

Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA
in
Automatica e Ricerca Operativa

Ciclo XXIII°

Settore scientifico-disciplinare di afferenza: ING-INF04

Soft Tissue Modeling for Virtual Reality Surgery
Simulator with Haptic Feedback

Presentata da: Dott. Ing. Gianluca De Novi

Coordinatore Dottorato

Prof. Paolo Toth

Relatore

Prof. Claudio Melchiorri

Esame finale anno 2011

Author's Web Page: <http://phd.deis.unibo.it/?phd=gianluca.denovi@unibo.it>

Author's e-mail: gianluca.denovi@unibo.it

Author's address:

DEIS – Dipartimento di Elettronica Informatica e Sistemistica

Alma Mater Studiorum – Università di Bologna

Viale Risorgimento 2

40 136 Bologna

Italia

tel. +39 051 2093001

fax. +39 051 2093073

web: <http://www.deis.unibo.it>

Dedicated to the little boy
who believes that sometimes
dreams can become reality

Contents

Introduction	1
1 – Soft Tissue Models for Surgery Simulation	
1.1 – The spring damper mass model	5
1.2 – 3D meshes	8
1.3 – Collision detection	11
1.4 – Cuttings and fractures simulation	15
2 – Metaballs and Implicit Surfaces	
2.1 – Metaballs	19
2.2 – Volume approximation using sphere trees	23
2.3 – Surface extraction	25
2.4 – Performances evaluation	28
3 – Fluid and Rigid Bodies Simulation	
3.1 – Particles approximation	33
3.2 – Particles collision detection model	35
3.3 – Rigid body simulation	38
4 – Soft Tissue Model	
4.1 – 3D mesh generation	41
4.2 – 3D mesh parameters assignment	47
4.3 – Blobby meshes vs FEM	51
5 – Local Interaction Model	
5.1 – Local deformation model	53
5.2 – Local interaction model	56
5.3 – Multi-body interaction	59
5.4 – Multilayered surfaces	62

5.5 – Haptic textures	63
6 – Cuttings and Fractures Simulation	
6.1 – Fractures simulation	67
6.2 – Cuttings simulation	70
6.3 – Cuttings optimization	75
7 – Conclusions and Final Remarks	
7.1 – Developed Software	78
7.2 – Some results	83
7.3 – Conclusions and future works	85
List of Figures	87
Bibliography	91
Acknowledgements	93

Introduction

During the last thirty years, minimal invasive surgery (MIS) became an important milestone in the surgery scenario, improving in a consistent way the patient's safety and reducing the time of permanence within hospitals. The MIS approach to surgery allows the surgeon to perform a classic operation (open surgery) avoiding opening the patient body, but just using few small holes where he/she can introduce particular surgical tools and, in some cases, a camera too. There are many different MIS surgical practices but the principal two categories are the laparoscopy (thorax and abdomen) and catheter-based procedures. The first typology of operations requires the use of a camera in order to see the surgical operation field and to control the surgical tools. The use of MIS tools requires very strong manual skills. As a result, surgeons need to spend a significant amount of time improving their skills surgical through the use of trainers. Usually, surgeons need to use trainers for new surgical procedures or for new surgical tools. There are different kind of trainers used to improve the surgeons skills:

- Cadavers.
- Animals.
- Trainer boxes.
- Virtual reality simulation systems.

The use of cadavers and animals raise ethical issues and are allowed in only a few countries. Often, their use requires special permissions. One issue with this method is represented by the fact that the animal should be alive because the behavior of the in-vivo tissues is completely different from the dead tissues. The trainer box represents an acceptable solution. It requires the use of a box in order to simulate the abdomen or the thorax of the patient and it includes the use of latex/silicon organ models. The trainer box results in an expensive educational tool because, after few simulations, the silicon models need to be replaced and the behavior of those models is not realistic.

Virtual reality simulators (VRS) represent an alternative training system provided with very interesting features. A computer usually produces the VRS's. Surgical tools are interfaced with the computer and simulation software able to create, in computer graphics (CG), a realistic surgical scene. With simulation software, it is possible to simulate (in an approximate way) the aspect and the behavior of real tissues. The VRS method presents interesting features like the replicability¹, the traceability², and the simulation of critical events. Moreover, through the use of haptic devices³ it is possible to interact with the virtual scene. This is expressed through feeling all contact forces and consequently increasing the realism of the experience. It is evident that the quality of the soft tissue model and the interaction model used for the simulation are paramount in order to achieve a superior realism for the simulation. This thesis presents a novel approach based on metaballs (a particle used as field generator) and iso-surfaces for modeling soft/rigid tissues. With this approach, the most efficient models can be obtained. Additionally, the performances of surgery simulators may be increased when the high complexity of the scene and of the surgical tasks is very demanding in terms of computational requirements. Moreover, the new model improves collision detection performances and adds the possibility to render multi-layer surfaces of different compliance, such as soft tissues and bones. The following approach represents an excellent choice

¹ Possibility to simulate a particular situation multiple times.

² Possibility to track the user's performance.

³ The haptic tool is a particular kind of robot able to apply to the end-effector (handle held by the user) a force (with a proper orientation and module) in order to re-create the sensation of the touch of a virtual object.

when the complexity of the scene models is more important than the accuracy of the tissue behavior.

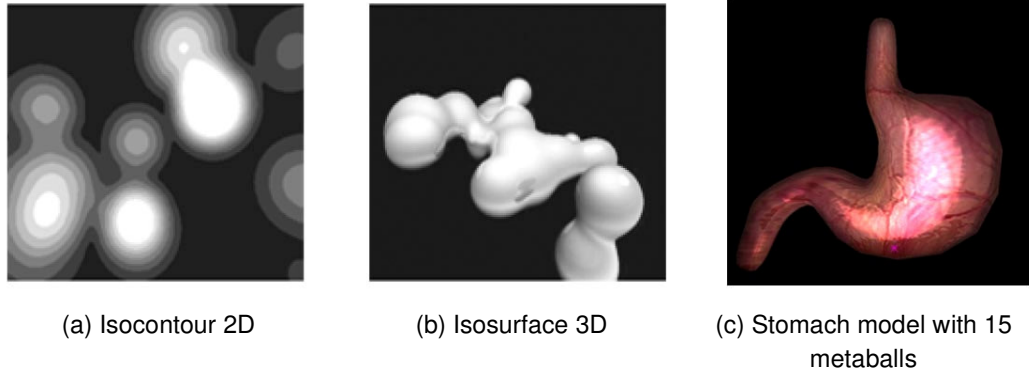


Fig. 1. Implicit Model.

Haptic devices and virtual reality environments are tools more and are more frequently and successfully used in training systems for surgical applications. In this context, in order to simulate a realistic surgical scene, it is necessary to represent a large number of entities, each of them often with very complex geometries. As shown in [1], in real-time simulation environments there are two main approaches for tissue representation: the first one consists of the spring-particle model; the second depiction is represented by the implicit model [2], [3]. The spring-particle (or spring-mass) model is more accurate than the implicit model, but it is quite expensive in terms of computational requirements and cannot be used in simulations where there are very complex geometries. On the other hand, the implicit model describes very complex geometries with a little number of particles, but does not allow local deformations of the surfaces if only few metaballs are used [4]. In this thesis we present an extension of the implicit model that creates virtual environments for surgical simulation with geometrically complex scenes, multi-body interactions, and haptic feedback. The basic idea is to simplify the overall dynamic model of the simulated entities by using the spring-particle model for the global deformations, and an extended implicit model for the local deformations. In surgery, the organs' geometry generally presents a rounded aspect, and in only a very few cases do they have sharp edges/flat planes. One of the most convenient approaches presented in the literature to create organic shapes is represented by the metaballs primitive (*Blobby*

Modeling) [5], [6]. Using Blobby Modeling, the organs can be represented with a set of few particles used as field generators and extracting an implicit surface (or iso-surface) by setting a proper field threshold value τ (see Fig. 1[a], 1[b], and 1[c]). One of the interesting features of the implicit approach is that it is possible to convert the geometry in a sum of factors (*particle fields*) and then it is possible to achieve a fast collision detection between different organs and between the organs and virtual surgical tools. By adding dynamic properties to the implicit model, it is possible to create geometrically complex scenes, with few particles (Fig. 1[c]) and with low interaction costs. This thesis is organized as follows.

The first chapter presents a quick look at the soft tissue model typically used in the surgical simulations through describing the principal features and limitations of them. In the second chapter, the metaballs approach and how to use this technique to approximate volumes in order to create very complex geometries for surgery simulations is introduced. The third chapter presents how to use the metaballs in order to simulate rigid bodies and fluids while the fourth chapter explains how extend the metaballs approach in order to simulate soft tissue and how to reduce the complexity of the geometry used for the simulation. In the fifth chapter, the interaction model used for the proposed approach and how to extend it, simulating multilayered surfaces using haptic textures is presented. The sixth chapter discusses the simulation of cuttings and fractures of the tissue. Finally, the seventh chapter shows the conclusions and the final remarks for the proposed approach.

Chapter 1

Soft Tissue Models for Surgery Simulation

This chapter briefly introduces the classic approach commonly used to develop virtual reality surgery simulations, trying to show the principal advantages and limitations of it.

1.1 - The Spring Damper Mass Model

The common approach used for commercial and academic simulations of soft tissues for surgery is based on the spring-damper-mass model (SDM). The SDM approach approximates volume using a 3D network of particles (points in $\epsilon\mathbb{R}^3$) connected by visco-elastic links (Fig 1.1). Using the SDM model, it is possible to assign the material properties the right value for the set of scalar variables m (*mass*), k (*stiffness or elasticity constant*) and h (*damping or viscosity constant*). As initial conditions in for the solution of the equations of the model, it is necessary to set the initial length of the visco-elastic link l . It is possible also to use functions (Hyper-Elastic material) instead of a constant value for k and h ($k(x)$ and $h(v)$, where x is the module of the

deformation and v is the module of the velocity), but usually, it is used a linear behavior in order to minimize the computational cost (reducing the accuracy). Another improvement for the model is achieved by defining two-threshold values τ_d as link deformation limit and τ_b as breaking limit.

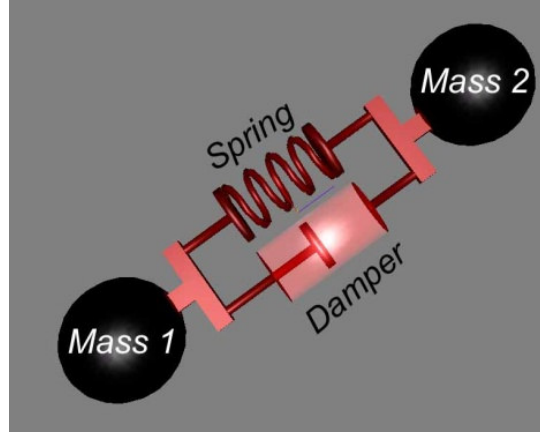


Figure 1.1 – Visco-elastic Link model

At this point it is possible to define two data structures used to implement the SDM model.

$$\begin{array}{l}
 \text{Particle} \left\{ \begin{array}{l}
 \text{mass } m \\
 \text{position } \mathbf{p} \\
 \text{speed } \left\{ \begin{array}{l} \text{internal } \mathbf{v}_i \\ \text{external } \mathbf{v}_e \end{array} \right. \\
 \text{acceleration } \left\{ \begin{array}{l} \text{internal } \mathbf{a}_i \\ \text{external } \mathbf{a}_e \end{array} \right. \\
 \text{force } \left\{ \begin{array}{l} \text{internal } \mathbf{F}_i \\ \text{external } \mathbf{F}_e \end{array} \right. \\
 \text{energy } E \\
 \text{ray } r
 \end{array} \right.
 \end{array}
 \quad
 \begin{array}{l}
 \text{Link} \left\{ \begin{array}{l}
 \text{stiffness } k \\
 \text{viscosity } h \\
 \text{deformation limit } \tau_d \\
 \text{breaking limit } \tau_b \\
 \text{length } h \\
 \text{p1 index} \\
 \text{p2 index}
 \end{array} \right.
 \end{array}$$

It is possible to see that the data structures require some extra fields used to implement the simulation algorithm in the right way. Considering only the basic model with the first three parameters and considering k and h as constants, it is possible to define the elastic component using the Hook law as:

$$\mathbf{F}e = -k\mathbf{x} \tag{1.1}$$

Where $\mathbf{F}e$ represent the elastic force and \mathbf{x} the deformation of the link.

$$\mathbf{x} = \|\mathbf{p}'_2 - \mathbf{p}'_1\| \cdot (|\mathbf{p}'_2 - \mathbf{p}'_1| - |\mathbf{p}_2 - \mathbf{p}_1|). \quad (1.2)$$

Where \mathbf{p}_2 and \mathbf{p}_1 are the initial positions of two particles and \mathbf{p}'_2 and \mathbf{p}'_1 the positions of the same particles after the deformation.

For the viscose component, the viscose force can be defined as

$$\mathbf{F}_v = -\lambda \mathbf{v}. \quad (1.3)$$

It is necessary to separate the forces, speed, and acceleration in two different contributes (internal and external) because it is necessary to apply the dumping only on internal forces and not, for instance, on the external forces. For example, if we apply the link damping on the external forces the movements of the body under the gravity effect, it is conditioned by the link damping and is not a realistic behavior. By keeping the two contributes separate, the integration scheme become the following:

$$\mathbf{v}_i(t) = \int \frac{\mathbf{F}_i}{m} dt$$

$$\mathbf{v}_e(t) = \int \frac{\mathbf{F}_e}{m} dt$$

$$\mathbf{p}(t) = \int \mathbf{v}_i(t) dt + \int \mathbf{v}_e(t) dt.$$

Considering a deformation at the instant t , the link deformation can be expressed by the following relations

$$d(t_0) = |\mathbf{p}_2(t_0) - \mathbf{p}_1(t_0)|; \quad d(t_1) = |\mathbf{p}_2(t_1) - \mathbf{p}_1(t_1)|;$$

$$\Delta d(t_1) = d(t_1) - d(t_0).$$

Therefore, using the (1.2) it becomes

$$\mathbf{x}(t_1) = \|\mathbf{p}_2(t_1) - \mathbf{p}_1(t_1)\| \cdot \Delta d(t_1). \quad (1.4)$$

This represents the direction of the elastic force. The forces on the two particles connected at the two ends of the link at the generic instant t will be:

$$\mathbf{F}_e(t) = -k\mathbf{x}(t); \quad \mathbf{F}_v(t) = -\lambda\dot{\mathbf{x}}(t).$$

$$\mathbf{F}_{1 \rightarrow 2}(t) = \mathbf{F}e(t) + \mathbf{F}v(t).$$

$$\mathbf{F}_{2 \rightarrow 1}(t) = -\mathbf{F}_{1 \rightarrow 2}(t).$$

These forces will be attractive if $\Delta d > 0$ and will be repulsive if $\Delta d < 0$ (figure 1.2). It is also important to study the behavior for the SDM model and the influences of the integration method used on it, considering a discrete time scale, but it is not relevant for this thesis focused on the optimization of the computational costs.

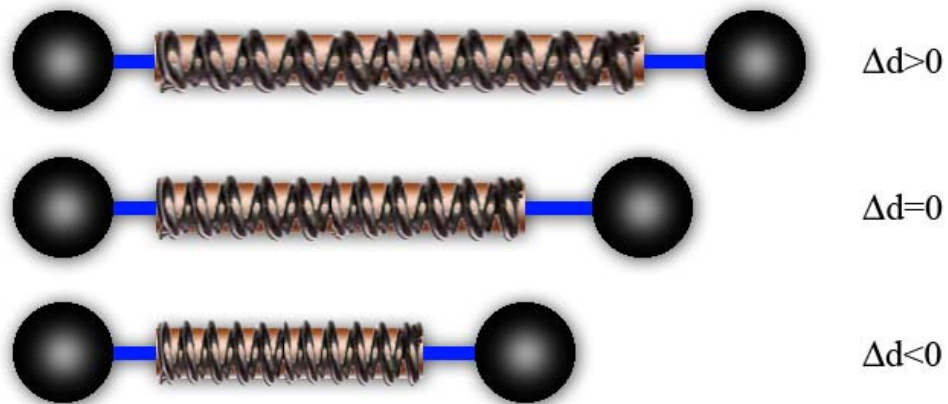


Figure 1.2 – Link deformation

1.2 - 3D Meshes

After the definition of the SDM model, it is important to define how to use it to simulate a volume of soft tissue. The typical approach is to approximate the volume as a network of particles and links (3D dynamic mesh). The connections scheme for the geometry usually is the tetrahedron, so the entire volume is tessellate, using as finite element a tetrahedron, as shown in the figure 1.3.

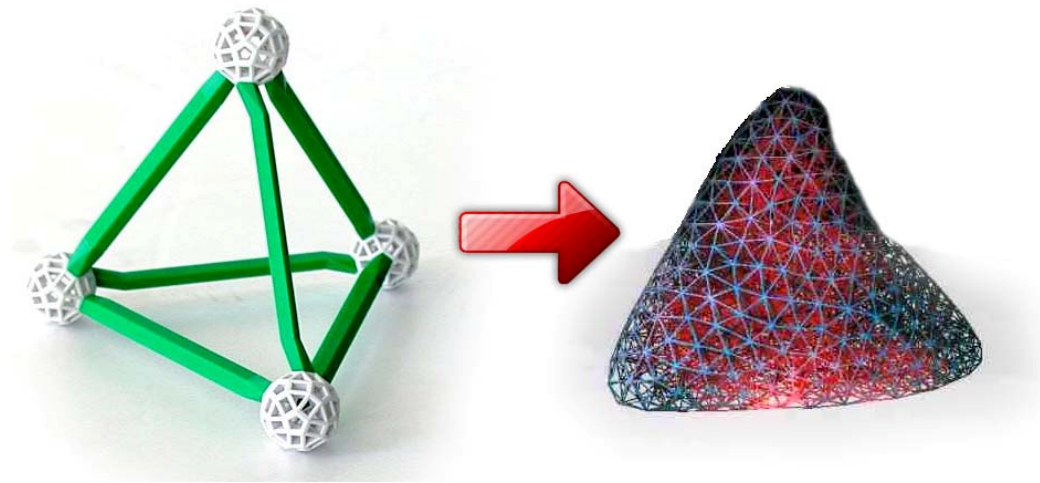


Figure 1.3 – Tetrahedron and 3D mesh based on tetrahedron

Using this approach, it is possible to approximate the behavior/deformations of the volume. The accuracy of the approximation depends from some parameters:

- Link's properties (k , h , etc.).
- Masses distribution.
- The tessellation resolution.

It is easy to understand that, through the use of a high mesh resolution, it is possible to obtain a more realistic behavior. Unfortunately, the computational cost for the simulation depends on the mesh complexity, so it is necessary to find a precise tradeoff between a good resolution and the desired performances of the simulation. The mesh resolution also can be constant or variable. It is possible to use different sizes of a tetrahedron; for example, one can increase the resolution close to the edges, where it is important to have a higher accuracy, and reduce the rest of the volume. For the surgery simulation, the typical approach is to use a constant mesh resolution, because it is easier to tune all mesh parameters and it make the cut simulation easier. When using a 3D mesh starting from a solid volume model for TAC, CTScans, and so on, it is possible to use the marching tetrahedron algorithm (derived from the marching cube [7] and extended to create volumetric meshes instead of surfaces). The marching tetrahedron algorithm is able to create a regular and uniform mesh that represents an excellent mesh for real-time simulations.

Another important task performed by the marching tetrahedron algorithm is to create the external surface for the model. This is necessary to draw the model as a normal polygonal surface. The 3D mesh generation is a task that needs to be performed just one time (off-line) and not during the simulation. All the mesh data is usually stored in the following data structures:

- Particles array
- Links array
- Polygons array

Using the mentioned data structures, the simulation pipeline is organized as shown in figure 1.4.

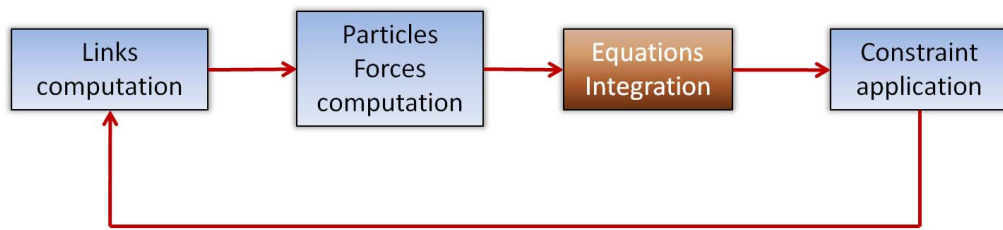


Figure 1.4 – Physic simulation loop

The first block of the pipeline computes the deformation of each link and, as a final result, the forces to apply to each particle. The second block added to the visco-elastic forces is the contribution of other kinds of forces: gravitational and magnetic, for instance. The third stadium is performed by the integration of all model equations using a proper integration method and selecting an appropriate value for the Δt . A right Δt value is important for the simulation stability.

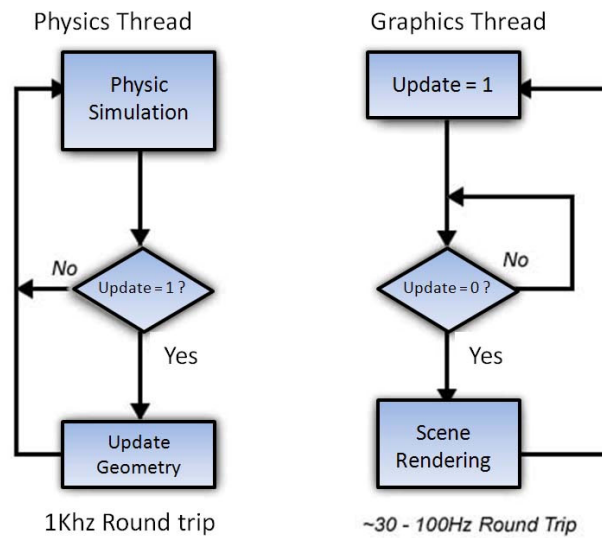


Figure 1.5 – Graphics-Physics Synchronization scheme

After the integration, it is possible to determine the position of each particle. As a result, it is possible to check if any particle violates the predetermined constraints conditions. If so, an appropriate algorithm will recover the correct position for the particle.

The physics loop must be interfaced with the graphics loop. Usually, these loops are located on different threads and the physics thread runs faster than the graphics thread. The synchronization of both threads work as shown in figure 1.5. The synchronization block is required because the physics thread is faster than the graphics thread and the geometry could change during the drawing task. Taking this into consideration, it is possible to create a backup copy of the geometry for a specific instant.

1.3 - Collision Detection

The collision detection task is very expensive to perform. In a surgery simulation, the collision detection task needs to be performed between the surgical tools and the anatomic tissues. In some cases, it is necessary to perform the collision detection between multiple organs. This is known as a multi-body simulation. There are many

different approaches to try in order to reduce the complexity of this task, but the common approach is to use a space partitioning optimization. The space partitioning algorithm reduces the number of tests that need to be performed between the surgical tool geometry and the anatomic model geometry (polygon by polygon). Usually, the surgical tool is approximated using a set of a few spheres. Consequently, the collision test needs to be performed between each sphere of the set and each polygon of the anatomic model.

