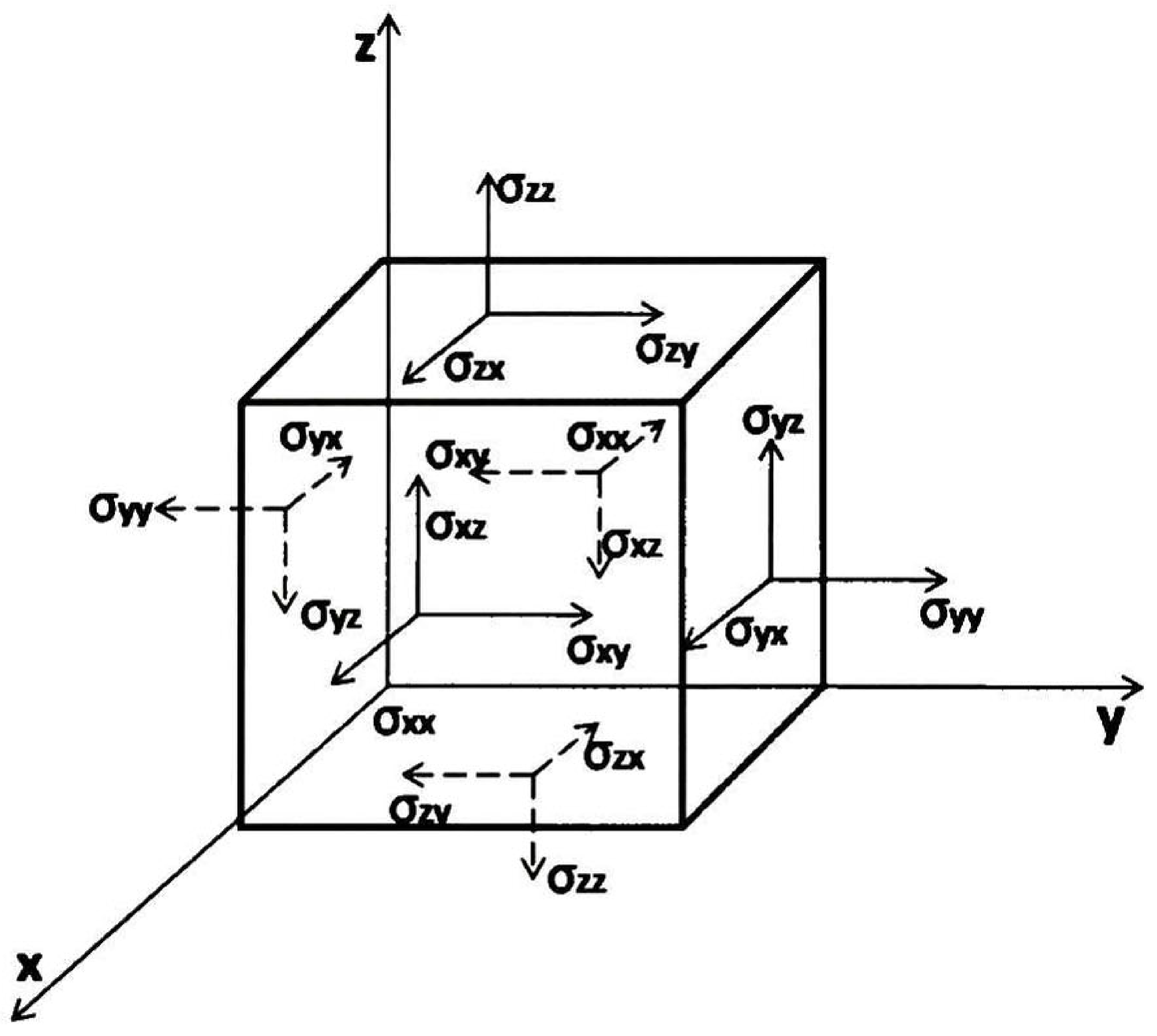

Figure 2.2: Six independent stress components at an internal point in an infinitesimalsmall cubic block.

$$\sigma_{xy} = \sigma_{yx}; \quad \sigma_{xz} = \sigma_{zx}; \quad \sigma_{zy} = \sigma_{yz} \tag{2.4}$$

Hence, at a point in the elastic object there are in total six stress components known as *stress tensor* and can be expressed in a vector form

$$\sigma^T = \{\sigma_{xx} \quad \sigma_{yy} \quad \sigma_{zz} \quad \sigma_{yz} \quad \sigma_{xz} \quad \sigma_{xy}\} \tag{2.5}$$

Before to show the relation between stress and strain we will define the strain for a 3D elastic object and its relations with the displacement.

Strain - Displacement Relations

In real elastic objects the strain varies throughout the geometry and similar to the stress, the strain can be composed of six independent components and can be expressed in the vector form of

$$\epsilon^T = \{\epsilon_{xx}\ \epsilon_{yy}\ \epsilon_{zz}\ \epsilon_{yz}\ \epsilon_{xz}\ \epsilon_{xy}\} \tag{2.6}$$

Let denote $u = u(x,y,z)$, $v = v(x,y,z)$, and $w = w(x,y,z)$ as the displacements in the x,y, and z coordinate directions respectively, then the components of the strain can be obtained from the derivatives of the displacements as follows:

$$\begin{aligned} &\epsilon_{xx} = \frac{\partial u}{\partial x}; \quad \epsilon_{yy} = \frac{\partial u}{\partial y}; \quad \epsilon_{zz} = \frac{\partial u}{\partial z}; \\ &\epsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}; \quad \epsilon_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}; \quad \epsilon_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \end{aligned} \tag{2.7}$$

To express the strain-displacement relations in matrix form, we define the displacement vector as

$$\mathbf{u} = \left\{ \begin{array}{c} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{array} \right\} \tag{2.8}$$

The six strain-displacement relationships in equation (2.7) can be expressed in the compact form

$$\epsilon = \mathbf{Lu} \tag{2.9}$$

where **L** is a matrix of partial differential operators obtained from equation (2.7) and is given by

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \end{bmatrix} \tag{2.10}$$

In computer graphics there are two common choices known as the *Green's nonlinear strain tensor* and the *Cauchy's linear strain tensor* defined in equations (2.12) and (2.11) respectively.

$$\epsilon = \frac{1}{2}(\nabla\mathbf{u} + [\nabla\mathbf{u}]^T + [\nabla\mathbf{u}]^T\nabla\mathbf{u}) \tag{2.11}$$

$$\epsilon = \frac{1}{2}(\nabla\mathbf{u} + [\nabla\mathbf{u}]^T] \tag{2.12}$$

where $\nabla\mathbf{u}$ is the gradient of the displacement field and is expressed in a matrix of the form

$$\nabla\mathbf{u} = \left\{ \begin{array}{ccc} u_{,x} & u_{,y} & u_{,z} \\ v_{,x} & v_{,y} & v_{,z} \\ w_{,x} & w_{,y} & w_{,z} \end{array} \right\} \tag{2.13}$$

Where the index x,y and z represents spatial derivatives. If the deformations of the body are large is used the Green's strain tensor, on the other hand for small deformations the Cauchy's strain tensor is used.

Stress-strain relations

The equations that give the relationship between stress and strain are known as the *constitutive equations* for a material. The equation (2.2) can be generalized as the Hooke's law for a general anisotropic material (i.e. the material property varies with direction) given in the following matrix form:

$$\sigma = \mathbf{c}\epsilon \tag{2.14}$$

where $\mathbf{c}$ is a matrix of material constants can be written explicitly as a symmetric matrix as follows.

$$\left\{ \begin{array}{c} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{yz} \\ \sigma_{xz} \\ \sigma_{xy} \end{array} \right\} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ & & c_{33} & c_{34} & c_{35} & c_{36} \\ & & & c_{44} & c_{45} & c_{46} \\ & & & & c_{55} & c_{56} \\ & & & & & c_{66} \end{bmatrix} \left\{ \begin{array}{c} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \epsilon_{yz} \\ \epsilon_{xz} \\ \epsilon_{xy} \end{array} \right\} \tag{2.15}$$

For isotropic materials the matrix **c** can be reduced to

$$\mathbf{c} = \begin{bmatrix} c_{11} & c_{12} & c_{12} & 0 & 0 & 0 \\ & c_{11} & c_{12} & 0 & 0 & 0 \\ & & c_{11} & 0 & 0 & 0 \\ & & & (c_{11}-c_{12})/2 & 0 & 0 \\ & & & & (c_{11}-c_{12})/2 & 0 \\ & & & & & (c_{11}-c_{12})/2 \end{bmatrix} \tag{2.16}$$

where

$$c_{11} = \frac{E(1-v)}{(1-2v)(1+v)}; \quad c_{12} = \frac{Ev}{(1-2v)(1+v)}; \quad \frac{c_{11}-c_{12}}{2} = G \tag{2.17}$$

in which E,v,G are Young's modulus, Poisson's ratio, and the shear modulus of the material, respectively, where the first two are independent. Given two of these three constants the other one can be calculated by the relation

$$G = \frac{E}{2(1+v)} \tag{2.18}$$

Equilibrium equations

To obtain the dynamic equilibrium equations, let us consider an arbitrary point in the body using an infinitesimal differential element, as shown in Figure 2.2. The equilibrium of forces is required in all directions and, as this is a generalization of a dynamic system, we have to consider the inertial forces of the element. The equilibrium of forces in the x directions gives

$$\begin{aligned} &\left(\sigma_{xx} + \frac{\partial \sigma_{xx}}{\partial x}\mathrm{d}x\right)\mathrm{d}y\mathrm{d}z - \sigma_{xx}\mathrm{d}y\mathrm{d}z + \left(\sigma_{yx} + \frac{\partial \sigma_{yx}}{\partial y}\mathrm{d}y\right)\mathrm{d}x\mathrm{d}z - \sigma_{yx}\mathrm{d}x\mathrm{d}z \\ &+ \left(\sigma_{zy} + \frac{\partial \sigma_{zx}}{\partial z}\mathrm{d}z\right)\mathrm{d}x\mathrm{d}y - \sigma_{zx}\mathrm{d}x\mathrm{d}y + \underbrace{f_x}_{\text{external force}} = \underbrace{\rho\ddot{u}\mathrm{d}x\mathrm{d}y\mathrm{d}z}_{\text{inertial force}} \end{aligned} \tag{2.19}$$

where the inertial force is the term on the right-hand side of the equation, and f_x is the external body force that affects the body as a whole (i.e. applied in the centre of the element). Hence, the equation (2.19) becomes one of the equilibrium equations, written as

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} + f_x = \rho\ddot{u} \tag{2.20}$$

in the same way, the equilibrium equations of forces in the y and z directions yields

$$\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z} + f_y = \rho \ddot{v} \tag{2.21}$$

$$\frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + f_z = \rho \ddot{w} \tag{2.22}$$

Defining a vector of the external body forces $\mathbf{f}_b$ in x, y and z directions as follows

$$\mathbf{f}_b^T = \{f_x \; f_y \; f_z\} \tag{2.23}$$

the equilibrium equations (2.20),(2.21) and (2.22) can be written in a concise matrix form as follows

$$\mathbf{L}^T \sigma + \mathbf{f}_b = \rho \ddot{\mathbf{u}} \tag{2.24}$$

To write the equilibrium equation (2.24) in term of the displacements can be done by using equations (2.9) and (2.14). Hence, can be obtained a general form of the dynamic equilibrium equation expressed as a matrix equation

$$\mathbf{L}^T \mathbf{c} \mathbf{L} \mathbf{u} + \mathbf{f}_b = \rho \ddot{\mathbf{u}} \tag{2.25}$$

if the loads applied on the body are static, then the static equilibrium can be obtained simply by dropping the dynamic term (inertial force) in equation (2.25)

$$\mathbf{L}^T \mathbf{c} \mathbf{L} \mathbf{u} + \mathbf{f}_b = 0 \tag{2.26}$$

2.3.2 FEM

This method is based in the applications of Rayleigh-Ritz method, it doesn't work over the complete domain of the problem; it only considers sub-domains called *elements*. In the FEM, a complex continuum system is discretized into simple and small geometric elements; some nodal points are specified in an element where the unknown values are expressed mathematically. An interpolation based on the values of the nodes is needed to obtain the unknown values inside an element. The interpolation or shape functions are complete set of polynomials.

This chapter presents the simple overview of the FEM, for a deep analysis of the method refer to specific FEM books, such as [28],[20] and [52].

FEM procedure

The standard FEM procedure consists of four steps: domain discretization, constructing shape functions, formation of the FE equations on local coordinate systems and assembly of global FE equations. This procedure is shown in Figure 2.3.

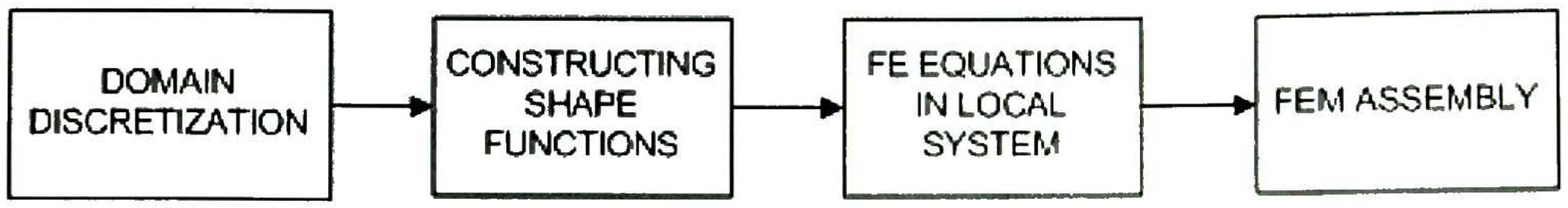

Figure 2.3: FEM procedure performed to simulate elastic objects.

Domain discretization The total domain (body) Ω is divided into n_e sub-domains Ω_e, called elements. This procedure is called *meshing* and is often carried out by preprocessors. All elements are connected by its nodes forming an entire domain without any gap or overlapping. A finer mesh generally generates more accurate results.

Constructing Shape functions The FEM formulation uses a local coordinate system that is defined for an element in reference to the global coordination system; i.e., it is possible that each element has its own orientation and to assemble all the elements it is required a coordinate transformation for each element, considering a global coordinate system. Based on the local coordinate system defined on an element, the displacements of an element is assumed by polynomial interpolation using the nodal displacements as follows

$$u(x) = \sum_{i=1}^{n} \Phi_i(x)\mathbf{u}_i^e \tag{2.27}$$

where n is the number of nodes forming the element e, $\mathbf{u}_i^e$ is the nodal displacements at *i-th* node, which can be expressed in a general form as

$$\mathbf{u}^e = \left\{ \begin{array}{c} \mathbf{u}_1^e \\ \mathbf{u}_2^e \\ \vdots \\ \mathbf{u}_n^e \end{array} \right\} \begin{array}{l} \rightarrow \text{displacements at node 1} \\ \rightarrow \text{displacements at node 2} \\ \qquad\qquad \vdots \\ \rightarrow \text{displacements at node n} \end{array} \tag{2.28}$$

where each of the $\mathbf{u}_i^e$ has n_f Degrees Of Freedom (DOF) that corresponds to the displacements and / or rotations at the *i-th* node (i.e. for a 3D body $n_f = 3$). Therefore, the total dimension of the vector of DOFs $\mathbf{u}^e$, is the same that the number of element nodes times the DOFs at a node, i.e. $n \times n_f$.

In equation (2.27), Φ_i are the element *shape functions*, which are predefined to assume the shapes of the displacement variations with respect to the coordinates. These functions are chosen in such way that the nodal positions take either a unitary or a null value, this property is known as the *Delta property*. In the case of a linear triangle the shape functions are identical to the barycentric coordinates as shown in Figure 2.4.

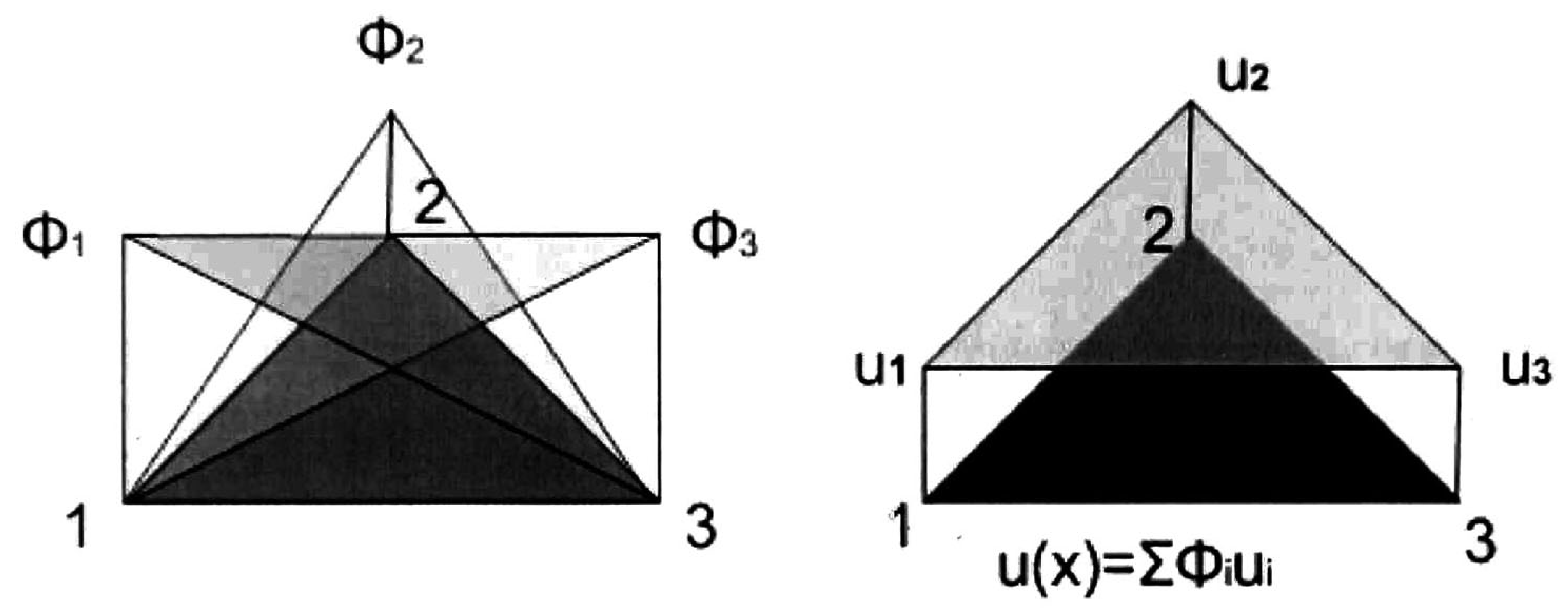

Figure 2.4: Shape functions of a sigle linear triangular element; these functions (Φ_i) are used to interpolate the nodal displacements $\mathbf{u}_i$ allowing to obtain of a continuous displacement inside an element ($u(x)$).

FE equations on local coordinate systems The potential energy in this context is the elastic strain energy. The strain energy in the entire domain of elastic solids can be expressed as

$$\Pi = \frac{1}{2}\int_V \epsilon^T \sigma \mathrm{d}V = \frac{1}{2}\int_V \epsilon^T \mathbf{c}\epsilon \mathrm{d}V \tag{2.29}$$

The kinetic energy of the entire problem domain is defined as follows

$$T = \frac{1}{2}\int_V \rho \dot{\mathbf{u}}^{\mathrm{T}} \dot{\mathbf{u}} \mathrm{d}V \tag{2.30}$$

Therefore, FE equations for an element can be formulated by substituting the interpolation of the nodes, (2.27) and the equation (2.9) which is the strain-displacement equation, into the equation (2.29) yields

$$\Pi = \frac{1}{2}\int_{V_e} \epsilon^T \mathbf{c}\epsilon \mathrm{d}V = \frac{1}{2}\int_{V_e} \mathbf{u}^{eT}\mathbf{B}^T\mathbf{c}\mathbf{B}\mathbf{u}^e \mathrm{d}V = \frac{1}{2}\mathbf{u}^{eT}\left(\int_{V_e} \mathbf{B}^T\mathbf{c}\mathbf{B}\mathrm{d}V\right)\mathbf{u}^e \tag{2.31}$$

The script e stands for the element, hence the volume integration is over the element. In the equation (2.31) $\mathbf{B}$ is called the *strain matrix*, defined by

$$\mathbf{B} = \mathbf{L}\mathbf{\Phi} \tag{2.32}$$

where $\mathbf{L}$ is the differential operator given in equation (2.10). Let define the element *stiffness matrix* as

$$\mathbf{K_e} = \int_{V_e} \mathbf{B}^T\mathbf{c}\mathbf{B}\mathrm{d}V \tag{2.33}$$

hence, the equation (2.31) can be rewritten as

$$\Pi = \frac{1}{2}\mathbf{u}^{eT}\mathbf{K_e}\mathbf{u}^e \tag{2.34}$$

The stiffness matrix is symmetric and its dimension is equal to the number of element nodes times number of DOFs at a node, i.e. $n \times n_f$

Substituting equation (2.27) into (2.30), the kinetic energy can be rewritten as

$$T = \frac{1}{2}\int_{V_e} \rho\dot{\mathbf{u}}^{\mathbf{T}}\dot{\mathbf{u}}\mathrm{d}V = \frac{1}{2}\int_{V_e} \rho\dot{\mathbf{u}}^{eT}\mathbf{\Phi}^T\mathbf{\Phi}\dot{\mathbf{u}}^e\mathrm{d}V = \frac{1}{2}\dot{\mathbf{u}}^{eT}\left(\int_{V_e} \rho\mathbf{\Phi}^T\mathbf{\Phi}\right)\dot{\mathbf{u}}^e \tag{2.35}$$

From previous equation we can define the mass matrix of the element, by denoting

$$\mathbf{m^e} = \int_{V_e} \rho\mathbf{\Phi}^T\mathbf{\Phi}\mathrm{d}V \tag{2.36}$$

where ρ is the material density. Finally the equation of the forces can be computed as the derivatives of the energy with respect to the nodal positions. A linearized relationship for an element between nodal forces and nodal positions can be expressed as

$$\mathbf{f^e} = \mathbf{K^e}\mathbf{u^e} \tag{2.37}$$

FEM Assembly The FE equations for all the elements can be assembled together, thereby, is obtained the deformation of the whole system. The global stiffness matrix is calculated as follows

$$\mathbf{K} = \sum_{i=1}^{n_e} \mathbf{K^e}_i \tag{2.38}$$

Using the linearize elastic forces, the linear algebraic equation of motion for the entire domain is expressed as

$$\mathbf{M\ddot{u}} + \mathbf{C\dot{u}} + \mathbf{Ku} = \mathbf{f}_{ext} \tag{2.39}$$

where $\mathbf{M}$ is the global mass matrix, $\mathbf{C}$ is the damping matrix and $\mathbf{f}_{ext}$ externally applied forces, $\mathbf{\ddot{u}}$ is the acceleration vector and $\mathbf{\dot{u}}$ is the vector of the velocity components.

2.3.3 Linear FEM

In computer graphics has been applied three kind of FEM approaches, *the linear FEM, the corotational method* and *the non-linear method*; for purposes of this thesis only the first two are described. For small displacements can be used the linear Cauchy's strain tensor shown in equation (2.11). Then the strain matrix $\mathbf{B}$ of the equation (2.32) can be expressed explicitly as

$$\mathbf{B}_i = \begin{bmatrix} \frac{\partial\Phi_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial\Phi_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial\Phi_i}{\partial z} \\ \frac{\partial\Phi_i}{\partial y} & \frac{\partial\Phi_i}{\partial x} & 0 \\ \frac{\partial\Phi_i}{\partial z} & 0 & \frac{\partial\Phi_i}{\partial x} \\ 0 & \frac{\partial\Phi_i}{\partial z} & \frac{\partial\Phi_i}{\partial y} \end{bmatrix} \tag{2.40}$$

Considering i and j the indices of the element nodes, the stiffness matrix can be defined of the form

$$\mathbf{K}_{ij} = \int_V \mathbf{B}_i^T \mathbf{c} \mathbf{B}_j \mathrm{d}V \tag{2.41}$$

2.3.4 Corotational FEM

The corotational method is based in the linear FEM using the Cauchy's strain tensor; on the other hand a reference state of the elements is store. To compute the deformation forces, these have to be translated to the reference state and rotate back to the current state. Therefore, the deformation forces are expressed as follows

$$\mathbf{f}_i = \mathbf{R}\sum_{j=1}^{n} \mathbf{K}_{ij}(\mathbf{R}^T\mathbf{p}_j - \mathbf{p}_{0j}) \tag{2.42}$$

where $\mathbf{R}$ is the rotation matrix is required to translate from the current to initial state, $\mathbf{p}$ are the positions of the element nodes in the current deformed state defined as $\mathbf{p} = \mathbf{p}_0 + \mathbf{u}$, in which $\mathbf{p}_0$ indicates the initial positions of the element nodes.

2.3.5 Time integration

In a simulation such as virtual surgery, the environment is interactive and the user exerts forces over the virtual objects in real time, therefore is necessary to compute the displacements dynamically employing the equation (2.39). The external forces are in equilibrium with the internal forces by solving the displacement vector $\mathbf{u}$. To compute the displacements at each time step are widely used the *direct integration method.*

There are two main types of direct integration method: *implicit* and *explicit.* Implicit methods are generally more efficient for a relatively slow phenomenon, and explicit methods are more efficient for a very fast phenomenon. A list of explicit and implicit methods is shown in Figure 2.5.

The most explicit methods are conditionally stable. This means that are limited to a time step Δt, and if it is exceeded the computed solution will become unstable and might grow without limit. Therefore, the time steps used in the explicit methods are 100 or 1000 times smaller than those used in implicit methods. Defining the acceleration as $\mathbf{a} = \ddot{\mathbf{u}}$ and the velocity as $\mathbf{v} = \dot{\mathbf{u}}$, the explicit Euler method is expressed as

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t \mathbf{v}_t \tag{2.43}$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \Delta t \mathbf{a}_t \tag{2.44}$$

Implicit methods are unconditionally stable, and the stability of the simulation is not dependent of size of the time step, however these methods require the evaluation solution of large system of equations. The implicit Euler method is expressed as

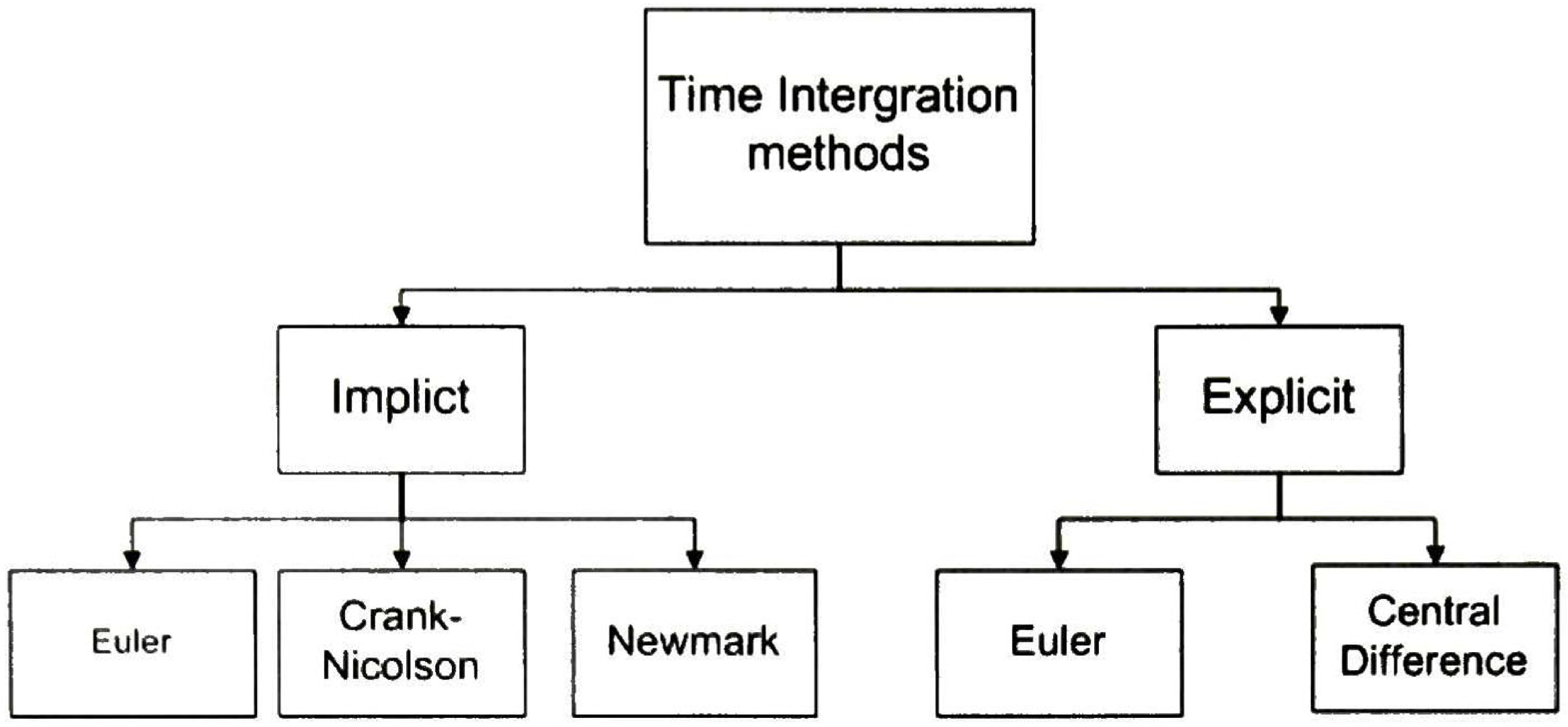

Figure 2.5: List of common time integration methods.

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t \mathbf{v}_{t+\Delta t} \tag{2.45}$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \Delta t \mathbf{a}_{t+\Delta t} \tag{2.46}$$

To solve these equations is required to redefine a new system of equations and employ a numerical method such as Newton-Raphson. If the derivative of the forces changes in time, the Newton-Raphson method must be executed each time step.

2.4 Extended Finite Element Method

The XFEM is in essence the FEM with adaptations appropriate for evolving boundary value. It basically represents the discontinuities with the creation of enrichment functions. It exploits the partition of unity property of shape functions, this property is expressed as:

$$\sum_{i=1}^{n} \Phi_i(x) = 1 \tag{2.47}$$

The main idea of exploiting the partition of unity property is to construct basis functions through products of classical shape functions and a local enriched basis [4]. This means that, when a discontinuity appears (e.g. an element has been cut) the related nodes are enriched by a global discontinuous enrichment function with the product of the shape functions and the local enriched basis. Therefore, the equation of the displacements (cp. equation (2.27)) now can be calculated as

$$u(x) = \underbrace{\sum_{i=1}^{n} \Phi_i(x)\mathbf{u}_i}_{\text{classical}} + \underbrace{\sum_{j=1}^{n} \Phi_j(x)\psi_j(x)\mathbf{a}_j}_{\text{enrichment}} \tag{2.48}$$

where the discontinuous enrichment functions are denoted by $\psi_j(x)$, and the new DOFs as $\mathbf{a}_j$, commonly the number of new DOFs is the same than the element DOFs. The shape functions $\Phi_j(x)$, given in the enrichment side of the equation (2.48), can be different of the classical shape functions $\Phi_i(x)$ (e.g. higher order), in this thesis are considered to be both the same.

The enrichment function $\psi(x)$ can be any discontinuous function. If the support of a node is cut, commonly is enriched with Heaviside function, which is defined as

$$\Psi(x) = H(x) = \begin{cases} +1 & \text{above the crack} \\ -1 & \text{below the crack} \end{cases} \tag{2.49}$$

Another option is to use the *shifted* enrichment functions defined as follows

$$\psi_i(x) = \frac{1}{2}(H(x) - H_i) \tag{2.50}$$

where H_i is the value of Heaviside function at the *i-th* node. The shifted function consists in use the enrichment contribution only inside of the discontinuous element and ignores the contribution on the borders of the element and also outside the element (cp. Fig. 2.6)

The inclusion of enrichment function break with the delta property, then the displacements of the enriched nodes has to be computed as the sum of the components $\mathbf{u}_i + \Psi_i\mathbf{a}_i$. The nodal DOFs $\mathbf{u}_i$, when the generalized Heaviside enrichment function is used, loose their physical meaning requiring some adaptations to the equations. On the contrary, the shifted enrichment function directly records the values of displacements (enriched and non-enriched nodes) in $\mathbf{u}_i$; therefore, in a shifted function the added DOFs $\mathbf{a}_i$ are only required to establish the displacement of a enriched element. Figure 2.7 shows the physical meaning of DOFs in a discontinuous element according to the enrichment function.

The facility of the implementation depends on the choice of the enrichment function, more over, this choice impacts on the numerical stability of the simulation [22].

The equation (2.48) can be generalized to consider multiple cuts, to this end is required the inclusion of new enrichment function each time that a discontinuity is created and also new DOFs. Considering a element that has been cut twice, in which H^I,H^{II} are the enrichment functions for the first and the second cut respectively, the displacements can be calculated as

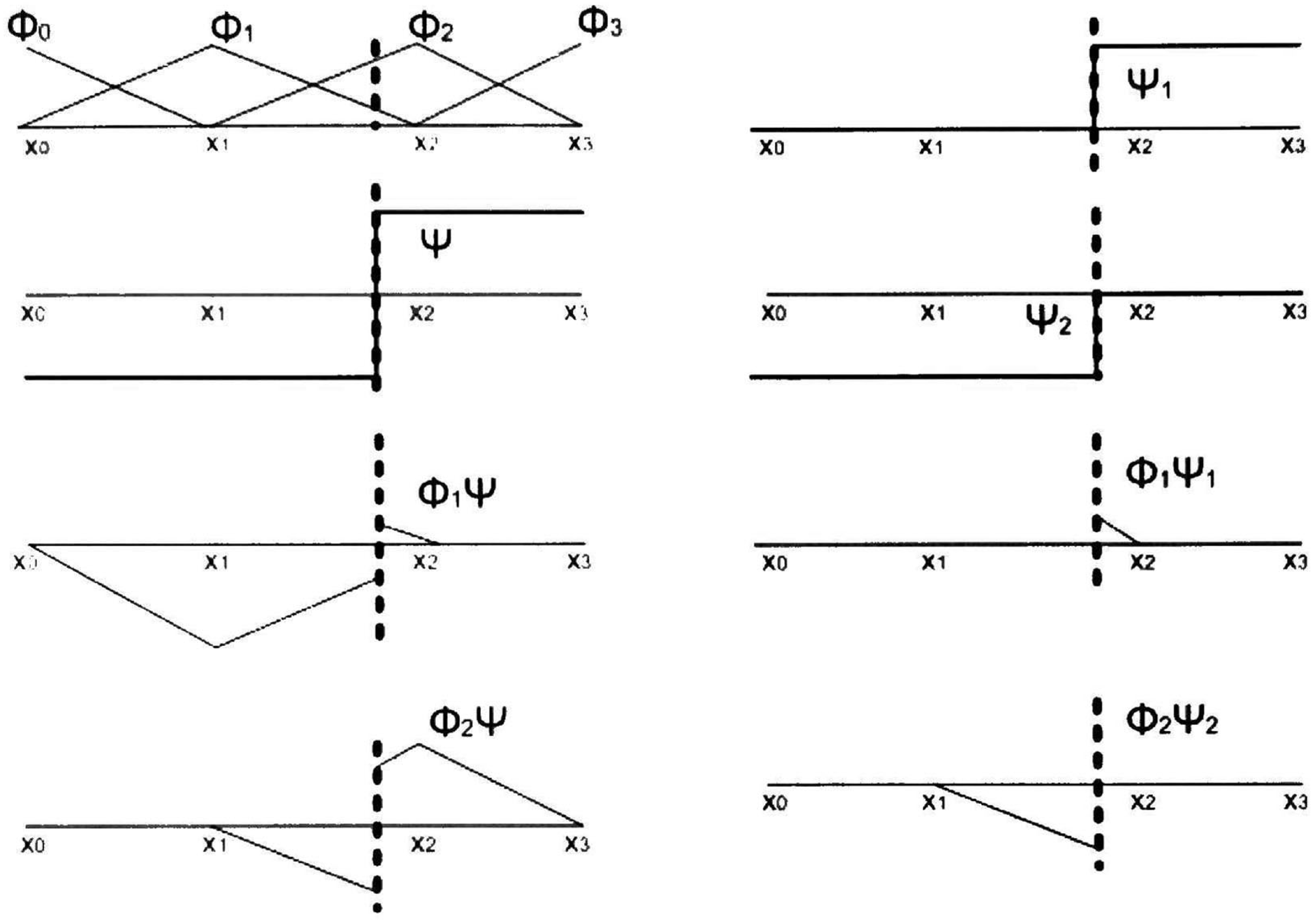

Figure 2.6: Enrichment functions. The Heaviside enrichment function on the left and the shifted function on the right.

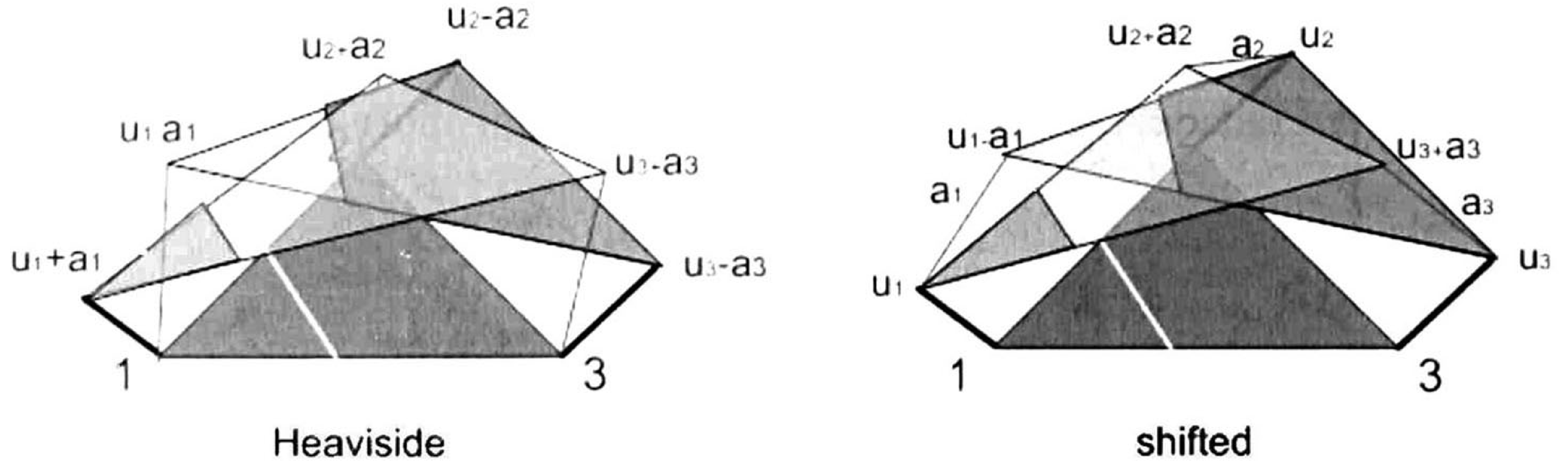

Figure 2.7: The enrichment functions have its own physical meaning of the nodal DOFs.

$$
\begin{aligned}
u(x) &= \sum_{i=1}^{n} \Phi_i(x)\mathbf{u}_i + \sum_{j=1}^{n} \Phi_j(x)\psi_j^I(x)\mathbf{a}_j^I + \sum_{j=1}^{n} \Phi_j(x)\psi_j^{II}(x)\mathbf{a}_j^{II} \\
&= \sum_{i=1}^{n} \Phi_i(x)\mathbf{u}_i + \sum_{j=1}^{m} \Phi_j(x)\psi_j(x)\mathbf{a}_j
\end{aligned}
\tag{2.51}
$$

where m represent the number of all added DOFs. The enrichment functions $\psi_j(x)$ are global as are H^I, H^{II}. The new cut must be independent from the previous cuts, this is shown in the Figure (2.8), in which H^{II} is zero on the below side of the cut I.

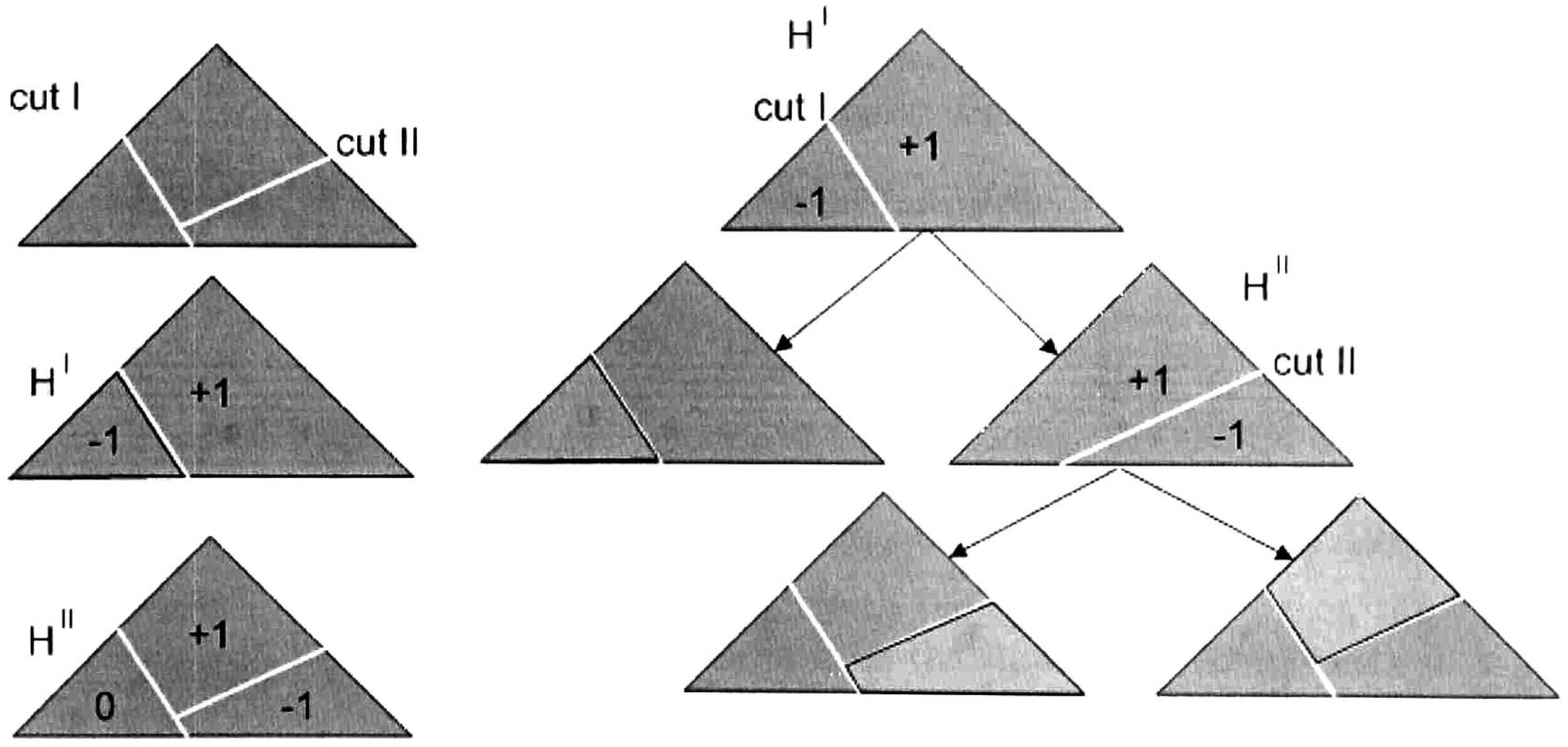

Figure 2.8: Multiple cuts per element. A new independent component is created for each cut (on the left); a hierarchical cut structure can be employed for more complex crack branching (on the right).

2.4.1 Linear XFEM

The vectors of the displacement (**u**) and force (**f**) of the linear XFEM, unlike the linear FEM, require to include new DOFs of those elements with discontinuities; these vectors can be expressed as

$$\mathbf{u}^X = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_n \ \mathbf{a}_1 \cdots \mathbf{a}_n]^T \tag{2.52}$$

$$\mathbf{f}^X = [\mathbf{f}_1 \ \cdots \ \mathbf{f}_n \ \mathbf{f}_1^a \cdots \mathbf{f}_n^a]^T \tag{2.53}$$

The stiffness matrix of the standard FEM, shown in equation (2.41), must consider the enriched strain matrix which is difined as

$$\mathbf{B}^X = [\mathbf{B}_1 \ \cdots \ \mathbf{B}_n \ \psi_1\mathbf{B}_1 \cdots \psi_n\mathbf{B}_n] \tag{2.54}$$

It is easily to see that the enrichment functions $\psi_i(x)$ is constant according to the side of the cut plane, which are the sides above (V_a) and below (V_b). Hence, equations can be divided into two components to integrate the volume of the whole domain. Thereby, the equations of the stiffness matrix yields

$$\mathbf{K}_{ij}^X = \underbrace{\int_{V_a} \mathbf{B}_i^{XT}\mathbf{c}\mathbf{B}_j^X \mathrm{d}V}_{\text{above}} + \underbrace{\int_{V_b} \mathbf{B}_i^{XT}\mathbf{c}\mathbf{B}_j^X \mathrm{d}V}_{\text{below}} \tag{2.55}$$

This enriched stiffness matrix, can also be express as a matrix of four components as follows

$$\mathbf{K}^X = \begin{bmatrix} \mathbf{K}^{uu} & \mathbf{K}^{ua} \\ \mathbf{K}^{au} & \mathbf{K}^{aa} \end{bmatrix} \tag{2.56}$$

where $\mathbf{K}^{uu}$ is the standard stiffness matrix that related to the original DOFs and $\mathbf{K}^{ua}$, $\mathbf{K}^{au}$ and $\mathbf{K}^{aa}$ are related to the new DOFs. Considering the strain matrix $\mathbf{B}$ as a constant, the four components of the stiffness matrix can be defined as

$$\mathbf{K}_{ij}^{uu} = \mathbf{K}_{ij} \tag{2.57}$$

$$\mathbf{K}_{ij}^{ua} = \left(\frac{V_a}{V}\Psi_{aj} + \frac{V_b}{V}\Psi_{bj}\right) \tag{2.58}$$

$$\mathbf{K}_{ij}^{au} = \left(\frac{V_a}{V}\Psi_{ai} + \frac{V_b}{V}\Psi_{bi}\right) \tag{2.59}$$

$$\mathbf{K}_{ij}^{aa} = \left(\frac{V_a}{V}\Psi_{ai}\Psi_{aj} + \frac{V_b}{V}\Psi_{bi}\Psi_{bj}\right) \tag{2.60}$$

where Ψ_{ai} and Ψ_{bi} indicates the value of $\Psi_i(x)$ above and below the cut plane respectively. When the shifted enrichment function, given in the equation (2.50), is used, the components of the enrichment stiffness matrix become

$$\mathbf{K}_{ij}^{uu} = \mathbf{K}_{ij} \tag{2.61}$$

$$\mathbf{K}_{ij}^{ua} = \begin{cases} -\frac{V_b}{V}\mathbf{K}_{ij} & \text{if } H_j = +1 \\ \frac{V_a}{V}\mathbf{K}_{ij} & \text{if } H_j = -1 \end{cases} \tag{2.62}$$

$$\mathbf{K}_{ij}^{au} = \begin{cases} -\frac{V_b}{V}\mathbf{K}_{ij} & \text{if } H_i = +1 \\ \frac{V_a}{V}\mathbf{K}_{ij} & \text{if } H_i = -1 \end{cases} \tag{2.63}$$

$$\mathbf{K}_{ij}^{aa} = \begin{cases} -\frac{V_b}{V}\mathbf{K}_{ij} & \text{if } H_i = H_j = +1 \\ \frac{V_a}{V}\mathbf{K}_{ij} & \text{if } H_i = H_j = -1 \\ 0 & \text{if } H_i \neq H_j \end{cases} \tag{2.64}$$

A disadvantage of the linear FEM is that only allows simulate deformations of small displacements. Therefore, the linear XFEM can be used to simulate partial cuts in stiff objects.

2.4.2 Corotational XFEM

The forces in the corotational FEM, shown in equation (2.42), must consider that the rotation of a discontinuous element will have different behavior in each part, then employing the equation (2.48), the deformation forces in an enriched element turn into

$$\begin{aligned} \mathbf{f}_i^X = & \sum_{j=1}^{n} \mathbf{R}_a \int_{V_a} \mathbf{B}_i^{XT}\mathbf{c}\mathbf{B}_j^X dV \; (\mathbf{R}_a^T \mathbf{p}_j^X - \mathbf{p}_{0j}^X) \; \} \text{ Above} \\ & + \sum_{j=1}^{n} \mathbf{R}_b \int_{V_b} \mathbf{B}_i^{XT}\mathbf{c}\mathbf{B}_j^X dV \; (\mathbf{R}_b^T \mathbf{p}_j^X - \mathbf{p}_{0j}^X) \; \} \text{ Below} \end{aligned} \tag{2.65}$$

Where the positions of $\mathbf{p}^X = \mathbf{p}_0^X + \mathbf{u}^X$ Considering an element, the rotation for the part above of the cut plane is denoted as $\mathbf{R}_a$ and the rotation below as $\mathbf{R}_b$; from above and using equation (2.49), the equations of the deformation forces of the standard DOFs and the added DOFs can be computed separately. Thus, the deformation force of the standar DOFs can be computed as follows

$$\mathbf{f}_i = \frac{V_a}{V}\mathbf{R}_a \sum_{j=1}^{n} \mathbf{K}_{ij}(\mathbf{R}_a^T \mathbf{p}_{aj} - \mathbf{p}_{0j}) + \frac{V_b}{V}\mathbf{R}_b \sum_{j=1}^{n} \mathbf{K}_{ij}(\mathbf{R}_b^T \mathbf{p}_{bj} - \mathbf{p}_{0j}) \tag{2.66}$$

where the positions of the element nodes are calculated

$$\mathbf{p}_{sj} = \mathbf{p}_{0j} + \mathbf{u}_j + \Psi_{sj}\mathbf{a}_j \text{ where } s = a \text{ or } s = b \tag{2.67}$$

where the subscript s indicates that the above equation can be applied for the parts above and below of the cut. The deformation force of the added DOF can be computed as follows

$$\mathbf{f}_i^a = \frac{V_a}{V}\mathbf{R}_a\Psi_{ai} \sum_{j=1}^{n} \mathbf{K}_{ij}(\mathbf{R}_a^T \mathbf{p}_{aj} - \mathbf{p}_{0j}) + \frac{V_b}{V}\mathbf{R}_b\Psi_{bi} \sum_{j=1}^{n} \mathbf{K}_{ij}(\mathbf{R}_b^T \mathbf{p}_{bj} - \mathbf{p}_{0j}) \tag{2.68}$$

Employing the shifted enrichment function (cp. equation (2.50)) into equation 2.68 , the deformation forces yield

$$\mathbf{f}_i^a = \begin{cases} \sum_{j=1}^{n} -\frac{V_b}{V}\mathbf{R}_b\mathbf{K}_{ij}(\mathbf{R}_b^T \mathbf{p}_{bj} - \mathbf{p}_{0j}) & \text{if } H_i = +1 \\ \sum_{j=1}^{n} \frac{V_a}{V}\mathbf{R}_a\mathbf{K}_{ij}(\mathbf{R}_a^T \mathbf{p}_{aj} - \mathbf{p}_{0j}) & \text{if } H_i = -1 \end{cases} \tag{2.69}$$

and the positions of the element nodes

$$\mathbf{p}_{aj} = \begin{cases} \mathbf{p}_{0j} + \mathbf{u}_j & \text{if } H_j = +1 \\ \mathbf{p}_{0j} + \mathbf{u}_j + \mathbf{a}_j & \text{if } H_j = -1 \end{cases} \quad \mathbf{p}_{bj} = \begin{cases} \mathbf{p}_{0j} + \mathbf{u}_j - \mathbf{a}_j & \text{if } H_j = +1 \\ \mathbf{p}_{0j} + \mathbf{u}_j & \text{if } H_j = -1 \end{cases} \tag{2.70}$$

For purposes of this thesis the nonlinear XFEM is not required to be described (the formulation is in [22]))

2.4.3 Mass Lumping Techniques

The stability and the accuracy of the simulation are harmed if there are sliver elements. Slivers can be caused if one part of the element is so small that the volume is almost zero. This problem happens in explicit and implicit methods. However, [21] declare that the simulation stability can be impacted by the choice of the enrichment functions and the mass lumping technique.

The enriched mass matrix can be represented with four components (similar to the stiffness matrix) as follows

$$\mathbf{M}^X = \begin{bmatrix} \mathbf{M}^{uu} & \mathbf{M}^{ua} \\ \mathbf{M}^{au} & \mathbf{M}^{aa} \end{bmatrix} \tag{2.71}$$

where each component is defined by

$$\mathbf{M}_{ij}^{uu} = \int_V \rho \Phi_i \Phi_j \mathrm{d}V \tag{2.72}$$

$$\mathbf{M}_{ij}^{ua} = \int_V \rho \Phi_i \Phi_j \Psi_j \mathrm{d}V \tag{2.73}$$

$$\mathbf{M}_{ij}^{au} = \int_V \rho \Psi_i \Phi_i \Phi_j \mathrm{d}V \tag{2.74}$$

$$\mathbf{M}_{ij}^{aa} = \int_V \rho \Psi_i \Phi_i \Phi_j \Psi_j \mathrm{d}V \tag{2.75}$$

Note that the lumped matrices $\mathbf{M}_{ij}^{ua}$ and $\mathbf{M}_{ij}^{au}$ are zero. Therefore, is only necessary to compute the lumped matrices of $\mathbf{M}_{ij}^{uu}$ and $\mathbf{M}_{ij}^{uu}$ There are three principal techniques to lump the matrices such as *row summation*, *weighted diagonal* and *enrichment lumping technique.* The lumped mass matrices using row summation yields

$$\mathbb{M}_{ii}^{uu} = \sum_j \mathbf{M}_{ij}^{uu} \tag{2.76}$$

$$\mathbb{M}_{ii}^{aa} = \sum_j \mathbf{M}_{ij}^{aa} \tag{2.77}$$

The weighted diagonal technique is computed as follows

$$\mathbb{M}_{ii}^{uu} = m \frac{\mathbf{M}_{ii}^{uu}}{\sum_j \mathbf{M}_{ij}^{uu}} \tag{2.78}$$

$$\mathbb{M}_{ii}^{aa} = m \frac{\mathbf{M}_{ii}^{aa}}{\sum_j \mathbf{M}_{ij}^{aa}} \tag{2.79}$$

where m is the element total mass Finally the enrichment lumping technique, proposed by [29], is applied in explicit integration time and is computed as follows

$$\mathbb{M}_{ii}^{uu} = \frac{m}{n} \tag{2.80}$$

$$\mathbb{M}_{ii}^{aa} = \frac{\rho}{n} \int_V \psi_i^2 \mathrm{d}V = \left(\frac{V_a}{V} \Psi_{ai}^2 + \frac{V_b}{V} \Psi_{bi}^2 \right) \mathbb{M}_{ii}^{uu} \tag{2.81}$$

where n indicate the number of element nodes.

[22] analyze the lumping techniques and the enrichment function in order to determine which produces more stability on the simulation. [22] conclude that using the shifted function and the enrichment lumping technique the slivers are avoided and therefore ensuring stability on the simulation.

2.4.4 Discussion

Even though in [21] had already employed the XFEM for virtual surgery, their proposal is generalized to simulate all kind of elastic materials, moreover ignores how the XFEM interacts with the others methods of CG (cp. Figure 2.1), in other words, they don't indicate how to establish a mapping between the visual mesh and the physical topology when the discontinuities appear, how to proceed if the cut touches over a tetrahedron vertex, how the forces of discontinuous elements are not affected when collide, how must be treated the XFEM equations considering discontinuous and non-discontinuous neighbors, how to realize self-collisions among discontinuous elements and how to do efficiently the cutting process.

In this work sets the XFEM as a core of the simulation and around it we select and create methods of CG that allow us to work in an efficient manner; that is, if the methods are customized to the XFEM requirements, the generated results will be better and more efficient; furthermore by this way we can design a framework specifically for a surgery simulation that demands more accuracy and in the same time interactivity in real time.

Has been decided to work with SOFA framework [2] because it contains a set of methods of CG allowing the selection and modification of the algorithms in accordance with our needs; in addition SOFA contains an intuitive structure that is possible to create and use new methods easily; SOFA is a project that continues in development, novel methods and more efficient will be included allowing in a future be adapted to work with the XFEM.

Chapter 3

XFEM framework for cutting soft tissue

In the previous chapters have been described different approaches that deal with topological changes; now, our objective is not to create one more application, instead, we aim to create the bases for all surgery simulations that include topological changes.

This chapter starts with a description of the problem to deal, beginning with a short resume of the research presented in previous chapter and concluding with specific questions that gives place to this research. Consequently, the methodology is described starting with the adaptations to the corotational XFEM; it is introduced since the FEM equations; right after, is explained the mapping method which is the key algorithm to unify all others algorithms with the XFEM. Also the algorithm for cutting is presented and finally the adaptations of the visual update and collision algorithms are explained.

3.1 Problem description

There are a wide range of surgical applications, and some of these are capable to cut the tissue; nevertheless, there are four principal factors required in a robust virtual surgery: *accuracy*, *interactivity*, *stability* and *real-time*; which are shown in Figure 3.1. The related work described in the previous chapter shows how the approaches achieve some of these factors but not all, remaining in most of the cases unstable or inaccurate simulations.