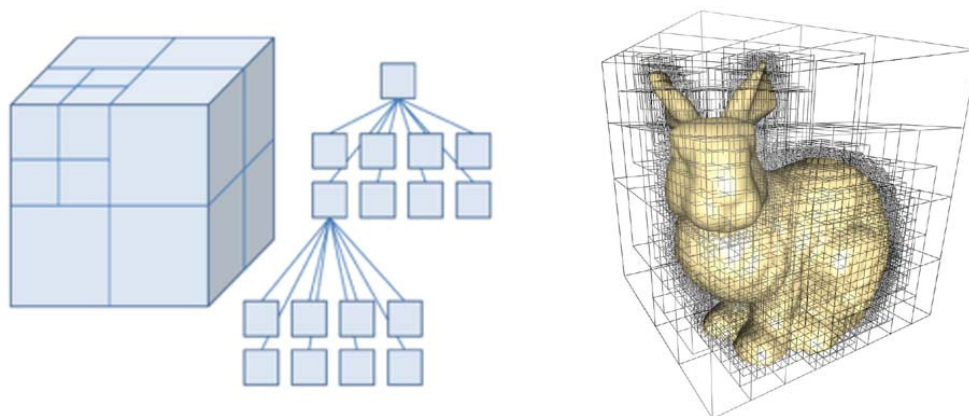


Figure 1.6 – Octree space partitioning

Using space partitioning, it is possible to reduce the number of polygons on which the test is performed. The classic space partitioning used for this task is the octree [8] that creates space partitioning as shown in figure 1.6. Using an octree, the first test performed is between the surgical tool and the first level bounding box; if the tool is inside that box, the test is recursively performed between it and the eight sub-boxes contained at the second level. If the tool is inside one of these, the test is performed again until the last level. In the event of the last box of the last level containing any polygons, the collision detection is performed between those polygons and the surgical tool. If a collision is recognized, a proper algorithm for the collision handling applies the appropriate reaction for the simulation. It is important to say that the surgical tool can be present in more than one box at the last level, so the resolution of the octree must be tuned on the right size in order to obtain the best results. In case the simulator provides haptic feedback, it is indispensable implement also contact forces between the tissue surface and the haptic tool. Assuming the

surface is a triangular mesh, the contact force \mathbf{F}_c in a generic contact point $\mathbf{p}(x,y,z)$ is represented as:

$$\mathbf{F}_c = \hat{\mathbf{n}} \cdot |\alpha\hat{\mathbf{i}} + \beta\hat{\mathbf{j}} + \gamma\hat{\mathbf{k}}|$$

with

$$\hat{\mathbf{n}} = \theta\hat{\mathbf{i}} + \vartheta\hat{\mathbf{j}} + \rho\hat{\mathbf{k}}$$

and

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} F_{1x} & F_{1y} & F_{1z} \\ F_{2x} & F_{2y} & F_{2z} \\ F_{3x} & F_{3y} & F_{3z} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

$$\begin{bmatrix} \theta \\ \vartheta \\ \rho \end{bmatrix} = \begin{bmatrix} n_{1x} & n_{1y} & n_{1z} \\ n_{2x} & n_{2y} & n_{2z} \\ n_{3x} & n_{3y} & n_{3z} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}.$$

Where $\hat{\mathbf{n}}$ is the normal on the triangle at the contact point \mathbf{p} , $\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{n}}_3$ are the normal vectors on the triangle vertices, $\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3$ (see figure 1.7) are the resultant forces on the triangle vertices (particles), and $\lambda_1, \lambda_2, \lambda_3$ are defined as

$$\lambda_1 = \frac{B(F + I) - C(E + H)}{A(E + H) - B(D + G)};$$

$$\lambda_2 = \frac{A(F + I) - C(D + G)}{A(D + G) - B(E + H)};$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2;$$

with

$$A = p_{1x} - p_{3x}; \quad B = p_{2x} - p_{3x}; \quad C = p_{3x} - p_x;$$

$$D = p_{1y} - p_{3y}; \quad E = p_{2y} - p_{3y}; \quad F = p_{3y} - p_y;$$

$$G = p_{1z} - p_{3z}; \quad H = p_{2z} - p_{3z}; \quad I = p_{3z} - p_z;$$

$\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, are the three vertices of the triangular face.

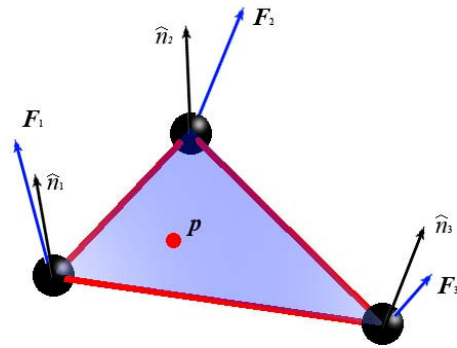


Figure 1.7 – Triangle normal and forces

After the definition of the tissue model and a basic interaction model, the physic simulation layer is almost complete. In order to be comprehensive, the simulation layer must be completed through adopting a proxy object algorithm (or God Object) [9]. The proxy algorithm is important in that it eliminates incorrect behavior in the haptic response (see figure 1.8).

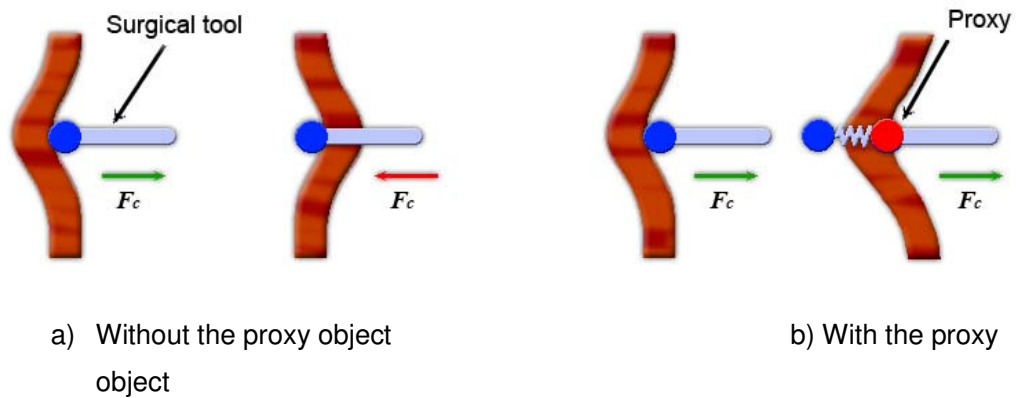


Figure 1.8 – Proxy object and soft tissue

In the figure 1.8a, without the proxy object, the surgical tool crosses the tissue section and the tissue deformation becomes inconsistent with reality. By adding the proxy object (figure 1.8b), however, the deformation results are coherent with reality and the feedback force.

1.4 - Cuttings and Fractures Simulation

The real-time cuts simulation represents an important issue today because it is very difficult to find a strategy that retain the low complexity of 3D dynamic mesh. Different approaches to this issue have advantages and disadvantages. The cut simulation typically adds complexity to the dynamic mesh, but the collision detection and the physics performances are influenced as well. There are two typical approaches used today in the commercial simulation systems:

- Tetrahedrons Split (TS) [10][11].
- Tetrahedrons Removing (TR) [12].

The first approach is based on the real geometric tetrahedron cut during the collision between the surgical tool and the dynamic mesh, as shown in figure 1.9a.

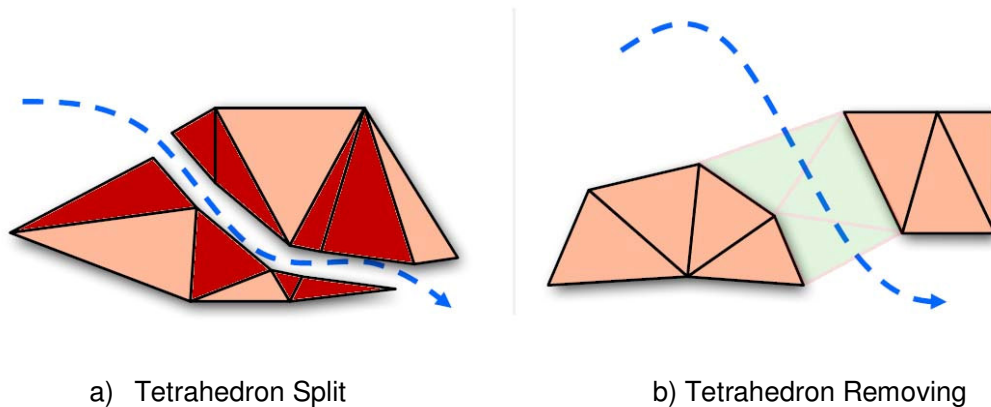


Figure 1.9 – Cut simulation

The TS represents a very accurate method but it is very expensive in terms of computational cost and it presents a very dangerous mesh complexity increment problem. Through subdividing each tetrahedron when the cutter touches the volume, the complexity of the mesh grows very quickly. A size threshold that subdivides the tetrahedron only if it is bigger than a specific size represents a possible solution; in case it is smaller, the tetrahedron can be deleted like in the TR. The TR approach decreases the complexity of the mesh by removing tetrahedron from the mesh. It is

easy to implement and it is also a fast solution. A possible problem, however, is to implement an algorithm able to recognize whether the cutter is touching an internal or an external face before to remove the tetrahedron. Another limitation for the TR is the fact that the mesh resolution should be very high in order to have a realistic cut. If the resolution is low, and in real time simulations the resolution is typically low, the incision looks serrate and not as smooth as a real cut (figure 1.10).

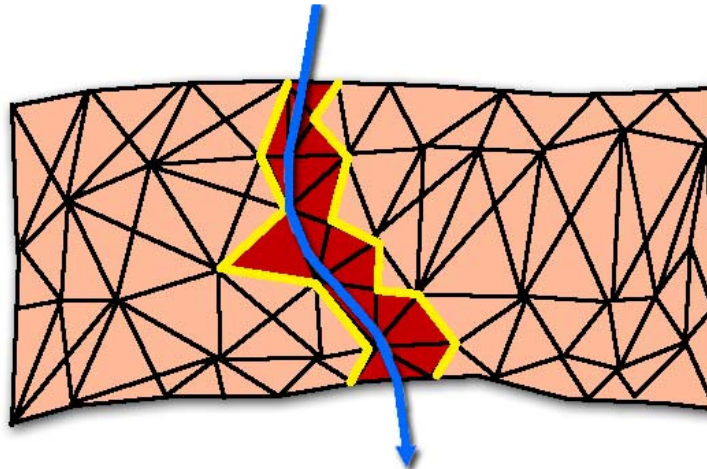


Figure 1.10 – Tetrahedron removing cut surface problem

A possible improvement of the TR approach could be to change the mesh resolution locally to the cut in order to minimize the effect artifact (figure 1.11).

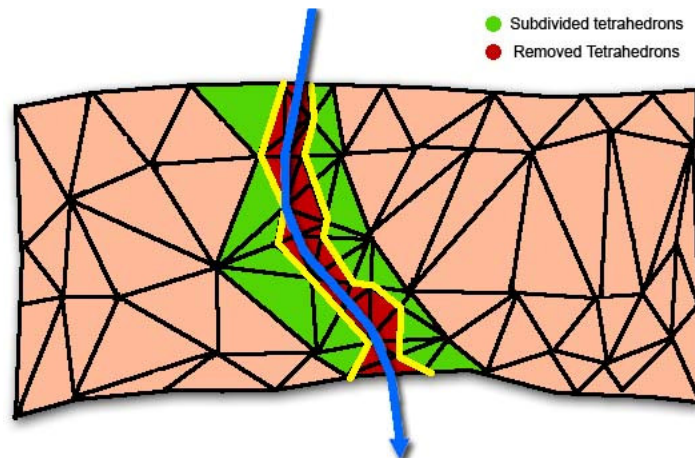


Figure 1.11 – TR improvement

After increasing the mesh resolution in proximity to the cut, the global mesh complexity grows. Assuming that the increase is acceptable when working on only a small part of the scene, the algorithm results are faster because each tetrahedron could be replaced by a specific template (figure 1.12), without calculate the right tool tissue intersection as in the TS.

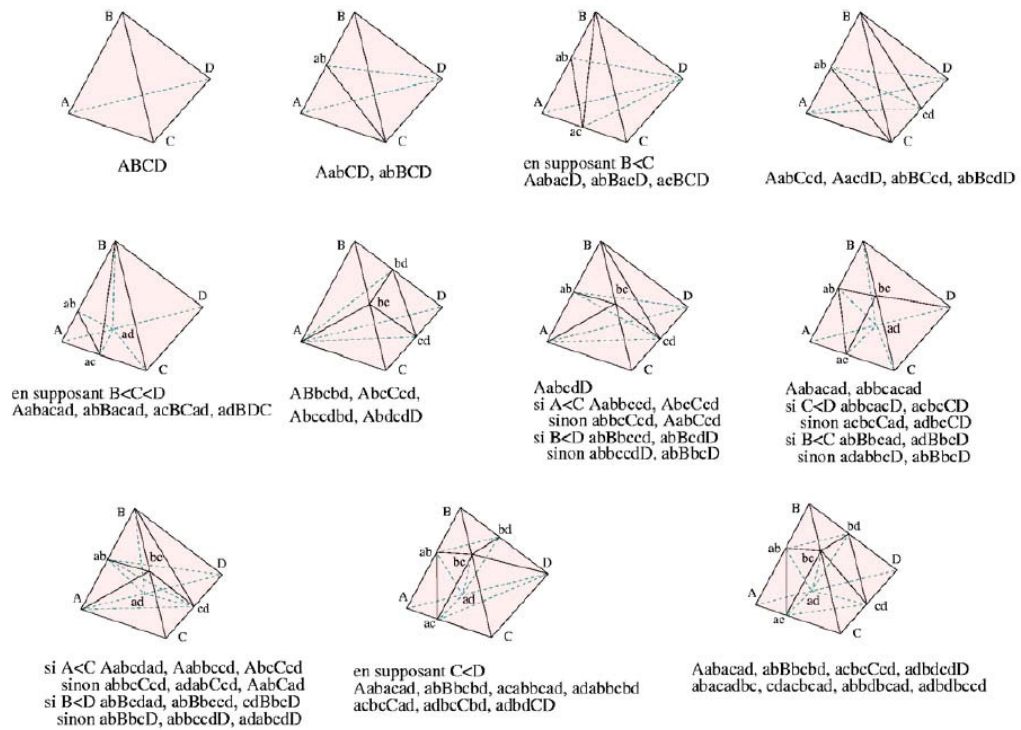


Figure 1.12 – Decomposition of tetrahedron by edge split

For the fracture simulation, the typical approach is to work on each link. The first test is performed in order to evaluate if the link deformation is between τ_d (deformation threshold) and τ_b (breaking threshold) in order to evaluate if the link is going to be deformed or broken. In case of deformation, the algorithm updates the link with a new length. This is usually a stretched length percentage. If the deformation is over τ_b , the link is deleted from the data structure along with all the tetrahedron containing that link. This adds new triangles on the surface (figure 1.13).

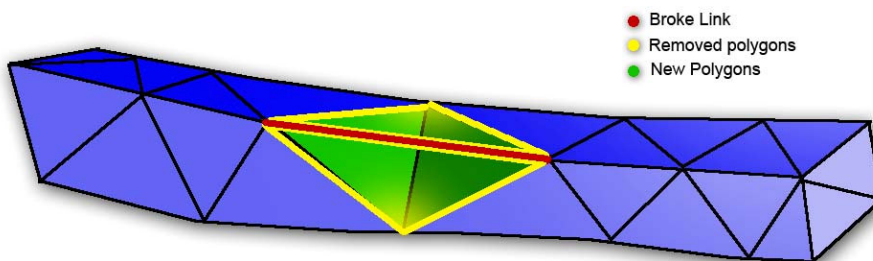


Figure 1.13 – Link breaking

For the fracture simulation, it is possible to have an incorrect behavior (**ghost connection**) on the links level when the only connection between two pieces of tissue is for example only a link, because drawing the polygonal surface there is not a connection between the two pieces (Figure 1.14).

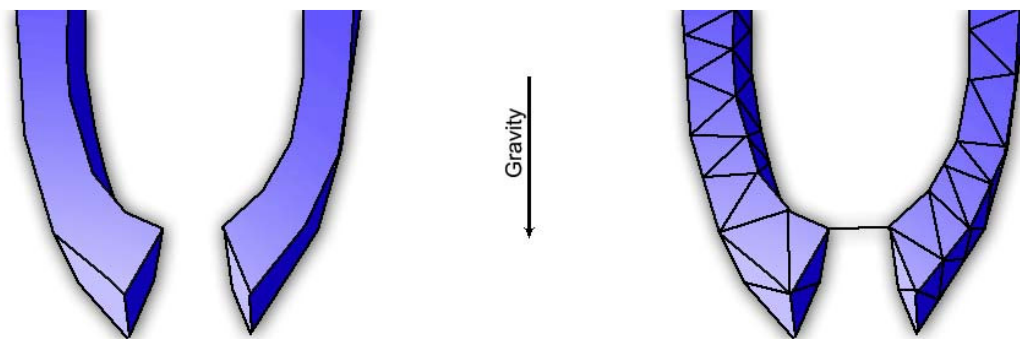


Figure 1.14 – Ghost connection

A solution to avoid a ghost connection is to check periodically the dynamic mesh in proximity to the broken links in order to remove eventual **ghost links**. Typically in the surgical simulation, the ghost connection issue is neglected.

Chapter 2

Metaballs and Implicit Surfaces

This chapter introduces the metaball concept, how to use it to approximate volumes, and how to extract the external surface in order to draw it. After the metaballs introduction, the chapter ends with an evaluation of the volume approximation task and a short analysis of all advantages that could be obtained using the presented model.

2.1 - Metaballs

After the discussion about the typical approach used for the soft tissue and organs modeling in the surgical simulation in the previous chapter, this chapter introduces an alternative approach, its advantages (in terms of computational cost, scene complexity and so on), and its limitations. The proposed approach works using the metaball [13] primitive in order to model the surgical scene. Before describing the general approach, it is indispensable to define the metaball and how it works.

A metaball can be considered as a field generator. Therefore, in order to define the metaball, we need the following information (assuming to work in a tridimensional space):

- The source position $\mathbf{p}_0 \in \mathbb{R}^3$.
- The field Intensity I .
- The field function $f: \mathbb{R}^3 \rightarrow \mathbb{R}$ or $f: \mathbb{R} \rightarrow \mathbb{R}$.

Then we can say that the field intensity in a generic point $\mathbf{p} \in \mathbb{R}^3$ respect to the source \mathbf{p}_0 could be expressed as:

$$f(\mathbf{p}) = f(x, y, z) = \frac{I}{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$$

or

$$f(r) = \frac{I}{r^2}. \quad (2.1)$$

Where r is the distance between the point \mathbf{p} and the source \mathbf{p}_0 . In general, it is possible to use a power greater than two, so the field function becomes:

$$f(\mathbf{p}) = f(x, y, z) = \frac{I}{\left(\sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}\right)^m}$$

or

$$f(r) = \frac{I}{r^m}. \quad (2.2)$$

where m represents the desired power. Considering the generic field function $f(x, y, z)$, and setting a proper threshold value τ , the metaball is obtained as an iso-energetic surface for this field. There are different kinds of field functions with different properties (figure 2.1 – table 2.1):

Quadratic	$f(r) = \frac{1}{r^2}$	
Bloby Molecules	$f(r) = ae^{-br^2}$	a, related to the height b, related to the standard deviation.
Meta-Balls	$f(r) = \begin{cases} a(1 - \frac{4r^6}{9b^6} + \frac{17r^4}{9b^4} - \frac{22r^2}{9b^2}) \\ 0 \end{cases}$	a, scaling factor b, is the maximum distance a control primitive contributes to the field.
Soft Object	$f(r) = \begin{cases} a(1 - \frac{3r^2}{b^2}) & 0 \leq r \leq b/3 \\ \frac{3a}{2}(1 - \frac{r}{b})^2 & b/3 \leq r \leq b \\ 0 & b \leq r \end{cases}$	a, scales the function b, each control primitive has no influence after a distance b.

Table 2.1 – Field function examples

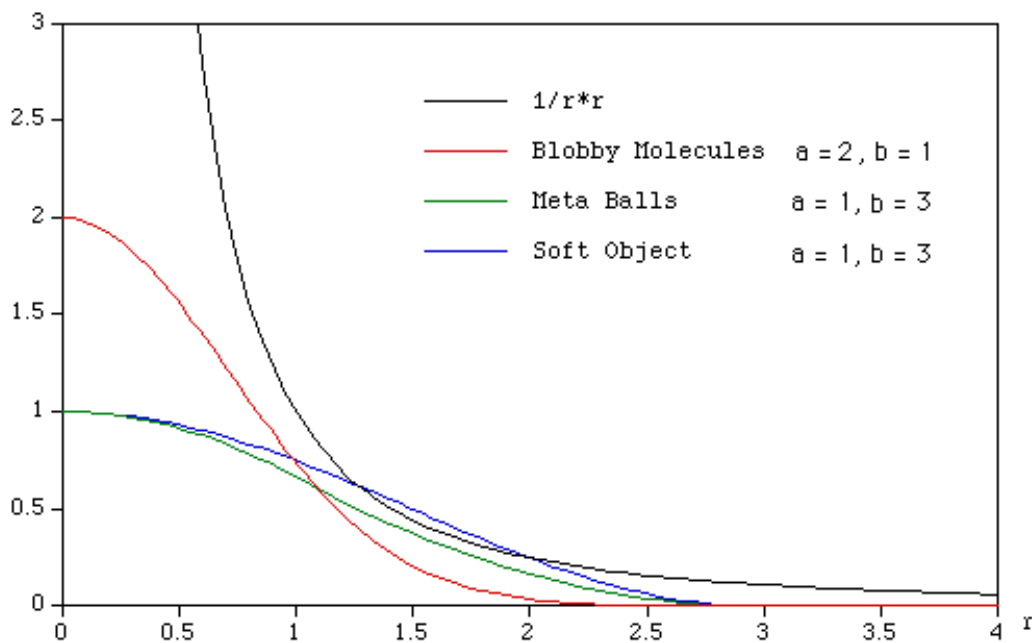


Figure 2.1 – Metaballs Field functions (see table 2.1)

The behavior of two metaballs at different distances is shown in figure 2.2.