At the beginning, the visualization of 3D models was the objective, where the models are constructed through images generated by a computer tomography, attempting to represent these models as real as possible and therefore, this kind of applications helps to analyze a specific patient (e.g. a tumor in the brain). Later, the research turn towards an interactive applications forcing to simulate the behavior of the models (e.g. interactive endoscopy). The

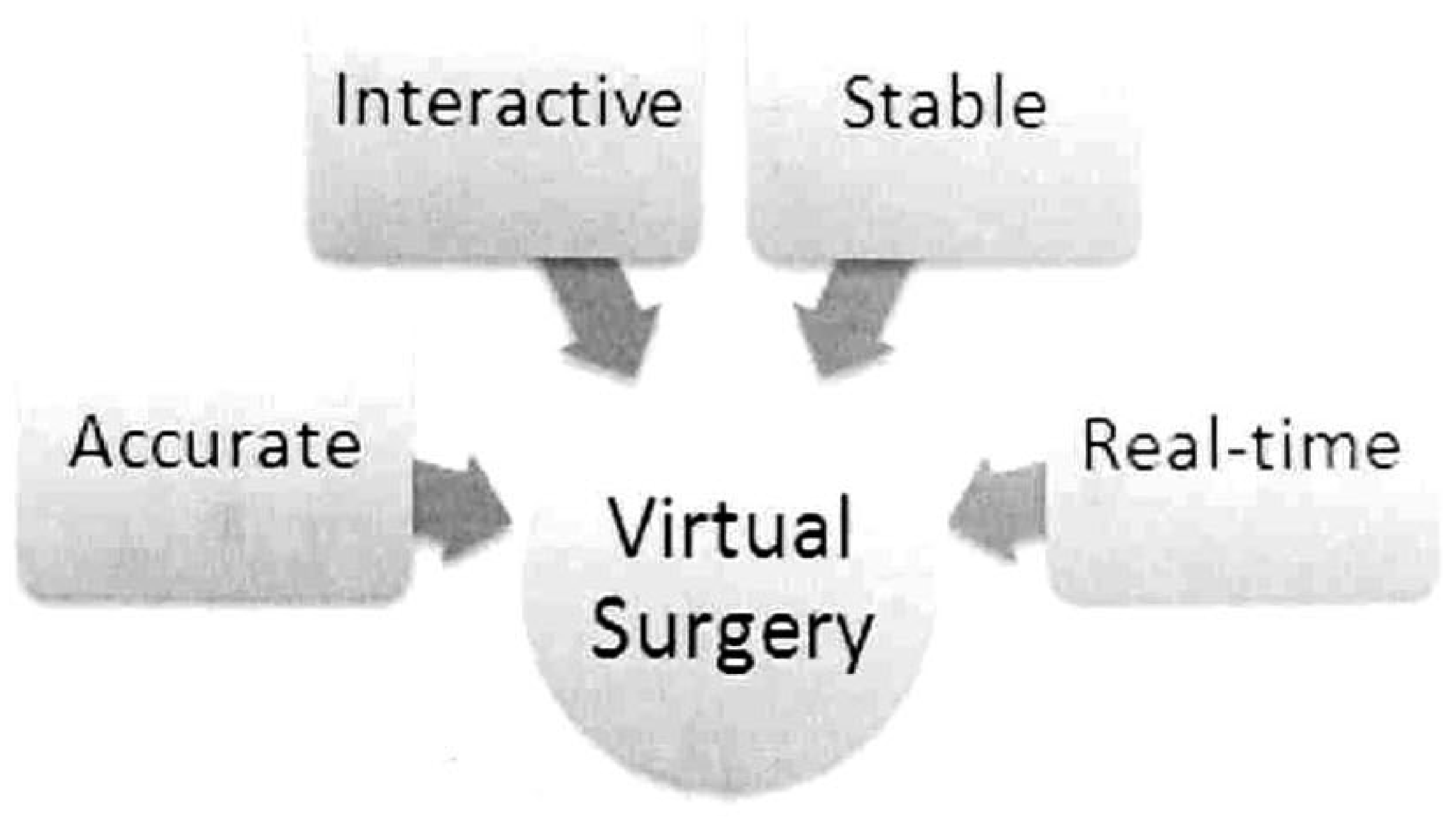

Figure 3.1: Principal factors that contains a robust simulation. All this factors are demanded for all surgery simulations.

desire of realistic simulations obligate to use accurate models boosting the use of physical based methods (cp. section 2.2.1). Nevertheless the use of those physical models is opposite with the interactivity in real time, then the research focuses on realistic animated simulations such as [32, 44]. However, the opportunity to create training simulations turned back the attention in interactive applications, resulting in many approaches that adequate the equations or make hybrid simulations in order to obtain interactivity in conjunction with physical models [12, 7]. Afterward, the medical simulations demand the inclusion of complex procedures that require the simulation of topological changes (e.g. cutting the skin [8, 42]); although, many approaches deal with this problems, the most of these obtain a simulation that fails with at least one of the factors shown in Figure 3.1.

[21] attempts to achieve these important aspects, leading into a stable and accurate simulation. Nevertheless, their approach is very general, explained for only one tetrahedral element and omitting how the XFEM must interact with other methods of CG without losing its advantages.

[27] create a framework for surgery simulation and also considers the XFEM for arbitrary splitting the elements, proposing algorithms for collision detection, visual update, cutting and the processing distribution in threads; although it is used in medical applications and employs the XFEM, this framework implements the linear XFEM which only support small deformations and moreover, the XFEM is not the base of the simulation, therefore, the algorithms proposed do not use the advantages provided by the XFEM.

Our principal objective is to create a specific framework that fully exploits the advantages of the XFEM and, by setting the XFEM as a core of the simulation, create more complex surgery simulations which we know will be stable, accurate and interactive in real-time. Therefore, this thesis, will give answer to the following questions:

- What kind of XFEM use?

- How works the discontinuous element with its neighbors (split/non-split)?
- How can be created an efficient mapping between the visual mesh and the physical mesh topology, considering changes in the topology specific to the XFEM?
- Which algorithm can we use to cut the tissue?

All the questions presented above have a direct impact in the performance and realism of the simulation, this means that the choice of linear, corotational or nonlinear XFEM will decide the capability of deformations; this decision can increment or decrement the performance. Besides, the manner that a discontinuous element associates its added DOFs to the neighbors, considering if those neighbors are continuous or discontinuous, can affect the computational effort by increasing the added DOFs when it is not necessary. Moreover, when an element is divided, the new "virtual elements" must update their corresponding displacement; also, it has to be considered that the physical mesh is associated to the visual mesh, which is updated each time step. Finally the cutting algorithm mush be designed looking for visual update in real time, therefore, this algorithms must be efficient for finding collided elements, identifying intersections between the interactive tool and the elements, and for allowing all the others algorithms to be executed in real time.

Basing the simulation in a robust method is the key to ensure efficient responses working together with others algorithms of CG; thereby, allowing to reach the requirements that a surgery simulation demands.

3.2 Methodology

The linear XFEM, used by [27], is fast but unfortunately does not allow to simulate large deformations; on the other hand,considering the nonlinear XFEM allows large deformation but it increase the computational effort, complicating the interactivity in real time. Therefore, we prefer the corotational XFEM, allowing large deformations without harming the performance.

If there are at least two discontinuous elements that share the split edge (i.e. both elements were cut in the same edge), these elements will share the added DOFs of the adjacent vertex, avoiding adding new DOFs. On the other hand, a discontinuous element with a non-discontinuous element are ignored the values of the forces on the added DOFs of the shared edge.

In order to create a mapping among the original physical topology, the visual topology and the collision topology considering the changes when a discontinuity appears, will be created a new alternative mesh topology that will control the discontinuous elements, and will update the value of the displacements calculated in the original mesh topology. This mesh topology will be associated with the visual mesh topology and thereby, allow us to show the corresponding portion of volume in the divided elements. The mapping between the visual mesh and the alternative mesh can be easily based on the barycentric coordinate of the tetrahedrons.

In order to the cut the tissue must be designed an algorithm that considers this mapping methods and exploits its capacity to search elements in conjunction with the collision detection algorithms. When the mesh is well refined a semi-progressive cut is visually acceptable. The cutting method creates a list that records the touched elements and quickly find a full cut of an element.

3.2.1 Adaptations to the corotational XFEM

The FEM corotational has been used for some approaches such as [30, 31, 18], showing that is physically accurate and suitable for real-time applications. However, it is possible to compute the rotation matrix in different manners. However, The estimation of the rotation matrix can impact the accuracy of the simulation. Therefore, we based the fundaments of the corotational FEM in the approach proposed in [37]. The rotation matrix for a tetrahedron (i.e. 3x3 matrix), is computed considering the positions of the vertices in the undeformed configuration.; in this configuration the barycentric coordinates of a point $\mathbf{p}$ satify

$$\begin{bmatrix} p_1^x & p_2^x & p_3^x & p_4^x \\ p_1^y & p_2^y & p_3^y & p_4^y \\ p_1^z & p_2^z & p_3^z & p_4^z \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = \begin{bmatrix} p^x \\ p^y \\ p^z \\ 1 \end{bmatrix} \tag{3.1}$$

where β_i are the barycentric coordinates of the point $\mathbf{p}$. The above equation can be expressed as $\mathbf{P}\beta = \mathbf{p}$. Thereby, a tetrahedron from the perspective of the deformed configuration yields

$$\mathbf{Q}\beta = \mathbf{q} \tag{3.2}$$

where $\mathbf{q}$ is the point equivalent to $\mathbf{p}$; that is, in the deformed and the undeformed configuration respectively. β represents the same barycentric coordinates that corresponds to the deformed tetrahedron. Using equations (3.1) and (3.2) yields

$$\mathbf{q} = \mathbf{Q}\beta = \mathbf{Q}\mathbf{P}^{-1}\mathbf{p} = \mathbf{A}\mathbf{p} \tag{3.3}$$

The transformation of the tetrahedron is described by the matrix $\mathbf{A} = \mathbf{Q}\mathbf{P}^{-1}$, which can be expressed as

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{t} \\ 0\,0\,0 & 1 \end{bmatrix} \tag{3.4}$$

where $\mathbf{t}$ stores the translational component and the components of rotation and stretching are stored in $\mathbf{B}$. To obtain the rotation matrix we do it by a polar descomposition of $\mathbf{B}$ as is proposed in [31].

Considering a discontinuous element divided into two parts, the rotation matrix is computed for each part; thereby, the number of rotations in a discontinuous element will be the number of subdivisions in this element as is show in equation (2.65).

In order to calculate the forces efficiently, as is propose in [37], these are computed in separate components; this means that the stiffness matrix will not be stored, instead of this; the rotations, the strain matrix, and the displacements are stored and computed in different times. In the case of the XFEM, for each part of a discontinuous element (i.e. above and below the cut plane), it is required to store all of these components.

3.2.2 Mapping Method

In CG is widely used the barycentric coordinates to create a mapping between two different meshes. The mapping consist of associate a specific points (i.e. vertexes of another mesh) with the elements of another mesh topology. Therefore, first is calculated the barycentric coordinates and consequently the bary-coefficients of the point are computed. The point associated must update its position each time the tetrahedron has changed (e.g. has been deformed or displaced). The barycentric mapping is shown in Figure 3.2.

The applications that only simulate the elastic behavior of the objects, i.e. do not consider topological changes, have a physical visual mapping where can be applied the barycentric mapping. Let's consider a cube model, which is divided in a tetrahedral and triangular mesh that corresponds to the physical and visual topology respectively, as is shown in Figure 3.3. The visual mesh that corresponds to the surface of the cube, must be updated according to the deformation of the cube, hence, each node of the visual mesh must be assigned to one tetrahedron (i.e. the closest tetrahedron). Note that one tetrahedron of the physical topology can have assigned more than one triangles of the visual topology; this situation depends of the refinement of both topologies.

Now, considering topological changes, the XFEM adds DOFs each time a discontinuity appears remaining the number of elements of the original mesh; thus, an element is divided

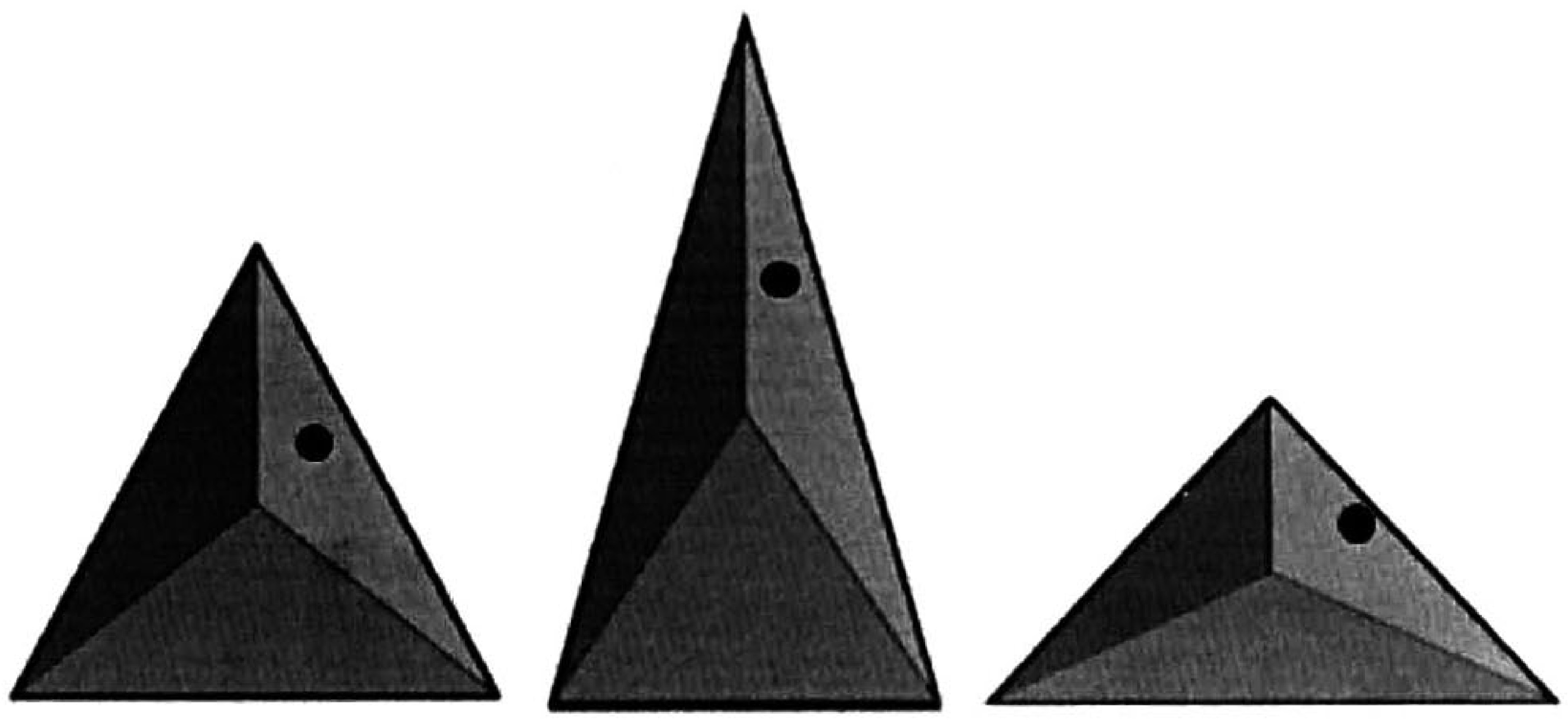

Figure 3.2: The black point is associated to the triangle by barycentric mapping. The barycoefficients are calculated to the point and when the triangle is deformed the barycoefficients indicate where the point should move.

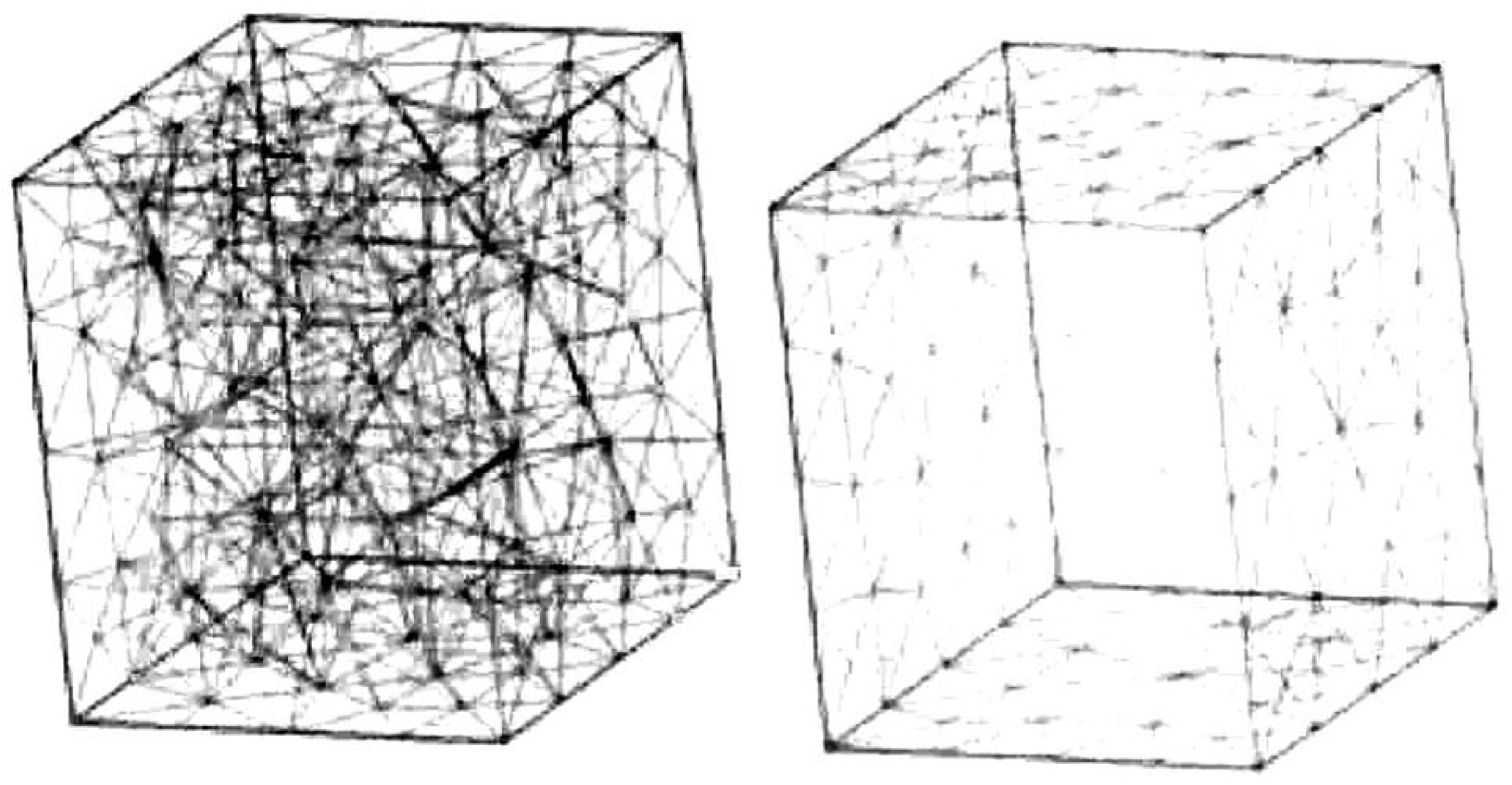

Figure 3.3: The same cube is divided into two different topologies: tetrahedrons for the physical topology (left) and triangles for visual topology (right)

using the values of the added DOFs. Then, how should be associated the triangular mesh to the tetrahedrons that do not explicitly exist? If the mapping is done to the original tetrahedron (that has been cut) the result will not be accurate because this tetrahedron doesn't show the same physical meaning of the original mesh anymore.

We propose a FEM-XFEM mapping that allows us easily control the discontinuous elements. This new mapping consists in creating an alternative tetrahedral mesh topology that inserts tetrahedrons each time the element is subdivided. The alternative mesh will have always the same number of DOFs that the original mesh; where the original DOFs are copied directly to the DOFs of the new topology, however the new topology will substitute the values of the added DOFs (i.e. only in the new topology) by true value of the vertexes of the new tetrahedrons (created by the split); i.e. the displacements of the *virtual elements* are composed by original and new DOFs and the total value is stored in the new DOFs location of the new mapping, thereby, there is not necessary to add more vertexes in the new mesh topology.

For example, considering a single tetrahedron the original mesh will have twelve DOFs (i.e. three for each vertex), if the element is cut into two parts, the original mesh will have 24 DOFs or 8 nodal displacements that can represent two tetrahedrons; where the first four nodal displacements have the values of the original tetrahedron vertexes and the other four contains variations on the displacements that helps to make the effect of splitting the element as is shown in Figure 3.4. The new mesh topology will conserve the first four nodal displacements because these values are directly applied to the position of the vertexes of the new virtual elements, and the other four nodal displacements will be substituted with the real position of the missing vertexes to specify.

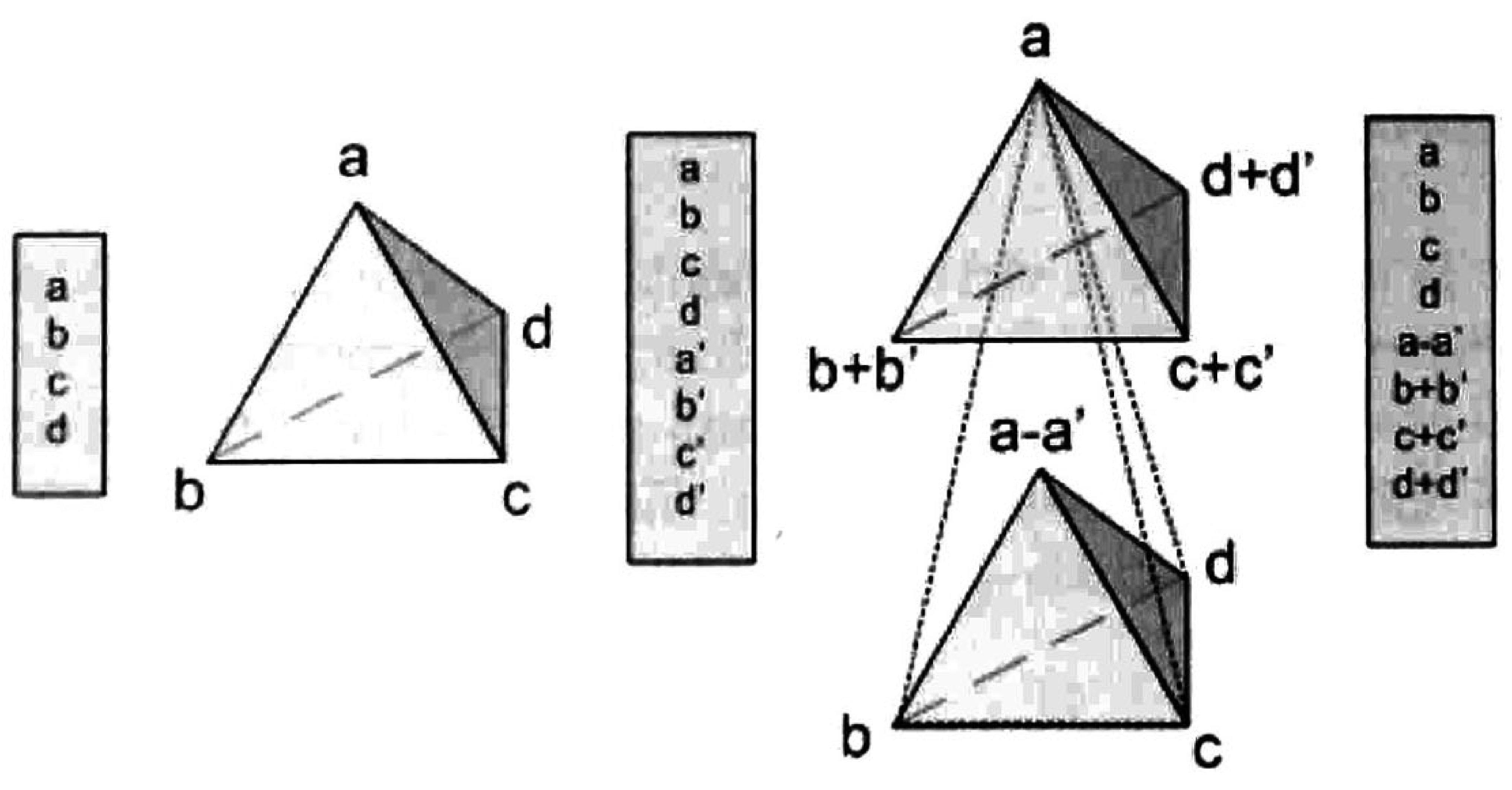

Figure 3.4: Added DOFs. The original nodal displacements (on the left) of a tetrahedron are four, when the tetrahedron is split new DOFs are added (list on the middle); in order to use the less memory, the alternative mesh stores in the place of the added DOFs the real position of the vertexes of the virtual tetrahedron(on the right).

Note that this *alternative mesh topology* is not the visual mesh; i.e. as the physical topology, the XFEM topology is not visible. The visual topology is mapped to the alternative topology and this last updates the values according to the original topology (that stores the results of the XFEM equations). The visual and collision meshes are mapped in the same manner with the alternative mesh.

It is important to mention that the mapping is not completely on the whole surface of the virtual elements, because the virtual elements will show just its corresponding portion of material; moreover, there are some situations that must be considered; for example, when an element has been cut, right after the virtual elements will appear exactly in same position; hence, the virtual elements must be classified according to the side of the cut plane (i.e. above or below). Thereby, the visual mesh node is assigned to the nearest tetrahedron that corresponds with the side of the cut.

Another important advantage of the alternative topology is its speed that takes to reassign the visual or collision mesh nodes to its corresponding physical tetrahedron when a cut occurs; in other words, if a tetrahedron is cut, there is a record of the points associated to the tetrahedron that will be destroyed in the alternative mesh, after are reassigned the new points to the corresponding virtual tetrahedron without checking all the tetrahedrons.

The process of using the alternative topology is shown in Figure 3.5.The alternative mesh is controlled directly from the physical topology; in this manner all the changes in the alternative mesh are quickly performed and also the search of associated elements and the copy of the DOFs are fairly straightforward. Another important remark about the alternative topology is that all its "DOFs", are never used in the equation (2.39) such as the original DOFs; thereby, the alternative mesh doesn't harm the performance of the simulation.

Controlling the forces

If an external force is applied to a virtual element, this force is directly propagated through the neighbors that shares its DOFs. Nevertheless, if an element is cut and all its neighbors aren't virtual elements then the virtual elements must be fixed to the adjacent edges (that are not cut); to this end, it is necessary to annul the forces on those DOFs that are connected to a non-discontinuos neighbor as is shown in Figure 3.6.

3.2.3 Cutting algorithm

There are three principal kinds of cutting algorithms, the non-progressive, progressive and semi-progressive. The non-progressive cut requires in the most of the cases the specification of the fist and the last point of cut, even though it is fast and efficient, it is not plausible for surgery simulations because is not truly interactive. The progressive cut shows the

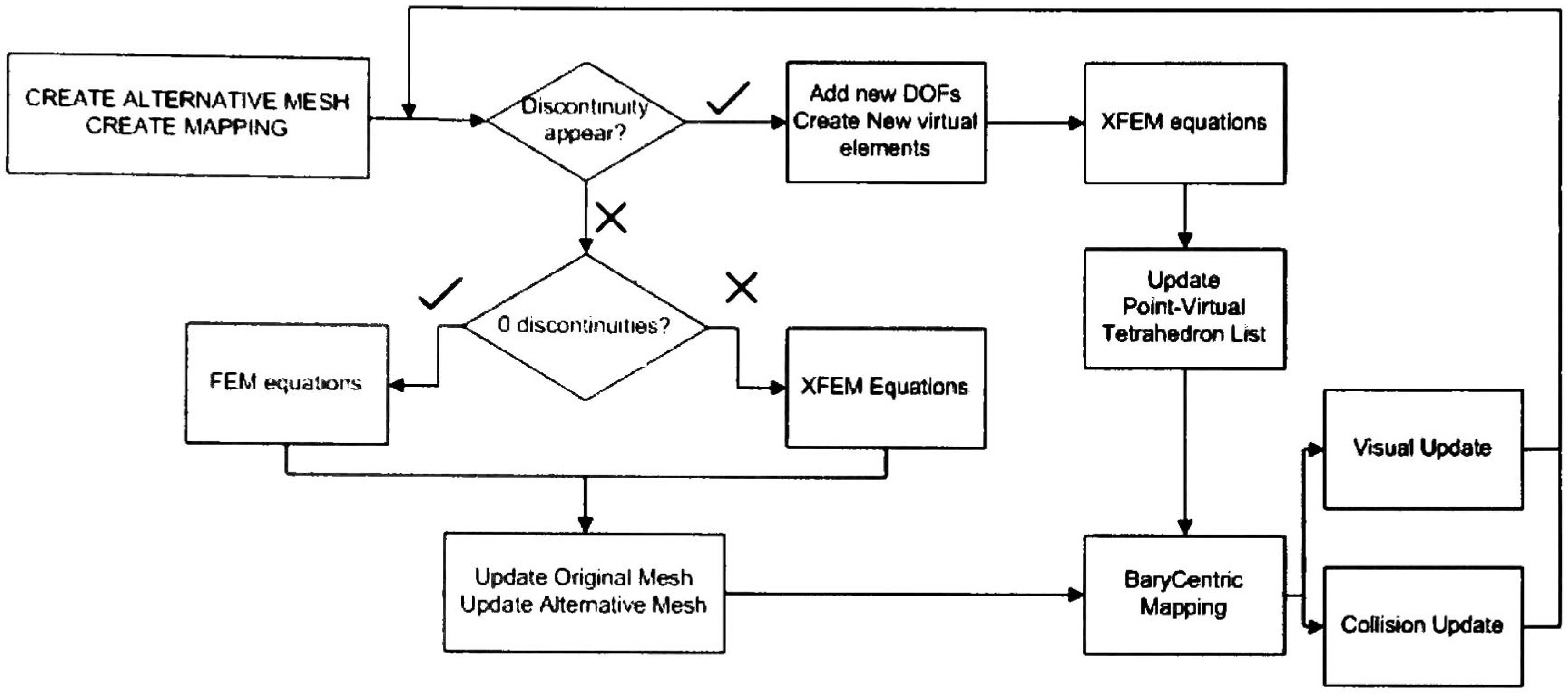

Figure 3.5: Basic process of the XFEM and the alternative mapping method.

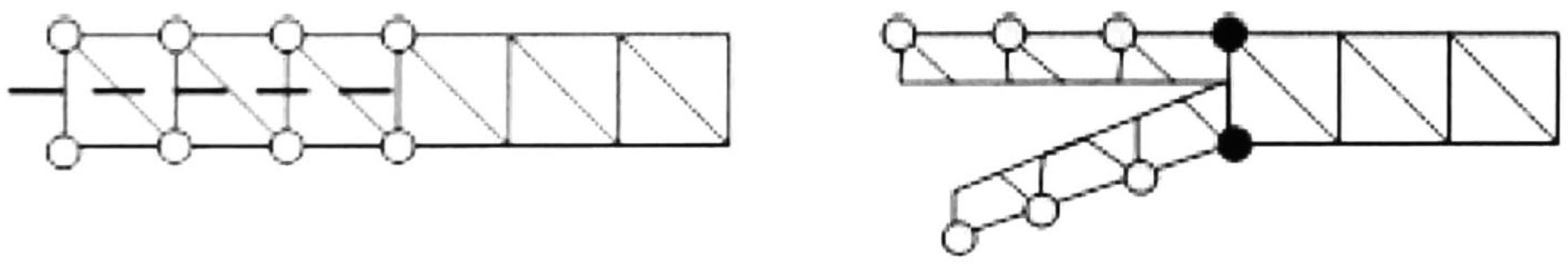

Figure 3.6: The original mesh is on the left; when the tool cuts the mesh, the discontinuous elements are opened, however, they must be fixed if one of its neighbors doesn't have a cut.

subdivision of the element at the same time while being cut; however, this technique is computationally expensive and may require the control of non-linear elements. On the other hand, if we split the tetrahedron when the tool crosses the whole element is considered semi-progressive cut because there is a delay before showing the incision. For a well refined mesh the delay in the semiprogressive cut can·be considered visually insignificant.

When the interactive tool touches a tetrahedron (indicated by the collision detection algorithm), firstly is obtained the collided point and is searched the nearest tetrahedron to this point, after that, all the neighbors of the tetrahedron are stored in a list which is used to find the next tetrahedron touched by the cutting tool; then, are obtained the new neighbors and also are stored in a list. Having two points of cut (both are very near) is created an intersection *quad* which is created using the deepness value. The last created list is used to find intersection between the tetrahedral edges and the quad, the intersection of a vertex are stored to identify if a tetrahedron has been completely cut, if this last happens, a new list will store the elements to split. After checking all the neighbors of the collided tetrahedron, are divided all the tetrahedrons of the list of elements to cut.

```
SEMI-PROGRESSIVE CUT

create L1 // list of tetrahedrons with one intersection
create L2 // list of tetrahedrons with two intersections
create L3 // list of tetrahedrons to cut

collisionEvent (collisionPoint)
{
  if the cut starts
    FirstPoint = collisionPoint
    T = nearest (Mesh, FistPoint) //return the tetrahedron neares to firstPoint
  else //while cutting
    listNT = neighbors(T); //list of neighbors of T
    Tc = nearest (listNT, fistPoint)
    SecondPoint = CollisionPoint
    semiProgressiveCut(Tc)
}

semiProgressiveCut(T)
{
  generatequad()// collision quad of the cut plane
  lT = neighbors(T)
  for each tetra i in lT
      if EdgeIntersection(T, quad)
```

```
        if exists T in list L2
          L3.add(L2.point1,L2.point2,intersectionPoint)
          L2.delete(T)
        else
          if exists T in list L1
            L2.add(L1.point,intersectionPoint)
            L1.delete(T)
          else
            L1.add(T,intersectionPoint)
            L1.add(T,intersectionPoint)


    for each tetra j in L3
        createXFEM(j)

    L3.clear
    FirstPoint = SecondPoint
}
```

As shows the previous algorithm, the expensive work is at the beginning of the cut, the reason is that the nearest tetrahedron is searched among all the tetrahedrons of the mesh. To avoid this, we can make use of the collision algorithm; i.e. the collision detections can give as an output an index of an element that corresponds to the collision mesh, therefore, as we have recorded the assignation of the points, we can find directly the tetrahedron associated to the collision element; thereby, is possible to extract quickly the list of neighbors for the collided region.

When an element is divided into two new virtual elements, the visual and collision mapping methods are called to be updated. Therefore, the visual and collision meshes must also divide its element according to the cut plane and be assigned to its corresponding virtual element.

3.2.4 Visual mapping

If the physical mesh changes its topology then the visual mesh also must be adapted to show this changes; with this objective, it is considered the barycentric mapping, which is applied to only show the portions of material associated to the virtual elements; however, to create the mapping, the portion of material is not truly analyzed, instead, is employed the cut plane. To show how the object is opening when is cut, it can be used the snapping or the subdivision. We choice for the subdivision can be performed faster than the snapping.

However, each time that an element is cut are created at the most four new points, despite the increment, the points are not DOFs and do not strongly harm the performance of the simulation.

Working with 3D models, must be considered the deepness, therefore, when a cut happens not only the surface must be mapped; also an internal subdivision must be created and mapped. In the Figure 3.3, the visual cube mesh is only the surface, therefore, there are new faces that must be created and displayed according to the deepness of the cut; to this end, we create the new mesh employing the intersections points of the edges of the virtual tetrahedrons, in this manner the remesh is straightforward by connecting the points as is shown in Figure 3.7

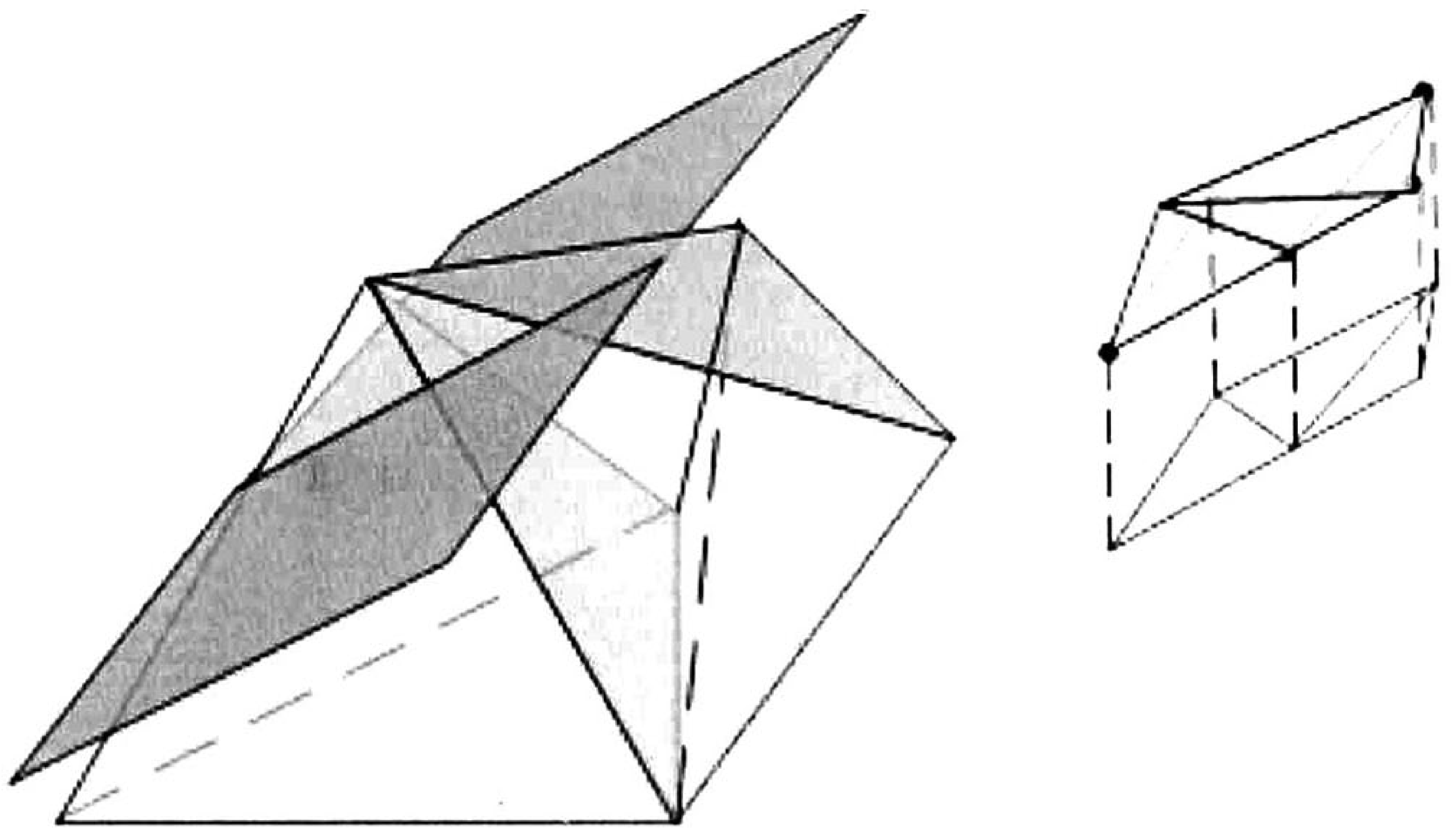

Figure 3.7: Connecting intersected points in order to create the internal mesh.

Note that if the material is very thin, it is possible that the cut crosses the both sides of the body, requiring the adaptation and the remapping of the both faces.

3.2.5 Collision Detection

The collision detection algorithm must be mapped to the alternative mesh and, while the user is interacting, if appears a collision, then the values of the forces will be sent directly to the corresponding tetrahedrons without worrying about the kind of element (i.e. if it is discontinuous or not) because the alternative mesh will propagate the force value to the corresponding original tetrahedron in order to follow with the XFEM equations. The mapping of the collision mesh considers only the visible parts of virtual elements; for this reason, the collision mesh do not covers the whole surface of the tetrahedron allowing to collide only in the active areas, as is shown in the Figure 3.8.

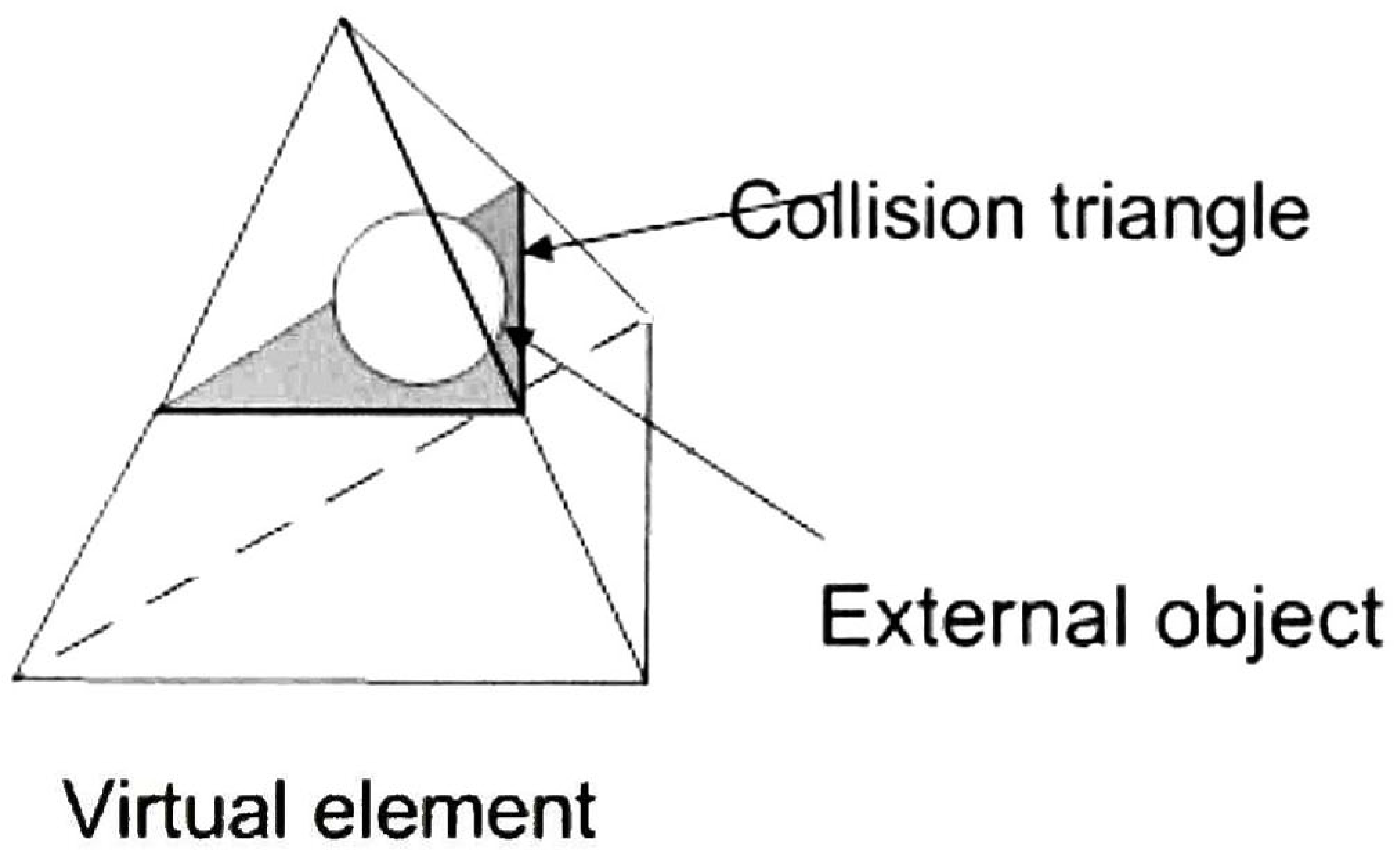

Figure 3.8: The collision model is mapped to the alternative mesh, not necessarily on the surface of the elements, it can be in the corresponding part of volume. The external object can overlap the elements, collides in the collision triangle.

Chapter 4

Implementation and Testing

This chapter has the objective to verify in specific simulations that the presented approach can be applied to generate robust simulations, especially for a surgery simulation. The factors described in the previous chapter (cp. Figure 3.1) are now considered to evaluate the approach. Moreover, all the algorithms presented in the previous chapter are analyzed.

Moreover, in this chapter is showed the implementation of the approach using the SOFA framework architecture. The chapter is divided as follows; it starts with the description of the case of study, followed by an explanation of how to implement the XFEM inside SOFA; afterwards the implementation of the XFEM in 2D and 3D is presented describing the important aspects considered to achieve the objective.

4.1 Case of study

In computer graphics there are a lot of soft materials that can be simulated as, e.g., hair, cloth and ruber; however, our perspective is another, we aim to achieve with the exigencies that a virtual surgery involves. In order to evaluate the approach in a medical context, the responses of the approach are tested in an open surgery. The approach can be analyzed specifically by simulating the skin cutting; thus, is possible to observe the responses of the XFEM exposed to many elements that are small and thin (i.e. low volume), and thereby, the robustness of the approach can be confirmed.

The simulation of the skin is important because the size of the elements impacts directly on the simulation performance when the elements are cut; i.e., if the cut is near to a vertex, this can generate sliver elements.

In order to interact with the simulation, the user can employ the mouse pointer in conjunction with the keyboard; in spite of the fact that the interaction can be limited to 2D, the angle of the user vision is stored; in this manner, the mouse can simulate a tool that is in the same direction to the angle of vision.

We create two different cases of study to analyze the existing approach; the first one is implemented in a two dimensional object, and the second is tested for 3D object.

The material parameters of the skin are described in the table 4.1, this values are copied from [38].

Parameter	**Notation**	**Value**
Young's modulus	E	$1.0 \cdot 10^4 Pa$
density	ρ	$1000 kg/m^3$
Poisson ratio	v	0.3
gravity	g	$9.8 m/s^2$

Table 4.1: This table shows the material parameters considered to test the approach.

4.1.1 2D: Human Face skin

A human face model, represented with triangular elements, is inserted in the 3D environment; in the case of 2D, the collision detection, the visual model and the physical topology are the same triangular mesh of the face.

The properties of the mesh comply the restrictions of FEM described in section 2.3.2 (i.e. all the elements are connected; without gaps and overlapping).

When the simulation starts the face model can be deformed interactively in real time and the user can specify where to cut the skin. To evaluate this case the following aspects are considered:

interactivity: the user should be capable to deform the model from any point and look the reactions at the same time it is doing.

cutting: analyze the delay of cutting a triangle while the user is cutting.

realtime: the simulation performance is examined while the user is cutting.

realistic physical responses: the responses has to be physically plausible, to this end, we compare the responses with other approaches completely based in the FEM but not fully interactive (i.e. snapping and subdivision).

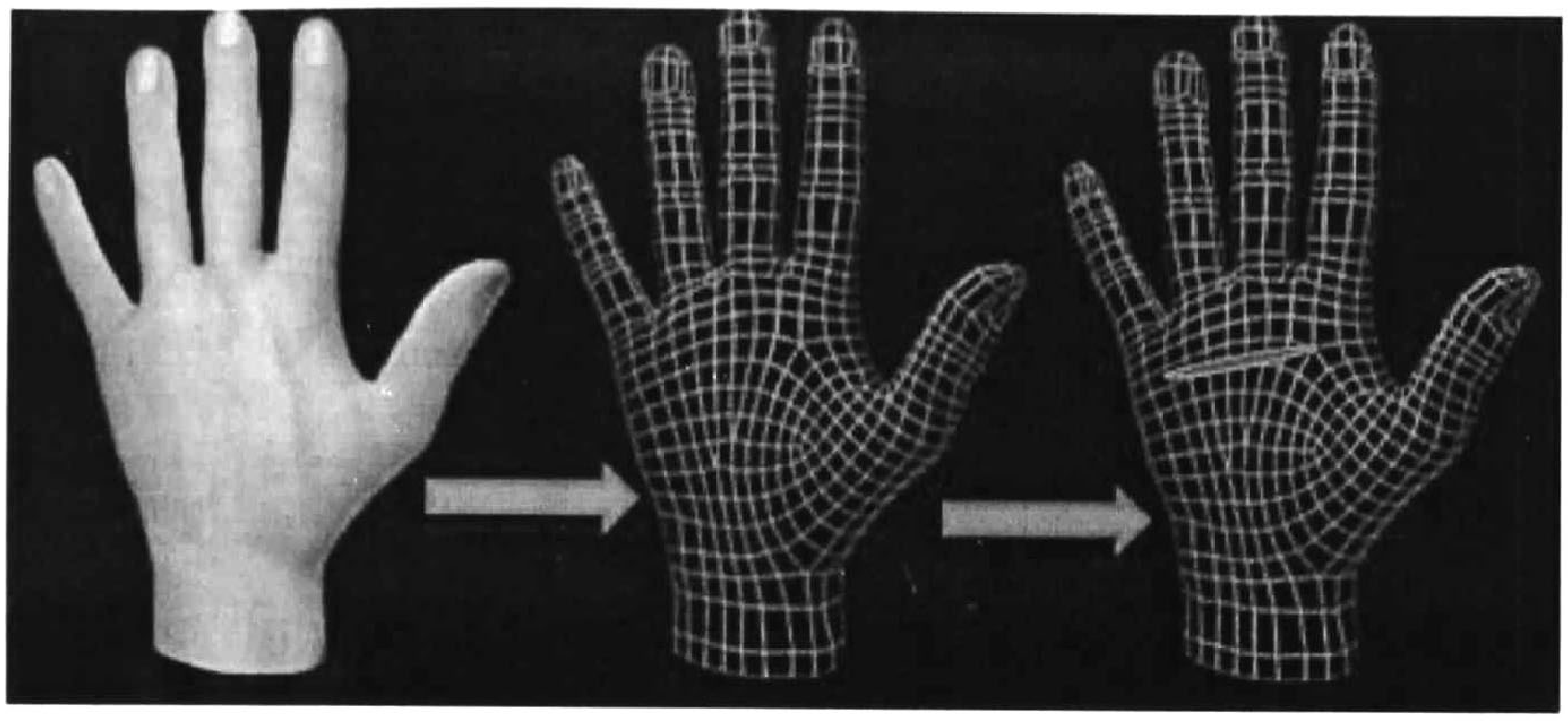
Figure 4.1: The hand skin is discretized and modeled in 3D; The simulation should allow the posibility to deform and cut the skin through the indications of the user.

4.1.2 3D: Hand Skin

The test consists of cutting the skin of the dorsal part of hand; this procedure can be applicable in different surgeries (e.g. lipoma removal). The procedure can be achieved using a straight line cut or a curved incision as is shown in Figure 4.1. The thickness of the skin of the hand is from 0.5 mm to 2 mm in soft regions (dorsal part) [40], in regions as the palm the thickness is from 1 to 4mm; in order to appreciate the 3D body model and mesh, we choice to use 2 mm as thickness of the skin.

In 3D must be considered the different topologies, the physical topology represented by a tetrahedral mesh of the skin of the hand; the visual and collision topologies are represented by a triangular surface mesh.

This case of study evaluates the following factors:

Performance: to analyze the impact on the simulation performance when the elements are cut.

Interaction: The user can interact with the hand model observing the physical reactions.

Cutting process: to analyze the delay of cutting a tetrahedron while the user is cutting; also is examined the creation of complex incisions (i.e. straight line cut and a curved incision).

Refinement:Testing different refinements of the meshes, observing which of these give better results considering the previous factors.

4.2 SOFA Framework

SOFA framework is a C++ library for physical simulation, designed for the medical context; this simulator, based on Open GL, is described in [2]. SOFA includes a set of algorithms of CG that can be combined to create complex simulations. In SOFA a single object can be represented by multiple geometrical models, these models are interactive and can be designed to be independent of the others; these commonly correspond to the physical model (mechanical model), visual model and collision model. The connections among these models is realized through mappings, this is shown in Figure 4.2

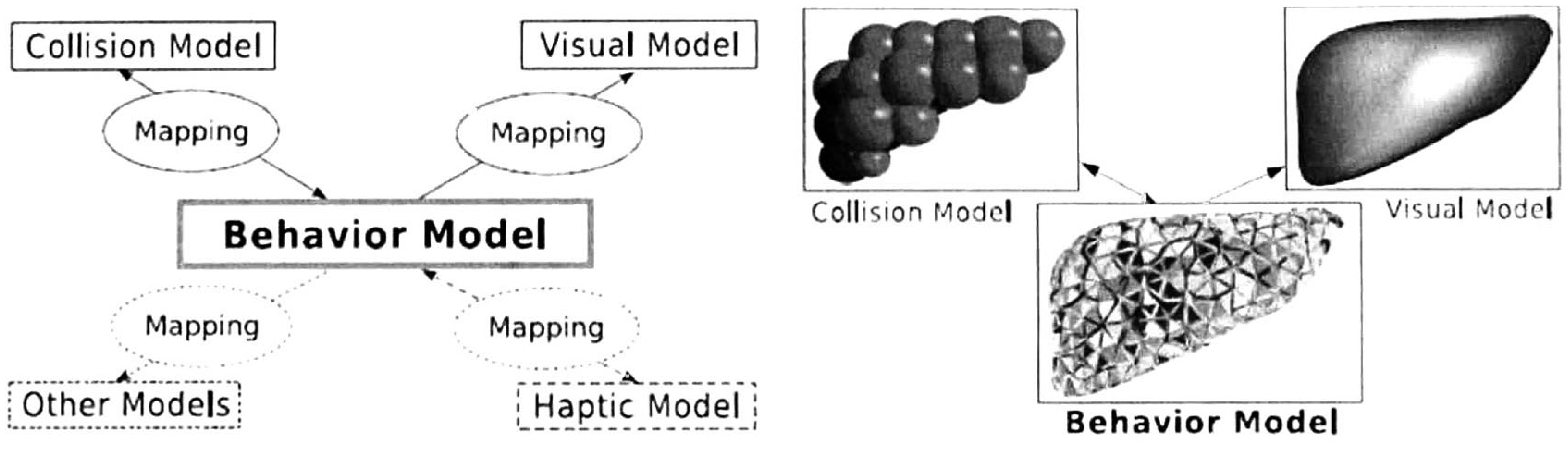

Figure 4.2: SOFA brings the capacity to create a simulation with multiple geometrical models that can be independent. The most commons are the behavior model (physical model), collision model and visual model.

Moreover, SOFA allows defining each model by combining independent components as, e.g., the physical model can formed by selecting a mass method; the force can be computed using the FEM equations or FFD, etc. The properties of each method can be easily modified, before of the simulation by editing the XML file or during the simulation.

Note that SOFA includes a lot of methods of CG, however the topological changes it's only enabled in 2D using subdivision and snapping allowing a non-progressive cut.

The simulation is constructed by the set of selected methods forming a tree structure which is known as *scene graph* as is shown in Figure 4.3

SOFA can be divided into three principal parts, the *core of SOFA* that involves all the classes that define an interactive object, i.e. physical, collision and mapping methods as is shown in Figure 4.4; the *Gui*, defines the graphical interface and the *simulation* management which generates all calls to functions to the methods included in the scene graph of the simulation into execution.

To develop the approach is only required to modify SOFA Component inside the core of sofa (cp. Figure 4.4); Hence, the XFEM class is designed as another force field component.

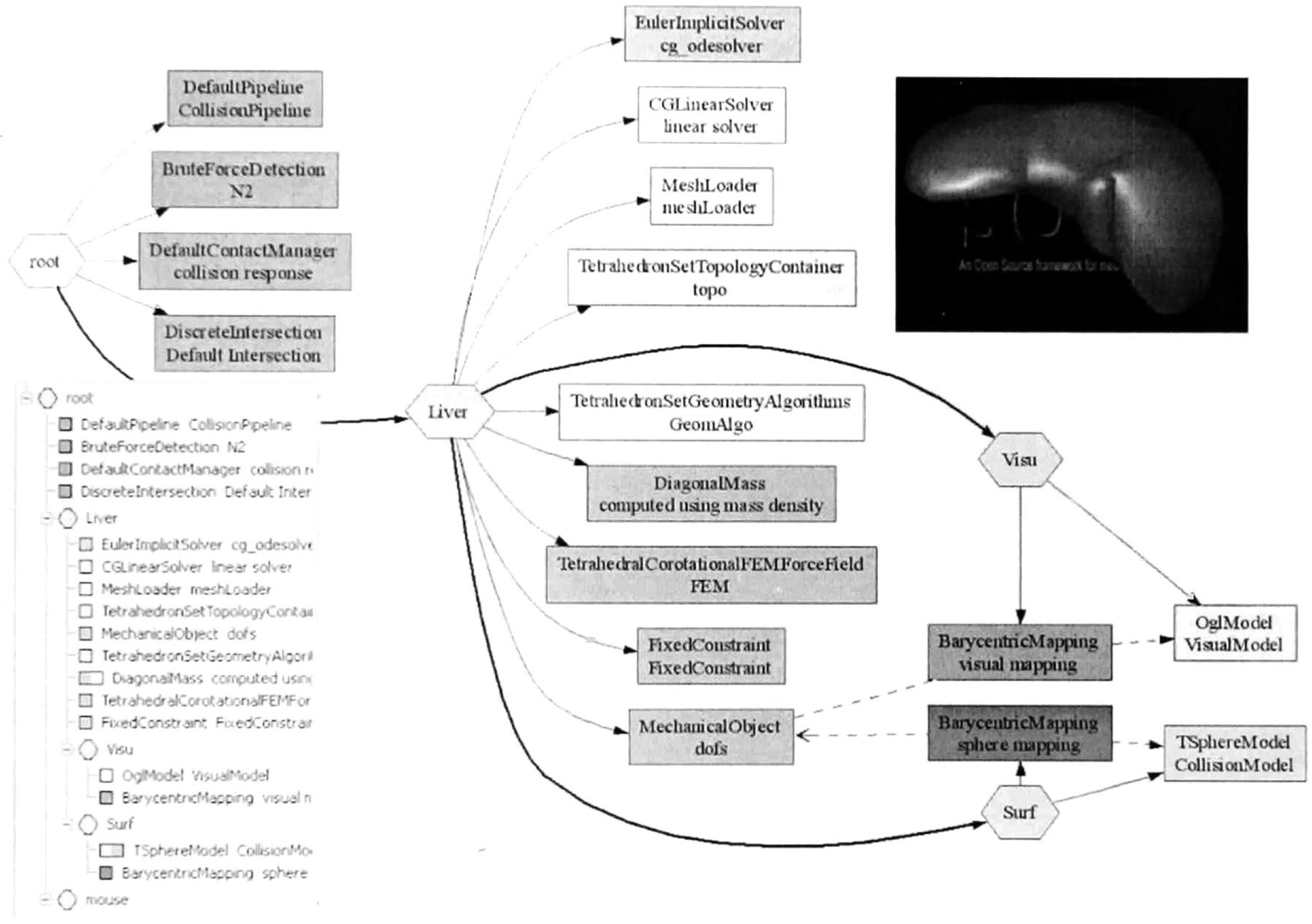
EulerImplicitSolver cg_odesolver
CGLinearSolver linear solver
MeshLoader meshLoader
TetrahedronSetTopologyContainer topo
DefaultPipeline CollisionPipeline
BruteForceDetection N2
DefaultContactManager collision response
DiscreteIntersection Default Intersection
root
Liver
TetrahedronSetGeometryAlgorithms GeomAlgo
DiagonalMass computed using mass density
TetrahedralCorotationalFEMForceField FEM
FixedConstraint FixedConstraint
MechanicalObject dofs
Visu
BarycentricMapping visual mapping
OglModel VisualModel
BarycentricMapping sphere mapping
TSphereModel CollisionModel
Surf
root
DefaultPipeline CollisionPipeline
BruteForceDetection N2
DefaultContactManager collision r
DiscreteIntersection Default Inter
Liver
EulerImplicitSolver cg_odesolve
CGLinearSolver linear solver
MeshLoader meshLoader
TetrahedronSetTopologyContai
MechanicalObject dofs
TetrahedronSetGeometryAlgori
DiagonalMass computed usin
TetrahedralCorotationalFEMFor
FixedConstraint FixedConstrai
Visu
OglModel VisualModel
BarycentricMapping visual n
Surf
TSphereModel CollisionMo
BarycentricMapping sphere
mouse

Figure 4.3: The set of methods are included in the scene graph, therefore, the inclusion of the different methods of CG in the simulation is straigthforward

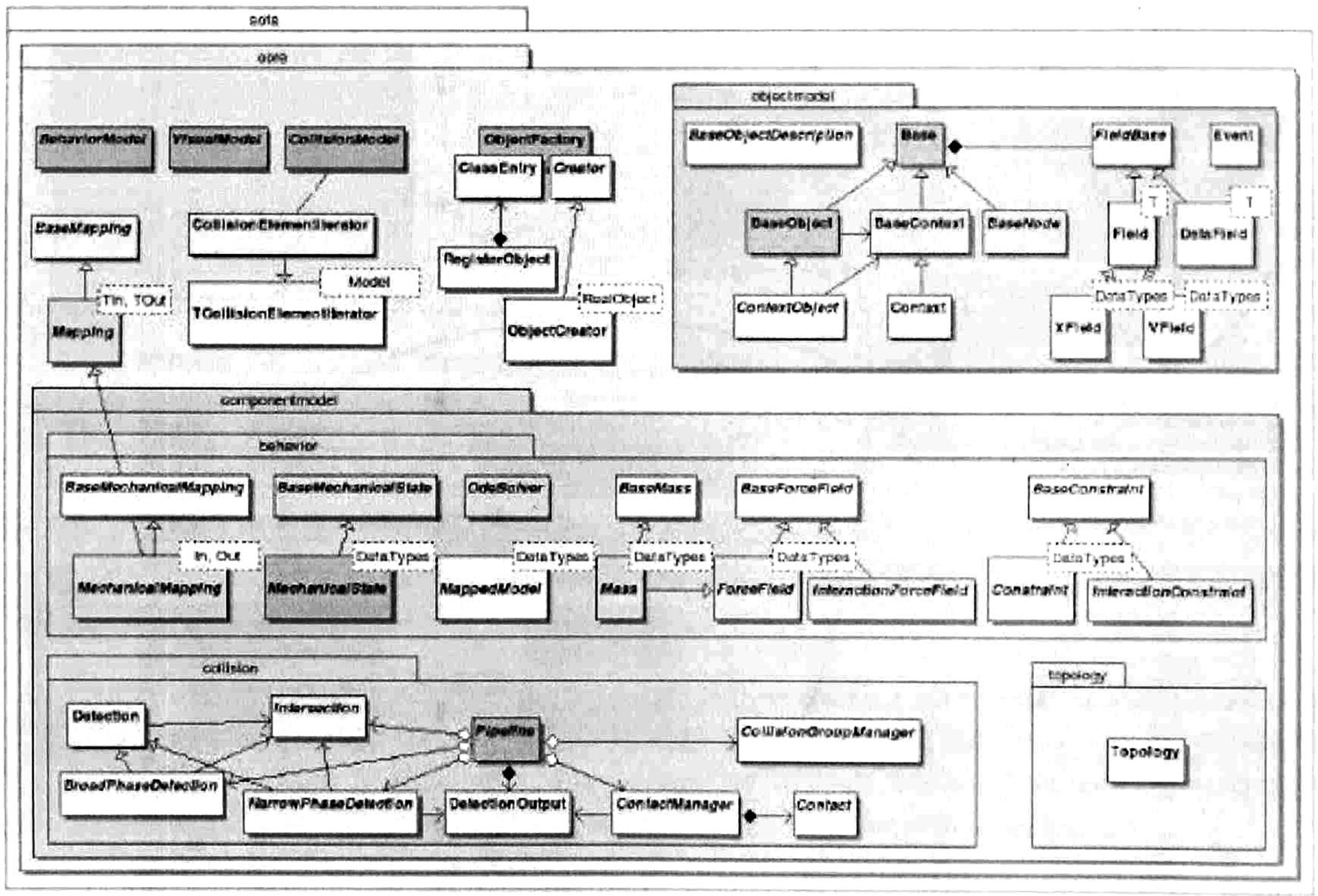

Figure 4.4: The core of SOFA contains all the methods required to simulate an interactive object, considering the physical, collision, mapping and visual methods.