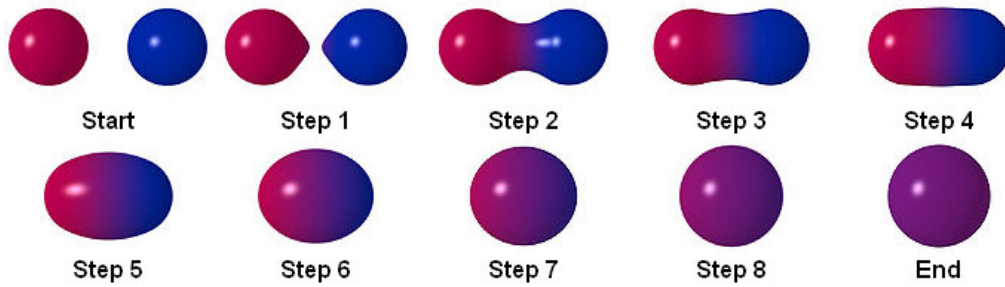


Figure 2.2 – Interaction of metaballs

Finally, the desired 3D model can be obtained with n metaballs as the volume defined by:

$$\mathcal{V} = \left\{ \mathbf{p}(x, y, z) \text{ such that } \sum_{i=0}^n f_i(x, y, z) \geq \tau \right\} \quad (2.3)$$

The iso-surface will be

$$\mathcal{V}_s = \left\{ \mathbf{p}(x, y, z) \text{ such that } \sum_{i=0}^n f_i(x, y, z) = \tau \right\}$$

At this point, it is possible to create different surfaces after collecting several metaballs (figure 2.3).

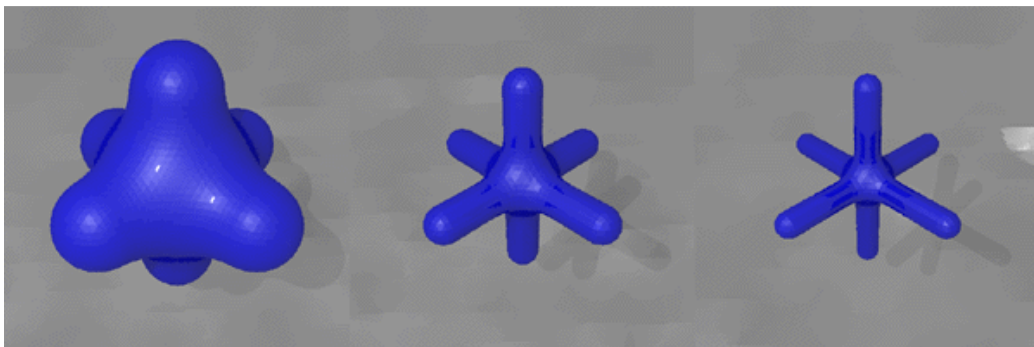


Figure 2.3 – Metaballs modeling

It is also possible to use field generators (metaballs) with negative field intensity, so the interaction between two metaballs with different signs look like in figure 2.4

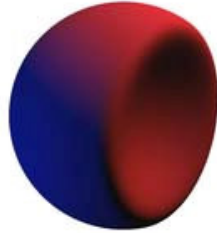


Figure 2.4 – Two metaballs with different sign

2.2 - Volume approximation using sphere trees

As discussed above, in our approach the 3D organ model must be created using metaballs. Then, the first step for the representation of an organ in a virtual environment is to obtain satisfactory 3D models optimized for this approach. The modeling task can be performed either manually, drawing the organ models and then tuning all the particle fields and setting proper threshold value, or by an automatic procedure for all these steps.

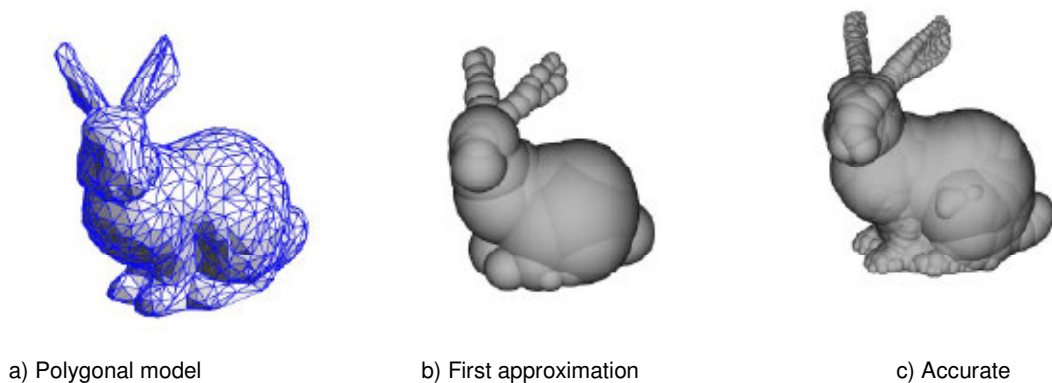


Figure 2.5 – Approximated 3D model [14]

The procedural approach requires an initial 3D model (polygonal or volumetric) generated by CAD programs or acquired by a 3D scanner (or a MRI/CTS device). The use of a sphere tree structure, as shown in Figure 2, can approximate the initial

geometry. Different techniques have been proposed for generating the sphere tree [15] [16] [8]; in our case, however, we need to use an algorithm able to generate a solid sphere model (and not a cave model) that minimizes the overall number of spheres. A recursive *octree* structure [8] can be used to generate a solid sphere tree, but in this case the obtained tree makes use only of spheres with a fixed diameter for each level of detail (LOD), (figure 2.6) and therefore, the number of the spheres used for the volume representation is not optimized.

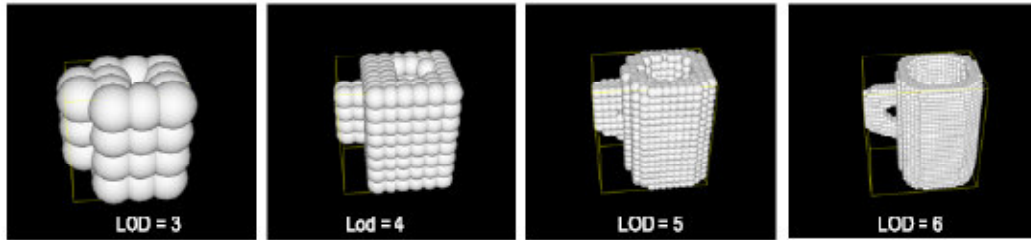


Figure 2.6 – Octree sphere tree for different levels of detail

The number of spheres can be minimized by means of the algorithm presented by [15], able to generate an optimized and solid sphere tree, in terms of number of spheres, the model fitting, and the spheres overlapping (figure 2.5). This tree can be used to generate the organ surface and also to detect collisions. At this point, it is necessary to convert the obtained sphere tree in a blobby model. This is achieved by converting each sphere, contained in the preferred level of the sphere tree, in a field generator (a metaball). Since the tree is made by spheres with different radius r_i , and since their surfaces are described by the same value τ , then a different field intensity I_i must be used in (2.2) for each metaball.

$$f_i(p) = \frac{I_i}{r_i^2} \rightarrow I_i = \tau r_i^2$$

In this way, each sphere of the octree is approximated with a spherical iso-surface with the same radius r_i . Since, according to figure 2.3, all the metaballs contribute to the global field, locally the distance of the iso-surface from its center c_i is normally

larger than r_i , and therefore an adjunctive tuning of the final value of τ may be necessary. For a proper value of τ , the resultant iso-surface appears as shown in figure 2.7.

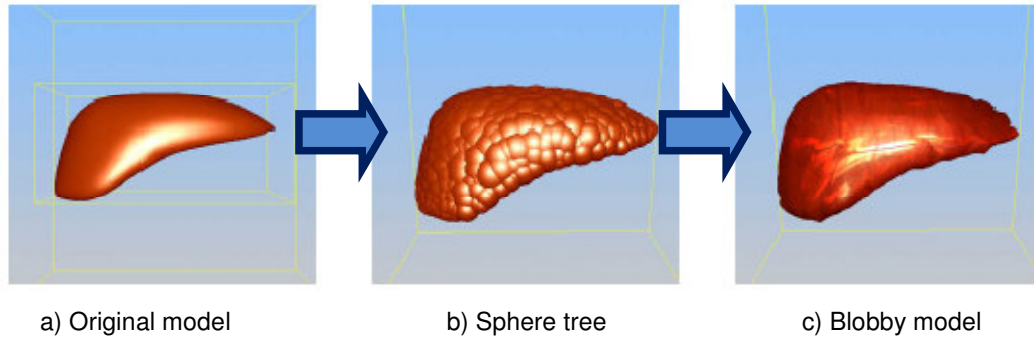


Figure 2.7 – Conversion steps from a sphere tree to a metaballs model

Once we obtained the conversion from sphere tree to metaballs model for the 3D model visualization, the resulting iso-surface, which approximated the original model surface, can be easily converted into a polygonal mesh by adopting the marching cube algorithm [7], or a derived algorithm optimized for real-time segmentation. It is important to observe that with few metaballs it is very easy to create very complex organ shapes. Experimental results have shown that the use of a power greater than two (2.2) allows better results in surface approximation. In our case, we used the (2.2) with $m = 6$. Obviously, the obtained blobby model is just an approximation of the real one and the quality depends on the LOD used in the sphere tree.

2.3- Surface Extraction

As shown in figure 2.7, there is a possibility to create an automatic algorithm able to convert an initial organ model into a blobby model. It is important to note that this process needs to be done just one time, and during the simulation the only part that needs to be done is the surface extraction (figure 2.8). The surface extraction task is usually performed by the marching cube algorithm (MC) [7]. The MC subdivides the model bounding box in a tridimensional grid and performs a test for each sub-cube inside the grid in order to recognize if it is empty or not. If the sub-cube is full, the

MC performs a test on each vertex in order to determine if there are vertices inside and outside the volume; if so, the MC places a polygonal pattern (excluding the combinations of completely full and empty cubes) (figure 2.9a).

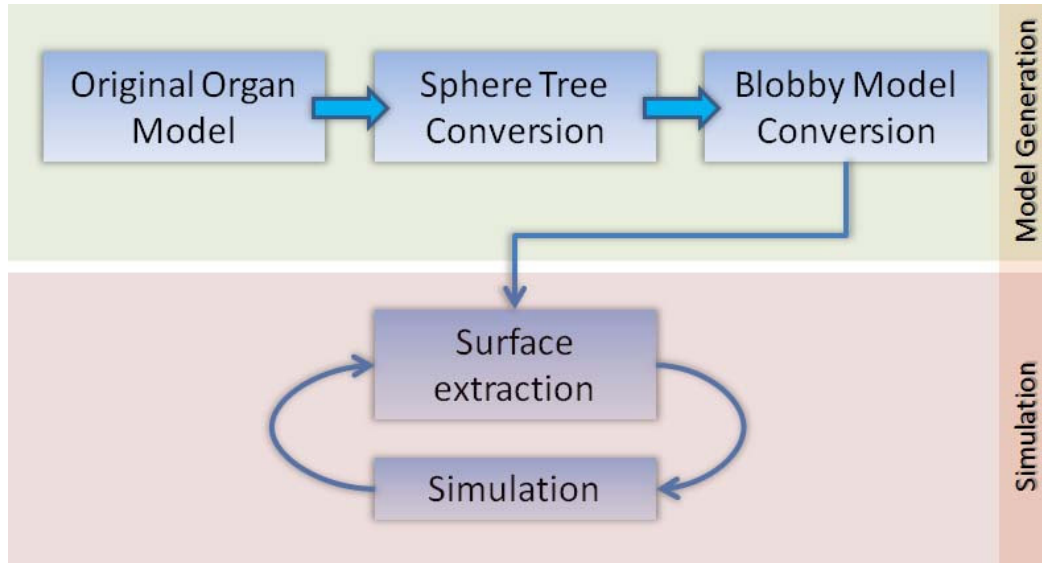


Figure 2.8 – Model synthesis/simulation workflow

Considering that there are eight vertices for the cube, there are 254 (256-2) combinations. In order to improve the performance of the algorithm, it is possible to subdivide all combinations in 15 groups (figure 2.9b) through counting the number of vertices inside the volume and adding a pattern orientation.

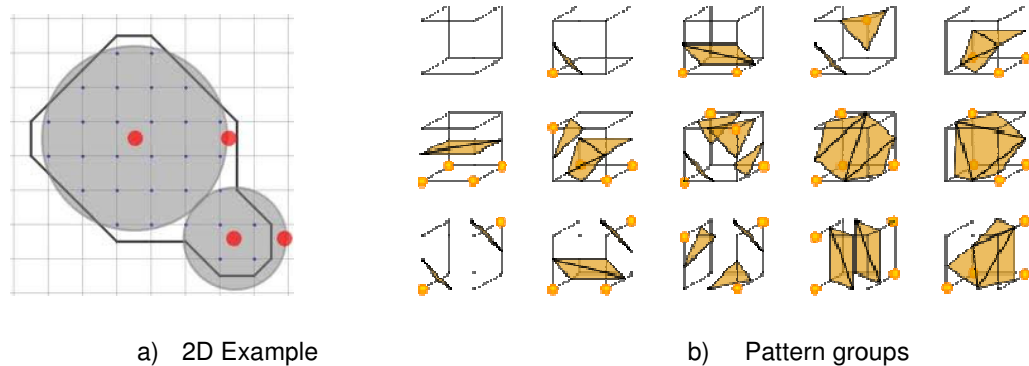
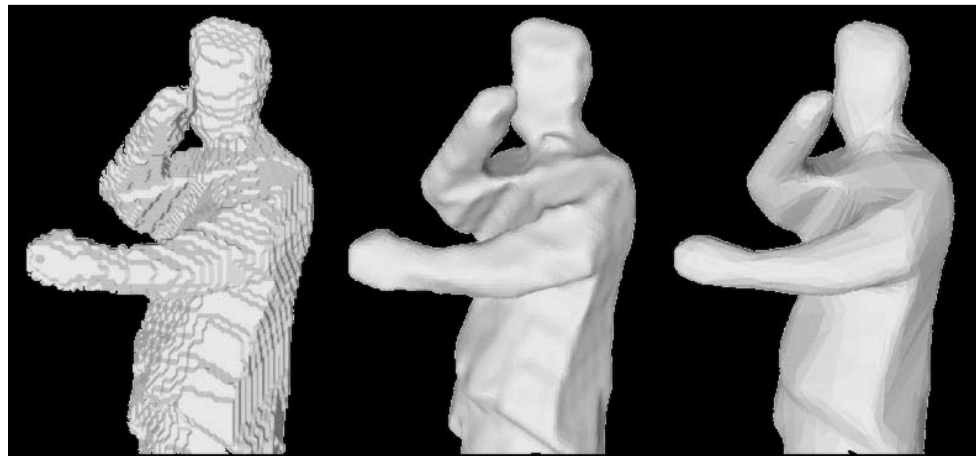


Figure 2.9 – Marching cube

Using a fixed pattern of polygons, the result of the polygonization process is not very accurate and the results are not as smooth as the expected result (figure 2.10a). It is necessary, then, to refine the polygonal pattern in order to fit the iso-surface in a better way using a surface-smoothing filter (figure 2.10b). Another problem for the obtained model could be the high resolution of the obtained polygonal mesh; it is possible to reduce it in order to work with a less complex geometry (figure 2.10c). There are many optimizations for the MC algorithm and it can extract very complex meshes starting from blobby models and volumetric data. Usually, it is used in medical imaging in order to extract surfaces from the DICOM files. DICOM files are the standard file format for the volumetric data acquired in CT-Scans, MRI-Scans and so on. Through extracting multiple surface layers (different gray levels processing the volumetric data), it is possible to obtain 3D models as shown in figure 2.11.



a) Marching cube surface b) Surface smoothing c) Polygon reduction surface

Figure 2.10 – Surface refinement

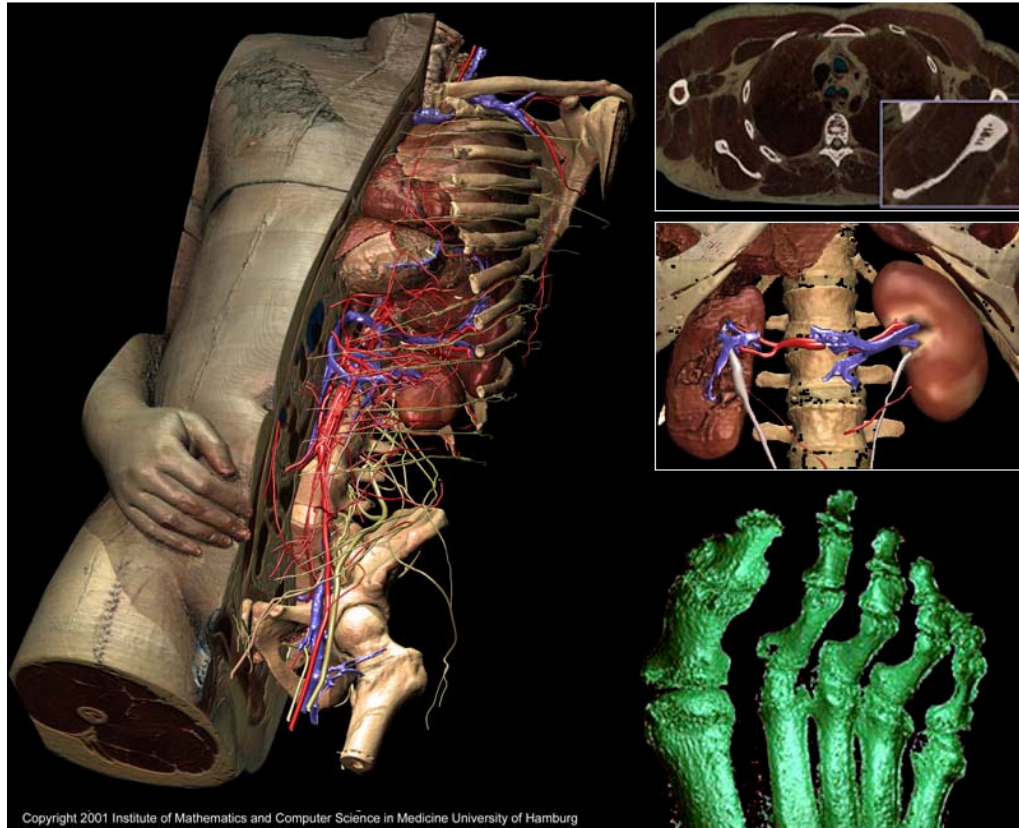


Figure 2.11 – Examples of marching cube application in medical imaging for tissue segmentation (Image by the Institute of Mathematics and Computer Science in Medicine University of Hamburg.)

2.4 - Performances Evaluation

As stated previously, the blobby model generation process needs to be performed just one time before the simulated operation. It is also important, however, to evaluate the performance for this task. A first evaluation may focus on the best approach used for the sphere tree generation. There are many sphere tree algorithms. This section will examine the performances of those algorithms when attempting to approximate a pattern of 3D models (Table 2.2).

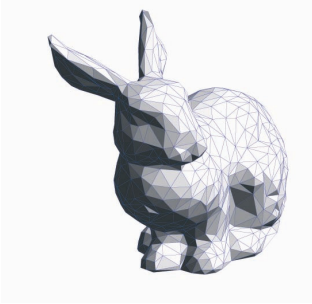
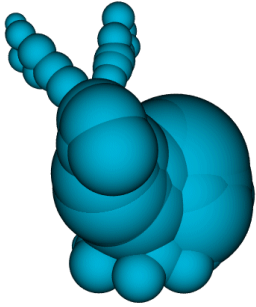
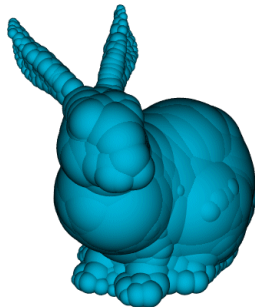
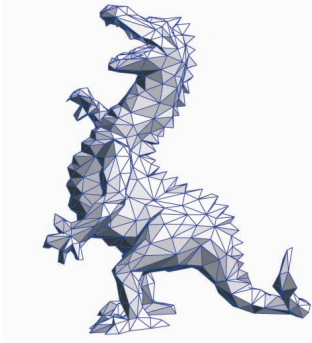
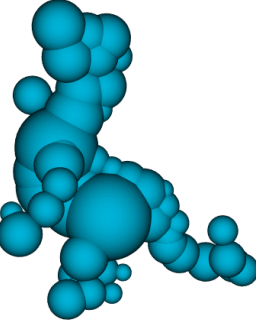
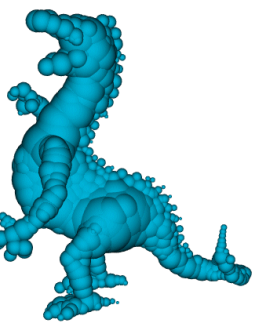
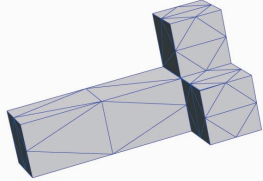
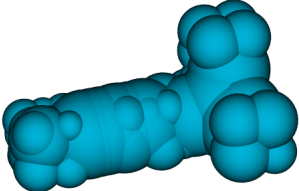
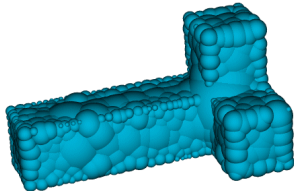
Original Model	Level 2	Level 3
		
		
		

Table 2.2 – 3D model pattern for test and the approximation using the [14] at level 2 and 3

As we use the sphere trees for real-time interactive simulations, we are most interested in the worst-case error for each level of the sphere tree. This occurs in an upper bound on the gap that will exist between two objects that are thought to be in contact. A number of additional algorithms are also included. The *Hybrid* algorithm is a post-processed version of the *Grid* algorithm; each sphere in the sphere tree has been replaced with one that covers the same region of the object but has minimum error rather than minimum volume. The *Optimised* algorithm is the *Combined* algorithm (both based on [15]) with a simplex based optimization that further

improves the fit of the sphere prior to its inclusion in the sphere tree. The "Opt 0%" and "Opt 5%" are also optimized versions of the "Combined" algorithm except that the algorithm is allowed to throw away spheres as long as the worst error is less than 100% or 105% of its original value.