To implement the XFEM we use the already implemented class in SOFA; this class is named as `TetrahedronFEMForceField`, which is the corotational FEM; therefore, the corotational XFEM inherits from `TetrahedronFEMForceField` as is shown in Figure 4.5

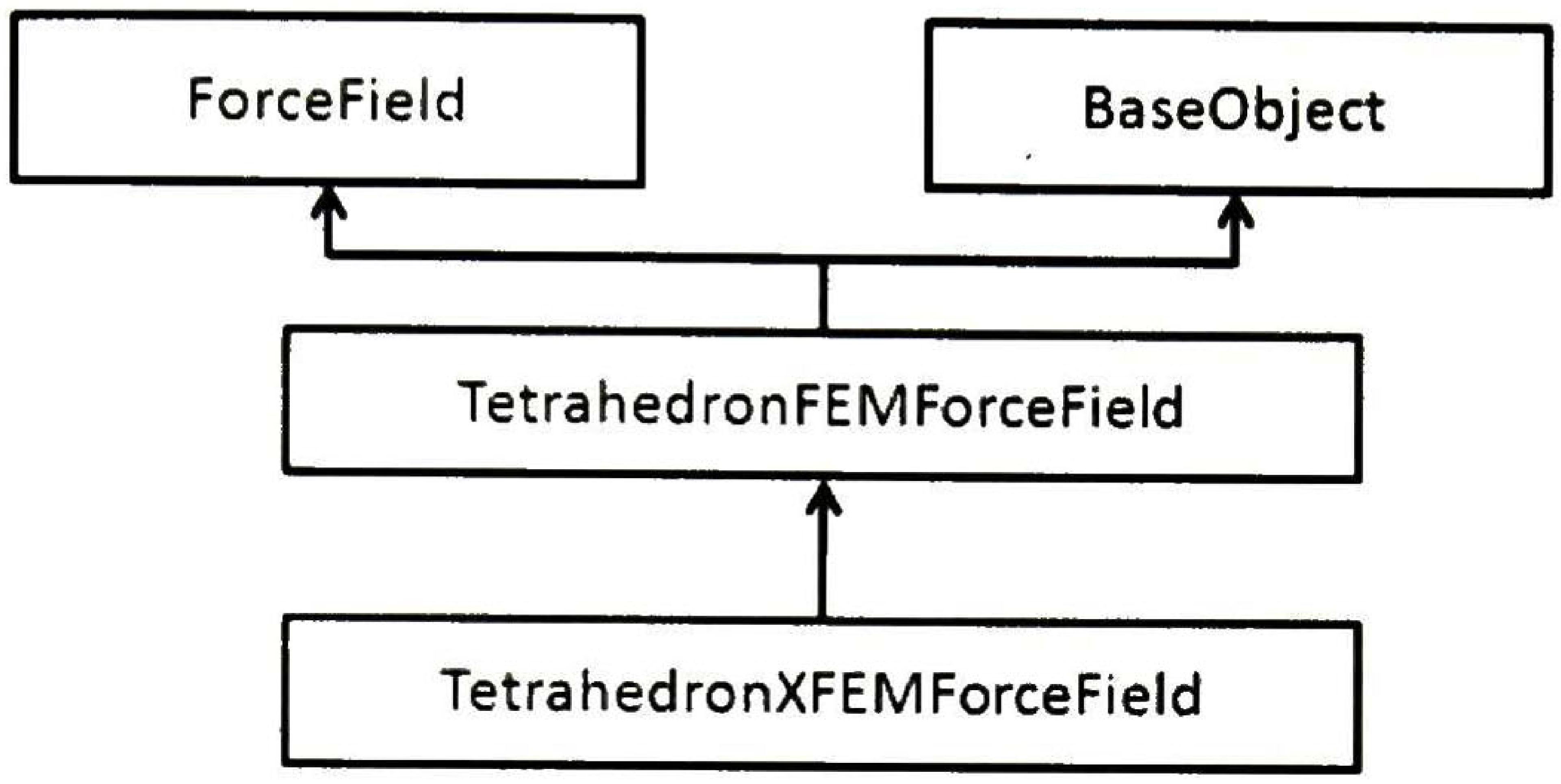

Figure 4.5: The corotational XFEM was implemented through the inheritance of the class of the corotational FEM, in this manner, the XFEM can call the functions of the FEM if there is no discontinuity.

Implementing the corotational triangle XFEM in the 2D case can be achieved similar as in 3D, using the class `TriangleFEMForceField` and creating the `TriangleXFEMForceField`.

4.3 Implementation of XFEM 2D

In the case of one triangle the XFEM is simply simulated using initially 3 nodal DOF, when a discontinuity appears, 3 nodal displacements are added, obtaining 6 nodal DOFs in total. The triangle of the Figure 4.6 is divided according to the cut plane specified by the user. The areas are computed and assigned to each part of the element.

For more than one triangle, the elements share to its added DOFs obtaining in total fewer than the double DOFs when all the elements are cut. Considering two triangles that are together and they have been cut with two different planes of cut (the planes intersect in an edge), they share the DOFs, forcing to fix the edges by sharing the forces. To Test the XFEM in a complex model, it has been used a higher mesh with more triangles as is shown in Figure 4.7, the square is dissected and the fragments fall by the gravity forces.

Employing the mouse the user can specify where to cut, by passing the mouse over the triangles that he wants to cut, the intersections are obtained and if a triangle is fully traversed (i.e. intersection of two edges) then it is converted into a XFEM element (virtual element) and the new DOFs are added; the semiprogressive cut algorithm is shown in Figure 4.8

Note that in 2D, the mapping method is only required when the topologies are not equal.

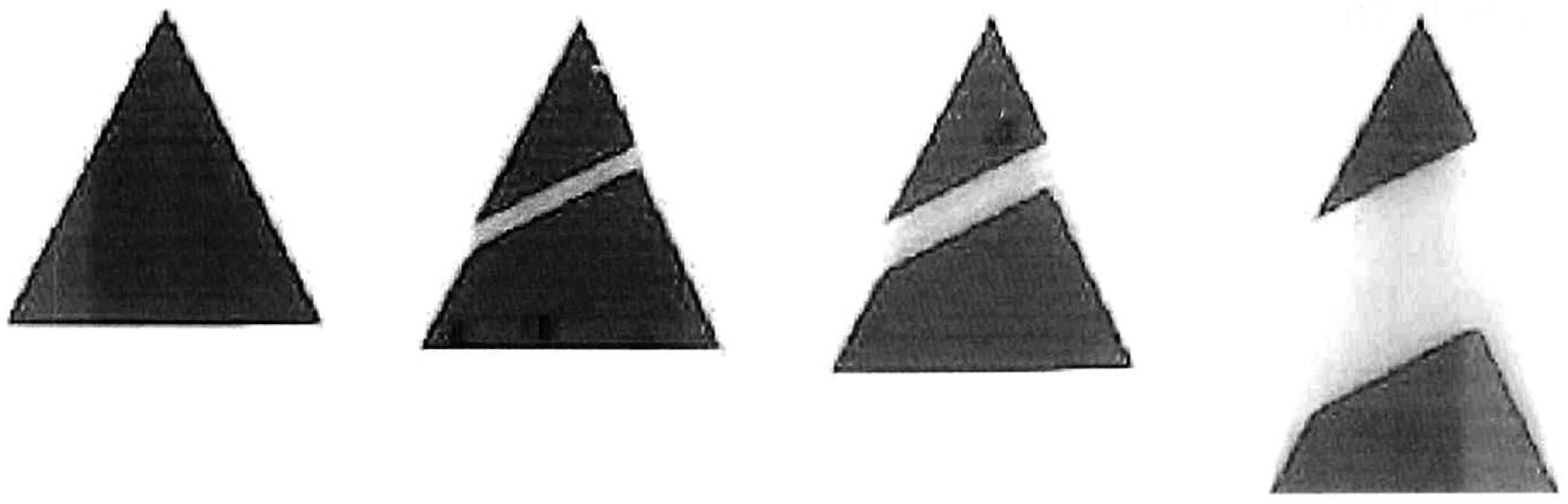

Figure 4.6: When a triangle is cut, two new virtual triangles are created by the added DOFs, however, only the corresponding part of its area must be displayed. Note that the original triangle is also hidden.

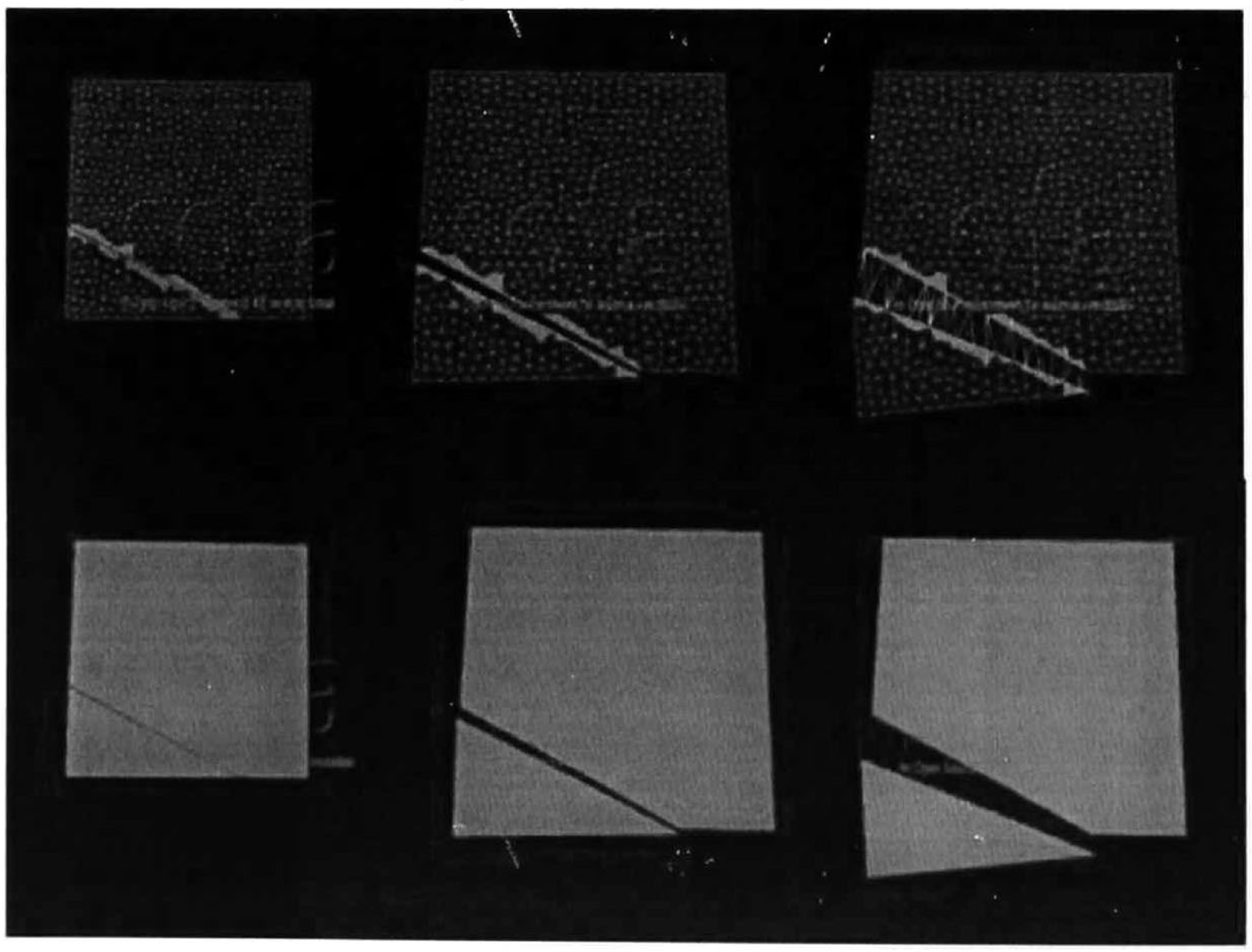

Figure 4.7: The square composed by triangular elements is dissected and all the triangles share their added DOFs with each discontinuous neighbor and due to the combination the forces, the discontinuous elements behave as one unified object.

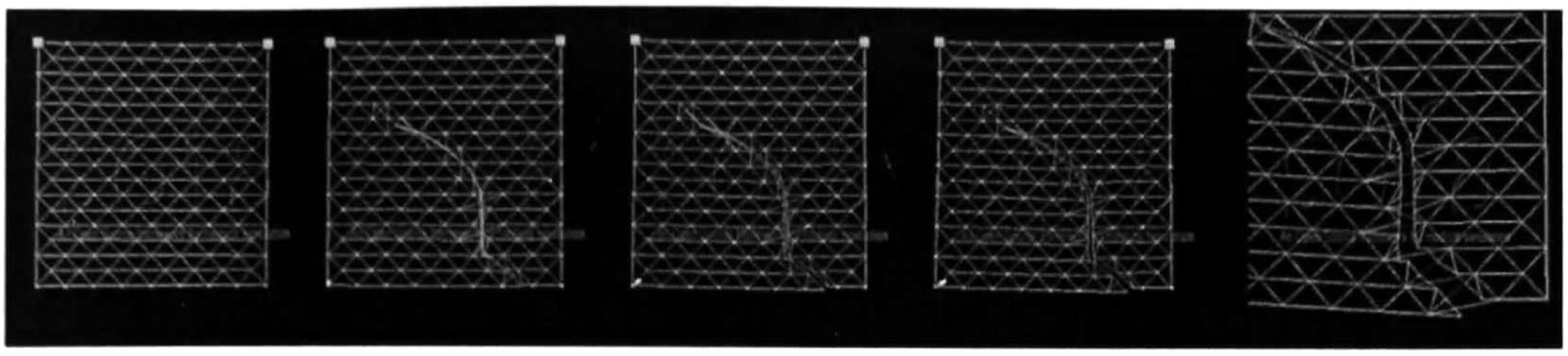
Figure 4.8: The user interacts with the mouse passing it through the mesh and if a triangle is fully crossed then is split. This figure shows the visual topology and the physical topology (hidden to the user). The visual mesh is generated by subdividing the triangles.

4.3.1 Evaluation of the approach in 2D

The face model is represented by a set of triangular elements as is shown in Figure 4.9, this model is imported to our simulation as the three topologies, i.e., the physical, collision and visual topologies will be the same mesh; in spite of the fact that it is not the best choice, because the simulation can give better results with fewer elements in the collision mesh than the visual mesh which requires more details; however, in order to compare with the existing approach (i.e. the snapping and subdivition), this choice makes easier the analysis.

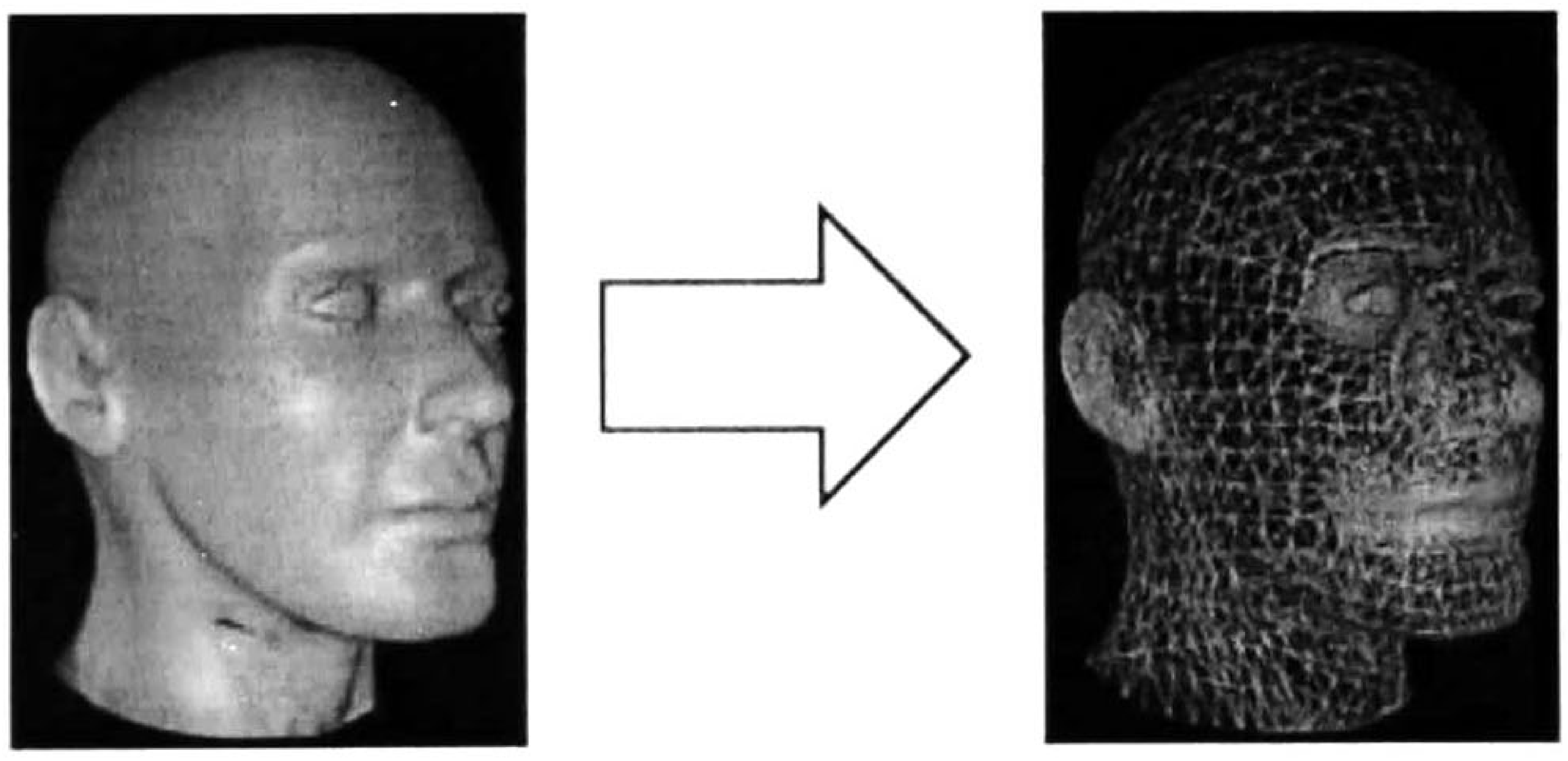
Figure 4.9: A human face is discretized into triangles and this model is imported in SOFA. In our case of study in 2D, the three models are triangular meshes.

The simulation basically consist of the following objects: a mechanical object that uses the corotational XFEM, the diagonal mass to compute the enrichment lumping approach described in section 2.4.3, a triangular collision model and these three models are mapped using the barycentric mapping. The time integration is realized with the implicit Euler method; the scene graph of the simulation is shown in Figure 4.10.

The force of gravity is annulated and the parameters are set as describe the table 4.1 (excepting the gravity). To test the approach in 2D has been employed a laptop with the

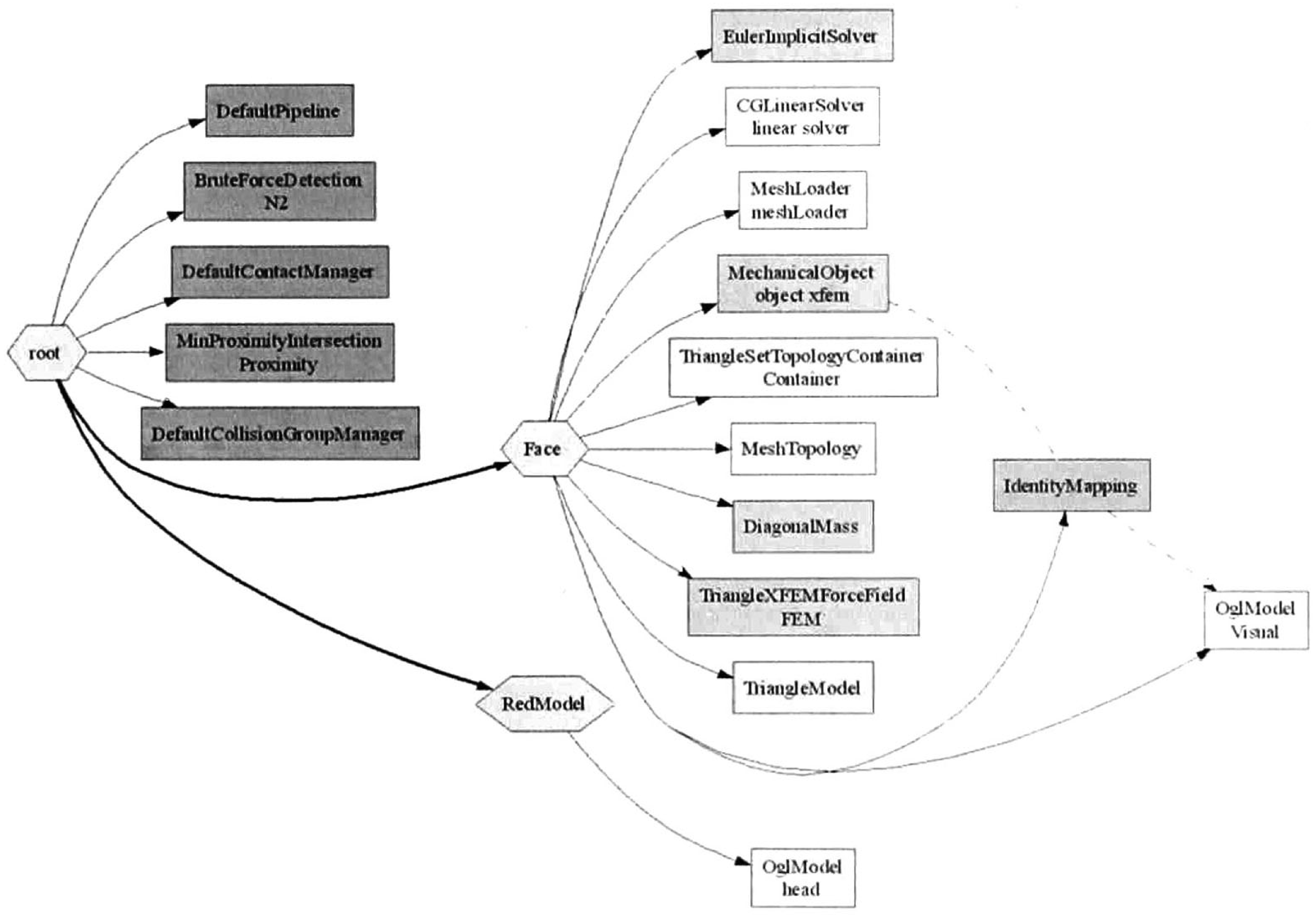

Figure 4.10: In order to compare with other methods, we select some basic components to include in our simulation. Some important methods, selected as part of our framework, are the implicit time integration required for ensure the stability of the simulation in every time step. Also, the collision detection algoritm that is based on bounding trees and especific for triangular elements.

Processor:	Intel core 2 duo 2GHz
Ram:	2GB
Video Card:	Mobile Intel (R) 965 Express, max memory 384MB

Table 4.2: Characteristics of the computer to execute the tests. A common equipment was selected to verify the portability of the approach, without requiring extra expenses.

characteristics of table 4.2, the characteristics described in section 4.1.1 are analyzed as follows.

When the simulation starts the model can be deformed exactly if the FEM where only used, because no one element is discontinuous. However, if the discontinuous elements appear in the simulation it is required to analyze if the interaction works in the same manner. The test was performed by simuling observing similar reactions in both

In 2D, we can compare with a method already included in SOFA which applies a combination of snapping and subdivision. Both are tested considering the same model (i.e. the face model) and components (cp. Figure 4.10), excepting the force field method (one method uses the FEM and the other the XFEM). Despite both allow cutting the meshes, snapping and subdivision is not fully interactive. The differences in the generation of nodal DOFs while the user cuts are shown in 4.3. It is possible to observe in Figure 4.11 that when the number of cuts increment there are more differences between both methods; moreover, the benchmark method generates one point in the initial and final points to fix the new elements with the edges of the neighbors, and that is not necessary in our framework by the sharing forces.

	accumulated		increment		
Cuts	XFEM	Snap and Sub.	XFEM	Snap and sub.	Difference
0 (initial)	3913	3913	0	0	0
10	3929	3933	16	20	4
50	3967	4020	54	107	53
100	4025	4133	112	220	108
500	4428	4955	515	1042	527
1000	5013	5836	1100	1923	823

Table 4.3: Comparative of the generation of nodal DOFs between the XFEM and snapping and subdivition in 2D.

The simulation using the XFEM allow to simulate physically the discontinuities without harming the performance; some images of the simulation are shown in Figure 4.12

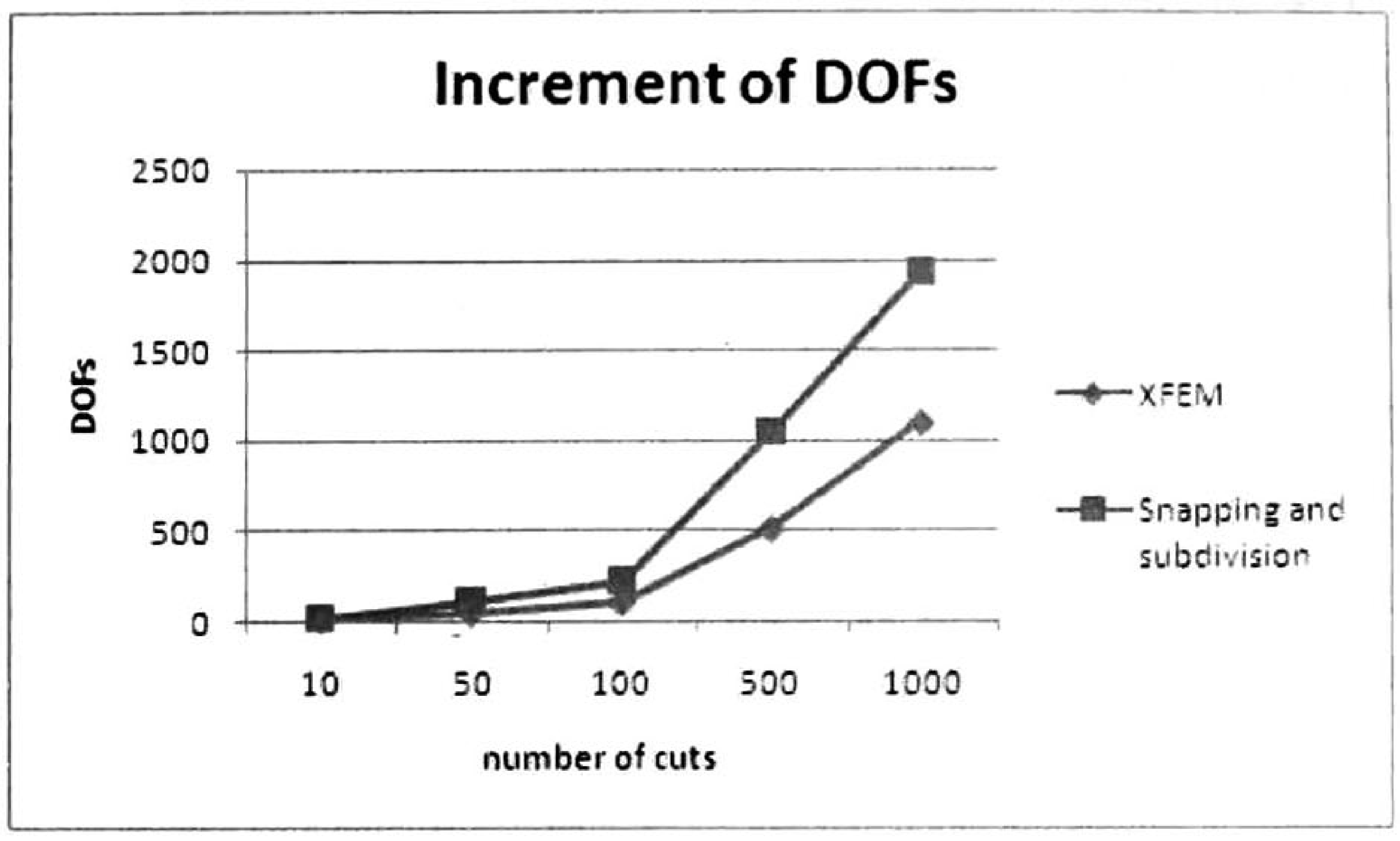

Figure 4.11: The XFEM generates less DOFs than other methods such as the snapping and subdivision, therefore, the XFEM doesn't impact strongly the simulation performance.

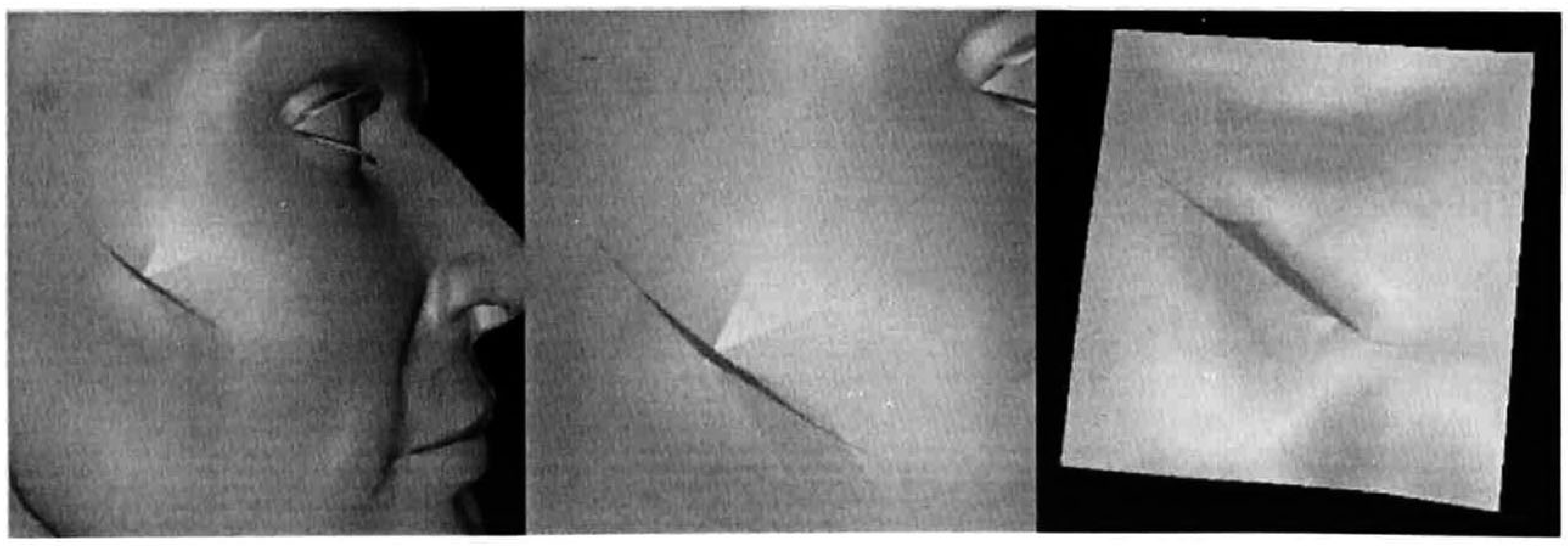

Figure 4.12: The skin of the face is cut interactively. The XFEM allows the modeling of incisions without the necesity to re-mesh the physical topology.

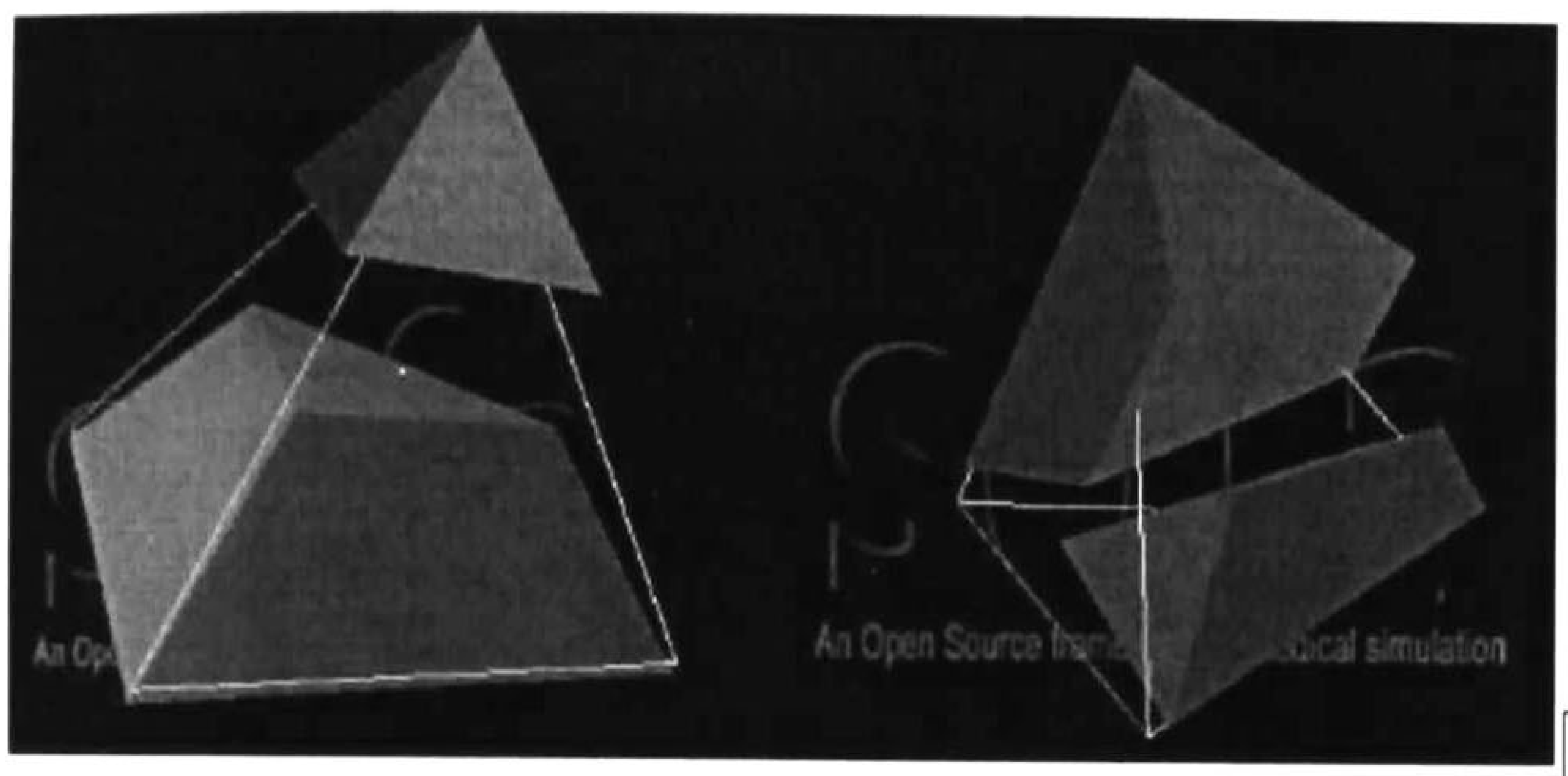

Figure 4.13: A single tetrahedron is fully crossed in two diferent ways: by intersecting three edges (left side) or intersecting four edges (right side). If a tetrahedron is split, the added DOFs allows to simulate other two elements, however, only the corresponding part of volume is displayed.

4.4 Implementation in 3D

Considering the XFEM with only one tetrahedron, it can be split similar as the triangle in 2D case, starting with 4 nodal DOFs, when a discontinuity appears another 4 nodal DOFs are added obtaining 8 nodal DOFs in total. However, a tetrahedral can be completely split in two ways, as is shown in Figure 4.13

The volume of each part is computed and stored, it is associated to the original tetrahedron element and its corresponding side of the element. Considering more tetrahedrons, it is similar as the 2D; the elements share its added DOFs to its discontinuous neighbors, in Figure 4.14 a cube is dissected in two parts.

When 3D models are use, it is required to consider that the topologies can be different; however, if we split a tetrahedron the collision and visualization of the tetrahedron can´t be achieve directly, therefore, it is necessary to use a mapping between these topologies.

4.4.1 Mapping method

When a tetrahedron has been cut, the original element is hidden and the both sides must be displayed, to do this, a mapping method must be designed between the FEM XFEM - visual and collision models as is shown in Figure 4.15.

The XFEM mapping was designed to be embedded with the original physical topology management; this mapping consist of the creation of an alternative mesh which is created dynamically while the user cuts; this mesh must interact with the other topologies and methods to react efficiently (cp. Figure 4.17). The associated components with the XFEM and

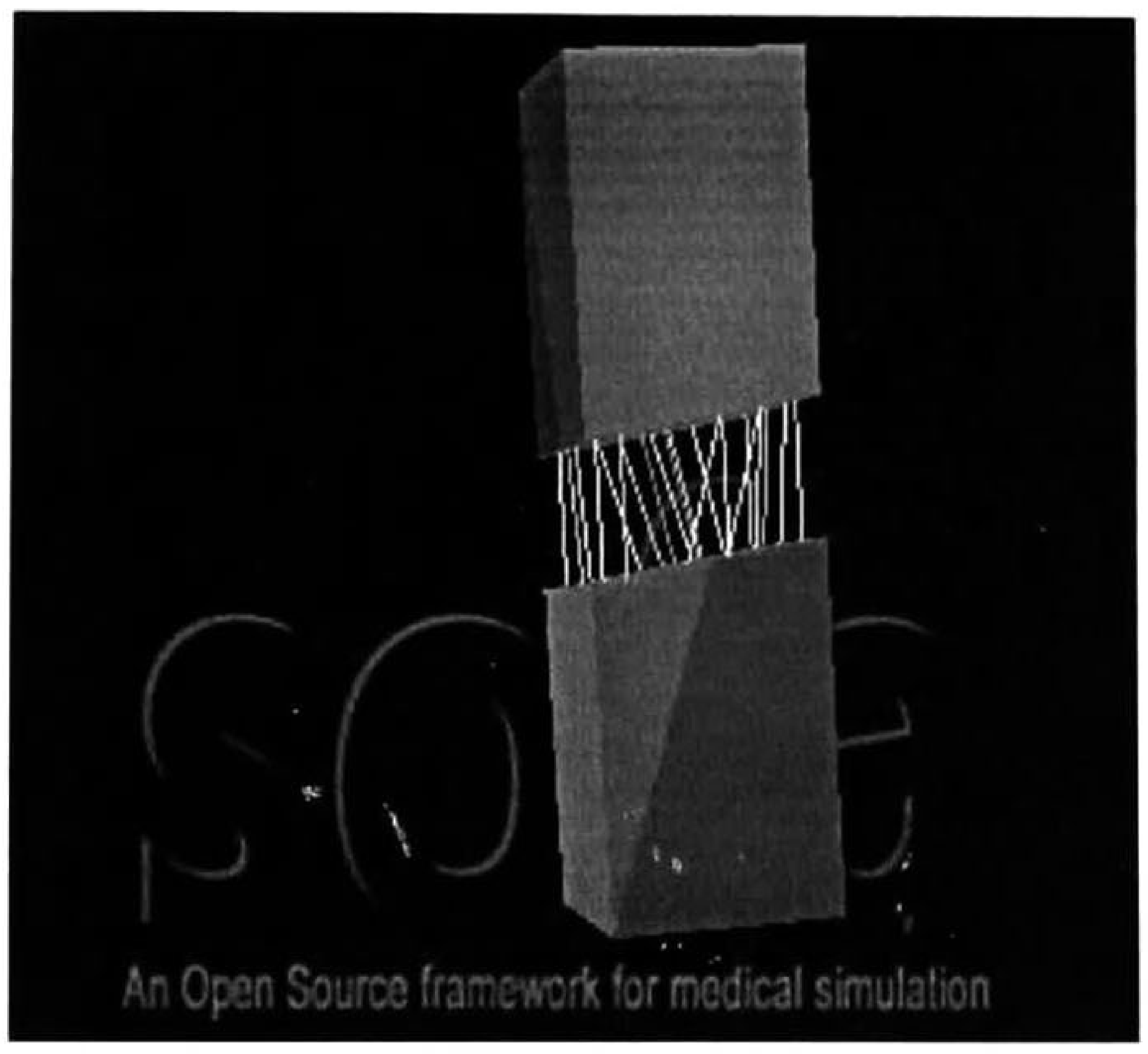

Figure 4.14: The dissection of an object is achieved by sharing the added DOFs between the neighbors of discontinuous elements.

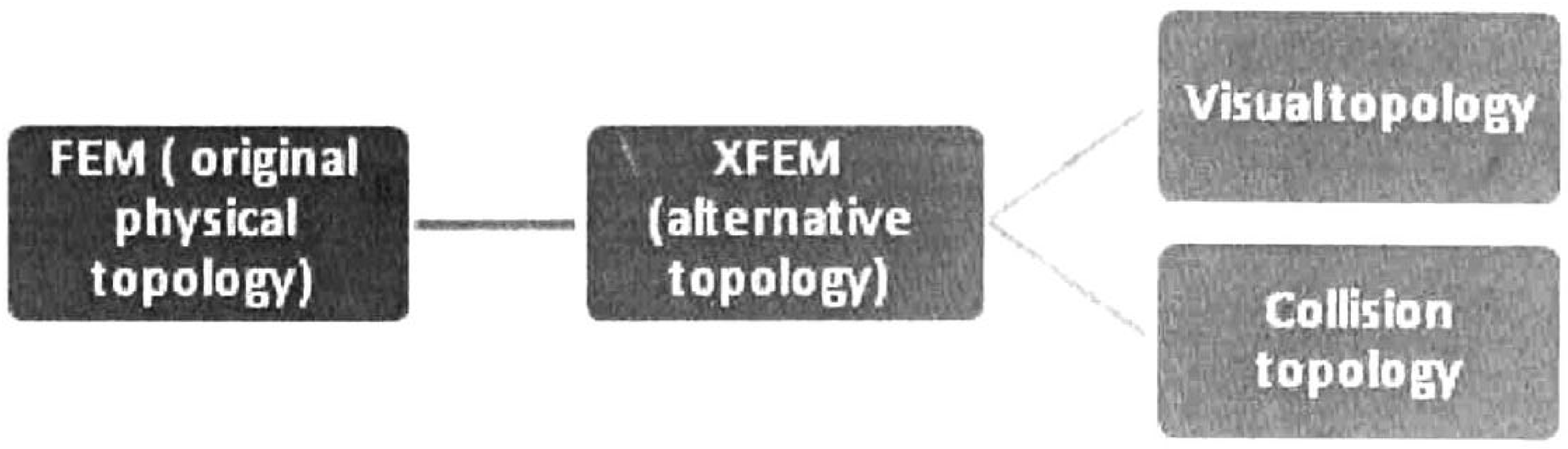
FEM (original physical topology)
XFEM (alternative topology)
Visualtopology
Collision topology

Figure 4.15: The original physical topology is controled by the FEM, when the discontinuities appear and alternative mesh is created, containing all the new virtual elements managed by the XFEM, and in this manner the barycentric mapping can be perfomed with the visual or collision methods; i.e., the alternative mesh works as a mediator between the behavior model and the other models when the XFEM is applied.

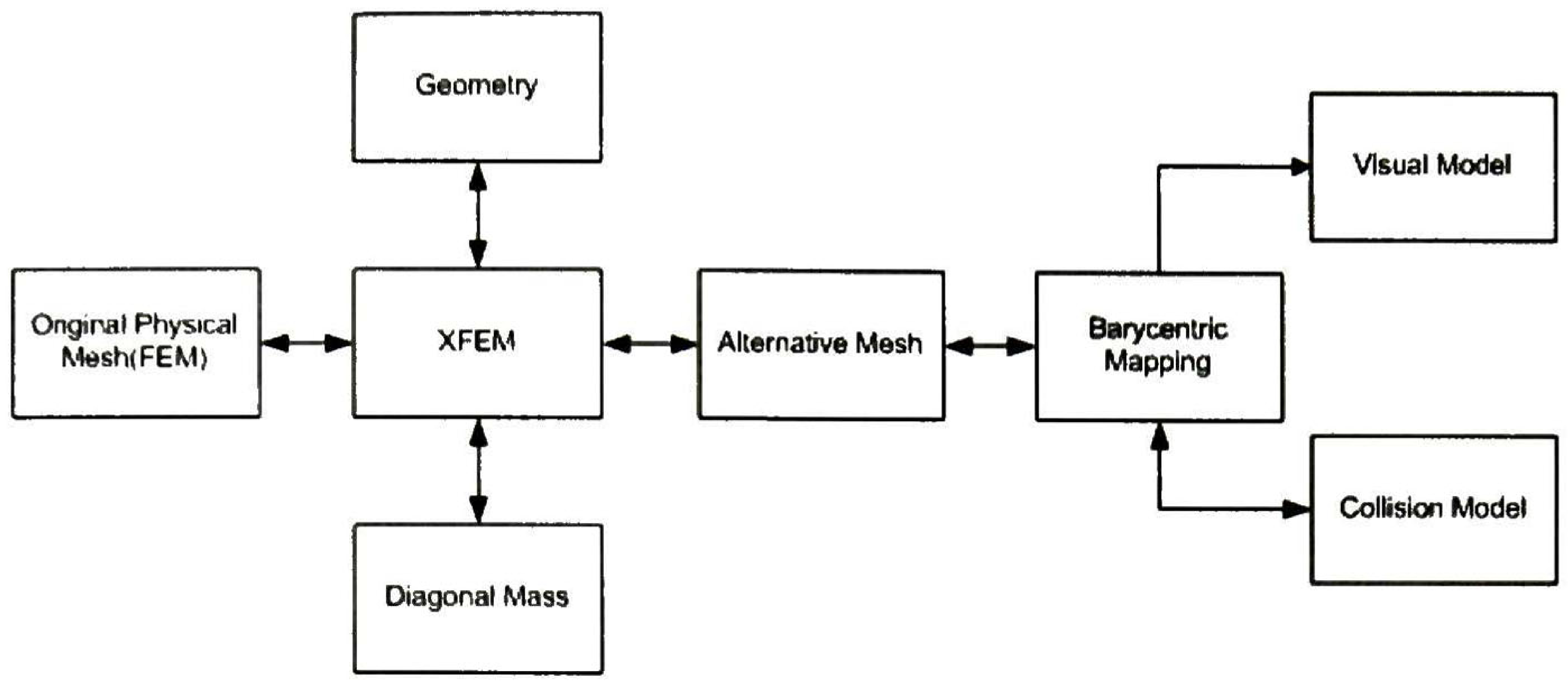

Figure 4.16: SOFA components employed to create the XFEM mapping.

the mapping are shown in the Figure 4.16; the visual and collision models can be associated to the alternative mesh through the barycentric mapping without any modification, since the mapping is controlled by the alternative mesh. When a tetrahedron is cut, the new virtual elements appear in the same position than the original element (because initially the added DOF are zero), therefore the mapping is not direct, it is required an association original-alternative meshes and the side of the cut plane, thereby, the mapping creates point-tetrahedron connections, where both are of the same side of the cut plane; the alternative mesh in a dissected cube is shown if Figure 4.18

Now, let's consider a cube that is dissected twice as is shown in Figure 4.19, the elements of the middle are below to the first cut, but above to the second cut, thus, one solution is to store the normal plane for every cut and compare all elements to remap each visual point with its corresponding virtual element. Nonetheless, this solution is too expensive when the number of cuts increases (i.e. when the semiprogressive cutting is applied). Therefore, the mapping stores the association of points with its nearest tetrahedron and the mapping is updated locally only when the current tetrahedron is cut.

The Figure 4.20 shows an example of the mapping process; the XFEM mapping stores the associations between original -virtual elements- visual points; thereby, the update is quickly performed.

4.4.2 Visual and Collision models

When the simulation starts, the visual mesh is only the surface of the 3D object, however, if the object is cut, there isn't an internal mesh that corresponds to the deepness of the object, as is shown in Figure 4.21. As was described section 3.2.4, the intersections with the tetrahedrons allow to generate the missing part of the mesh, as is shown in Figure 4.22.

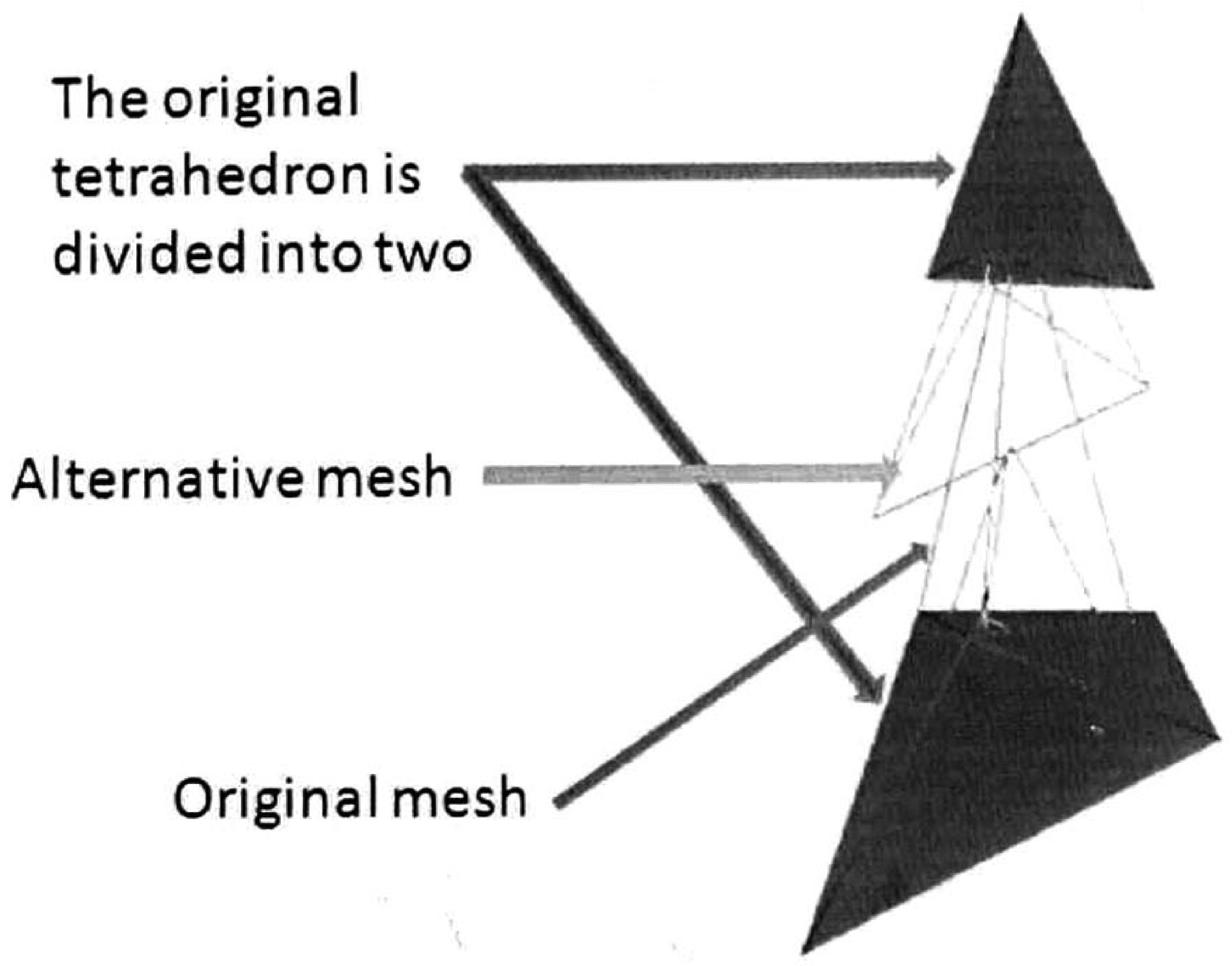

Figure 4.17: When an element is divided, the alternative mesh is generated with the new virtual elements formed by the added DOFs, this mesh allows to create a visual mapping to show the corresponding volume. Note that in the alternative mesh the virtual elements can overlap, however, this mesh is only required to make easier the association of topologies.

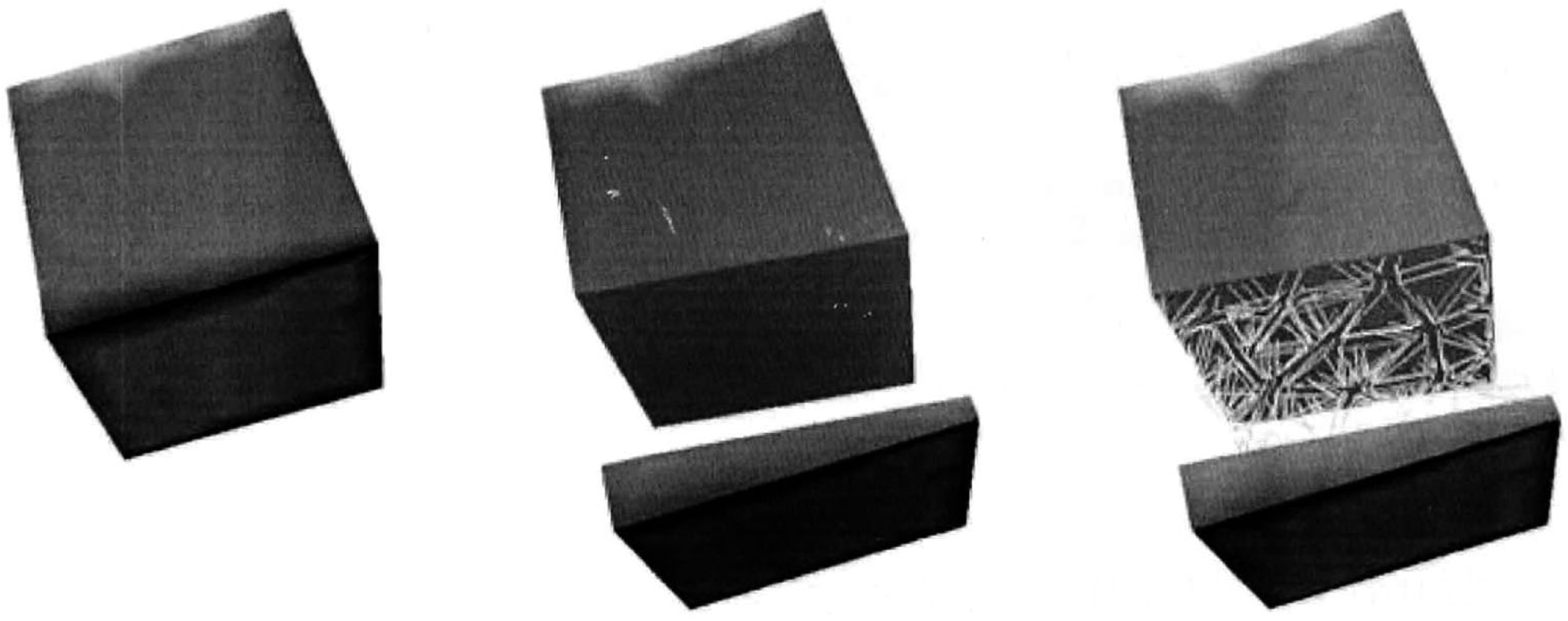

Figure 4.18: The alternative mesh helps to map the visual elements with its corresponding physical elements, this mesh is displayed in the right side.