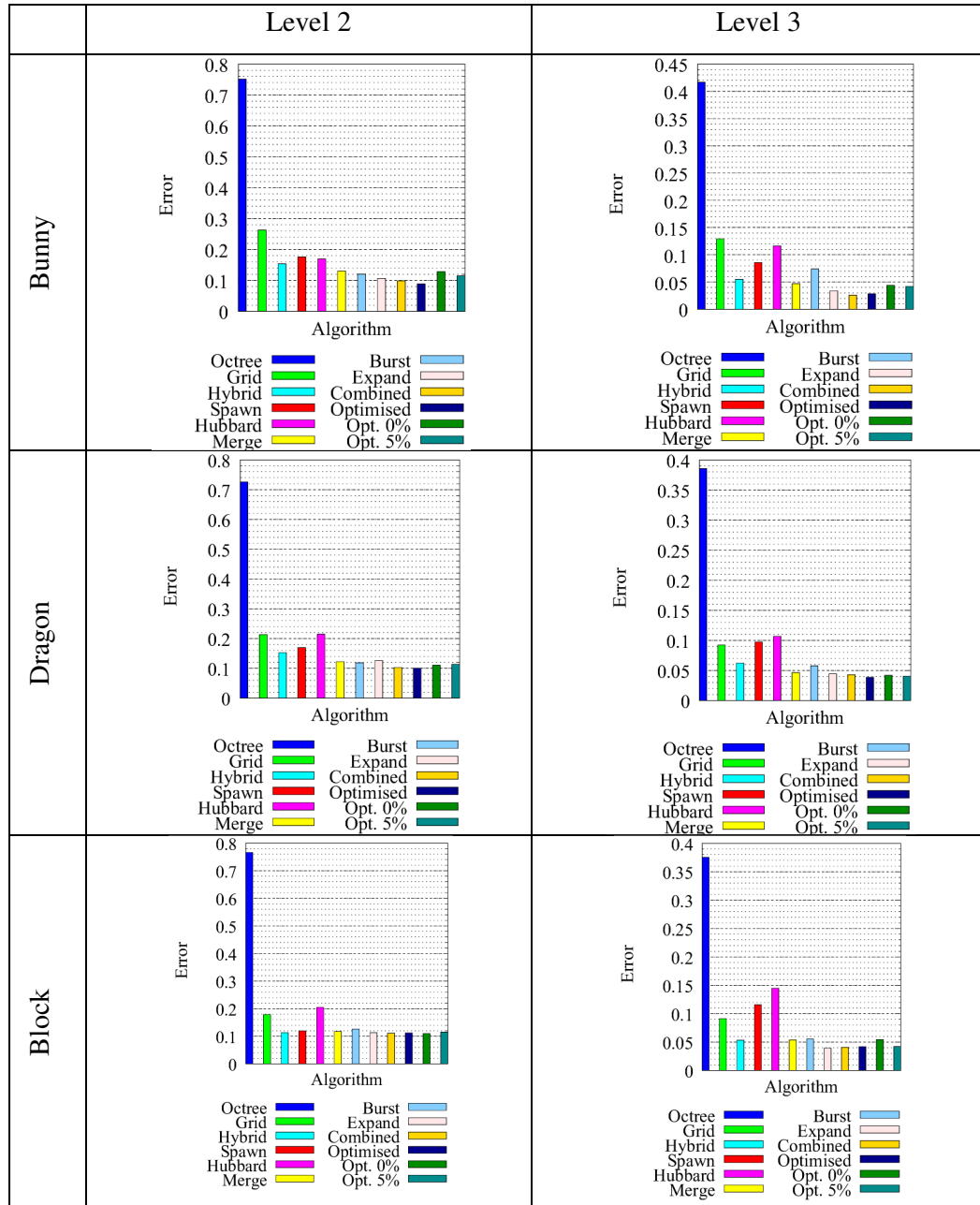


Table 2.3 – Sphere tree approximation algorithm benchmark¹

¹ The Benchmark is by G.Bradshaw on the website <http://isg.cs.tcd.ie/spheretree/>

After investigating the table, it is easy to understand that the best result is obtained with the Optimised algorithm that is based on the medial axis approximation algorithm [15]. Another test has been performed between the octree and the Medial Axis Approximation (MAA) in order to show the the sphere number reduction and also the quality of the obtained mesh after the polygonization process [17].

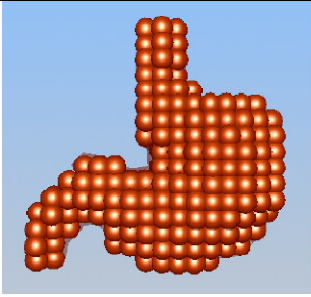
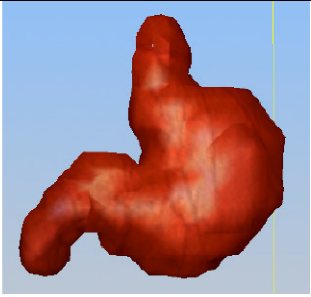
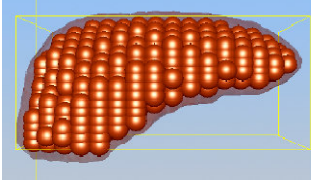
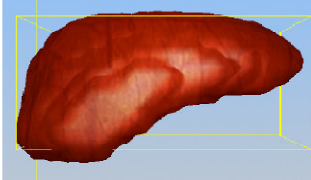
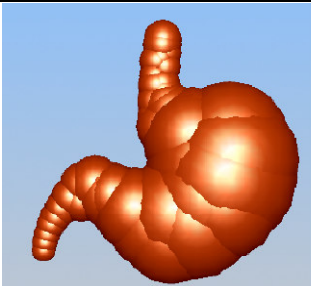
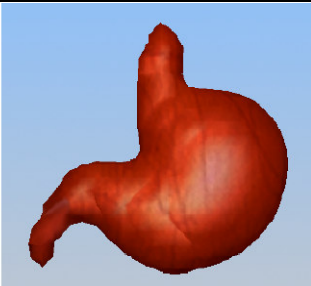
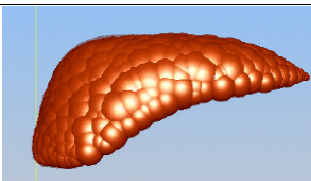
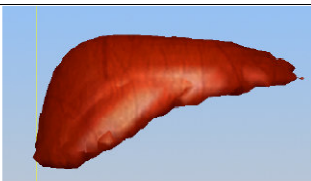
	Sphere Tree	Blobby Model	Settings
Stomach			Sphere =412 LOD =4 <i>m</i> = 6 algorithm=octree
Liver			Sphere =707 LOD =4 <i>m</i> = 6 algorithm=octree
Stomach			Sphere =48 LOD =3 <i>m</i> = 6 algorithm=MAA
Liver			Sphere =366 LOD =3 <i>m</i> = 6 algorithm=MAA

Table 2.4 – Octree vs MAA comparison in terms of number of sphere and shape quality.

Looking at table 2.4, it is also easy to recognize that the MAA offers best results in terms of number of spheres and in terms of surface approximation. It creates a smooth surface that fits the original surface in the best way.

Chapter 3

Fluids and Rigid Bodies Simulation

After the description of the blobby model generation, it is also interesting to understand how to use metaballs in order to simulate fluids and rigid bodies. In surgical simulations, the use of rigid body simulation is not as important as the soft tissue. It needs to be considered, however, because in the body there are also bones. The fluid simulation is important as well in order to simulate blood and other kinds of physiological fluids.

3.1 – Particles Based Fluid Model

A reason to use metaballs for surgery simulations is born of the fact that metaballs are commonly used for the fluid simulation (figure 3.1). Moreover, during the operations, blood and other kinds of physiological fluids are often present, so it is very important to have a fluid model. If the fluid model is similar to the tissue model, it is an advantage in terms of complexity reduction for the simulation. As told in the chapter two, the metaballs are points (or particles) in the space with a scalar field associated; therefore, the fluids can be simulated using the metaballs and the implicit

surfaces. Before using metaballs, a new value for each particle, the ray r , must be added.



Figure 3.1 – Fluid simulation using metaballs.

It is important to note that the ray r is not equivalent to the threshold value τ . At this point, it is possible to define a data structure for each particle:

$$Particle = \begin{cases} Acceleration \\ Speed \\ Position \end{cases}$$

All data present in the structure are required for the physic simulation, but the ray is important for the collision detection between the fluid particles. Usually, the ray is the same for all the particles. For this reason, it is defined a single general structure for the fluid and another for each particle, storing all particles inside an array.

$$fluid = \begin{cases} number\ of\ particles \\ particles\ ray \\ Particles\ mass \\ \tau\ field\ threshold \\ k\ compressibility \\ \eta\ viscosity \end{cases}$$

After the physical simulation for each particle, it is possible to use the marching cube algorithm in order to draw the fluid mesh. Using a real-time reflection and refraction, a variety of fluids can be simulated in a very realistic way (figure 3.2).



Figure 3.2 – Realistic fluid simulation (www.maiani.eu)

3.2 - Particles Collision Detection Model

The theoretical fluid model is an incompressible fluid and the real fluids are quasi-incompressible; in order to obtain a quasi-incompressible behavior from our model, a collision model between all fluid particles must be defined.

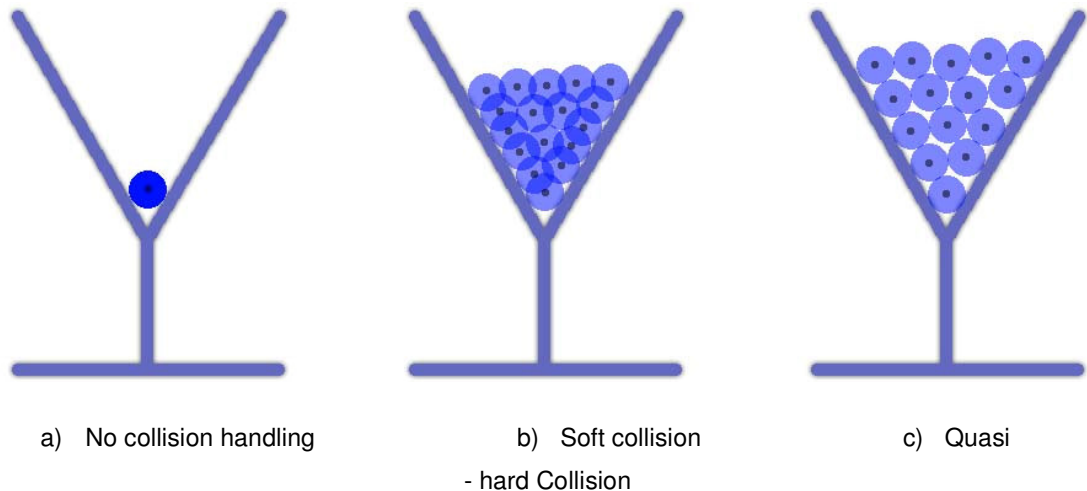


Figure 3.3 – Fluid approximation using particles under the gravity effect

If a collision-handling algorithm is not used, a realistic behavior will not be obtained because the particles could be overlapped (Figure 3.1a). It is possible to define the collision model as an elastic collision, in order to allow a proper particle overlapping. Additionally, it can be used in order to simulate a realistic behavior (figure 3.1b) and it can create a stiff elastic constant a quasi-incompressible fluid (figure 3.1.c). The

contact force module between two particles i , in collision with the particle j in the collision model should as follows.

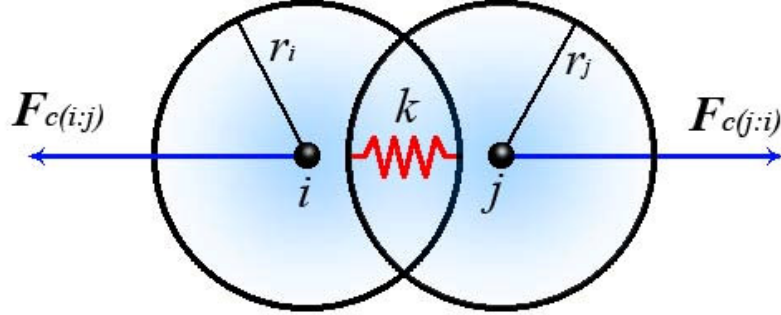


Figure 3.4 – Fluid particles contact model

$$F_c = \begin{cases} (r_i + r_j) < \|\mathbf{p}_i - \mathbf{p}_j\| \rightarrow 0 \\ (r_i + r_j) \geq \|\mathbf{p}_i - \mathbf{p}_j\| \rightarrow k[(r_i + r_j) - \|\mathbf{p}_i - \mathbf{p}_j\|] \end{cases}$$

That force will be distributed on the two particles using their masses:

$$\mathbf{F}_{c(j:i)} = (\mathbf{p}_i - \mathbf{p}_j) \frac{m_i}{m_j + m_i} \cdot F_c; \quad \mathbf{F}_{c(i:j)} = (\mathbf{p}_j - \mathbf{p}_i) \frac{m_j}{m_j + m_i} \cdot F_c;$$

considering the same mass value

$$\mathbf{F}_{c(j:i)} = \frac{(\mathbf{p}_i - \mathbf{p}_j) \cdot F_c}{2}; \quad \mathbf{F}_{c(i:j)} = \frac{(\mathbf{p}_j - \mathbf{p}_i) \cdot F_c}{2};$$

then

$$\mathbf{F}_{c(j:i)} = -\mathbf{F}_{c(i:j)}$$

It is easy to observe how the simulation pipeline is similar to the soft tissue pipeline, but here the particle collision is handled inside the constraint application block (figure 3.4).

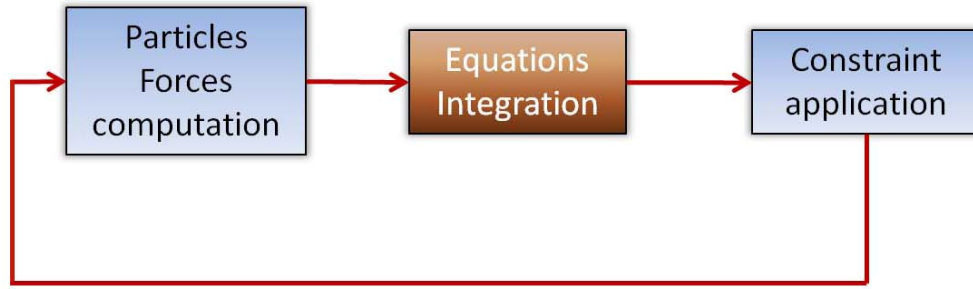


Figure 3.5 – Fluid simulation block

Regarding the fluid, it is possible to simulate also other properties of the fluid (density, adhesion, cohesion and so on). It is evident that collision detection represents a possible problem in terms of computational performances, because a collision test should be performed between each particle. A possible solution for this problem is represented by the use of an octree space partitioning, in order to perform the test just between few particles per time. There is also a cohesion force \mathbf{F}_{ch} between the fluid particles, so if the distance is shorter than a specific distance threshold τ_d , there is an adjunctive elastic forces that keep the fluid particles close.

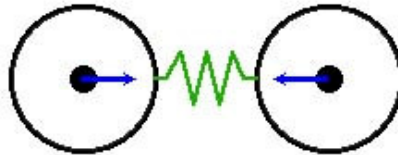


Figure 3.6 – Cohesion force between particles

$$\mathbf{F}_{ch(i;j)} = (\mathbf{p}_i - \mathbf{p}_j) \cdot \begin{cases} d \leq (2r + \tau_d) & -\delta k_{ch} \\ d > (2r + \tau_d) & 0 \end{cases}$$

where

$$\delta = \frac{\tau_d - (d - 2r)}{(\tau_d - 2r)}$$

and

$$d = \|\mathbf{p}_i - \mathbf{p}_j\|.$$

Through the interaction model, complex fluid properties are easily simulated. The same model used for the fluid simulation is also used for the smoke simulation (In MIS, a particular kind of surgical tool is utilized to burn the tissues; often this releases smoke or steam.).

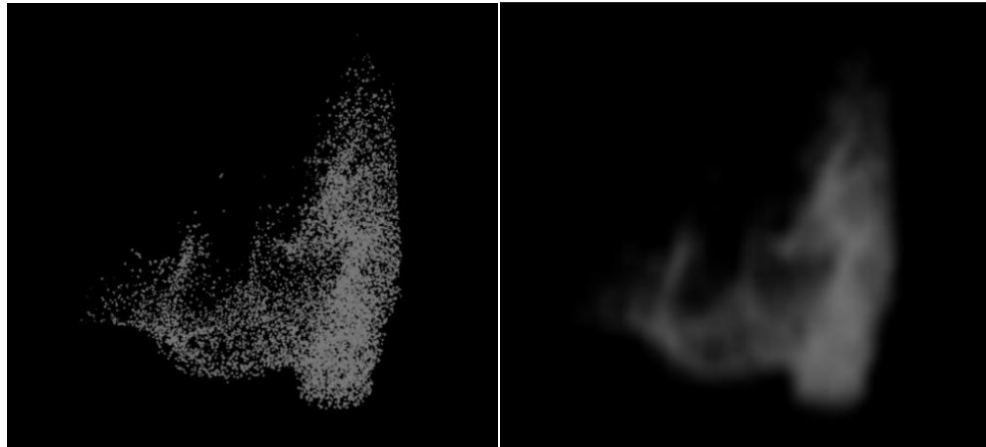


Figure 3.7 – Smoke simulation using particles and a quadratic blur effect.

3.3 – Rigid body simulation

Using metaballs, it is possible to simulate rigid body, and the approach is the standard used for polygonal meshes, but calculating collision between spheres instead of polygons. For the rigid body simulation, the definition of a data structure is required in order to keep all dynamic and structural data (source 3.1)

```
typedef struct
{
    //constants
    float Mass;           // total mass

    Matrix_3x3 Ibs;      // Inertial matrix
    Matrix_3x3 InvIbs;  // Inverse Inertial matrix
}
```

```

//Status variables
Vector    Position;    // the body position in the 3D space
Vector    CM;         // center of mass
Matrix_3x3 Rotation;  // rotation matrix
Vector    P;         // Linear momentum
Vector    L;         // angular momentum

// auxiliar variables
Vector    Vl;        // linear speed
Vector    Va;        // angular speed

//Q.tà calculate
Vector    Force;     // Force
Vector    Torque;    // Momentum
}Body;

```

Source 3.1 – Rigid body data structure

After the data definition, there is the simulation flow that is structured as a loop (in a thread) where several operations are performed (figure 3.5). The collision detection task is exactly the same used for the fluid simulation and, when a collision occurs on a generic point \mathbf{p} on the rigid body, the resultant force is applied on the entire body as shown in the source 3.2.

```

void AddForce(Body * body, Point3D v, Vector F)
{
    float d;
    Vector b, u;

    b = (v - body->CM);
    body->Torque+=F^b;

    u = b;
    d = b.Module();
    u.Scale((b*F)/(d*d));
    body->Force+=u;
}

```

Source 3.2 – Rigid body collision handling

After the definition of collision handling, the last step is equation integration. Rigid bodies are usually not used in the surgical simulation field; the focus is on an accurate way to simulate soft tissues. The rigid bodies are usually treated using classic approaches, are less interesting from a simulation point of view, and are easy to simulate with good results; for those reasons, rigid bodies are not a goal for this current thesis.

Chapter 4

Soft Tissue Model

After an introduction regarding the use of the metaballs for the fluid simulation and a quick introduction to the rigid body simulation, this chapter will explain how to extend the metaballs model in order to simulate soft tissues.

4.1 – 3D mesh generation

As shown in the chapter one, in order to simulate a dense soft tissue organ, it is indispensable to create a tridimensional mesh of particles and visco-elastic links. Using the metaballs approach, the creation of the tridimensional mesh can be difficult when the resolution of the sphere tree is low. The first topic that needs to be discussed is the strategy used for an automatic network creation starting from the sphere tree. As we know, the behavior of a soft tissue model can be strongly influenced by the connection topology used for the tridimensional mesh. In some cases, a wrong connection topology could create an odd behavior that does not match with the desired (real) one.

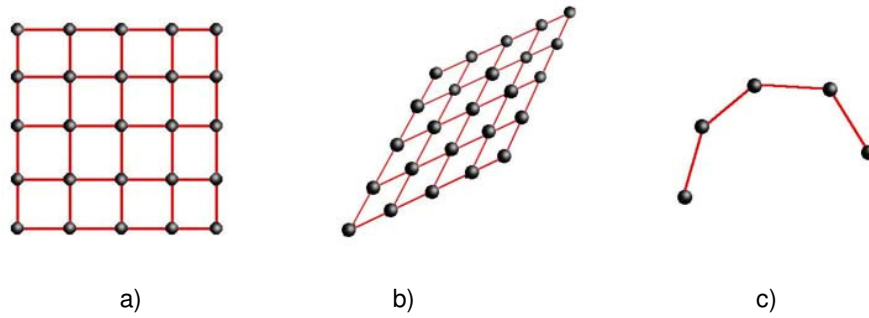


Figure 4.1-Cloth model

If in an elastic cloth simulation, we use a mesh model as shown in figure 4.1a, we will obtain two wrong behavior (non-elastic deformations) that do not match with the behavior of a real elastic cloth. In figure 4.1.b, the elastic cloth collapses in a single segment without deforming any link (the deformation is stable); in the figure 4.1.c, the elastic cloth can be deformed also without an elastic deformation. Adopting a connection scheme, as shown in figure 4.2a using cross-links (green and violet links) and an optimized version in 4.2b, is a possible solution to the non-elastic deformations for the elastic cloth simulation.

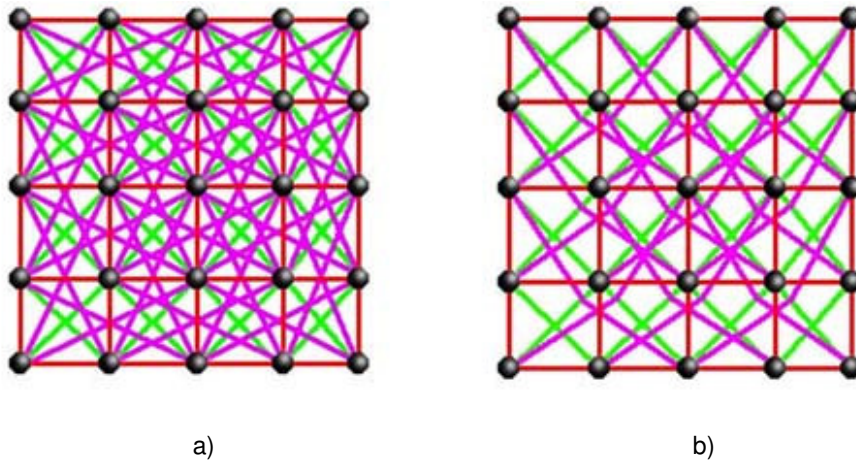


Figure 4.2 – Full Elastic deformable cloth model

For tridimensional meshes, the topology connection requires a complex building process. In our case, after the conversion from volumetric model (or polygonal model) to a sphere tree, we obtain only a set of spheres. A first approach to create a tridimensional mesh on the sphere tree could be to connect, with visco-elastic links, all the overlapped spheres (neighbor connection strategy - figure 4.3a).

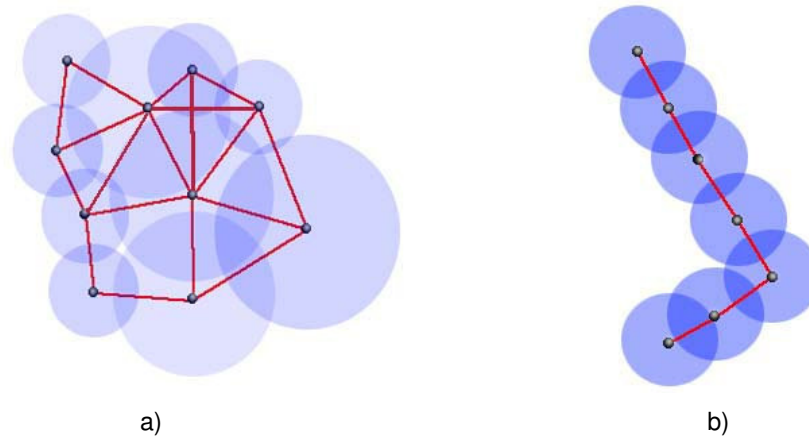


Figure 4.3 –Neighbor connection strategy

The neighbor connection strategy works well and is easy to implement, but it is not able to avoid non-elastic deformation (figure 4.3b). Usually, the non-elastic deformation is obtained for all spheres connected with less than three links (not aligned). It is possible to setup a proper algorithm able to recognize if a specific sphere is connected in the right way with the neighboring spheres, and to connect it with the closest sphere in case there are less than three connection links. In general, this approach needs to be integrated with a manual procedure, because it still presents some problems, such as the ghost links as shown in figure 4.4a. During the creation of the mesh and the creation of the correct minimal number of connections for each particle, through choosing the closest particles, it is possible to create ghost links, connecting a specific particle with another one that is outside the local body surface.

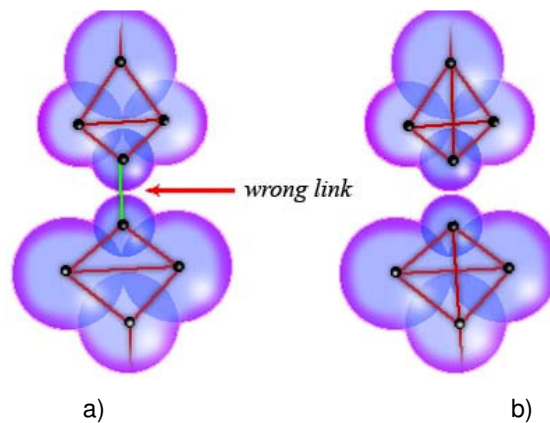


Figure 4.4 – Ghost link problem

As discussed in chapter one, the avoidance of the ghost link is an expensive task for the fracture simulation; however, in this case, we need to perform the mesh generation task only one time, before to start the simulation. As a result, it is possible to implement a proper algorithm able to recognize and fix ghost links through creating a new proper connection scheme. During mesh generation, there are also other situations where the neighbor approach does not produce the best connection topology. For a right connection scheme, all the links (the first three) for each particle should be connected with other three non-coplanar particles, creating a tetrahedron (figure 4.5a). If the three closest particles are coplanar with the first one, the connection topology does not work well because the tridimensional deformations cannot have the same elastic behavior on a direction normal to the particles plane. All forces applied in that direction will not be correctly balanced by the mesh (figure 4.5b).

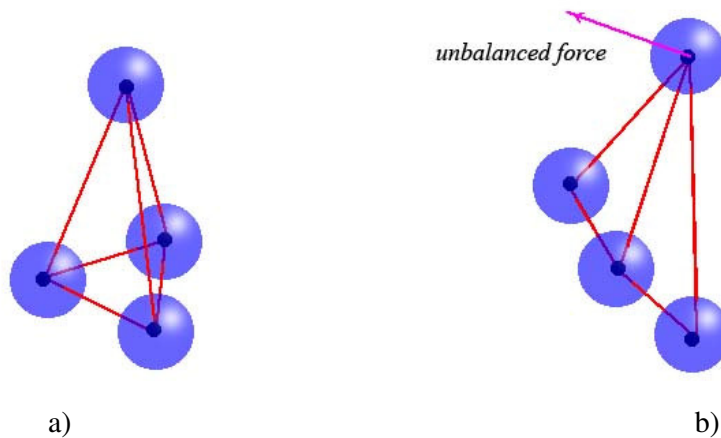


Figure 4.5 – Local topological connections

Nevertheless, the opportunity to create tetrahedral connections is not always possible. In some specific cases, a coplanar connection is the only possible scheme. If we consider, for example, a minimal blobby model (using a minimal sphere set) for the intestines, the only possibility is to use a segment of spheres. In this last case all spheres present two connections, except the first and the last sphere (figure 4.3b); the chain connection scheme does not work because it allows non-elastic deformation. In this case, the neighbor strategy produced a satisfactory result, introducing cross-links (figure 4.5).

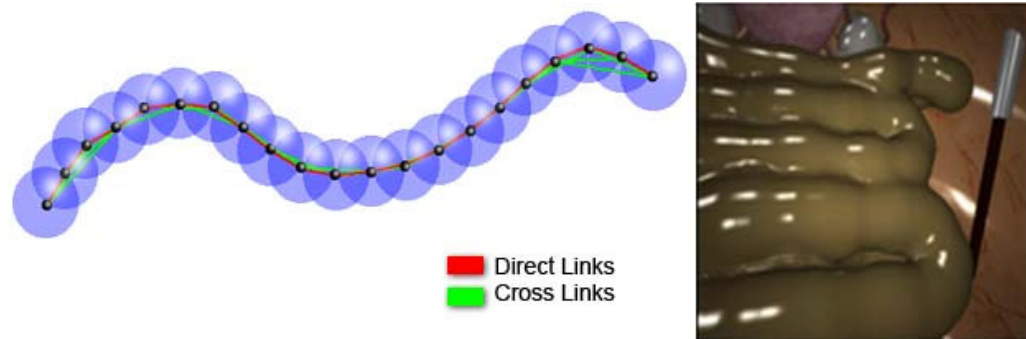


Figure 4.6 – Intestine simulation model using cross-links [18]

Through the use of cross-links in a chain, it is possible to avoid the non-elastic deformation. This occurs because when trying to bend a short section of the chain, the cross-links create a resistance force to balance the bending force.

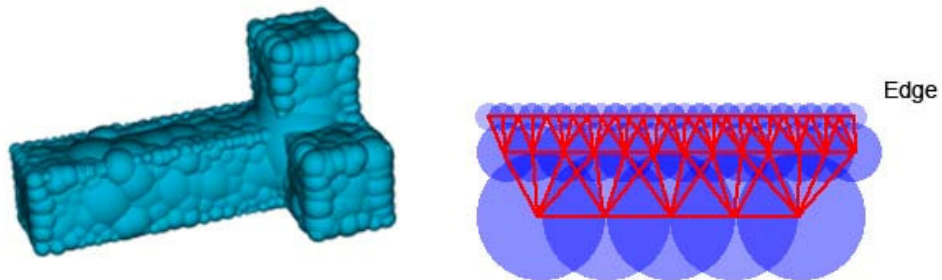


Figure 4.7 – Mesh resolution growth on the edges

An interesting property of the sphere tree approximation using the [14] is that the resolution of the sphere tree grows close the edges, increasing the number of spheres and reducing the ray in order to fit the original model in the best way. As told in chapter two, the blobby approach works well for the surgical simulations, because, by having rounded organ models, the number of spheres tends to be low. The high sphere tree resolution on the edges allows the simulation to indirectly describe all parts of the models where it is easier to have a deformation (more deformable areas), providing a better accuracy (figure 4.7). In some cases in the educational surgery simulations, the main goal is not the accuracy of the model but the simulation of a realistic behavior. In those cases, the neighbor connection strategy extended with the

non-elastic deformation avoidance is not the best approach, because it is not able to assure a correct elastic behavior. Consider a semi-rigid chain of metaballs: using the neighbor strategy it is not possible to achieve realistic elastic behavior, because the chain is not able to react to the orthogonal forces and the result is the bending of the chain. A possible solution is represented by the possibility to use a Bounding Connection Set that is an external set of links connected between each particle and the edge of the model-bounding box (figure 4.8).

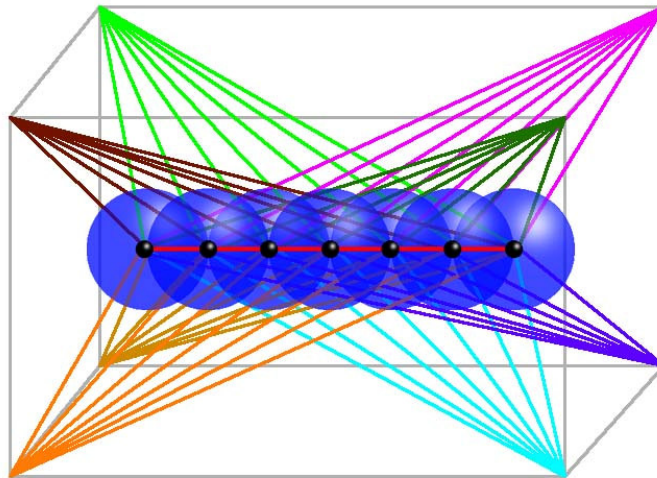


Figure 4.8 – Bounding Connection Set

Using the Bounding Connection Set, it is possible to keep a reasonable stiffness for shapes thin and long. During the simulation flow, the bounding box vertices will not be considered as particles; this means that all the link force will load on the connected particle. Also, the bounding box vertices will not be considered for collision detection. Instead, they will be considered as “ghost particles”. The bounding box does not have mass. There are different types of connection strategies and, as mentioned at the beginning of this section, the results in terms of computational performances and in terms of behavior depends from the right choice of the kind of model and when considering the required dynamics. By setting all tissue parameters inside the mesh and deciding on the best connection strategy, the final behavior during the simulation should be able to satisfy the initial requirements creating bubble deformation objects (figure 4.9).

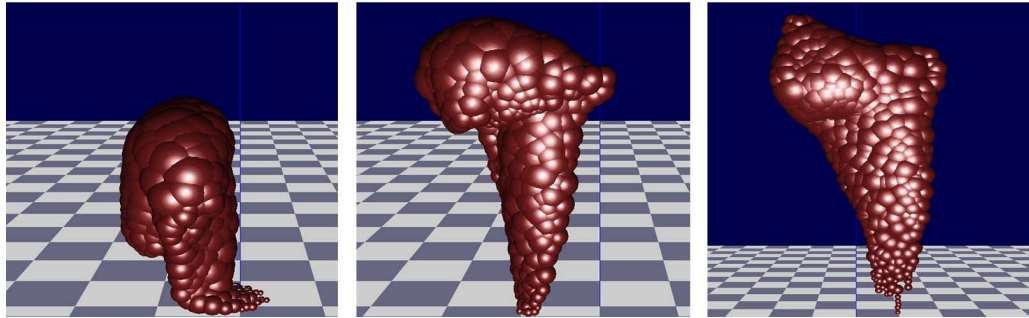


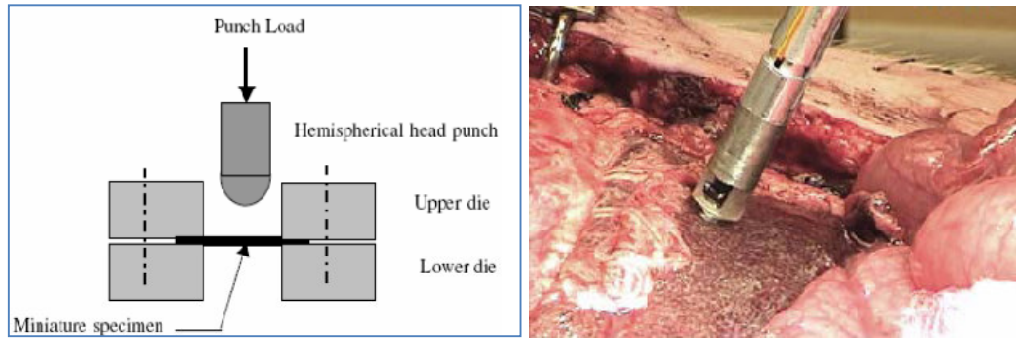
Figure 4.9 – Sphere tree deformation using neighbor connection strategy

4.2 – 3D mesh parameters assignment

After the tridimensional mesh generation, in order to simulate the tissue in the accurate way, it is indispensable to assign to the dynamic structure the right parameters (m , k , h , etc). The first step before starting the soft tissue properties assignment is to estimate the visco-elastic properties of different kinds of tissues through the use of apposite devices. There are different possible methods to characterize a specific tissue:

- In vitro rheology.
- In vivo rheology.
- Elastometry.
- Solving inverse problems.

The in vitro rheology is performed on an in vitro tissue sample using a punch load that deforms the tissue and using a load cell to recognize the tissue response (figure 4.10a). The technology to perform this kind of test is mature and it can be performed in a laboratory, but the results are not very realistic for soft tissues (perfusion of the tissues).



a) In vitro rheology.

b) In vivo rheology (CIMIT).

Figure 4.10 – Tissue characterization by rheology

The in vivo rheology is performed on living tissues can provide stress/strain relationship at several locations, but the results are influenced by the boundary conditions and are not well understood (figure 4.10b). The elastometry (MR, Ultrasound) measure property inside any organ in not invasive, but is valid today only for linear elastic materials. An elastometry require a complicate registration task between the ultrasound and the MRI datasets before the dataset merging. The inverse problem approach is well suited for surgery simulation (computational approach), but requires the geometry before and after the deformation in order to estimate all mechanical properties of the tissue (figure 4.11).

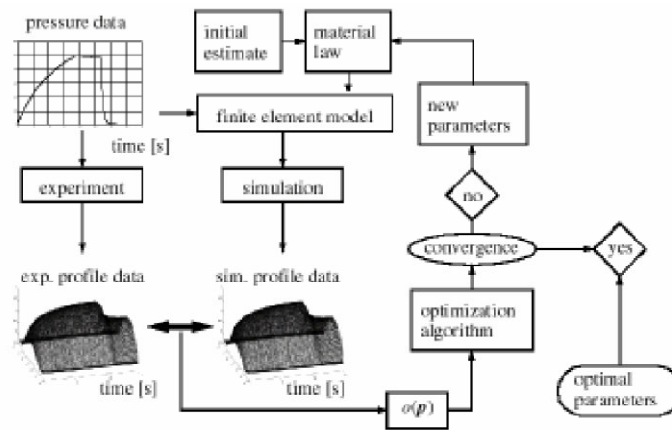
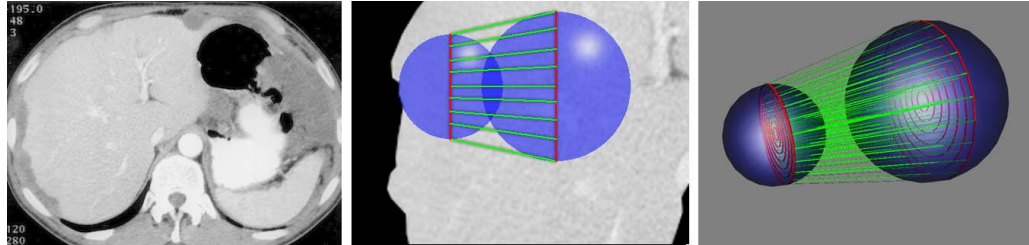


Figure 4.11 - Inverse problem approach for the soft tissue parameters estimation (INRIA)

After the tissue characterization process, a methodology to place the real data into the simulated model in the correct way needs to be identified. A good example for this procedure is the creation of the simulated model, starting from a volumetric dataset, acquired with a CTScan and/or a MRI device (figure 4.12a).



a) CTScan Slice b) Liver-metaballs overlapping c) visco-elastic paths evaluation

Figure 4.12 – Material properties extraction from a volumetric dataset

Assuming that we have a tridimensional dataset with all mechanical properties encoded in each voxel, we can start overlapping the sphere model to the dataset, grabbing all the mechanical/structural properties. For the masses for example, we can estimate the mass of the tissue contained in a single sphere:

$$M_{si} = \frac{4}{3} \pi r^3 dm$$

Where dm represents the estimated mass of a single voxel. In this case, a possible error is represented by the sphere intersection, because the shared volume is considered multiple times; in order to reduce this last error, the sphere mass M_{ni} could be estimated as

$$M_{ni} = V_{ni} M_{tot}$$

where V_{ni} is the volume for the sphere i normalized by the entire model volume V_{tot}

$$V_{ni} = \frac{V_i}{V_{tot}};$$

$$V_{tot} = \sum_1^n \frac{4}{3} \pi r_i^3.$$

M_{tot} is the entire model mass (obtained summing the weight of each voxel). As mentioned earlier, assuming that all mechanical properties are encoded in each voxel, it is possible to estimate the mechanical properties for the visco-elastic link between two spheres. For example, consider estimating average properties of a set of links connected between two parallel planes orthogonal to the spheres connection link (figure 4.12b, 4.12c). Taking into account only a single link inside the set, it is possible to assign the link elastic property as a sum of each voxel contribute.

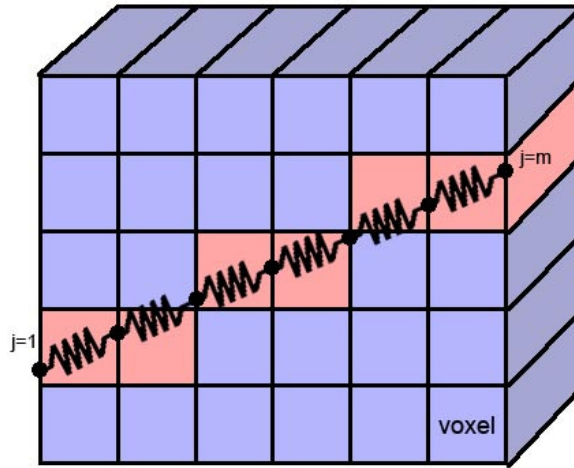


Figure 4.13 – Link properties estimation using voxels

Assuming a linear elastic behavior (constant elastic coefficients) and a set of m voxels using a j index, the entire link elasticity K_{li} can be expressed by

$$K_{li} = \frac{\prod_{j=1}^m k_j}{\sum_{j=1}^m k_j}.$$

The breaking threshold τ_{lb} (and for the deformation threshold τ_d) could be used for the lower or the average

$$\tau_{lb} = \min\{\tau_{b1}, \dots, \tau_{bm}\} \quad \text{or} \quad \tau_{lb} = \frac{\sum_{j=1}^m \tau_{bj}}{m}.$$

$$\tau_{ld} = \min\{\tau_{d1}, \dots, \tau_{dm}\} \quad \text{or} \quad \tau_{ld} = \frac{\sum_{j=1}^m \tau_{dj}}{m}.$$

For the viscosity constant h is using the average value too.

Today, the parameter assignment task for a volume of non-linear and anisotropic tissue is still an open issue. For a normal trainer (not for surgical planning), usually the tissue model is approximated with a linear and isotropic behavior. In the case of surgical planning applications, the approach considers more realistic behavior but still uses strong approximations and the tissue model remains the FEM, because it can provide more accurate results due the mesh regularity.

4.3 – Blobby meshes versus FEM

As shown before, using metaballs and the described procedure, it is possible to generate a tridimensional model able to simulate a soft tissue volume. In regards to the FEM (finite element Model), there are some differences. A first difference is that in the FEM (for example meshes of tetrahedrons), the finite element is a tetrahedron; each tetrahedron can be deformed by simply deforming its links. Using the blobby meshes, the “finite element” is just a sphere that cannot be deformed, and the model deformation is solely performed on the mesh links. This last feature means that there is a limit for the model deformation, and over this limit it is not possible to deform the body, introducing a behavior not coherent with the reality. The normal FEM model, under a large force, can collapse to a zero thickness model, allowing another non-realistic situation.

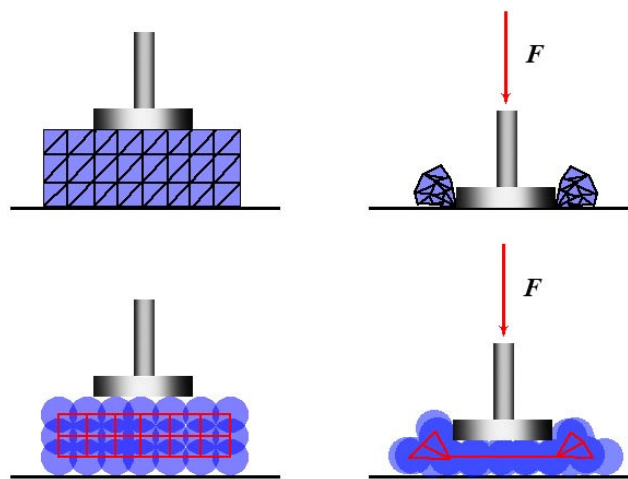
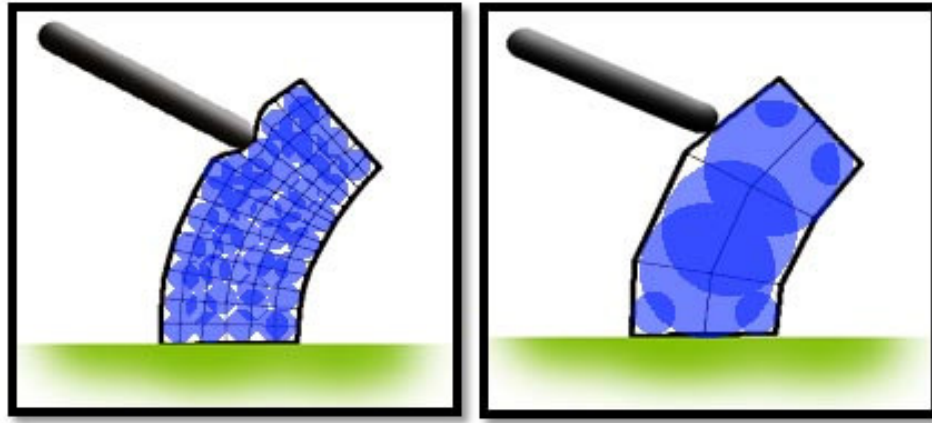


Figure 4.14 – FEM vs Blobby mesh compression

Another important limitation of the blobby meshes is that by using a MAA sphere tree, the resultant model does not allow some types of deformations (i.e. local deformations) due the low resolution of the dynamic mesh (figure 4.15a, 4.15b).



a) Octree blobby mesh.

b) MAA blobby mesh.

Figure 4.15 – Local deformation on mesh reduction

From a different point of view, the reduction of the mesh resolution determines a growth of the computing performances, and it could be an important feature for all simulation applications where accuracy is not the most important feature.

Chapter 5

Local Interaction Model

In the previous chapters, the metaballs approach for the soft tissue modeling was introduced, showing its advantages and limitations. This chapter will explain techniques able to improve that approach, adding very important features for the surgery simulation development, and also providing haptic feedback.

5.1 – Local Deformation Model

As explained in the previous chapter, the metaballs approach to the soft tissue presents some limitations using the MAA for the creation of the sphere tree. Through obtaining a low resolution mesh, it is possible to keep a global behavior, but it is not possible to obtain local deformations. During a surgery simulation, the global model behavior is very important, but the local deformations are important too: they increase the realism of the interaction and, in some cases, it is indispensable for particular kinds of surgical tools. The local deformations, however, are usually hidden behind the surgical tool and it is possible to recognize them just in proximity to the surgical tool. The fact that the local deformations are usually hidden means

their accuracy can be unimportant in terms of visual perception. Starting from this last consideration, we can expect to introduce a new approximation about the body deformations that is not coherent with reality but is acceptable in terms of computational cost reduction for the simulation. The introduced assumption is to consider the body deformations as a contribute of two different kinds of deformation

(Figure 5.1):

- Local deformation.
- Global deformation.

By expecting to split the deformations in two different contributes, it is possible to keep the global deformation provided by the low resolution mesh and it is possible to simulate “fake” local deformations using a trick.

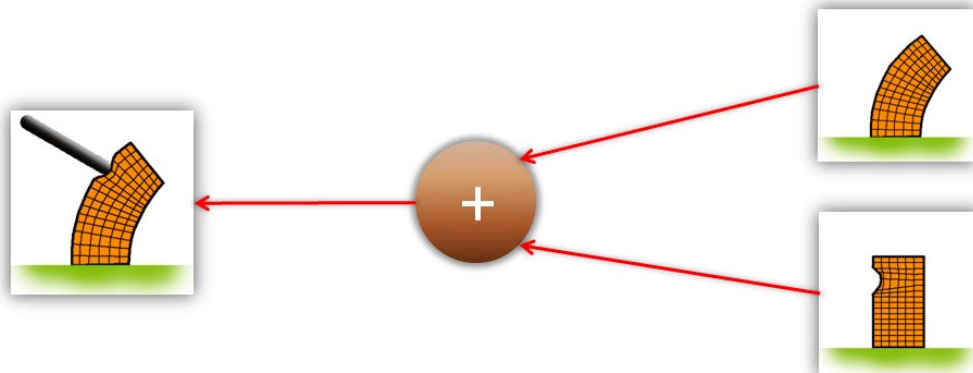


Figure 5.1 – Tissue deformation as sum of a local and a global body deformation

As mentioned in the chapter two, when working with metaballs it is possible to use metaballs with different signs, and the effect on the iso-surface is a local deformation (using a metaball of a proper ray). The local deformation is exactly what we need in order to simulate the interaction with a surgical tool (figure 5.2). Assuming the use of positive field generators for the organ model, the only actions needed is to associate a negative metaball to the surgical tool. When the surgical tool touches the iso-surface it will look deformed, simulating a local deformation with an acceptable realism.

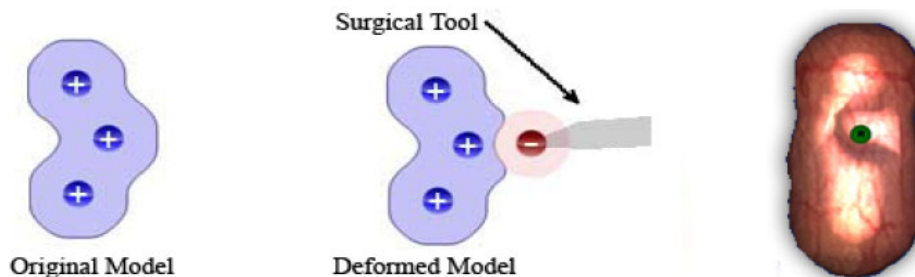
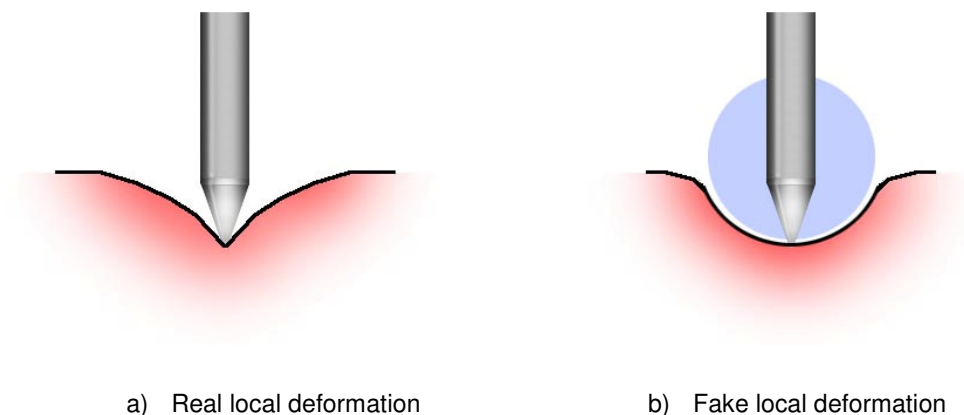


Figure 5.2 – Simulation of a local deformation using metaballs

Considering a single point tool model, the local surface deformation that we can obtain has a concave profile, completely different from the real deformation that we should obtain in a real case (Figure 5.3a and 5.3b).



a) Real local deformation b) Fake local deformation

Figure 5.3 – Real single point deformation vs. fake local deformation

Adopting a multipoint tool model, the local deformation shape can be improved; in fact, the local deformation will copy the exact shape of the surgical tool (Figure 5.4).

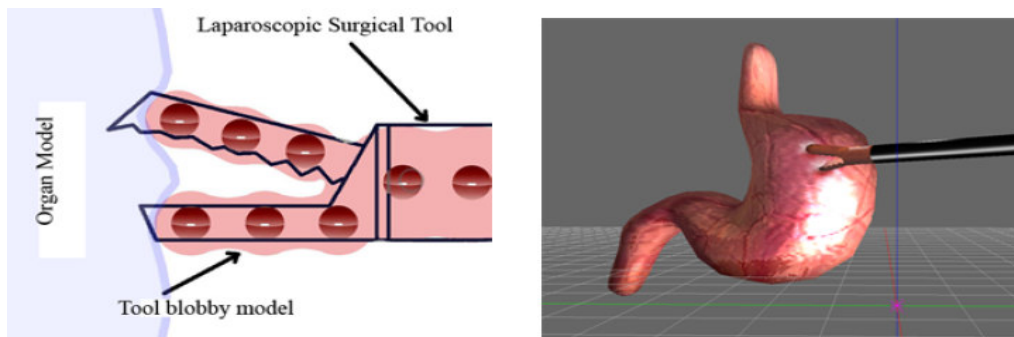


Figure 5.4 – Multipoint local deformation

The proposed local deformation technique can simulate local deformations only from a visual point of view. In order to work in an accurate way, it needs to be supported by a physic point of view with the intention of generating feedback forces. By using the proposed approach, switching the tool's particle signs, and modeling the surgical tool with a proper particle set, it is possible to simulate very complicated local deformations, to improve the realism of the simulation, and to keep the simulation operating at optimal speed.

5.2 – Local Interaction Model

After we improved the local deformation between the surgical tool and the organ model, it is indispensable to define a collision detection model optimized for the metaballs approach and able to exploit all features discussed in the past chapters. In general, a blobby model is just a way to generate a potential field and an iso-surface with a desired shape. Inside the iso-surface, the field intensity will be higher than outside, defining a scalar collision detection function implicitly able to evaluate if a collision occurs in a generic point $\mathbf{p} \in \mathbb{R}^3$ is inside as:

$$\sigma(\mathbf{p}) = \begin{cases} 1, & \Phi(\mathbf{p}) > \tau & \textit{Inside} \\ 0, & \Phi(\mathbf{p}) = \tau & \textit{On the surface} \\ -1, & \Phi(\mathbf{p}) < \tau & \textit{Outside} \end{cases}$$

where $\Phi(\mathbf{p})$ is the field function able to describe the field intensity in a generic point \mathbf{p}

$$\Phi(\mathbf{p}) = \sum_{i=0}^n f_i(\mathbf{p})$$

$f_i(\mathbf{p})$ is the field function for the metaball i in the point \mathbf{p} . In this manner, it is possible to understand if the generic point \mathbf{p} is inside ($\sigma(\mathbf{p}) = 1$) or outside ($\sigma(\mathbf{p}) = -1$) the volume, or if it is on the iso-surface ($\sigma(\mathbf{p}) = 0$). Moreover, it is also possible to approximate the minimal distance $\delta\tau$ between the point \mathbf{p} with $\Phi(\mathbf{p}) = \varepsilon$ and the iso-surface calculating.

$$\delta\tau(\varepsilon) \approx |f^{-1}(\varepsilon) - f^{-1}(\tau)| = \left| m \sqrt{\frac{Ic}{\varepsilon} - \frac{Ic}{\tau}} \right|$$

considering as Ic the intensity of the closest field generator. The approximation introduces an error on the minimal distance computation, but by using a high value for m , it can be neglected. This last value could be used, in case of haptic rendering, for the interaction force modulation, approximating the surgical tool such as a point. In case a collision occurs, i.e. $\sigma(\mathbf{p}) = 1$, it is possible to define a local reaction force vector f_r , with amplitude given by the Hook Law as:

$$\|f_r(\mathbf{p})\| = k\delta_t(\varepsilon)$$

where k is related to the local stiffness of the object, and direction as

$$\hat{f}_r(\mathbf{p}) = \frac{\sum_{i=0}^n \left[\frac{\mathbf{p} - \mathbf{c}_i}{\|\mathbf{p} - \mathbf{c}_i\|} \right]}{\left\| \sum_{i=0}^n \left[\frac{\mathbf{p} - \mathbf{c}_i}{\|\mathbf{p} - \mathbf{c}_i\|} \right] \right\|} \quad (5.1)$$

that follows the field anti-gradient $-\nabla\Phi(\mathbf{p})$ direction. This force can be used for haptic rendering purposes, e.g. for the god object method, [9]. In terms of collision response, it is possible to use the same approach discussed in the section 3.2, modeling the collision as a soft collision, distributing the contact forces to the surgical tool and on the soft tissue.

```

/* Selective field function for the metaball I
   in the point p */
float GetFieldValue(int i, point p)
{
    return metaball[i].Ic/(( p - metaball[i].c).Module);
}

```

```
// Global field function for the entire organ model
float GetFieldIntensity( point p )
{
    float field=0;
    /* Sum of the contributes of each metaball
       In the variable field */
    for(int i=0; i<metaballs; i++)
    {
        field += GetFieldValue(i, p);
    }
    return field;
}

/* Collision detection function evaluate the field in a
   specific point in the space */
int CheckCollision( point p )
{
    float f;
    f = GetFieldIntensity(p);
    if(f < SurfaceThreshold) return -1; //Outside
    if(f > SurfaceThreshold) return 1; //Inside
    if(f == SurfaceThreshold) return 0;
    /* On the iso-surface */
}
...
//somewhere inside the code
if(CheckCollision(p) >= 0)
{
    // collision response algorithm
}
...
```

Source 5.1 – Collision detection and handling

Moreover, the surface friction can also be modeled on the iso-surface by means of classic formulations, considering f_t and f_n , the tangential and normal component to the surface of the force vector f_r . For example, a static condition is obtained if

$$|f_t| \leq \mu_s |f_n|,$$

while a viscous force friction can be computed as

$$|f_d| = -\mu_d |f_n| \hat{v}_t,$$

where μ_s ; μ_d are the static and dynamic friction coefficients, and \hat{v}_t the direction of the tangential velocity. From the algorithm point of view, the collision detection can be written as in the source 5.1. When looking at the source code, it is evident that by using the metaballs approach, the collision detection is very fast because it does not need to perform complicated polygons intersections; the collision detection requires only the evaluation of a field function in a specific point of the space, with a linear complexity. From a computational point of view, this last result is very important, because using the classic approach (meshes of FEM), the collision detection was very expensive and tricky to implement. At this point, the principal advantages achieved using the metaballs approach can be represented by the possibility of reducing the mesh complexity, keeping the local deformation and a very fast collision detection; these are two important results that save a considerable amount of time that can be spent on the execution of other tasks such as improving the speed and the complexity of the simulation. The source code 5.1 does not implement any optimization, but in the case of complex geometries, with a big number of metaballs (i.e. $n > 50k$), a spatial subdivision could be implemented in order to improve the computational performances. It is necessary to remark that with more than 50k metaballs it is possible to create very complex geometries.

5.3 – Multi-Body Interaction

There are several advantages obtained through the use of the metaballs approach; it is possible to save a great deal of computational power that could be spent improving the realism of the surgical scene. A first important feature that can be implemented is the multi-body simulation, allowing the simulation to manage multiple soft/hard bodies. The multi-body extension is based on the same technique used for the local interaction: using different signs for the local deformation and a soft particle collision response. The first difference, however, is that by having more than two bodies, the use of two signs is not a possible solution, so the multi-body extension requires the use of a body ID for each metaball. This is necessary to determine if a specific metaball belongs to a specific body. By using the body ID, it is possible to sum the contribute of all metaballs with the same ID and to subtract the contribute for the others.

```
...  
  
// Global field function for the entire organ model  
float GetFieldIntensity( int BodyID, point p )  
{  
    float field=0;  
    /* Sum of the contributes of each metaball  
       In the variable field */  
    for(int i=0; i<metaballs; i++)  
    {  
        if(BodyID == metaball[i].ID)  
            field += GetFieldValue(i, p);  
        else  
            field -= GetFieldValue(i, p);  
    }  
    return field;  
}
```

```
/* Collision detection function */
int CheckCollision( int BodyID, point p )
{
    float f;
    f = GetFieldIntensity(BodyID, p);
    if(f < SurfaceThreshold) return -1; //Outside
    if(f > SurfaceThreshold) return 1; //Inside
    if(f == SurfaceThreshold) return 0;
    /* On the iso-surface */
}
...
//somewhere inside the code
if(CheckCollision(Body[i].ID, p) >= 0)
{
    /* collision response algorithm */
}
...
```

Source 5.2 – Multi-body Collision detection and handling

Through extending the simulation with the multi-body feature, the field function and the collision detection changes as shown in the source 5.2 (all modifications are in red). The last modification for the collision detection slightly increase the complexity when a single body was just linearly dependant to n (number of metaballs) to $n \times m$ where m is the number of bodies inside the simulation.

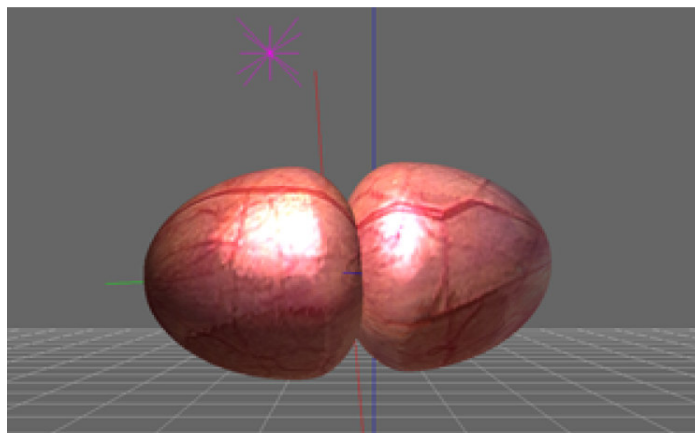


Figure 5.5 – Multi-body contact

When using two metaballs with different ID's, the contact should look like in Figure 5.5. Another important modification is for the drawing task. This needs to be modified because in the multi-body case, each body should be drawn separately, spending more time extracting each body surface separately. In order to improve the multi-body collision detection and response in a scene with a large number of metaballs, the octree space partitioning represents a good solution. A single metaball as field generator generates an infinite field from the source position; knowing this, before using an octree, it is indispensable to define a cutoff threshold in order to identify where the field should be approximate to zero. In this way, it is possible to say that each metaball field has effect on a specific distance and over that distance the field effect can be ignored.

5.4 – Multilayered Surfaces

Another possible improvement that could easily be achieved for the interaction model is the multilayered model. The human body is made of organs (e.g. muscles) overlapped with bones or other tissues with different visco-elastic properties. In the multilayer case, the local interaction force f_r needs to be calculated considering all the contributions of the different layers. With the metaballs approach, the multi-layer extension can be easily obtained by considering more threshold values (or using particular rules in the field generation algorithm, but in this thesis we only consider the multi-threshold case for simplicity). If we consider n iso-surfaces, we can define the local interaction force as obtained by n springs, each of them with different properties, connected in series, see figure. 5.6.

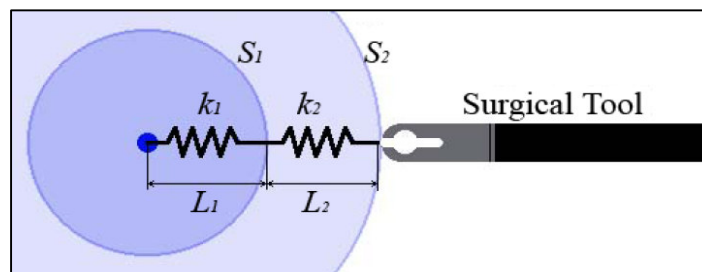


Figure 5.6 – Multilayered local contact model

As an example, consider two surfaces S_1 and S_2 , respectively with thickness L_1 ; L_2 , and stiffness constants k_1 ; k_2 (in this case, linear spring are used for simplicity, but more complex functions could be considered as well). If a deformation Δx is applied on the external organ surface, the deformation on each spring can be computed from:

$$\Delta x_1 = \frac{k_2}{k_1 + k_2} \Delta x, \quad \Delta x_2 = \frac{k_1}{k_1 + k_2} \Delta x.$$

Considering the equivalent stiffness k_{eq}

$$k_{eq} = \frac{k_1 k_2}{k_1 + k_2}$$

the amplitude of the resultant elastic force is

$$|f_r| = k_{eq} \Delta x = k_{eq} (\Delta x_1 + \Delta x_2).$$

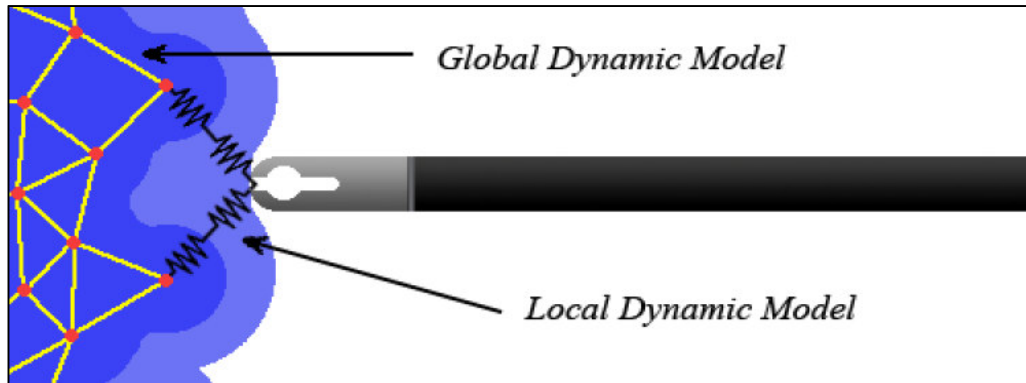


Figure 5.7 – Global-local interaction

while its direction is the same as $\hat{f}_r(\mathbf{p})$ for the haptic tool, see (5.1). The computed force is projected also on the closer particles and then to their visco-elastic links, resulting in a deformation of the global model, figure 5.7. The global deformation can be obtained only through a local interaction and deformation; the feedback force provided to the haptic tool will be the sum of two contributes: a local interaction

force and a global deformation force that will be computed as the resultant force applied by the tridimensional mesh on the touched particles.

5.5 – Haptic Textures

The last improvement proposed, using metaballs, is the use of haptic textures. When using a haptic tool, what we can perceive is usually only the shape of the object; on good haptic simulations, friction can be perceived as well. The haptic textures are a technique able to improve the realism of touch through the addition of the perception of the surface texture (roughness, regularity, and so on). There are different ways to simulate the surface texture:

- Procedural textures.
- Lookup texture.

The procedural textures are computed at runtime, using usually the model surface coordinate system (a bi-dimensional coordinate system), adopting a function of two coordinates able to perturb the surface geometry, stiffness and friction, adding an offset.

$$f_t: \mathbb{R}^2 \rightarrow \mathbb{R}.$$

A lookup texture is a bi-dimensional matrix (like a bitmap), mapped on the model surface and is similar to a common graphic texture (figure 5.8).

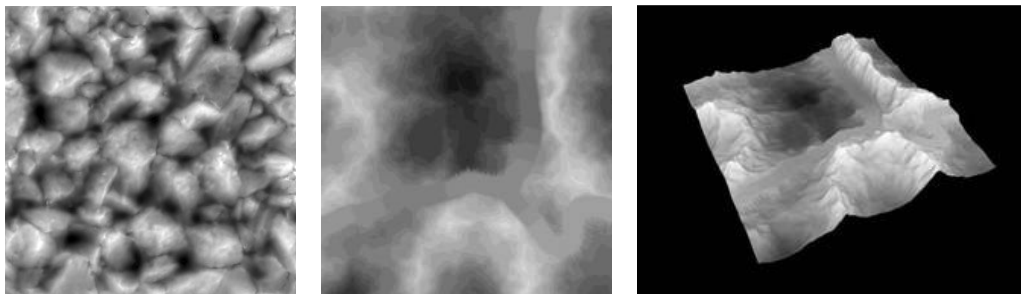


Figure 5.8 – Lookup Textures

The lookup textures require a mapping function too. As anticipated, this is a haptic texture.

In case the haptic texture is used as a displacement map, adding an offset on the surface geometry (we should perceive that also from the graphic rendering using a bump-map²), for the metaballs approach perturbs the field function before the surface extraction, then the local field function becomes:

$$f(r, \theta, \phi) = \frac{I}{r^m} o_t(\theta, \phi).$$

Where $o_t(\theta, \phi)$ is the texture value at the coordinate θ, ϕ (respect to the center of each metaball, or to the center of the object, or using a iso-surface parameterization).

In this case, the computation of the elastic force is more complex because the contribution of the texture is calculated for all the metaballs close to the haptic tool.

Considering the use of two stiffness haptic textures (for two overlapped iso-surfaces) on the surfaces, it is possible to express k_1, k_2 as a sum of two contributions

$$k_1 = k_{s1} + k_{t1}(\theta, \phi); \quad k_2 = k_{s2} + k_{t2}(\theta, \phi);$$

where k_{s1}, k_{s2} , are the constant stiffness values of the tissues, and $k_{t1}(\theta, \phi), k_{t2}(\theta, \phi)$, are the haptic texture values, at the coordinates θ, ϕ (again respect to the center of each metaball, or to the center of the object, or using a iso-surface parameterization).

Finally, considering the texture for the friction case, the friction constants (only the external surface can be considered) should be modeled in the following way.

$$\mu_D = \mu_d + \mu_{td}(\theta, \phi); \quad \mu_S = \mu_s + \mu_{ts}(\theta, \phi);$$

² In computer graphics, the bump-map or normal-map is a particular kind of texture able to simulate the roughness of a polygon perturbing the surface normal and adding a local shading effect.

where μ_d, μ_s are the respectively the dynamic and static friction constants values, and $\mu_{td}(\theta, \phi), \mu_{ts}(\theta, \phi)$, are the respectively the dynamic and static friction texture values, at the coordinates θ, ϕ .

Chapter 6

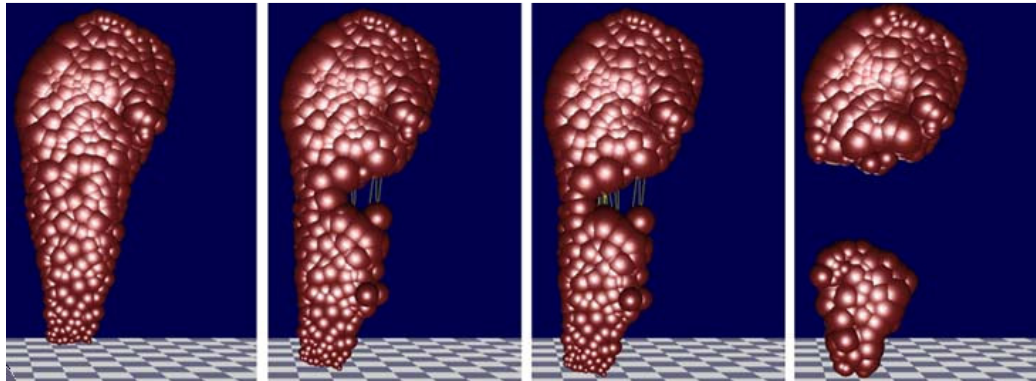
Cuttings and Fractures Simulation

Fractures and cuttings are the last two issues discussed in this thesis. This chapter will close all requirements for a surgical simulation. How the metaballs approach can improve the development of a surgery simulation, allowing an easy cut and fracture simulation, will be examined. Other tasks required for surgical operations, such as the suture task, are exactly the same used for alternative approaches so they will be not discussed inside this thesis, which is focused on the identification of a general methodology for the soft tissue simulation in the surgical field.

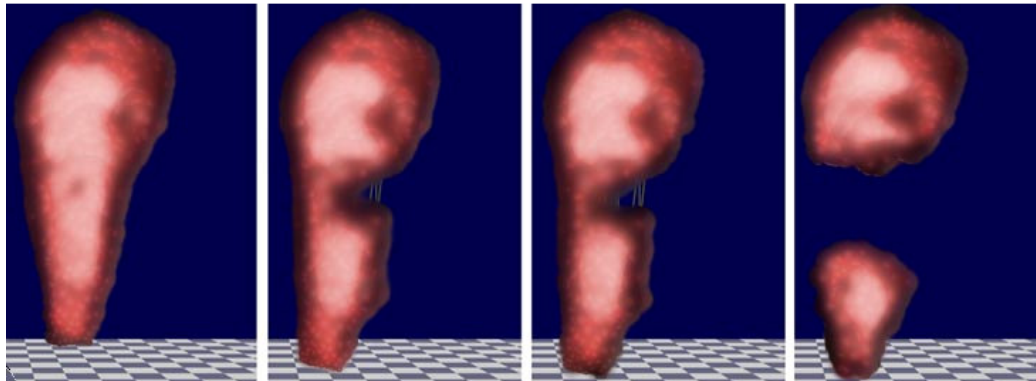
6.1 – Fracture Simulation

Fractures simulation is a task that is not often performed during operations, because the usual way to remove pieces of tissue is using cutting tools in order to avoid tissue damages. Fracture simulations are performed, however, where it is not possible to cut directly the tissue. For the metaballs approach, the adopted fracture model can be exactly the same used for the classic FEM meshes, working directly on the connection links. As mentioned in the chapter four, each link has a breaking threshold that is used in order to recognize if the link should be broken during the

deformation. With respect to the FEM model, in the fracture simulation, using the breaking threshold works better because when the links are broken and the volume is subdivided; it is not necessary to rebuild the fracture internal surface, because the surface will automatically generated during the iso-surface extraction with the marching cube (figure 6.1a, 6.1b).



a) Fracture simulation on the sphere tree



b) Fracture simulation from the iso-surface point of view

Figure 6.1 – Fracture simulation using the breaking threshold

It is possible to perform the fracture simulation saving precious time that can be spent on improving other elements. Unfortunately, when using the same technique for the fracture simulation, it is not possible to avoid ghost links (figure 6.2). If the model geometry is not extremely complex, it is possible to implement a proper algorithm for ghost link avoidance. A possible limitation for the fracture simulation using metaballs is the extremely rounded aspect of the fractures; this does not match

with reality that usually is not rounded and looks more similar to the FEM fracture simulation.

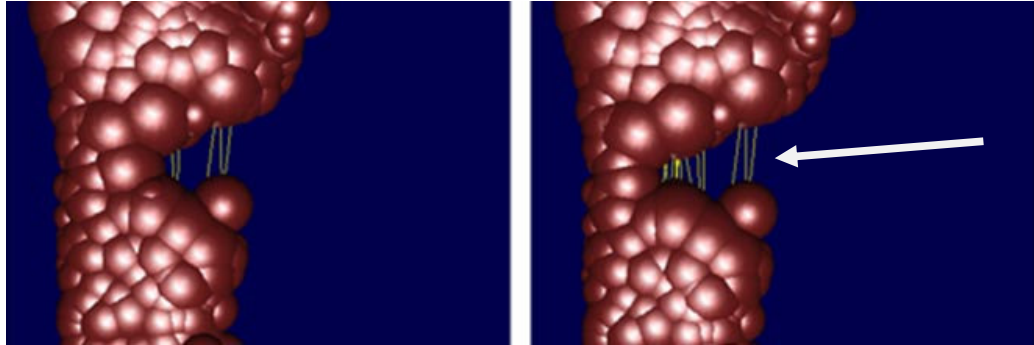


Figure 6.2 – Ghost links problem on metaballs

This fact happens because if the broken link connects two big metaballs, the fracture will have as profile, the metaball profile, and will look rounded. In the case of FEM fractures, the fracture follows the tetrahedral mesh and the fracture assumes an irregular aspect and severs with very clear edges. Usually, the fractures in surgery are applied on small pieces of tissues, then under this last hypothesis the rounded aspect could be acceptable, considering the other advantages achieved using this method. In terms of algorithm complexity, the fracture simulation could be implemented as shown in the source 6.1 (C++ code snippet for the link computation block, see figure 1.4).

```
TMetaball * Metaballs; // metaballs pipeline
TLink * link;          // link pipeline
unsigned int Linknum;  // number of links in the pipeline

/*Remove the link[index] from the model pipeline*/
int RemoveLinkfromPipeline(int index);

/*Compute the link[index] deformation and the forces for the
  metaballs p1 and p2 connected to the link */
void ComputeDeformationForce(int index);
```

```
...
// link forces computation block, inside the simulation loop
void ComputeLinkDeformation(...)
{
    float Stress; // measure the normalized link stress
    // links computation loop
    for(int i=0;i<Linknum;i++)
    {
        ...
        // calculating the stress factor
        Stress =((link[i].p2-link[i].p1).Module() )/link[i].len;

        /* if the stress factor is greater than tb the link
           Will be broken*/
        if(Stress>=link[i].tb)
        {
            RemoveLinkfromPipeline(i);
        }
        else // else the deformation force will be computed.
        {
            ComputeDeformationForce(i);
        }
        ...
    }
}
...
```

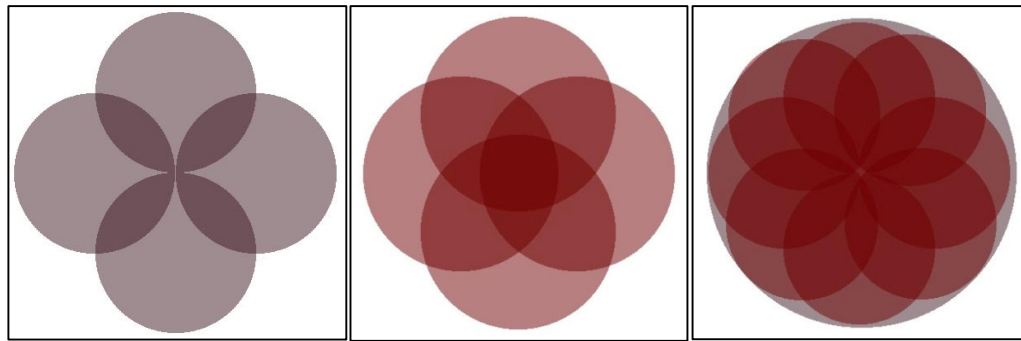
Source 6.1 – Fracture simulation

6.2 – Cuttings Simulation

The cutting simulation is still an open issue today in the surgery simulation field because is a very expensive and complex task. In the first chapter, there is a brief discussion about this topic and an introduction on the related problems. Usually, using the classic approaches, it was possible to perform cuts only on small portions

of the scene, in order to reduce the complexity and the computational costs. Using the metaballs approach, the complexity of the cuttings simulation is strongly reduced, offering the possibility to perform cuts everywhere inside the surgical scene.

The first difference between the metaballs approach and FEM is that, in FEM, the cut is performed on the mesh element (tetrahedron or different one), but in the metaballs case, it is performed on the sphere, just because in this case the sphere is the minimal volume element. The cut on the sphere is performed by recursively splitting the sphere into a pattern of sub-spheres (reducing the sphere size). In case the spheres are of a specific smaller size, the sphere is simply removed. As an example, using a 2D version of this algorithm, we can assume to have a circles pattern (2D version of the spheres pattern) like the pattern in figure 6.3.



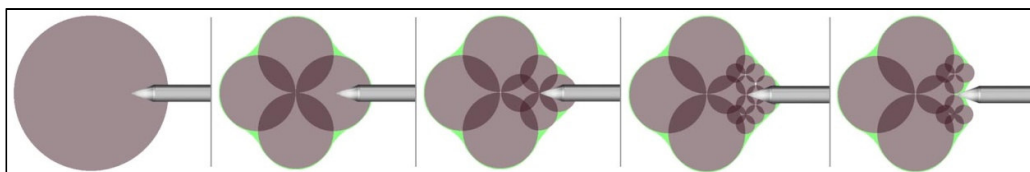
a) 4 circles.

b) 4 Circles.

c) 8 circles.

Figure 6.3 – Circles pattern

When a circle that is greater than a specific size is touched by a surgical tool, the circle is replaced recursively by a circles pattern with a proper size (the pattern will be contained inside the starting sphere, figure 6.4a, 6.4b, 6.4c, 6.4d); if the circle is smaller than the minimum size, the sphere is deleted (figure 6.4e).



a

b

c

d

e

Figure 6.4 – Circle splitting simulating the cut

Applying the splitting technique on a more complex soft tissue model, under the gravity effect, the cut can be performed as shown in figure 6.5.

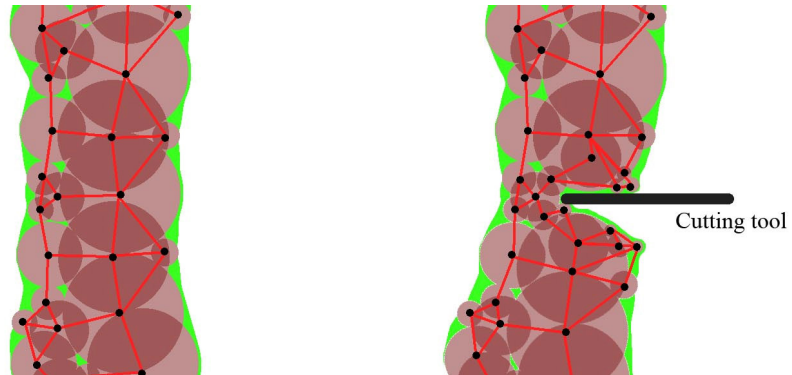


Figure 6.5 – Model-cutting simulation using metaballs splitting

Extending the model to the tridimensional case the spheres pattern could be as in figure 6.6.

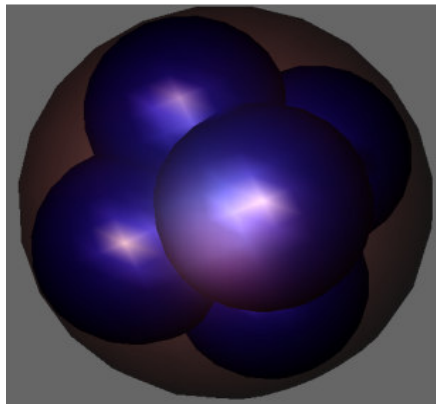


Figure 6.6 – Splitting spheres pattern

The metaball split is not enough for the cutting simulation, because the replaced metaball was connected with visco-elastic links to other metaballs (see figure 6.5). After the split, it is necessary to connect the new spheres pattern to the existing metaballs mesh. The new spheres pattern will come already connected internally and each sub-sphere will be connected with the external spheres previously connected with the replaced one (figure 6.7).

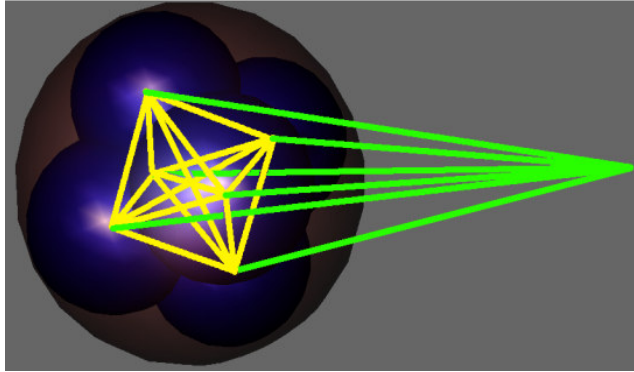


Figure 6.7 – Spheres pattern internal and external connections

Remarking in few words the cutting algorithm can be expressed in few rules:

- i. Each sphere touched by the cutting tool will be replaced by a specific pattern internally connected in a specific way.
- ii. Each sphere of the pattern will be connected with all external spheres previously connected with the replaced sphere and the elasticity constant will be subdivided for each new link.
- iii. The mass of a replaced sphere will be subdivided between all new spheres.
- iv. The recursive splitting will be stopped on a minimum sphere size (minimum ray); after that, each sphere and its link will be removed.

In terms of implementation, the C++ snippet shows the very low complexity of the described algorithm.

```
// minimum sphere ray
#define SPHERE_MIN_RAY 0.25f

Point3D CutterPos; // Cutter position

/* Split a metaball, remove the old metaball
   and add the new pattern at the end of the metaballs
   array creating the metaballs connections*/
void SplitMetaball(int index){...}
```

```
// Remove a metaball from the array
void RemoveMetaball(int index){...}

/* Routine for the Cutting simulation using a metaballs
   splitting approach */
void CutModelSim()
{
    for(int i=0; i<metaballsnum; i++)
    {
        if((CutterPos - metaball[i].c).Module<metaball[i].ray)
        {
            if(metaball[i].ray >= SPHERE_MIN_RAY)
            {
                SplitMetaball(i); /* New metaballs will be
                                   added at the end of the
                                   metaballs array */
                i--; /*because the sphere set will be
                     shifted by 1 to the left*/
                metaballsnum+=5; /* 6 spheres in the
                                   pattern minus the old one*/
            }
            else
            {
                RemoveMetaball(i);
                metaballsnum--;
                i--; /*because the sphere array will be
                     shifted by 1 to the left*/
            }
        }
    }
}
```

Source 6.2 – Cutting simulation using metaballs splitting

From the source code analysis, it is easy to understand that the algorithm implementation is not difficult, because the cut simulation is working on the sphere collision, not on complex polygon intersections, and can be implemented in a without issue.

6.3 – Cuttings Optimization

The metaballs cutting, such as the FEM mesh cutting (using the tetrahedron splitting), increases the mesh complexity for each metaball split. A first difference is that a single metaball split increases the mesh complexity faster than the tetrahedron split. For example, in the metaball split, using a six sphere set, the metaball number is increased by five and the internal links number increases by fifteen plus five times the number of external links (table 6.1).

Split	Before the Split	After the Split	Total
Metaballs	1	6	+ 6
Internal Links	0	15	+15
External Links	5	30	+35

Table 6.1 – Single metaball split complexity

Considering that when we split a metaball to simulate a cut, the old sphere is removed and at least one sphere of the set is removed; the table for the cut become such as the table 6.2.

Cut	Before the Cut	After the Cut	Total
Metaballs	1	6-1	+ 4
Internal Links	0	15-5	+10
External Links	5	30-5	+20

Table 6.2 – Single metaball cut complexity

Finally, the result is that for a single metaball connected with five external links a single cut increase the metaballs by five and the links by thirty. At the second level of cutting, each single sphere will be connected with four old internal links plus five old external links, and then the complexity will grow quickly. A possible solution for the proposed approach is to connect each external link after the sphere replacement, not with all the new metaballs, but just with the closest one.

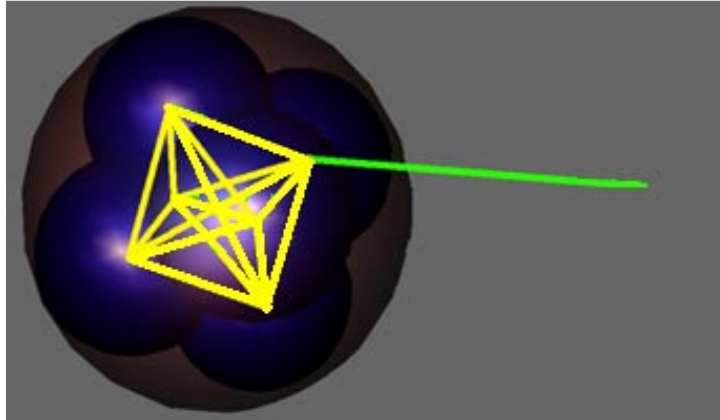


Figure 6.8 – Closest link optimization

Adopting the proposed optimization, it is possible to reduce the complexity growth, but this solution does not assure the non-elastic deformations avoidance. Another possible solution could be to connect sequentially each external link with the closest unconnected metaball. Even so, considering that the number of metaballs is very low, thanks to the MAA, the complexity growth represents an unimportant problem (usually < 10k for a complex surgical scene).

Usually after a cut, inside the mesh there may be residual metaballs, very small and completely contained in other greater spheres connected with them; if so, it is possible to remove those spheres. In order to understand if a sphere is contained inside another one, a test must be performed considering the maximum link extension before the breaking threshold (figure 6.9). If a sphere is contained inside another one at the maximum link extension, that sphere will be removed from the mesh, reducing the mesh complexity. This last test will be performed at runtime, on only the new spheres introduced by the pattern used for the replacement; this is inexpensive.

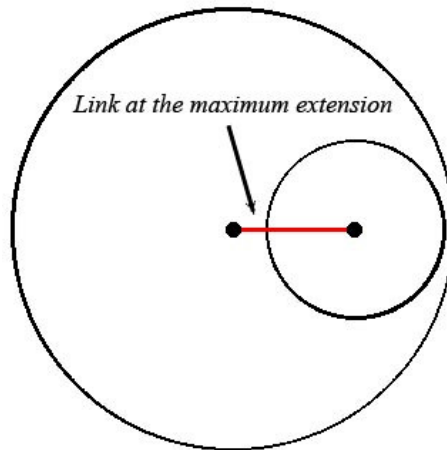


Figure 6.9 – Sphere bounding test

After this last discussion, it is easy to understand that the metaballs approach for the soft tissue simulation can create a lot of advantages for surgery simulation development, allowing the implementation of very sophisticated simulations, keeping a low complexity and requiring a less expensive computational load.

Chapter 7

Conclusions and Final Remarks

Fractures and cuttings are the last two issues studied during this research, which will close all requirements for a surgical simulation. This chapter will discuss about the achieved results (showing quickly the developed software) and about the future development for the metaballs approach.

7.1 – Developed Software

During the study about the proposed approach for soft tissue modeling, some software was developed in order to measure the achieved performances and in order to improve the general model finding specific optimization. The first system developed was an upgrade to existing software (developed for the master degree) and was a surgery simulator based on mesh of tetrahedrons. The development of this first simulation software was motivated by the need to investigate all the limitations for the FEM model for real-time application. The name of the first simulation system was LapLab (figure 7.1).

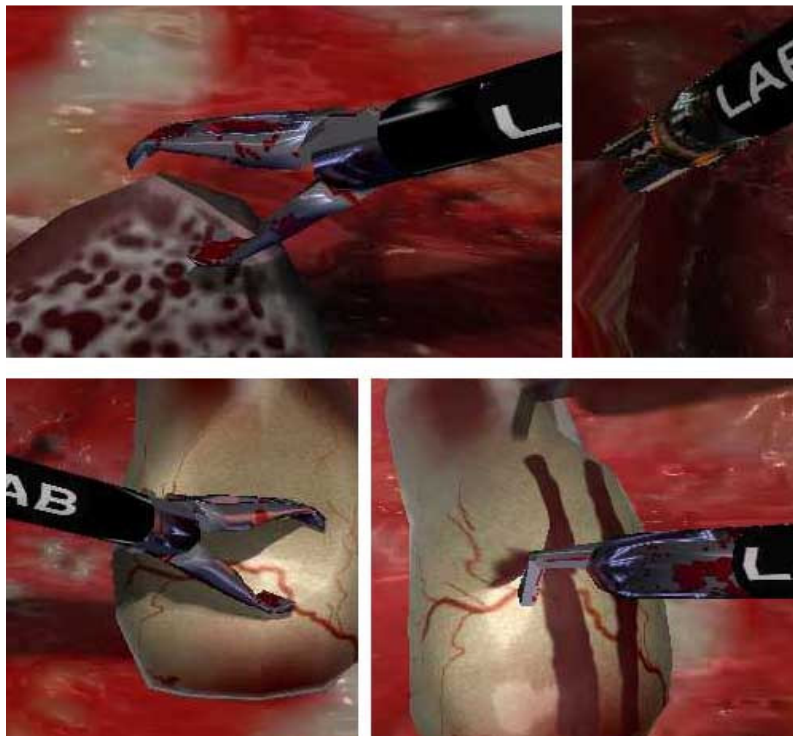


Figure 7.1 – LapLab laparoscopic surgery simulator

The LapLab simulator was able to provide haptic feedback using two phantoms Omni (by SensAble) connected to two real laparoscopic surgical devices (figure 7.2).

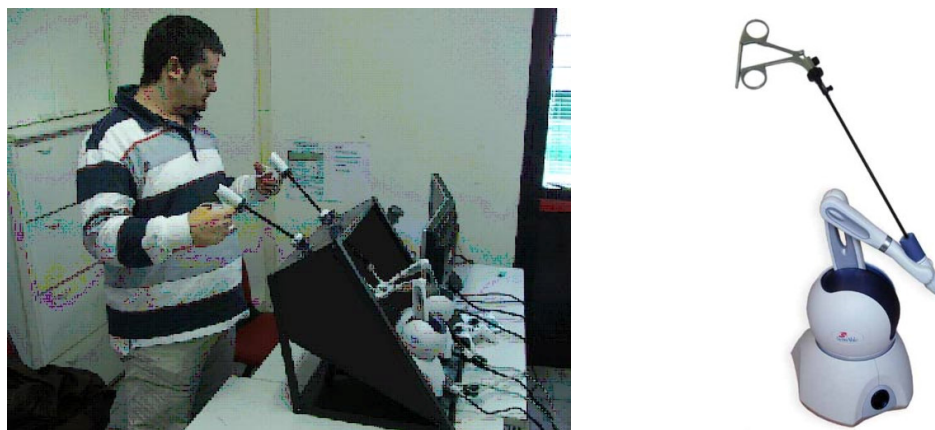


Figure 7.2 – Haptic tools interface

The implementation of a first simulation system based on FEM evidenced all problems and limitations due the complexity of the collision detection algorithm and response. The only way to work in real time was to create a surgical scene not

completely active where it was possible to perform specific tasks only on specific regions and the rest of the scene was simply a static mesh. Again, during the studies, other programs were developed in order to evaluate advantages and disadvantages for the metaballs and FEM approaches (figure 7.3).