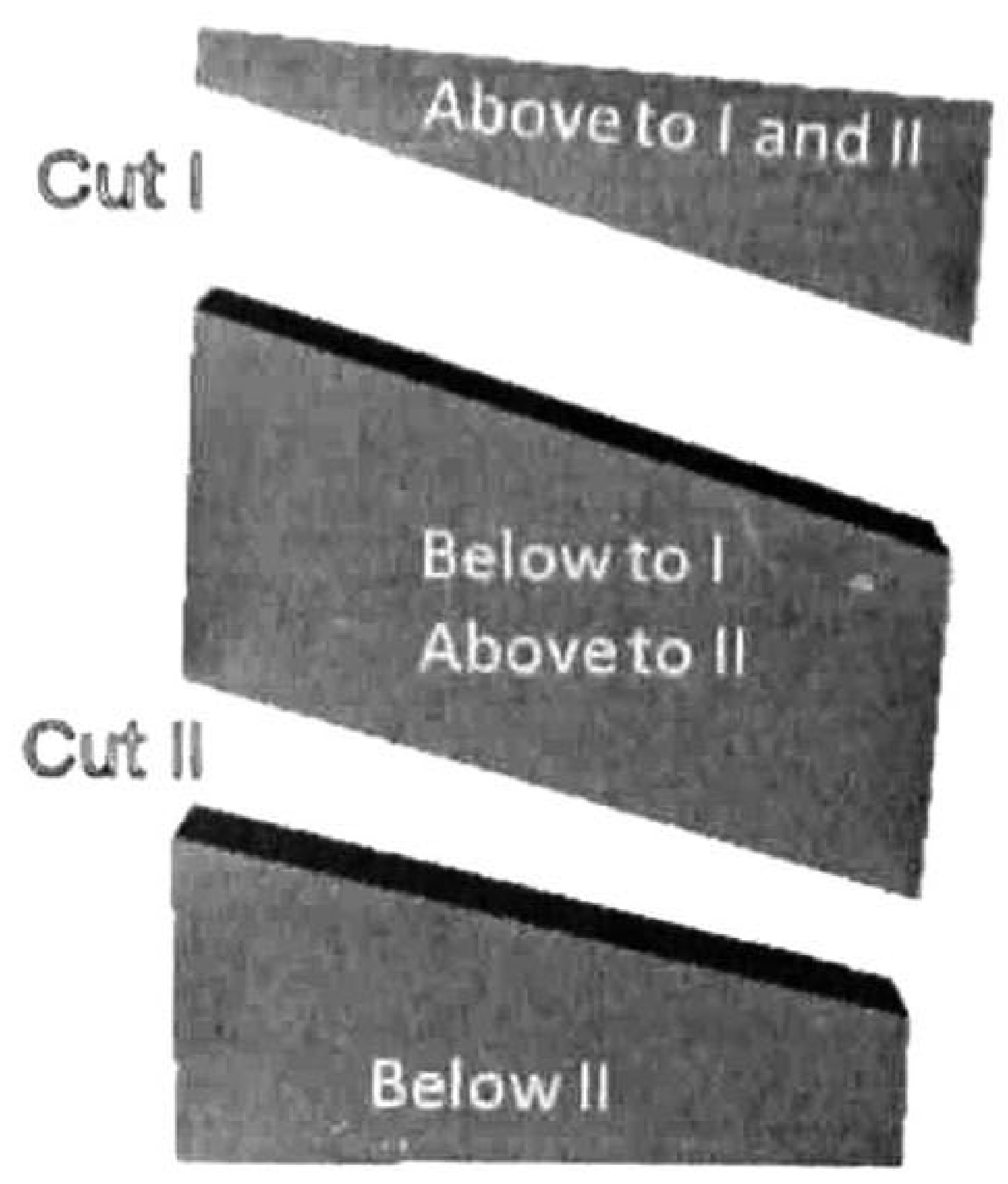

Figure 4.19: In this figure an object was divided twice, if the mapping is performed considering the cut plane, then in this case the elements in the center can be mapped in a wrong way. One option is to store all the cut planes however it is time consuming comparing all the tetrahedrons with the cut plane, therefore, if only the associations between visual points and the virtual elements are stored, these can be updated quickly, comparing only with the last cut plane.

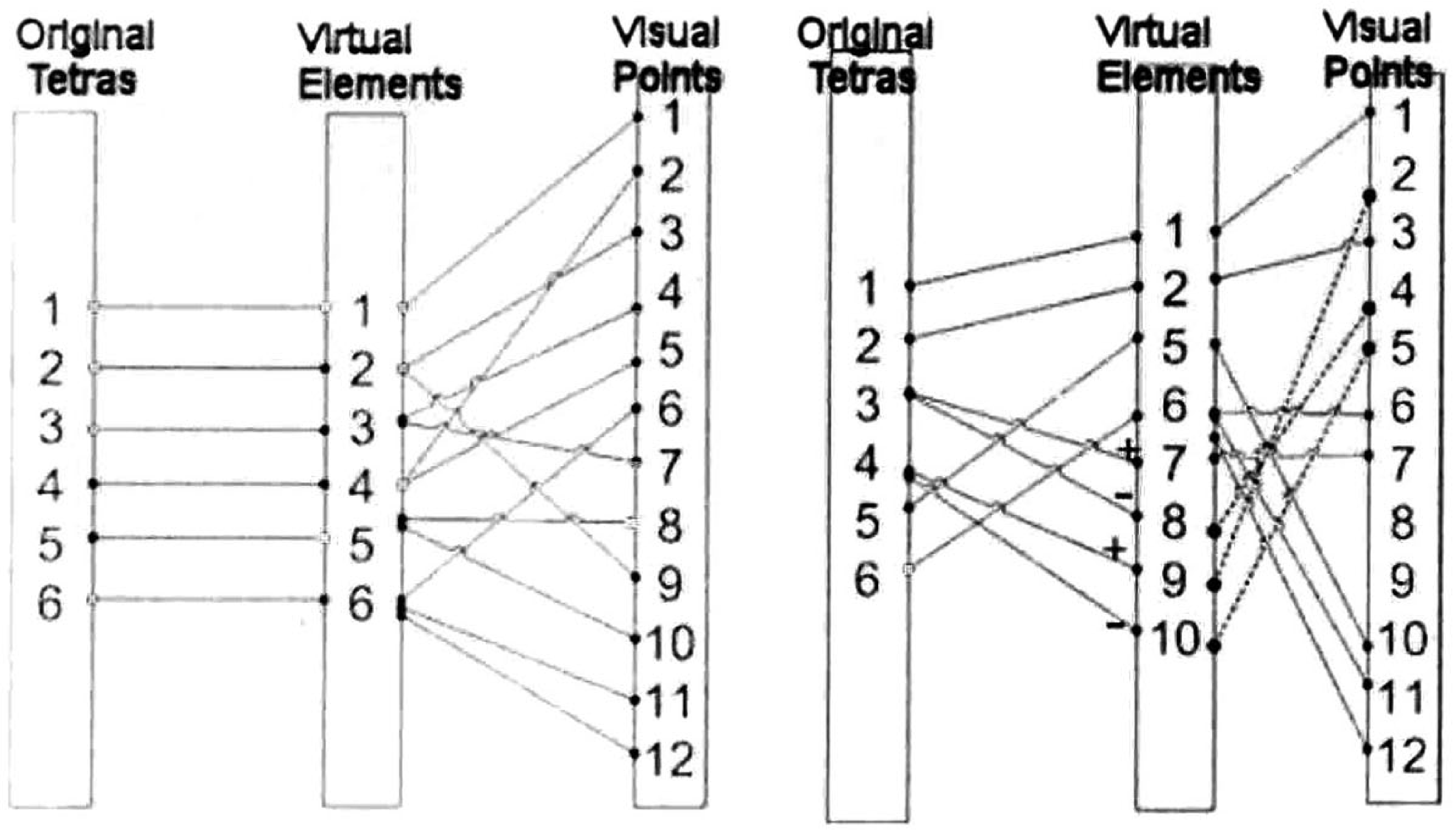

Figure 4.20: Initially the associations of points are direct, and are updated when an element is cut. For instance the elements 3 and 4 are cut, then two new virtual tetrahedrons are created for each element (4 in total); after, the visual points (only those connected with the discontinuous elements) must be re-associated to its tetrahedrons considering the cut plane (above + or below −).

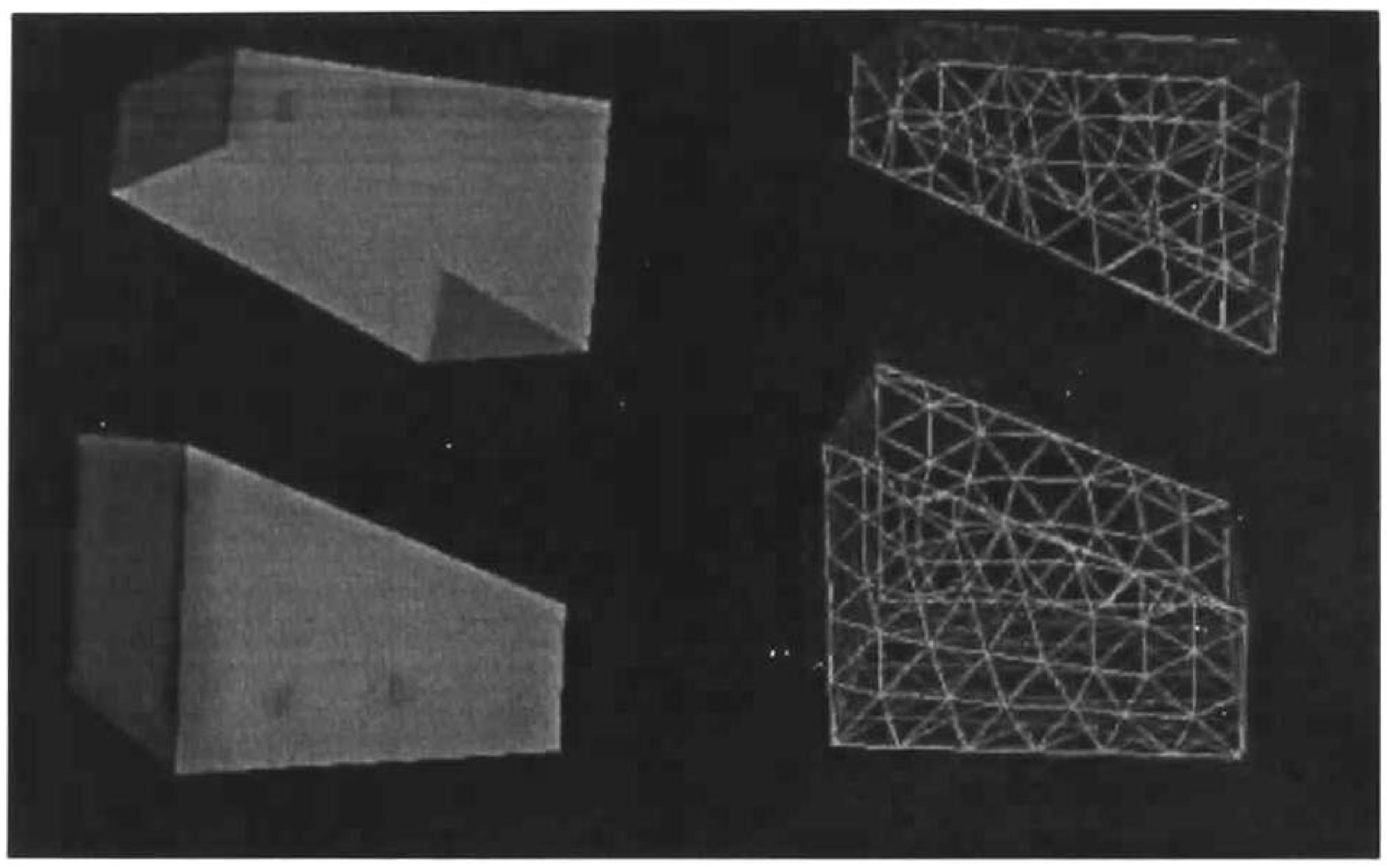

Figure 4.21: When the simulation starts, the visual mesh corresponds only to the surface of the object.

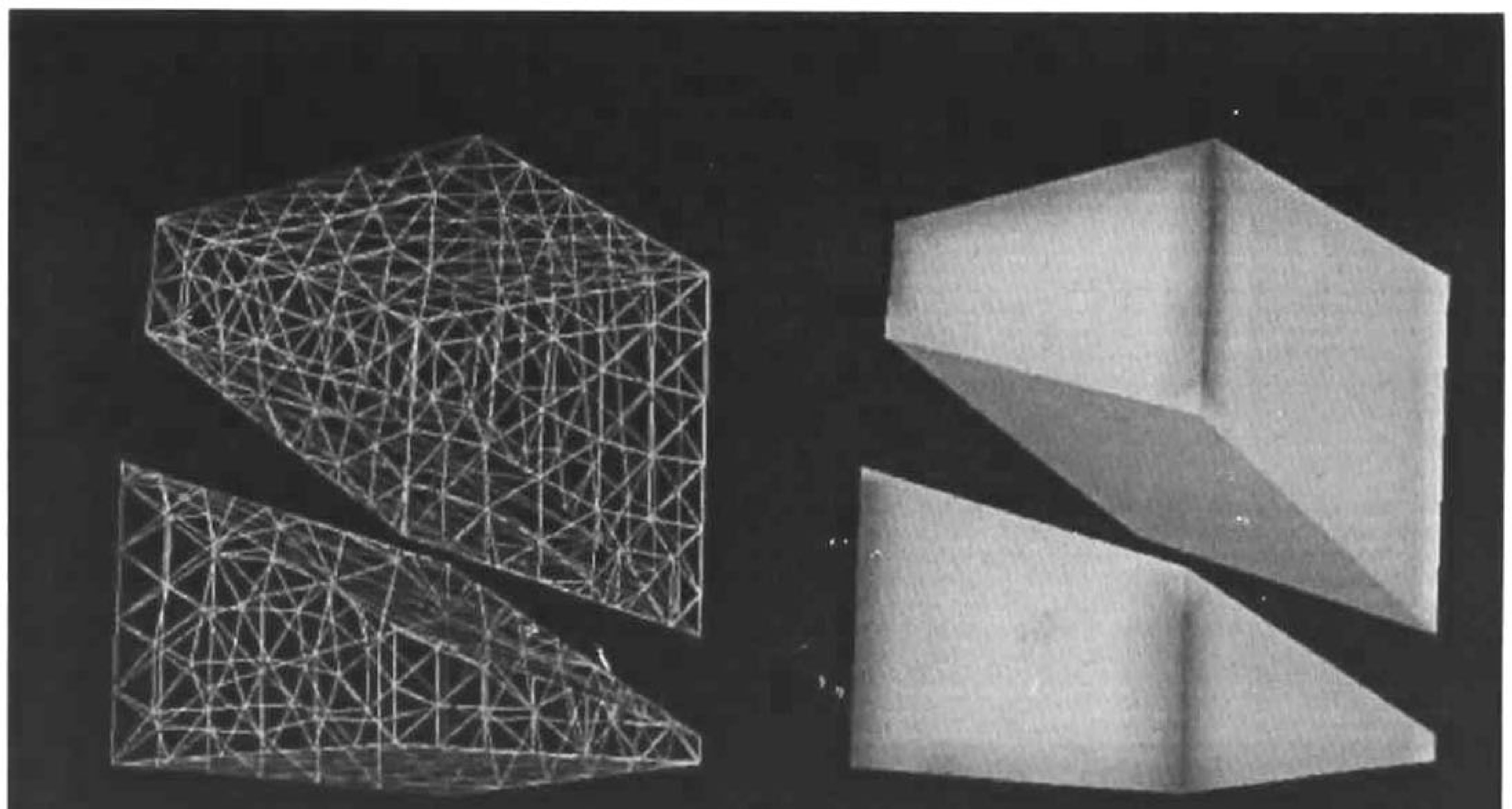

Figure 4.22: The internal mesh is created using the intersections of the cut plane with the internal tetrahedrons, creating new triangles in the visual mesh.

The internal collision mesh is generated similar to the visual mesh; it is also mapped to the surrounding virtual elements. The generated bounding trees considers all the parts as only one model, however, the collisions are obtained separately (cp. Figure 4.23).

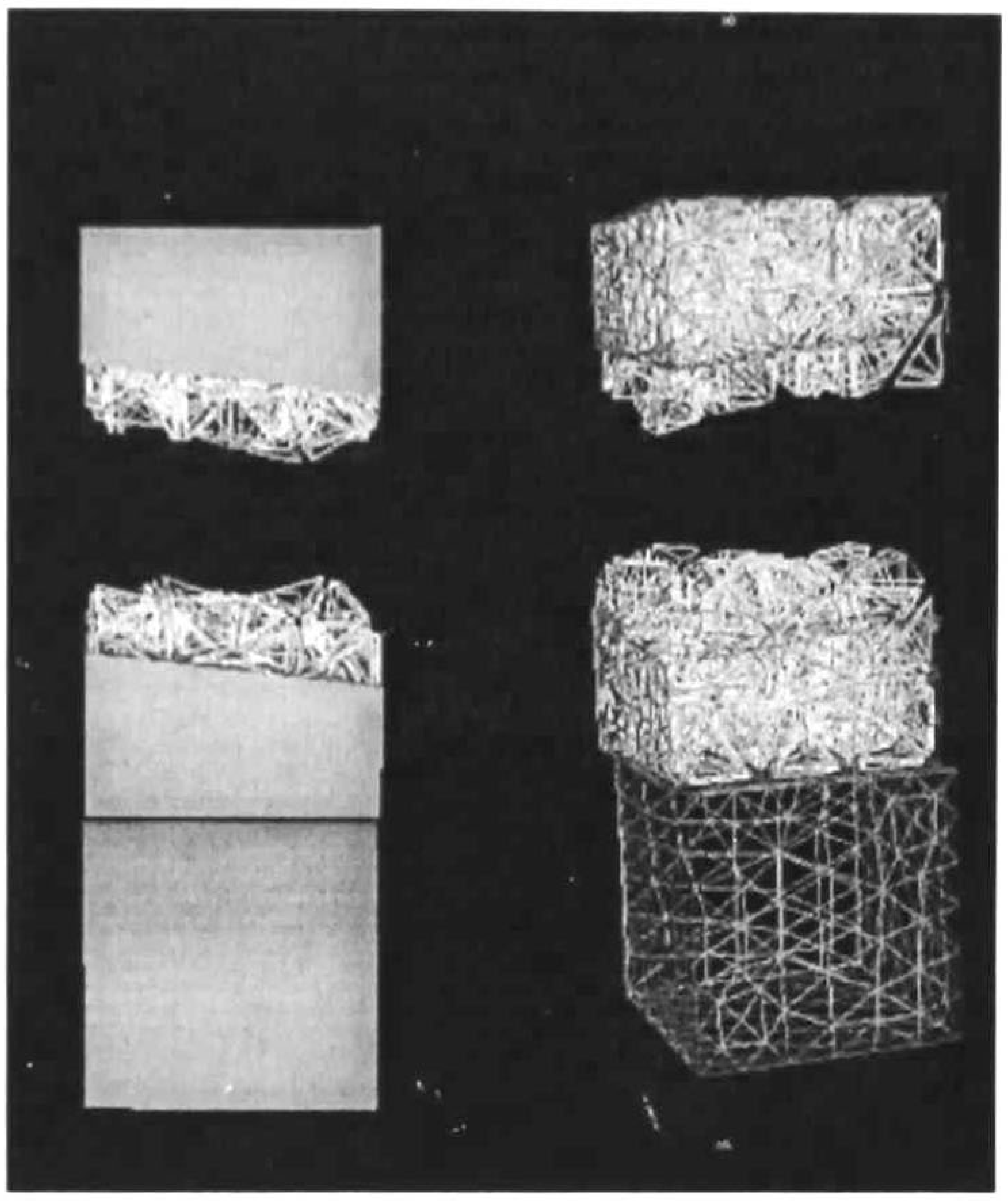

Figure 4.23: The collision detection algorithm (based on bounding trees), considers the object as unique, however when the object is cut, the intersections are given independently. The method works with triangle surfaces mapped to the alternative mesh.

4.4.3 Evaluation of the approach in 3D

In order to test our approach in a simulation with 3D objects, it has been developed the model of a hand as is shown in Figure 4.24 (a), the hand model is imported in SOFA considering the material parameters expressed in table 4.1. This model is tetrahedralized employing the mesh generator called *TetMesh-GHS3D*, this generator contains a command that applies a FEM correction, which consists of replacing overconstrained elements. Thereby, the physical and alternative topologies are tetrahedral meshes and the visual and collision topologies are triangular elements as is shown in Figure 4.24

Considering the 3D model, we don't have other method that works as a benchmark of cutting tetrahedral meshes; however, this comparison has been realized in 2D in section 4.3.1. Now, we analyze the performance, the cutting method, the interaction and the refinement of the mesh; all of this with the objective to get better results in our simulation. The scene graph considered to test the approach in 3D is shown in Figure 4.29.

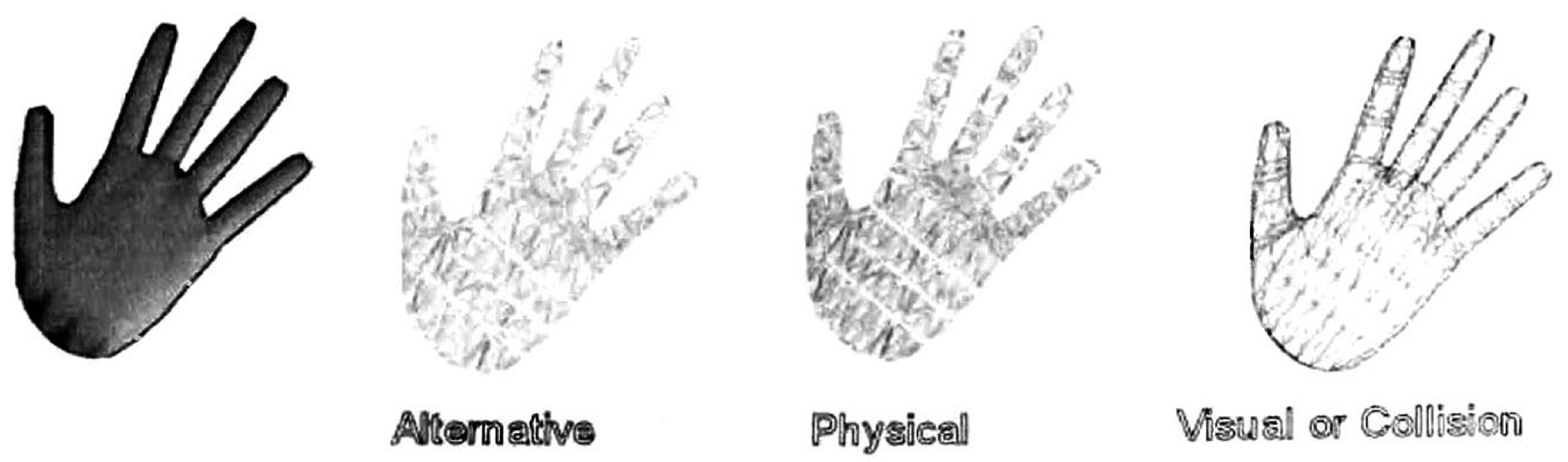

Figure 4.24: In 3D, our simulation is performed using multiple models, the behavior model (physical) and the alternative mesh are conformed by tetrahedrons; the visual and collision meshes are with triangular elements.

Performance

The objective of this test is to analyze how much time takes to process the new virtual elements; i.e. when many elements have been cut, the number of DOFs increments and consequently the time of computing the equations also increments. The simulation time and the real time relationship depends strongly on the charge of work (i.e. the number of DOFs). We are looking for a real-time simulation; therefore, the simulation time must be as closest as possible to the real world time. For this reason, we compare a simulation second with a real second and analyze how the performance is impacted when number of split elements (virtual elements) increase.

The test has been realized with the following procedure:

- When the user interacts the test starts.
- The simulation and system times are recorded.
- The simulation time is monitored and when it increments in one second of the previous recoded time, the system time also is obtained and the difference is recorded in a list.
- The procedure is repeated *nTest* iterations and the average is computed.

This procedure has performed for different meshes to analyse the performance while the increment of The results are shown in table 4.4 and in the Figure 4.25. The values indicate that the biggest increment is at the first 50 cuts and after this time increments in a constant value.

At first sight, these values analyzed together with the table 4.5, seem that have a negative impact; If we compare the time for 144 DOFs (i.e. 50 cuts in the mesh of 200 elements) with

No. Tetrahedrons	no cuts	50 cuts	100 cuts	150 cuts	200 cuts
200	0.28435	0.5156	0.64065	0.76325	0.843
400	0.38125	0.73045	0.9	1.0617	1.28675
600	0.60625	1.13515	1.3258	1.54455	1.6992
800	0.74065	1.51405	1.675	1.925	2.19145
1000	1.0164	2.2789	2.31175	2.6234	2.78515

Table 4.4: time that takes one simulation second, considering cuts and different meshes($\Delta t = 0.5$)

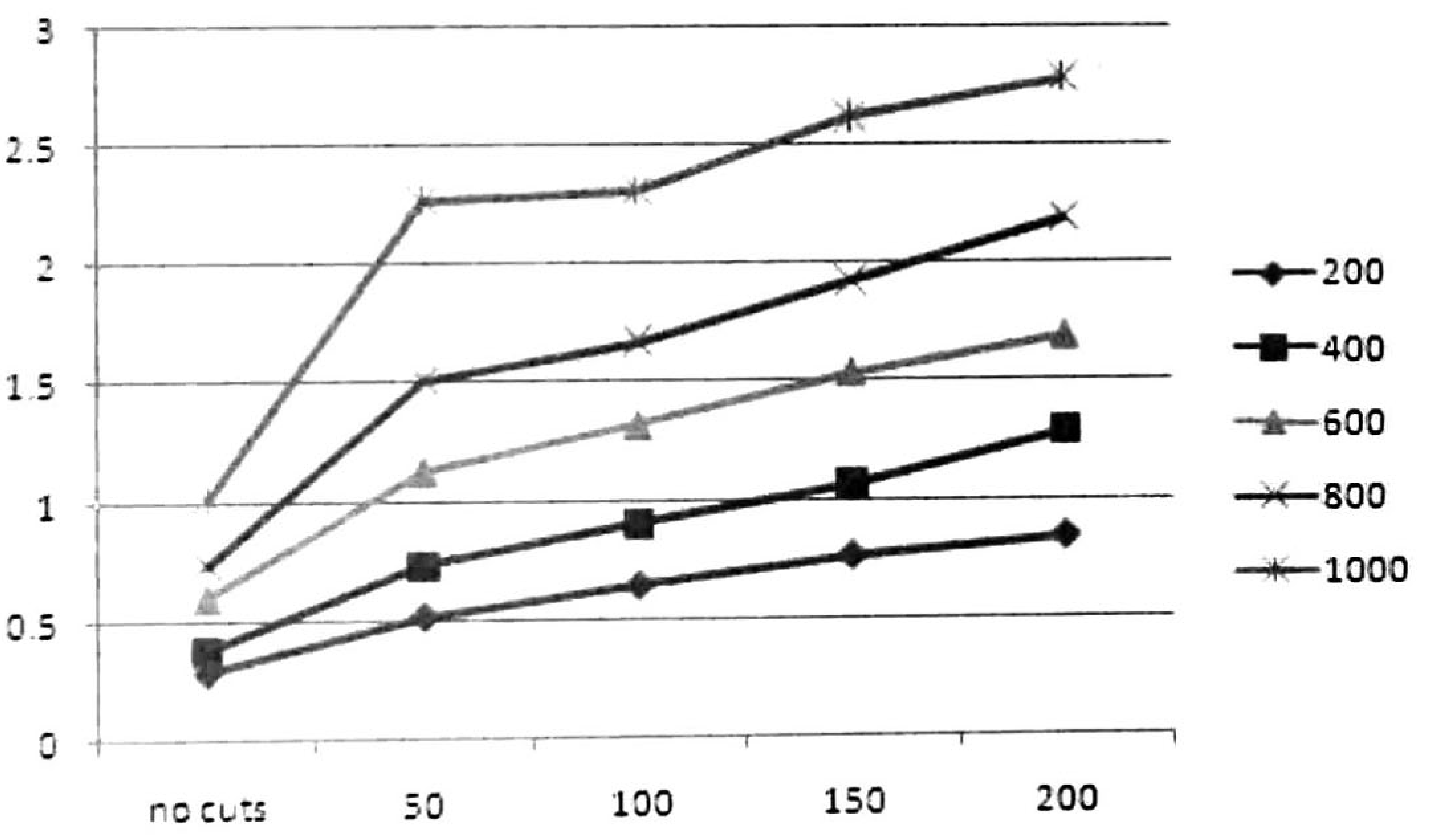

Figure 4.25: This graphic shows the relation of one simulation second and the real second, considering cuts and different physical meshes. ($\Delta t = 0.5$)

elements	no cuts	50 cuts	100 cuts	150 cuts	200 cuts
200	94	144	194	244	294
400	160	210	260	310	360
600	236	286	336	386	436
800	308	358	408	458	508
1000	370	420	470	520	570

Table 4.5: Number of nodal DOFs, considering an average increment of 50 new nodal DOFs in 50 cuts.

respect to the 160 (i.e. no cuts in the mesh of 400 elements) the first one is slower than the second indicating a slightly overhead by the mapping in ranges of 0.1 to 0.3 seconds; however, other methods increment for each cut in more than 4 nodal DOFs, harming more the performance than just implementing the mapping.

Cutting

As part of the approach, the semiprogressive cutting is implemented (see section 3.2.3); it is important to be evaluated because of the perspective of a virtual surgery; i.e., the cut of an element mush be as quick enough that the delay should be almost imperceptible. For that reason, the time (real world time) that takes to cross an element is analyzed; nevertheless, the time of crossing and element depends on the size of the element and the speed of the user hand while he is cutting, therefore, three speeds have been tested considering different meshes. The procedure to perform the test is as follows:

- When a tetrahedron is added to the list L1 the system time is recorded in tList0.
- If the same tetrahedron is added to the list L3 the system time is compared with the recorded in the tList1 and the difference of time is recorded in the list difRecord.
- When the difRecord is of size $nTest$, these values are stored in a file, consequently the average time is computed This procedure is realized for each speed and mesh. Moreover, in order to obtain the worst case, the elements are attempted to be cut through the center. The resultant values are shown in table 4.6 and Figure 4.26.

When the number of elements in the mesh increments the size of these elements decrements; hence, cutting a mesh with more elements allow better results. However, if the cut is fast and there are many elements (i.e. Fast and 1000 elements), the time that takes to process all the nodes also impacts on the process of finding the intersected element and subsequently it increments slightly the time of cutting.

Elements	Slow	Normal	Fast	Average
200	0.7124	0.4265	0.24685	0.46191667
400	0.63345	0.38135	0.16575	0.39351667
600	0.59765	0.27425	0.11095	0.32761667
800	0.513275	0.2129	0.111775	0.27931667
1000	0.453175	0.1459	0.140225	0.24643333

Table 4.6: Delay (in seconds) that takes to fully-cross one element, considering different speeds.