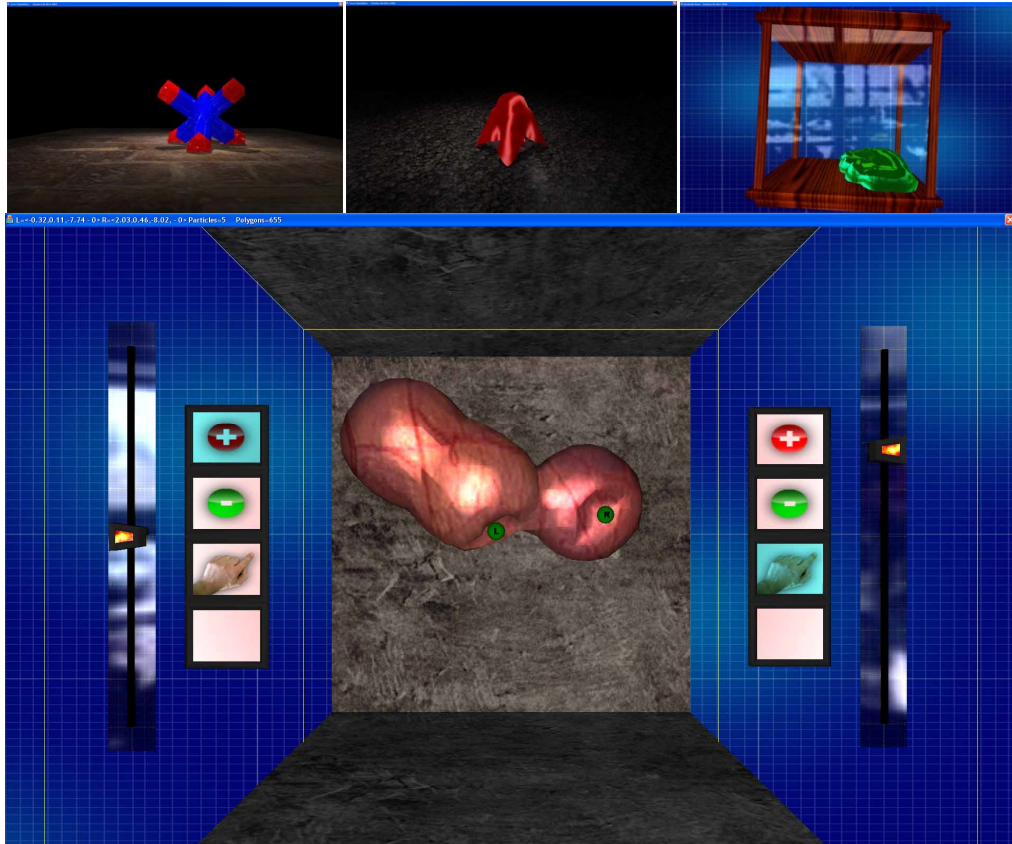


Figure 7.3 – Extra software for the FEM and metaballs performances testing

For the metaballs approach, a simulation system was developed also for the DaVinci's robot (by Intuitive Surgical), where it was possible to interact with the whole surgical scene, keeping good computational performances (figure 7.4).

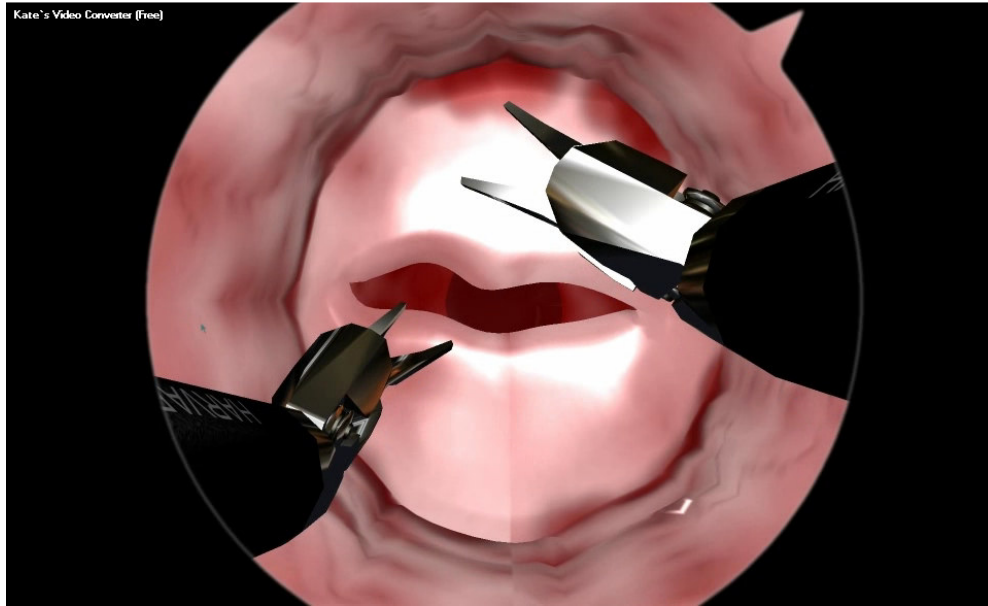


Figure 7.4 – DaVinci's robot simulation using metaballs

For the conversion process between polygonal to blobby model, an existing software “*SphereTree*” was downloaded from the MAA author's website (figure 7.5).

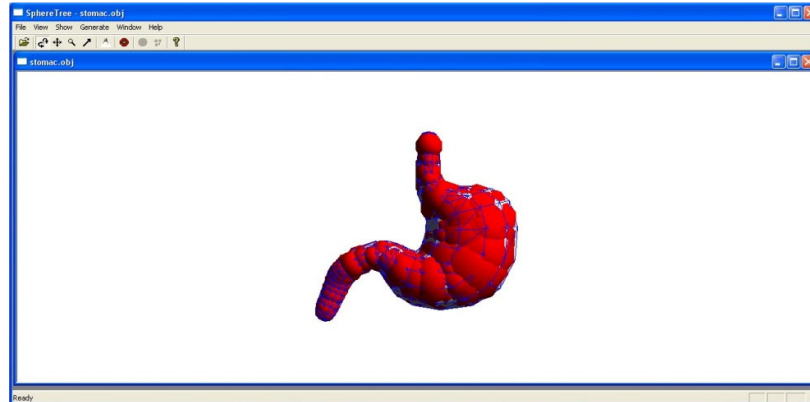


Figure 7.5 – Sphere tree conversion tool

For the volumetric dataset manipulation, a real-time viewer was developed (figure 7.6).

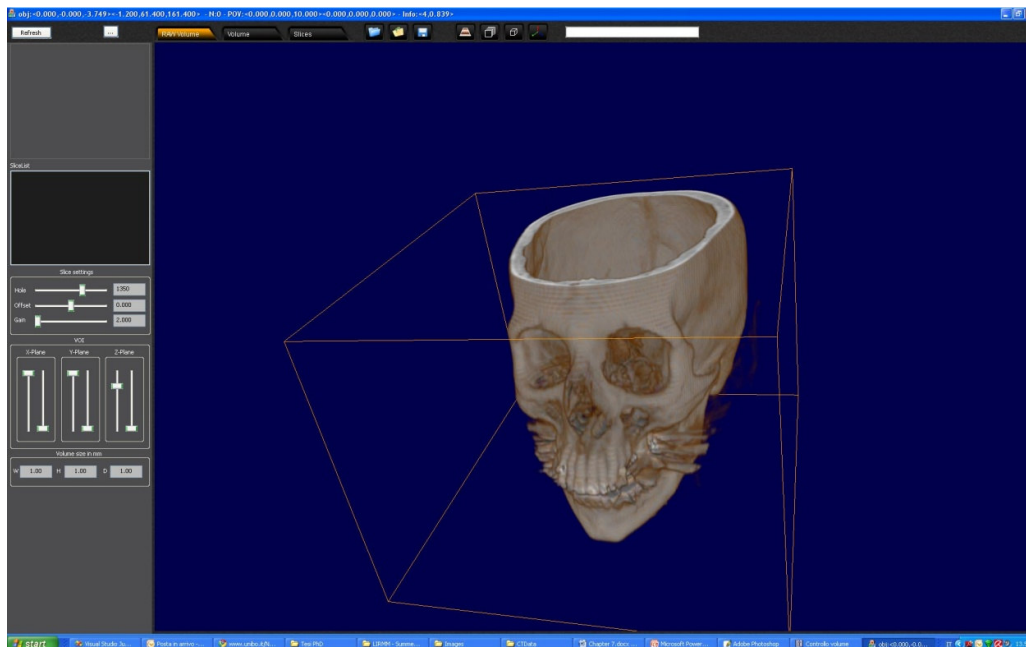


Figure 7.6 – Real-time volumetric dataset viewer

The generated sphere tree is an ASCII file containing all spheres information for the organ model and is encoded as shown in the list 7.1.

```

LOD=2 SOHERES=338
X Y Z RAY
-0.245934 0.238859 0.002782 0.162877
0.717871 0.980993 0.040382 0.604956
0.999624 0.933256 0.149999 0.605926
0.696139 0.839686 0.441099 0.676269
0.801164 0.744722 0.643430 0.640624
1.076958 0.848909 0.375355 0.631199
1.117090 0.756500 0.601770 0.604258
0.437312 0.744141 0.635375 0.681932
1.014425 0.435329 1.025270 0.462862
0.687898 0.201090 1.219617 0.349213
0.857836 0.685575 0.740987 0.659986
0.654323 0.482980 0.981736 0.580711
0.853320 0.367978 1.045842 0.421638
0.429632 0.532271 0.921596 0.635001
1.153475 0.493751 0.966038 0.487104
1.540440 0.347572 1.065334 0.365552
    
```



```
1.121983 0.273116 1.157629 0.388830
1.660789 0.700315 0.629767 0.560564
1.382047 0.763577 0.543778 0.606373
1.405957 1.055469 0.355325 0.355389
1.302149 0.559545 0.892073 0.549431
1.316268 0.673130 0.737208 0.565536
...
```

List 7.1 – Sphere object file

All blobby models are encoded in spheres files; at the simulation startup, all metaballs data are generated before starting the simulation loop.

7.2 – Results

Thinking about the basic requirements for a surgery simulator, the metaballs approach for soft tissue modeling was developed. The principal result of this research is that the proposed approach is able to satisfy all requirements. In terms of computational performances, some tests were performed, using a liver and a stomach model and modeling the surgical tool with one and two particles.

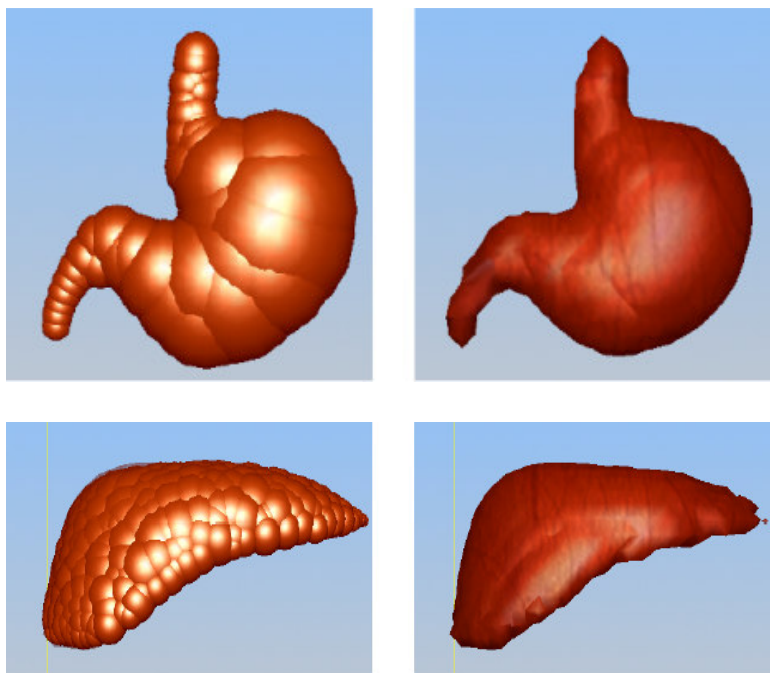


Figure 7.7 – Blobby models used for the benchmark

The tests were performed in order to evaluate the computational load when increasing the number of metaballs on the model side and on the tool side. Performing the test using a slow computer and without an implementation on GPU, the results are very positive, showing that the proposed approach really increases the potential for the development of a surgery simulation system (see table 7.1).

Model Type	Total Mtbls	Contact Points	Dynamic	Graphic
Stomach	48	1	~1000Hz	~68Hz
Stomach	48	2	~996Hz	~68Hz
Liver	366	1	~893Hz	~66Hz
Liver	366	2	~888Hz	~65Hz
Mitral Valve	421*	8	~714Hz	~49Hz

* Performed merging together a metaballs model and a little FEM model.

Table 7.1 – Metaballs performances benchmark

The hardware platform used for the test was equipped with a 1.8 Ghz 32bit CPU, 4Gb RAM, GPU nVIDIA GeForce 7950GTX, bus PCI-Express. The developed software is based on OpenGL graphic library, and runs on Windows XP Professional. From the data reported in table 7.1, it is possible to see that the performances of the dynamic model are not strongly influenced by the number of metaballs used for the organ model, but are more affected by the number of particles used for surgical tool representation. The reasons for these results are reported in the following subsections.

A. Local interaction model performances

The local interaction model can be influenced by the number of metaballs used for the tool representation and by the number of tissue layers. In particular, the performances obtained simulating multi-layer bodies depended linearly on the number of layers. The multi-point contact feature’s performance depended on the product between the number of particles in the tool and the number of particles of the organ. The local deformation of the iso-surface obtained by the interaction between

the surgical tool and the organ model's performance was linearly dependant on the sum of particles used for the organ model and surgical tool representation.

B. Global interaction model performances

For the global interaction model, the number of links used for interconnecting metaballs drastically influenced the performances. The number of metaballs used in the surgical tool, in this case, is not relevant because the forces exchanged between the tool and the global model are generated by the local interaction model. The multi-body interactions can be managed using the sphere tree used for the model generation, introducing a logarithmic dependence from the number of particles used for each body.

C. Graphic rendering performances

The graphics performances are influenced by a sum of factors. The first one is the marching cube algorithm that scanned a fixed volume in order to extract the external iso-surface. The cost of this algorithm is typically a constant for each frame (for non-optimized implementations). The second contribute to the performances' degradation is represented by the number of polygons used for the surface representation. Other factors contributed as well: the number of textures, the number of lights, and the complexity of shaders (if used) for procedural texturing. From the data reported in Table II, the obtained frame rate is sufficient for a fluid graphic representation of the scene.

7.3 – Conclusions and Future Works

The metaballs approach results are quite interesting because it is easy to implement and allows for satisfactory performances in surgical simulations. The proposed approach reduces the complexity of the 3D models, a fact that can be translated on the other hand in a reduction of accuracy for local/global deformations. The loss of accuracy, however, can be acceptable for some kinds of simulations, considering that it allows an easy handling of multi-body interactions, multilayer tissues, and multi-point contacts and deformations. The techniques described in this thesis have been

used for the development of a platform for training minimally invasive surgery operations [19]. The main components of this system are the Virtual Reality simulator and two haptic interfaces for force rendering to the operator. An important future work will be new implementation using the CUDA⁶ technology in order to improve the algorithm performance. Another important future work will be the study of an automatic procedure for the blobby mesh parameters tuning, grabbing all mesh parameters directly from a volumetric dataset, but with a better accuracy. For simulations of local deformations, it is also possible to increase the accuracy through dynamically splitting the metaballs close to the interaction point. This new idea needs to be investigated as well because it may increase the accuracy of the simulated model.

⁶ CUDA is a technology developed by the nVIDIA Inc. that uses GPUs for high performance general purpose parallel computing.

List of Figures

1	Implicit Model	3
1.1	Visco-elastic Link model	6
1.2	Link deformation	8
1.3	Tetrahedron and 3D mesh based on tetrahedron	9
1.4	Physic simulation loop	10
1.5	Graphics-Physics Synchronization scheme	11
1.6	Octree space partitioning	12
1.7	Triangle normal and forces	14
1.8	Proxy object and soft tissue	14
1.9	Cut simulation	15
1.10	Tetrahedron removing cut surface problem	16
1.11	TR improvement	16
1.12	Decomposition of tetrahedron by edge split	17
1.13	Link breaking	18

1.14	Ghost connection	18
2.1	Metaballs Field functions	21
2.2	Interaction of metaballs	22
2.3	Metaballs modeling	22
2.4	Two metaballs with different sign	23
2.5	Approximated 3D model	23
2.6	Octree sphere tree for different levels of detail	24
2.7	Conversion steps from a sphere tree to a metaballs model	25
2.8	Model synthesis/simulation workflow	26
2.9	Marching cube	26
2.10	Surface refinement	27
2.11	Examples of marching cube application in medical imaging for tissue Segmentation	28
3.1	Fluid simulation using metaballs	34
3.2	Realistic fluid simulation	35
3.3	Fluid approximation using particles under the gravity effect	35
3.4	Fluid particles contact model	36
3.5	Fluid simulation block	37
3.6	Cohesion force between particles	37
3.7	Smoke simulation using particles and a quadratic blur effect	38
4.1	Cloth model	42
4.2	Full Elastic deformable cloth model	42
4.3	Neighbor connection strategy	43
4.4	Ghost link problem	43
4.5	Local topological connections	44
4.6	Intestine simulation model using cross-links	45
4.7	Mesh resolution growth on the edges	45
4.8	Bounding Connection Set	46
4.9	Sphere tree deformation using neighbor connection strategy	47

4.10	Tissue characterization by rheology	48
4.11	Inverse problem approach for the soft tissue parameters estimation	48
4.12	Material properties extraction from a volumetric dataset	49
4.13	Link properties estimation using voxels	50
4.14	FEM vs Blobby mesh compression	51
4.15	Local deformation on mesh reduction	52
5.1	Tissue deformation as sum of a local and a global body deformation	54
5.2	Simulation of a local deformation using metaballs	55
5.3	Real single point deformation vs fake local deformation	55
5.4	Multipoint local deformation	55
5.5	Multi-body contact	61
5.6	Multilayered local contact model	62
5.7	Global-local interaction	63
5.8	Lookup Textures	64
6.1	Fracture simulation using the breaking threshold	68
6.2	Ghost links problem on metaballs	69
6.3	Circles pattern	71
6.4	Circle splitting simulating the cut	71
6.5	Model cutting simulation using metaballs splitting	72
6.6	Splitting spheres pattern	72
6.7	Spheres pattern internal and external connections	73
6.8	Closest link optimization	76
6.9	Sphere bounding test	77
7.1	LapLab laparoscopic surgery simulator	79
7.2	Haptic tools interface	79
7.3	Extra software for the FEM and metaballs performances testing	80
7.4	DaVinci's robot simulation using metaballs	81
7.5	Sphere tree conversion tool	81

List of Figures

7.6	Real-time volumetric dataset viewer	82
7.7	Blobby models used for the benchmark	83

Bibliography

- [1] H. Delingette, "Toward Realistic Soft-Tissue Modeling in Medical Simulation". *Proc. of the IEEE*; vol.86, no 3, march 1998.
- [2] M. Cani-Gascuel, M. Desbrun, "Animation of Deformable Models Using Implicit Surfaces", *IEEE Trans. on Visualization and Computer Graphics*; vol. 3,no. 1,march 1997.
- [3] T. Nishita, E. Eihachiro, "A Method for Displaying Metaballs by using Bezier Clipping", *Computer Graphics Forum, Vol.13, No.3, pp.271-280, 1994-9.*
- [4] Blinn, FA. James, "Generalization of Algebraic Surface Drawing", *ACM Trans. on Graphics 1(3)*; July 1982, pp. 235-256.
- [5] R. Ott, D. Thalmann, F. Vexo, "Organic Shape Modeling through Haptic Devices", *Computer-Aided Design Applications*; Vol. 3.
- [6] J. Shen, D. Thalmann, "Interactive shape design using metaballs and splines", *Proc. Implicit Surfaces '95, Grenoble, France, 1995.*
- [7] WE. Lorensen, HE. Cline, "Marching cubes: A High Resolution 3D Surface Construction Algorithm". *Computer Graphics, (Proc. SIGGRAPH87) 21(4);(July), 1987, 163-169.*
- [8] H. Sammet, R. Webber, "Hierarchical Data Structures and Algorithms for Computer Graphics", *IEEE Comp. Graphics and Applications, Vol.8 No. 3 48-68.*

- [9] CB. Zilles, JK. Salisbury, "A constraint-based god-object method for haptic display", *Intelligent Robots and Systems 95*, apos; *Human Robot Interaction and Cooperative Robots*apos; *Proc. 1995 IEEE/RSJ International Conf., Volume 3, Issue , 5-9 Aug 1995 Page(s):146 -151 vol.3.*
- [10] Daniel Bielser , Pascal Glardon , Matthias Teschner , Markus Gross, "A state machine for real-time cutting of tetrahedral meshes", *Journal of Graphic Models – Special Issue on pacific graphics 2003 – volume 66 issue 6, November 2004.*
- [11] H.W. Nienhuys and A. F. van der Stappen. "Supporting cuts and finite element deformation in interactive surgery simulation", In W. Niessen and M. Viergever, editors, *Procs. Of the Fourth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'01).*
- [12] C. Forest H. Delingette N. Ayache, "Removing Tetrahedra from a Manifold Mesh", In *Computer Animation (CA'02).*
- [13] J. Shen, D. Thalmann, "Interactive shape design using metaballs and splines", *Proc. Implicit Surfaces '95, Grenoble, France, 1995.*
- [14] G. Bradshaw, C. O'Sullivan, "Sphere-Tree Construction using Dynamic Medial Axis Approximation". *Image Synthesis Group, Dept. of Computer Science, Trinity College, Dublin, Ireland.*
- [15] G. Bradshaw, "Bounding Volume Hierarchies for Level-of-Detail Collision Handling", *PhD thesis, Trinity College Dublin, Ireland.*
- [16] S. Quinlan, "Efficient distance computation between nonconvex objects", *Proc. International Conf. on Robotics and Automation 3324-3329.*
- [17] Gianluca De Novi and Claudio Melchiorri, "Surgery Simulations and Haptic Feedback: a new Approach for Local Interaction Using Implicit Surfaces", *International Conference on Applied Bionics and Biomechanics, Venice, October 2010.*
- [18] L. France, A. Angelidis, P. Meseure, M. Cani, J. Lenoir, F. Faure, C. Chaillou, "Implicit Representation of the human Intestines for Surgery Simulations", *ESAIM: Proc. November 2002, vol.12, 42-47.*
- [19] M. Lima, C. Melchiorri, G. Ruggeri, G. De Novi, T. Gargano, M. Mogiatti, G. Mazzero, "A New Robotic Platform for Endoscopic Skill Training", *International Cong. on Endoscopic Surgery, June 2009.*