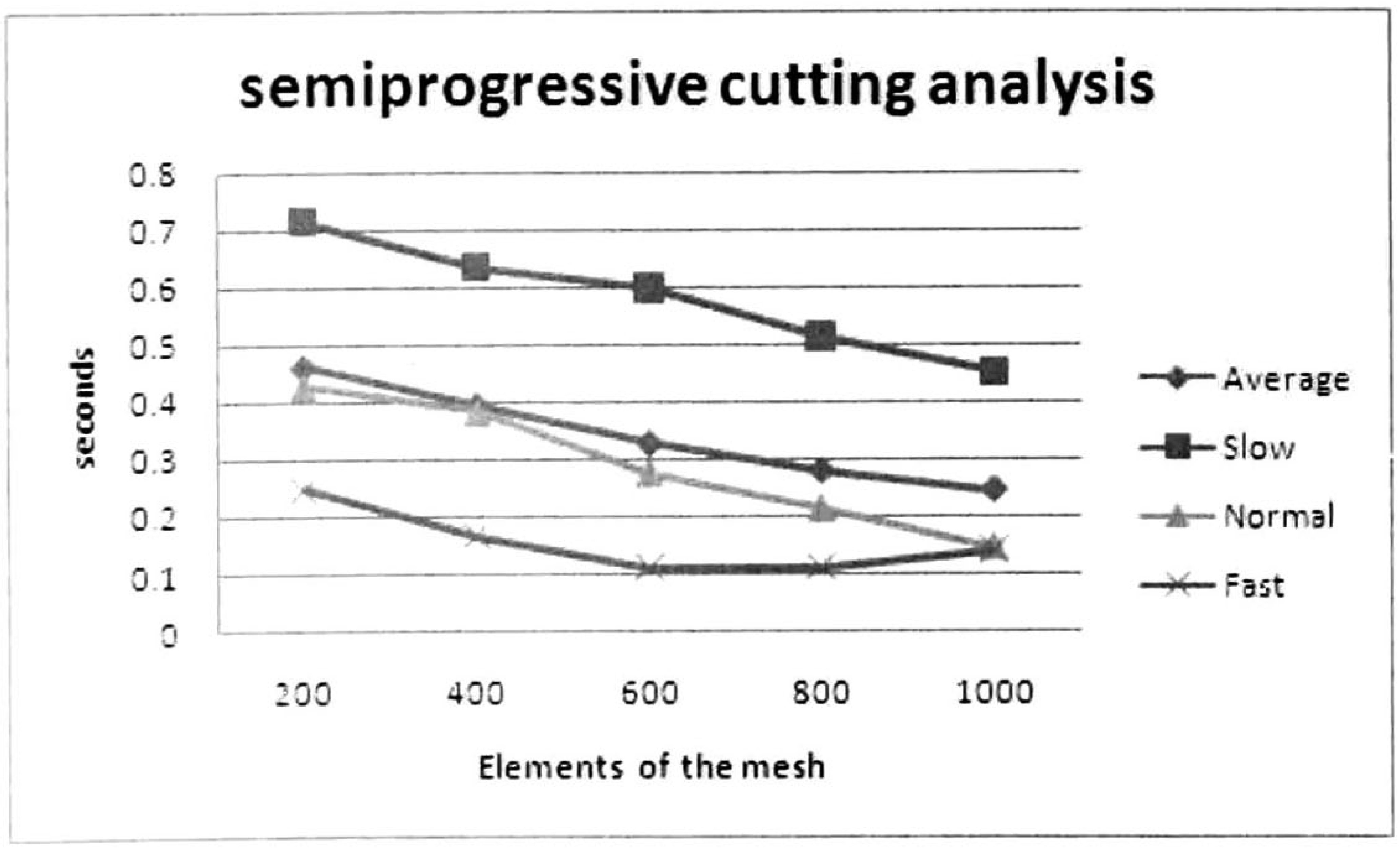

Figure 4.26: The delay of crossing an element considering different speeds and meshes.

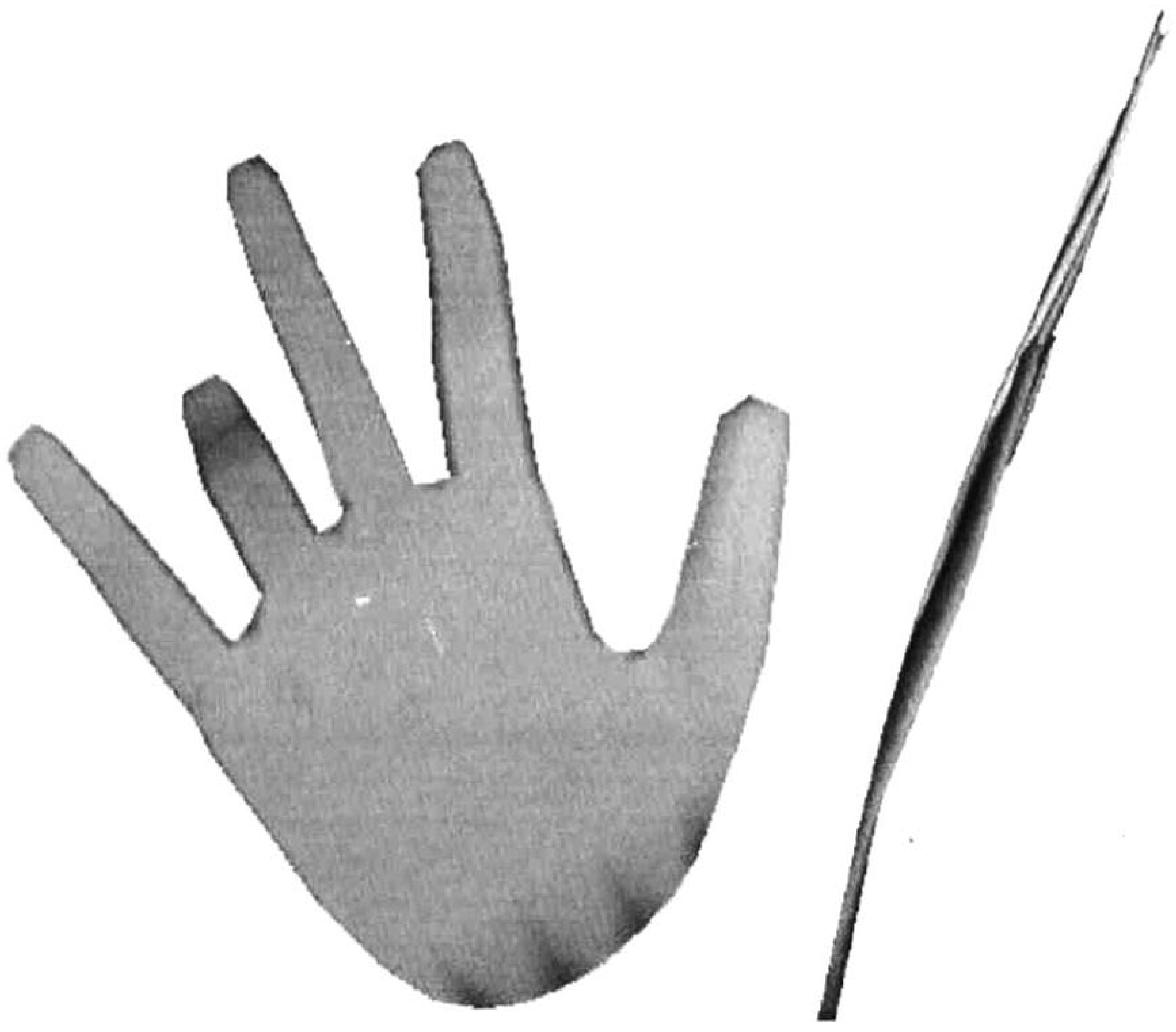

Figure 4.27: The skin mesh thickness measure $0.2mm$ and it is capable to be deformed interactively by the user.

Interaction

All virtual surgery must allow the user to interact with the model; this interaction can be performed by deforming the model or by cutting the elements, both interactions are always allowed on the simulation. In our case of study, the model consists of the dorsal skin of the hand, allowing to simulate an open surgery, the skin can be deformed and dissected as is shown in Figures 4.27, 4.28.

refinement

The evaluation of the refinement can be analyzed from the first two tests; it is necessary to find a trade-off between the DOFs and the speed of cutting. Another factor to consider is the average number of cut tetrahedrons to achieve with the desired surgery, this factor, depends on the distance of the incision. A long incision will be at the most of 6cm, this size will depend on the size of the elements, however, in all the analyzed meshes the number of cut elements does not exceed of 50 elements. Therefore, specifically for the employed equipment an efficient simulation is between 600 and 800 elements.

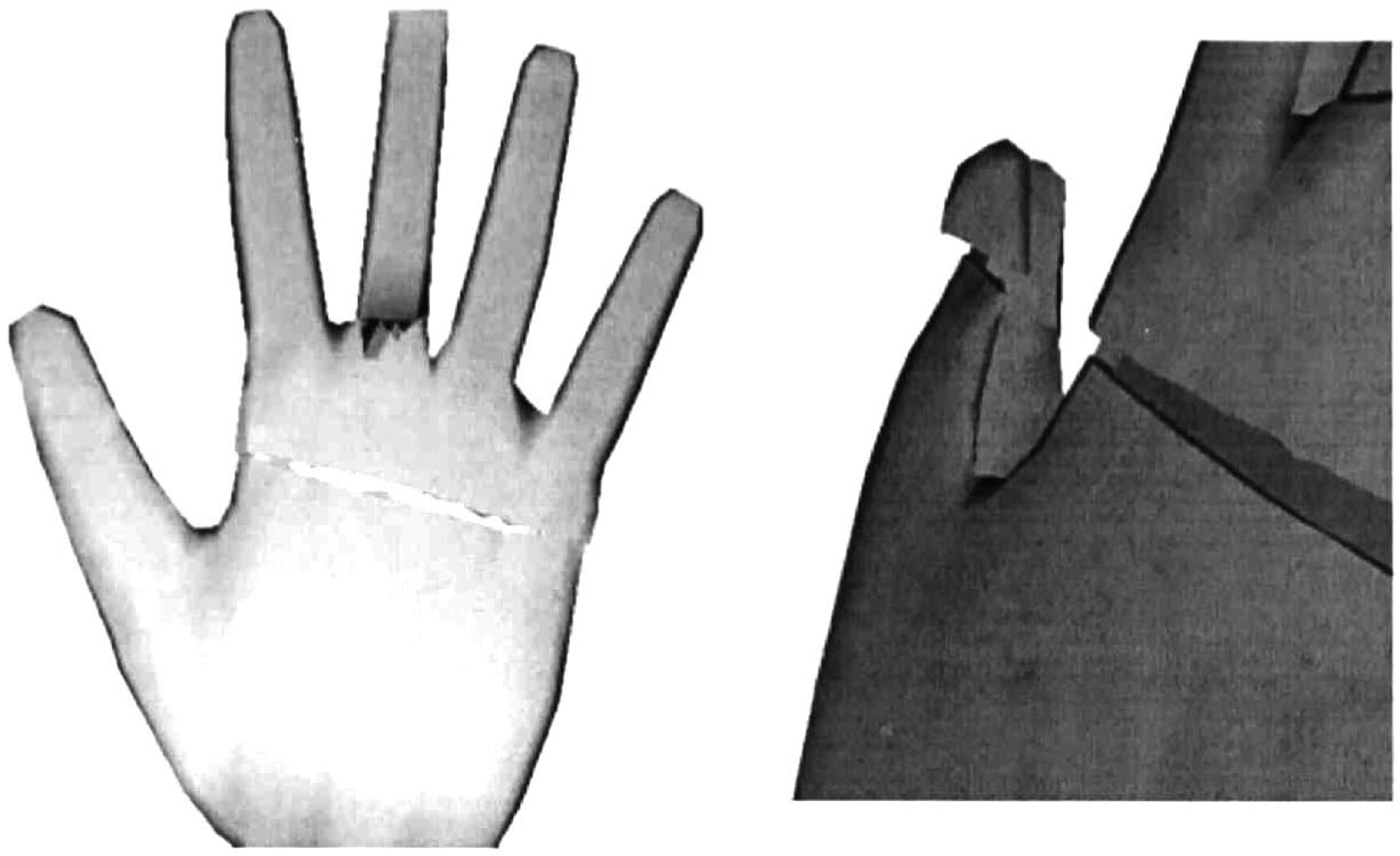

Figure 4.28: Cut of the skin of the hand.

4.4.4 Other remarks

The use of an implicit method gives better results than the explicit method, and it also ensures the stability for all the supported simulation time steps. However, to generate a real time simulation, considering the increment of the DOFs generated by the cutting process, it is necessary to modify the time step. We can achieve this problem by two forms, *measuring the time* and *time constant'for a single DOF.*; the first one consists in computing the differences between real world time (i.e. operative system time) and the simulation time (similar as the test of performance) and modify the time step each time the user cuts an element. The other option is to store in a constant *DOFtime*, the value of time step required for one single DOF (using mapping), when the user cut the model the time step is computed by $n \times DOFtime$ giving an approximation of the time step required to have a real time simulation, for our purposes the second options its suitable for surgery simulation.

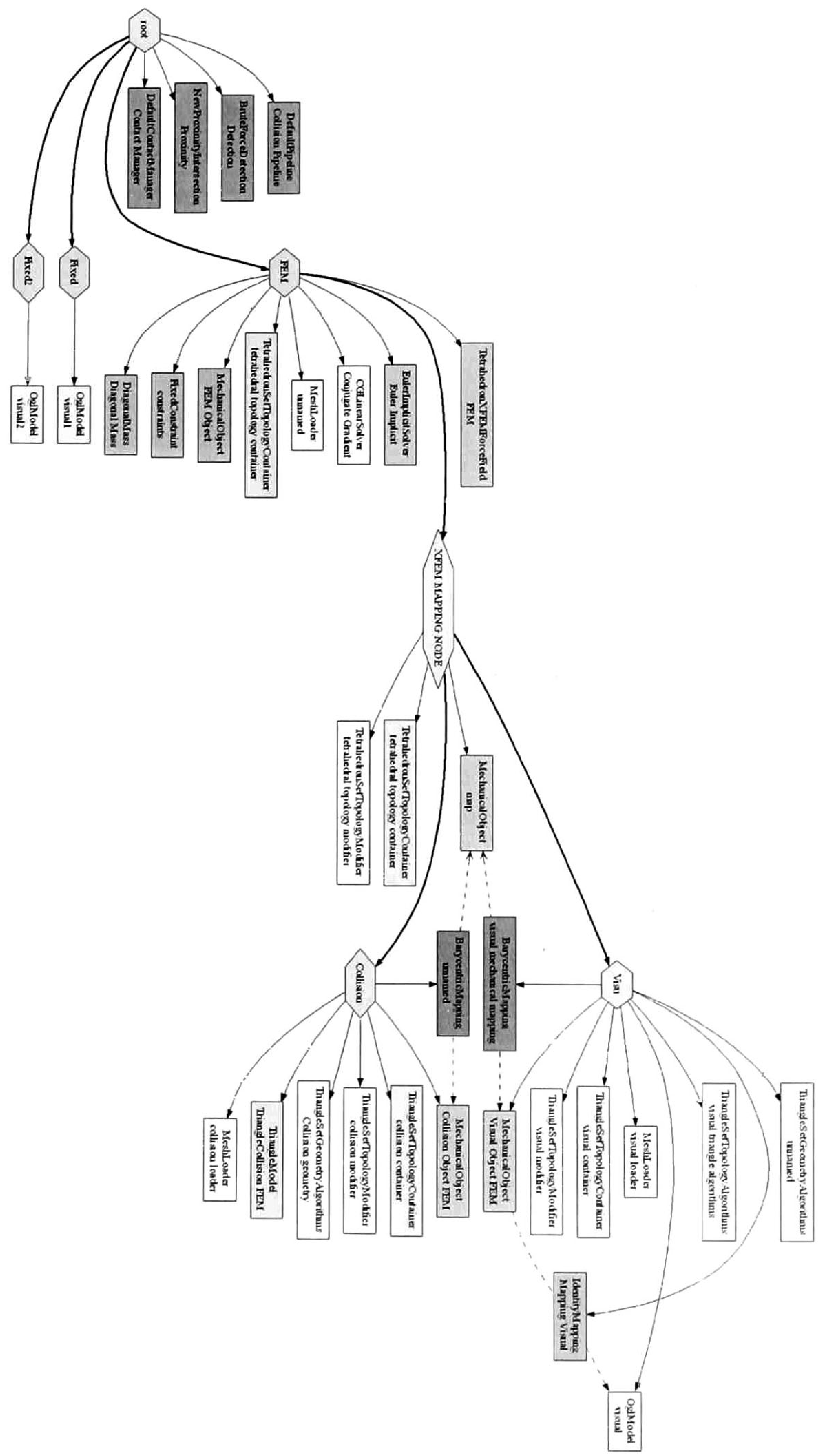
root
DefaultPipeline Collision Pipeline
BruteForceDetection Detection
NewProximityIntersection Proximity
DefaultContactManager Contact Manager
Fixed2
Fixed
OglModel visual2
OglModel visual1
FEM
DiagonalMass Diagonal Mass
FixedConstraint constraints
MechanicalObject FEM Object
TetrahedronSetTopologyContainer tetrahedral topology container
MeshLoader unnamed
CGLinearSolver Conjugate Gradient
EulerImplicitSolver Euler Implicit
TetrahedronXFEMForceField FEM
XFEM MAPPING NODE
TetrahedronSetTopologyModifier tetrahedral topology modifier
TetrahedronSetTopologyContainer tetrahedral topology container
MechanicalObject map
BarycentricMapping unnamed
BarycentricMapping visual mechanical mapping
Collision
Visu
MeshLoader collision loader
TriangleModel TriangleCollision FEM
TriangleSetGeometryAlgorithms Collision geometry
TriangleSetTopologyModifier collision modifier
TriangleSetTopologyContainer collision container
MechanicalObject Collision Object FEM
MechanicalObject Visual Object FEM
TriangleSetTopologyModifier visual modifier
TriangleSetTopologyContainer visual container
MeshLoader visual loader
TriangleSetTopologyAlgorithms visual triangle algorithms
TriangleSetGeometryAlgorithms unnamed
IdentityMapping Mapping Visual
OglModel visual

Figure 4.29: Scene graph, testing the skin of the hand in 3D

Chapter 5

Conclusion

In this thesis, it has been designed a framework focused on the efficient application of the XFEM; this framework includes the embedded design of an alternative mapping method, that allows to associate, in an easy and direct manner, the diverse topologies when the cut of an element appears. The mapping also stores the association of point-tetra, which makes faster the update of the assignation of visual or collision points (i.e. vertices), with its corresponding virtual element while the user is cutting; all these, considering the side of the cut plane.

Also, it has been created a semiprogressive cutting algorithm, which can be employed in the medical context if the working model contains an appropriate refinement. This algorithm uses the advantages of the mapped method, obtaining in this manner a fast update of the association between topologies. Moreover, this method simulates a cutting tool by the creation of a ″quad of collision" and its length can be controlled by the user.

In order to acquire more realism to the simulation, some problems were affronted such e.g., the generation of an internal mesh, the dynamic adaptation of the simulation time according to the system time (real world time), the association of discontinuous elements and its neighbors, and also were considered that the meshes may be different and therefore the association of elements is not directly (eg. many visual triangles in a facet of one tetrahedron).

SOFA framework has been employed to implement the approach, making it easy by the possibility of reutilize the code of the FEM through the inheritance of classes. Furthermore, SOFA provides the virtual environment and also the opportunity to include tools and haptics.

The approach has been tested on an open surgery simulation through a machine with characteristics accessible to the people, consequently, with a better computer, the refinement of the meshes can be improved and ,in that way, the delay of the semiprogressive cutting can be ignored.

Additionally, a variety of meshes in 2D and 3D where tested obtaining a considerable differences in the creation of nodal DOFs, in consequence, it is proved that the XFEM do not impact strongly the simulation performance, allowing real-time simulations. However, during the creation of the approach many questions have been generated that can be considered in the future work, some o these questions are:

- How to control efficiently the self-collisions between discontinuous elements?
- How can be improved the simulation with the use of multiresolution methods?
- How to control textures taking into account the internal mesh generation?
- Is it possible to create an efficient fully-progressive cutting algorithm?

Furthermore, it is contemplated the possibility to exploit the computer resources through the parallelism methods.

This work was completely focused on the XFEM, providing practical methods to implement it in a simulation that fulfill with stability, accuracy, interactivity and real time; which are the properties required for a virtual surgery simulation. An open surgery simulation was presented and by simulating the skin we show that, with very small elements, these properties are maintained, avoiding slivers and ensuring compatibility between the different topologies.

The inclusion of this framework will make possible the generation of more complex simulations, in which can be possible the interaction of diverse models (organs) that interact together and, in this manner, design simulations with mayor impact in the medical area, such as the extraction of a tumor or the fully physical modeling of one part of the body.

Bibliography

[1] National library of medicine: The visible human project website. http://www.nlm.nih.gov/research/visible/. July 2005.

[2] Jérémie Allard, Stéphane Cotin, François Faure, Pierre-Jean Bensoussan, François Poyer. Christian Duriez, Hervé Delingette, and Laurent Grisoni. Sofa - an open source framework for medical simulation. In *Medicine Meets Virtual Reality (MMVR)*, 2007.

[3] Michel A. Audette, Alexander Fuchs, Oliver R. Astley, Yoshihiko Koseki, and Kiyoyuki Chinzei. Towards patient-specific anatomical model generation for finite element-based surgical simulation. In *IS4TH*, pages 340–352, 2003.

[4] I. Babuska and J. M. Melenk. The partition of unity method. *International Journal of Numerical Methods in Engineering*, 40:727–758, 1997.

[5] T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *International Journal of Numerical Methods in Engineering*, 45(5):601–620, 1999.

[6] T. Belytschko and R. Gracie. On xfem applications to dislocations and interfaces. *International Journal of Plasticity*, 23:1721–1738, 2007.

[7] Jeffrey Berkley, George Tukiyyah, Daniel Berg, Mark Ganter, and Suzanne Weghorst. Real-time finite element modeling for surgery simulation: An application to virtual suturing: An application to virtual suturing. *IEEE Transactions on Visualization and Computer Graphics*, 10(3), 2004.

[8] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. In *Pacific Graph.*, pages 377–386, 2003.

[9] Daniel Bielser and Markus Gross. Interactive simulation of surgical cuts. In *Pacific Graph.*, pages 116–125, 2000.

[10] Grigore C. Burdea. Haptics issues in virtual environments. In *Proceedings of the Computer Graphics International*, pages 295–302, 2000.

[11] Hüseyin K. Cakmak and Uwe Kühnapfel. Animation and simulation techniques for vr-training systems in endoscopic surgery. In *Proceedigns of the Eurographics Workshop on Animation and Simulation (EGCAS)*, 2000.

[12] Jianyung Chai, Jian SUN, and Zesheng Tang. Hybrid fem for deformation of soft tissues in surgery simulation. *Medical Imaging and Augmented Reality*, pages 298–303, 2001.

[13] Kup-Sze Choi, Hanqiu Sun, and Pheng-Ann Heng. An efficient and scalable deformable model for virtual reality-based medical applications. *Artificial Intelligence in Medicine*, pages 51–69, 2004.

[14] Stéphane Cotin, Hervé Delingette, Nicholas Ayache, Inria Sophia Antipolis, and Route Des Lucioles. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(7):437–452, 2000.

[15] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from a manifold mesh. In *Computer Animation (CA'02*, pages 225–229. IEEE Computer Society, 2002.

[16] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from manifold tetrahedralisation : application to real-time surgical simulation. *Medical Image Analysis*, 9(2):113–122, April 2005.

[17] Matthias Harders, R. Hutter, A. Rutz, P. Niederer, and Gábor Székely. Comparing a simplified fem approach with the mass-spring model for surgery simulation. In *Proceedings of medicine Meets Virtual Reality*, 2003.

[18] Michael Hauth. *Visual Simulation of Deformable Models.* PhD thesis, University of Tübingen, 2004.

[19] Richard Paul Holbrey. *Virtual Suturing for Training in Vascular Surgery.* PhD thesis, University of Leeds, 2005.

[20] David V Hutton. *Fundamentals of Finite Element Analysis.* McGraw-Hill, 2004.

[21] Lenka Jeřábková and Torsten Kuhlen. Stable cutting of deformable objects in virtual environments using xfem. *IEEE Comput. Graph. Appl.*, 29(2):61–71, 2009.

[22] Lenka Jeřávková. *Interactive Cutting of Finite Elements based Deformable Objects in Virtual Environments.* PhD thesis, RWTH Aachen University, Zlín, Czech Republic, 2007.

[23] A.R. Khoei, S.O.R. Biabanaki, and M. Anahid. Modeling holes and inclusions by level sets in the extended finite element method. *Computer Methods in Applied Mechanics and Engineering*, 197:1100–1114, February 2008.

[24] Olaf Körner and Reinhard Männer. Implementation of a haptic interface for a virtual reality simulator for flexible endoscopy. In *Proceedings of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator(HAPTICS 2003)*, pages 278–284. IEEE, 2003.

[25] U. Kuhnapfel, H. K. Çakmak, and H. Maaß. 3d modeling for endoscopic surgery, 1999.

[26] Christian Laugier, César Mendoza, and Kenneth Sundaraj. Towards a realistic medical simulator using virtual environments and haptic interaction. *Robotics Research: STAR*, 6:289–306, 2003.

[27] Alex Linblad and George Turkiyyah. A physically-based framework for real-time haptic cutting and interaction with 3d continuum models. *SPM*, 2007.

[28] G. R. Liu and S. S. Quek. *THE FINITE ELEMENT METHOD A practical course.* Butterworth-Heinemann, 2003.

[29] Thomas Menouillard, Julien Réthoré, Alain Combescure, and Harriddh Bung. Efficient explicit time stepping for the extended finite element method (x-fem). *International Journal for Numerical Methods in Engineering*, April 2006.

[30] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proceedings of Siggraph*, pages 49–54, 2002.

[31] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of the 2004 conference on Graphics interface*, pages 239–246, 2004.

[32] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc*, 23:385–392, 2004.

[33] Andrew B. Mor and Takeo Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Proceedings of Medical Image Computing & Computer Assisted Intervention*, pages 598–607, 2000.

[34] N. Moës. J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal of Numerical Methods in Engineering*, 46(1):131–150, 1999.

[35] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Eurographics 2005 State of the Art Report*, 2005.

[36] Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. Preserving topology and elasticity for embedded deformable models. In *ACM Transactions on Graphics (Proc. of SIGGRAPH)*. ACM, august 2009. to appear.

[37] Matthieu Nesme, Yohan Payan, and François Faure. Efficient, physically plausible finite elements. In J. Dingliana and F. Ganovelli, editors, *Eurographics (short papers)*, august 2005.

[38] Han-Wen Nienhuys. *Cutting in deformable objects.* PhD thesis, University of Utrecht, 2003.

[39] Bernhard Reitinger, Alexander Bornik, Reinhard Beichel, and Dieter Schmalstieg. Liver surgery planning using virtual reality. *IEEE Computer Graphics and Applications*, 26(6):36–47, 2006.

[40] Hans-Martin Schmidt. *Surgical anatomy of the hand.* Thieme, 2003.

[41] Mert Sedef, Evren Samur, and Cagatay Basdogan. Real-time finite element simulation of linear viscoelastic tissue behavior based on experimental data. *IEEE Computer Graphics and Applications*, 2006.

[42] Guy Sela, Jacob Subag, Alex J. Lindblad, Dan Albocher, Sagi Schein, Gershon Elber, and George Turkiyya. Real-time haptic incision simulation using fem-based discontinuous free form deformation. In *SPM 06: Proceedings of the 2006 ACM symposium on Solid and Physical Modeling*, pages 75–84. ACM Press, 2006.

[43] D. Serby, M. Harders, and G. Székely. A new approach to cutting into finite element models. In *Medical Image Computing and Computer Assisted Intervention (MICCAI), number 2208 in LNCS*, pages 425–433. Springer-Verlag, 2001.

[44] Eftychios Sifakis, Kevin G. Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *2007 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 73–80, August 2007.

[45] Denis Steinemann, Matthias Harders, Markus Gross, and Gabor Szekely. Hybrid cutting of deformable solids. In *Proceedings of the IEEE Virtual Reality Conference*, pages 425–433, 2006.

[46] N. Sukumar, D.L. Chopp, N. Moes, and T. Belytschko. Modeling holes and inclusions by level sets in the extended finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190(46):6183–6200, September 2001.

[47] F.P. Vidal, F. Bello, K.W Brodlie, N.W. John, D.Gould, R. Philips, and N.J. Avis. Principles and aplications of computer graphics in medicine. *Computer Graphics*, pages 113–137, 2006.

[48] Lara M. Vigneron, Jacques G. Verly, and Simon K. Warfield. Modelling surgical cuts, retractions, and resections via extended finite element method. In *Proceedings of Medical Image Computing & Computer Assisted Intervention*, volume 7 of LNCS, pages 311–318. Springer Verlag, 2004.

[49] Lara M. Vigneron, Jacques G. Verly, and Simon K. Warfield. On extended finite element method (xfem) for modelling of organ deformations associated with surgical cuts. In *ISMS*, volume 3078 of LCNS, pages 134–143. Springer Verlag, 2004.

[50] Zhennan Yan, Lixu Gu, Pengfei Huang, Sizhe Lv, Xiao Yu, and Xianming Kong. Soft tissue deformation simulation in virtual surgery using nonlinear finite element method. *IEEE EMBS*, 2007.

[51] Shaoting Zhang, Lixu Gu, Weiming Liang, Jingsi Zhang, and Feng Qian. Real-time virtual surgery simulation employing mm-model and adaptive spatial hashing. *ICAT*, 2006.

[52] O. C. Zienkiewicz and R.L. Taylor. *The Finite Element Method for Solids and Structural Mechanics.* Elsevier Butterworth-Heinemann, sixth edition, 2005.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Cortando Tejido Suave usando el XFEM: una Perspectiva de Cirugía Virtual Cutting Soft Tissue using the XFEM: a Virtual Surgery Perspective

del (la) C.

LUIS FERNANDO GUTIERREZ PRECIADO

el día 28 de Agosto de 2009.

[illegible]
Dr. Pablo Moreno Villalobos
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

[illegible]
Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

[illegible]
Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

[illegible]
Dr. Mario Angel Siller González Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara