UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

DANIELE FERNANDES E SILVA

A Levels-of-Precision Approach for Physics-based Soft Tissues Modeling

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Anderson Maciel
Advisor

Porto Alegre, March 2015

CIP – CATALOGING-IN-PUBLICATION

Silva, Daniele Fernandes e

A Levels-of-Precision Approach for Physics-based Soft Tissues Modeling / Daniele Fernandes e Silva. – Porto Alegre: PPGC da UFRGS, 2015.

73 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2015. Advisor: Anderson Maciel.

1. Deformable model. 2. Physics-based animation. 3. Haptic feedback. 4. Heat diffusion. 5. Minimally invasive surgery. I. Maciel, Anderson. II. Title.

”Just keep swimming. Just keep swimming. Just keep swimming, swimming, swimming. What do we do? We swim, swim.”

— DORY (FINDING NEMO)

ABSTRACT

Computational simulation of surgical environments have been widely used usually for trainings, improving essential skills and minimizing errors in surgical procedures. As these environments are always looking for a more realistic behavior, it is important to use high-precision techniques while ensuring a real-time simulation. In order to better manage this trade-off between efficiency and effectiveness, we present a hybrid and adaptive environment that combines a set of methods to achieve good accuracy and performance for a simulation. Our system merges physically deformation methods (Finite Elements Method and Mass Spring Damper) with a non-physical method that approximates the formers behavior (Green Coordinates), being able to use the appropriate method depending on the situation. To simulate an approximation of a complete surgical environment, we also implement interaction tools, such as picking, burning, and haptic feedback. Our system provides great immersion for the user, consuming less computational resources and increasing update rates.

Keywords: Deformable model, physics-based animation, haptic feedback, heat diffusion, minimally invasive surgery.

Uma Abordagem de Níveis de Precisão para Modelagem de Tecidos Moles Fisicamente baseados

RESUMO

Simulação computacional de ambientes cirúrgicos têm sido amplamente utilizados, normalmente para treinamentos, ajudando no desenvolvimento de habilidades essenciais e minimizando erros em procedimentos cirúrgicos. Para estes ambientes, é essencial a obtenção de um comportamento mais realista, sendo importante o uso de técnicas com alta precisão, além de uma simulação em tempo real. A fim de melhor controlar este trade-off entre eficiência e eficácia, apresentamos um ambiente híbrido e adaptativo que combina um conjunto de métodos para alcançar uma boa precisão e desempenho na simulação. Nosso sistema mescla métodos físicos de deformação (Método de Elementos Finitos e Mass-Mola) com um método não-físico que aproxima o comportamento dos primeiros (Green Coordinates), sendo capaz de utilizar o método apropriado dependendo da situação. Para melhor simular um ambiente cirúrgico completo, foram implementadas ferramentas adicionais para interação, permitindo pegar e manipular, queimar, e sentir os objetos do cenário. Nosso sistema proporciona grande imersão ao usuário, consumindo menos recursos computacionais e aumentando as taxas de atualização da simulação.

Palavras-chave: modelo deformável, animação baseada em física, retorno de força, difusão de calor, cirurgia minimamente invasiva.

LIST OF ABBREVIATIONS AND ACRONYMS

MIS	Minimally Invasive Surgery
DOF	Degree of Freedom
MSD	Mass-Spring Damper
FEM	Finite Element Method
PAFF	Point-Associated Finite Field
GC	Green Coordinates
RK4	Runge-Kutta 4
GUI	Graphical User Interface

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1 INTRODUCTION

The learning process in medicine, especially in surgical area, occurs by observing the surgery act and the direct participation through trial and error. In order to capacitate professionals in surgery, trainings are performed to improve skills as motor coordination, agility, and precision to handle surgical instruments of manipulation, such as scissors, retractors, and graspers.

Minimally invasive surgery (MIS) is becoming more common in hospitals. This procedure aims at maximum preservation of anatomy and minimal aggression to the organism, in order to fix the problem (WICKHAM, 1987; VECCHIO; MACFAYDEN; PALAZZO, 2000). These surgical procedures are performed through tiny incisions instead of one large opening, reducing the damage to human tissue. It normally takes longer than conventional surgery, but it includes many benefits, such as: reduced surgical scar, less pain, lower complication rates, quicker recovery, shorter hospital stay, and patient comfort.

Computational environments allow students and professionals to familiarize with a real procedure, and for training (STOYANOV et al., 2003). However, applications require a finer degree of complexity in order to appear realistic, and lots of efforts have been made in this medical application areas.

In medical applications, the internal organs of the patients are simulated as rigid and nonrigid structures. Simulations for surgical training and planning are composed of a virtual environment where the visualization and interaction are preferable to be as close as possible to real situations (SHAH; DARZI, 2001).

The physical modeling of a simulator includes the modeling of contacts between virtual instruments and soft tissues, as well as the deformation of these tissues. Each organ is simulated using different parameters of stiffness, elasticity, and viscosity attributes. Then, the soft tissues are deformed according to the desired behavior.

User interaction is often required in such applications, as it is crucial for the immersion of the user within the environment. In this way, it is needed an environment capable of performing deformations when external forces are applied on the deformable body. Also, in many cases, the object geometry is modified due to complex physical properties that compose the material, and this must be taken into account. When the interaction between an anatomical structure and a surgical instrument happens, the shape of the tissue models are updated and, depending on the nature of the device tool, a piece of tissue is removed, burned, etc.

Such applications are of great importance in minimizing errors of surgical procedures. Therefore, they should ensure a high degree of realism. Moreover, these applications can greatly benefit from haptic feedback (BASDOGAN et al., 2004). Haptic simulation, in turn, requires high update frequencies from the model, to be able to deliver a concise and

accurate feedback. Several mathematical models have been proposed in the literature, the choice of a deformation method must take into account two contradictory elements: the simulation realism and the processing cost of computing this model.

In this thesis, we present a hybrid and adaptive environment that simulates surgical procedures. Our system treats the problem of *performance vs precision* by allowing the use of different (physically) deformation models along with a fast non-physical model. In this way, the proposed system can save lots of computational resources by managing the complexity of the computations. Furthermore, our environment provides haptic feedback, and features common to surgical procedures, such as picking and heating. We show that our system indeed achieves better update rates than using a single deformation method, while keeping a realistic (and good) precision in the interactions.

1.1 Motivation and objectives

Simulation environments are widely used for training and evaluating risky situations. Surgical training is an emerging area that focuses on immersing the professional (or student) in a situation similar to a real procedure. This allows the user to develop important skills and intuition about specific situations.

When working in a simulated environment, one of the most important requirements is the accuracy of the environment. Therefore, to ensure a pleasing accuracy, it is necessary the use of complex physical models, which in turn demand high computational cost. On the other hand, using accurate models may imply in a non-real-time simulations, making the environment unable to simulate the reality properly. Thus, the main challenge is to deal with the accuracy-efficiency problem in a way that the user can experience the most realistic environment.

Usually, surgery simulations use a single deformation method to model all the tissues of the virtual body. It is usually based on some finite elements model (DELINGETTE; AYACHE, 2005). However, some tissues do not require highly precise modeling as others do. In this context, we propose an integrated methodology that brings a two-fold contribution. One goal is to create a hybrid and adaptive environment where important tissues, i.e. the ones that require more precision, can be modeled by a more complex physical model and the not so important tissues (e.g., enclosure tissues) can be modeled by a faster/more efficient method. Furthermore, when tissues do not suffer direct interaction, we propose a method that approximates the deformation with an even faster method. This allows us to focus our computational resources on the desired tissues when needed, in addition to creating a faster application with a more pleasing interaction. The decrease in the computational cost not only allows less powerful computers to run the application, but it also contributes by enabling us to simulate an increasing number of objects.

1.2 Structure of this document

The remaining of this thesis is organized as follow: in Chapter 2 we introduce some applications for virtual surgical trainers. Chapter 3 discusses methods for physical-deformation modeling, which are going to be used to describe the tissues behaviors. In Chapter 4, we explain the methods used in our application. Chapter 5 presents some features of our application. In Chapter 6 we evaluate our system in a simple and in a complex scenario, being the last one an MIS simulation. And finally, the Chapter 7 summarizes this work and points some ideas for future work.

2 VIRTUAL REALITY-BASED SURGICAL TRAINERS

Surgical simulation aims at modeling a safe environment for training and evaluation of cognitive and motor skills development for professional surgeons. To improve the quality of the patient care, surgeons are conditioned to perform specific methodology training. Advanced minimally invasive surgical (MIS) techniques are even more complex than traditional surgery. Due to that complexity, training MIS is very time and resource consuming. Training simulators brought a great help in developing and spreading these techniques. There is a special program that focuses on those training: "The Fundamentals of Laparoscopic Surgery" (FLS). This educational program was developed by the Society of American Gastrointestinal and Endoscopic Surgeons (SAGES) for teaching and evaluating cognitive, technical skills, and surgical decision-making, in a scientifically accepted format.

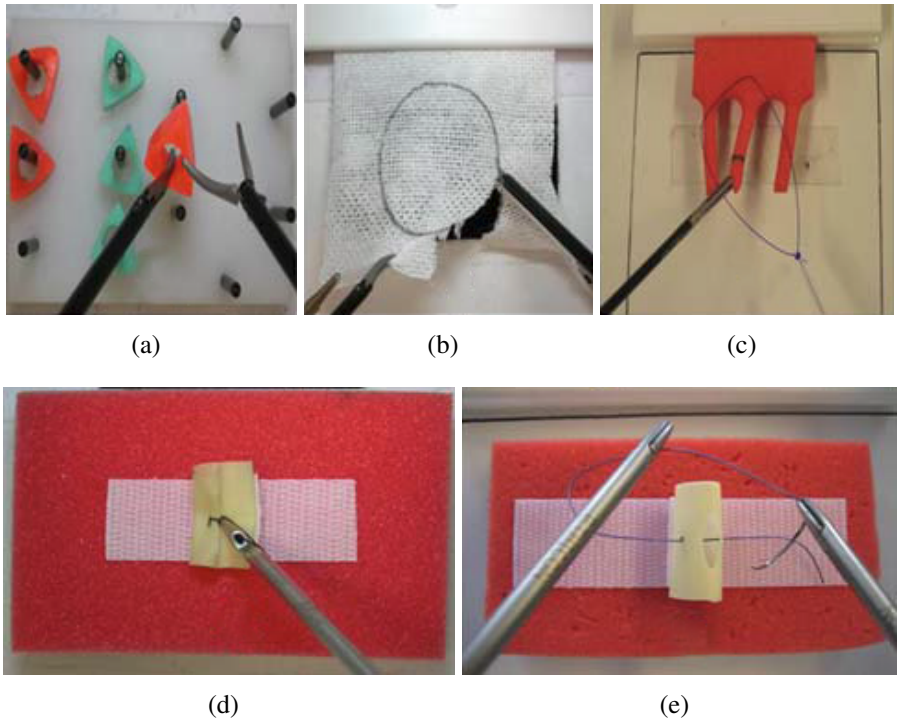
The primary goal in training laboratory is to know and learn how to manipulate the equipment. In practice, the surgeon manipulates a three-dimensional equipment, inside an enclosure environment, visualizing a two-dimensional field through a display. Training is, therefore, necessary to familiarize this change of field of vision beyond the specific material.

According to the FLS program, the surgeon has previously web-based study and skills on manual tasks (normally acquired in a box training). Thus, the goal is to manipulate the forceps and dissectors without direct visualization, using the mirrors to the dissociation between hands and eyes. Figure 2.1 shows the manual tasks that include peg transfer, pattern cutting, ligating loop, and extracorporeal and intracorporeal suturing (SCOTT et al., 2008; OKRAINEC; SMITH; AZZIE, 2009).

Virtual reality training has provided advantages over other techniques, such as box training, animals, or human cadavers. The computer simulators are able to model a full surgical procedure, calculate metrics to evaluate the training and score them, without the supervision of a specialist surgeon (CHOY; OKRAINEC, 2010; ARIKATLA et al., 2013).

Currently, several simulators are available as commercial products. However, they present several limitations, both on graphical and physical aspects of the simulation. Besides, a number of research works have been published focusing on physics-based methods for surgery simulation. Some aim specifically on the deformation problem (MOROOKA et al., 2013; SULAIMAN et al., 2013), others in the interaction with the tissue (CIRIO et al., 2011; LIM et al., 2006), and others on the visual and other aspects of organs and instruments (QUEIROS et al., 2011; PARK; WILSON; HOWARD, 2013; MARSHALL; PAYANDEH; DILL, 2005). We review some of these works in the next section.

Figure 2.1: Training skills using five tasks: (1) peg transfer: using two forceps to move a plastic object from one forcep to other. (2) pattern cutting: the operator is required to cut a circular pattern previously drawn on a gauze. (3) ligating loop: a ligating loop is placed and secured at the base of a foam appendage on a premarked line. (4) extracorporeal suture: a silk suture is placed through premarked points in a slitted penrose drain. Three knots are tied in an extracorporeal technique using a knot pusher. (5) intracorporeal suture: similar to extracorporeal suture, except that a precut smaller silk suture is used and the knots are tied using an intracorporeal technique.



(a) (b) (c)

(d) (e)

2.1 State of the Art

In a surgery context, the simulation is defined as a modulated system, where, for each task, the system uses different approaches to describing the visual and physical interaction. We present some research works which address the problems in a surgical simulator.

2.1.1 Haptic interaction

Responsible for providing the tactile sense from a virtual world, haptic interaction is the most important feature in surgery simulation, giving approximations for a real tactile response and interaction with virtual objects. A precise haptic interaction is essential for tasks such as suturing, incisions, and palpation.

Due to the variety of structures inside the human body, Cirio et al. proposed an haptic rendering to simulate different tactile sensations at the same time (CIRIO et al., 2011). Their approach uses a single model to interact with any structure, such as fluid, and deformable or rigid bodies, in a single scenario. Based on a smoothed-particle hydrodynamics, this work aims the modeling of different tissues states on the same method, defining the state of each tissue with its corresponding properties (stiffness for deformable bodies, viscosity for fluids).

(LIM et al., 2006) presented a multimodal system to simulate different contact re-

sponse between user tools and tissues of the environment. Their work describes methods to simulate palpation and cutting. They ensure a realistic force response and visual deformation by the point-associated finite field, a meshfree approach to model tissues of palpation. To the other scenario, they represent cutting by an algorithm based on node snapping.

2.1.2 Deformable bodies

In a surgery, many organs and tissues are described differently based on their own characteristic. To simulate these properties, it is necessary to model these tissues using a physical method to ensure their realistic behavior.

(DUAN et al., 2013) aim at modeling the contact forces among deformable objects for tetrahedral meshes. This ensures the preservation of volume and properties of the object. Using an implicit numerical integration scheme and appropriate constraints, a post procedure is performed to avoid the so-called "super-elastic" effect, which makes the system non realistic.

(SULAIMAN et al., 2013) presented an optimization of a physically-based model. The configuration of the parameters of the model that governs the deformable object, i.e. the liver, allows a more realistic behavior. However, the mass-spring model used is optimized in order to adjust these parameters without interfering with the stability of the system. In conclusion, this liver-tissue model is more suitable for real-time interaction with lower computational cost, being more accurate, realistic and acceptable to be used in the near future.

(MOROOKA et al., 2013) approach the problem of navigation system, in surgery context, using a FEM based analysis. Considering the tissue deformation by biomechanical behavior, their method describes the deformation by a real-time nonlinear FEM analysis using neural network. This approach uses several markers put on the surface of the tissue, which are used to generate the network and estimate the deformation, based on the position of the markers.

We have discussed some recent approaches for body deformation. Note that each system works using a single method, which in turn brings advantages and drawbacks (which we are going to cover deeply in the following chapters). Our method in the other hand, provides an environment that can take the advantages of each method, while making them to interact in a transparent way between each other.

2.1.3 Surgery interface and task training

Medical applications are usually very expensive, due to the device used for user interaction. Recent researches aim at reducing costs by using alternative ways, such as accessible sensors. Queiros et al. 2011 propose a motion tracking system, that monitors the movement performed by the specialist and assists the user/trainee in the manipulation of the 6 DOFs of the instrument. It uses a set of inexpensive sensors, such as accelerometer, gyroscope, magnetometer, and flex sensor, attached to specific laparoscopic instruments.

Park et al. approach the problem of device cost in a different way (PARK; WILSON; HOWARD, 2013). Their work uses the Microsoft Kinect 3D camera to detect the user's hand and arm. In this way, the application can correlate with the surgical instruments, designing an environment to describe a real MIS procedure.

In a suturing case, where the surgeon uses a specific tool for manipulating a string, this task presents a common problem, to model the suture line and simulate the knot by the procedure. (MARSHALL; PAYANDEH; DILL, 2005) present a simulator system

that assists residents in the development of skills, like manipulating surgical graspers and interacting with deformable tissue. A set of algorithms is used to solve problems of the deformation method, collision detection, and haptic feedback.

Currently, there are a some surgical applications being commercialized on the market, such as LapMentor (SIMBIONIX, 2014), LapSim (SCIENCE, 2014), and daVinci (INTUITIVE SURGICAL, 2014). All these private applications assist novice surgeons to develop and train their skills for a real situation.

2.2 Discussion

The major goal in medical applications is to achieve realistic appearance in real time. However, precise modeling methods imply high-computational cost. Surgery simulation carries the problems previously discussed, such as choosing the physically-based method to model behavior for each organ, collision detection, interaction between tissues and with a external force (e.g., applied by an user), and haptic feedback to ensure tactile sense. Each of these features reduces the application performance, when it demands more and more realism. Thus, some features are put aside in order to obtain lower computational cost.

Haptic feedback requires high frequency update, which is not achievable in more complex applications. For this reason, most of the commercially available applications do not include this feature, using only stable models to simulate the behavior of different tissues from the human body.

3 STUDY OF SOFT TISSUES MODELING

Animated virtual objects can behave in several ways such as rigid movement, resizing, or in soft bodies, deformation. Models based on physical deformation are important to create virtual representations of deformable solid structures realistically. In many applications where the user interacts with the virtual world, it is necessary that those objects behave with a high degree of similarity to the real world. We now briefly discuss the most used methods for model deformation.

3.1 State of the art

In this Section, we present a study that briefly discuss physically-based methods for deformation, but focus deeply on discussing about methods used for numerical integration. These numerical-integration methods dictate how fast a deformation method can compute each iteration, as well as how robust is each one.

3.1.1 Physically-based methods

3.1.1.1 Mass-spring damper

The Mass-Spring Damper (MSD) is described as an ordinary differential equation system which uses mathematical methods, explicit or implicit, to solve numerical approximations (PROVOT, 1996; GIBSON; MIRTICH, 1997; BOURGUIGNON; CANI, 2000; GEORGII; WESTERMANN, 2005; SELLE; LENTINE; FEDKIW, 2008; MESIT; GUHA; FURLONG, 2010). This method uses a discretized object and, in order to simulate the deformation behavior, uses motion equations that are built around the masses points and the springs of the model. In this model, a mass is the information of each nodal point, storing the position, velocity, and acceleration values of each one during the time steps of the simulation. The springs are the connections between neighbor nodal points, each spring also contains its stiffness value and rest size value.

3.1.1.2 Finite Elements Method

Different from the MSD, the Finite Element Method (FEM) works on the continuum mechanics to approximate interpolation functions (BATHE, 1982; JOHNSON, 1987; CHEN; ZELTZER, 1992; BRO-NIELSEN; COTIN, 1996; YIN et al., 2009; KAUFMANN et al., 2009a; BECKER; IHMSEN; TESCHNER, 2009; KAUFMANN et al., 2009b). In order to use an irregular grid for simulation of deformable objects, the object is defined as a set of finite elements consisting of a set of nodes. The method computes a function for each element, such function satisfy an equilibrium equation.

The advantage of FEM is that complicated geometry, general boundary conditions,

and variable or non-linear material properties, can be handled relatively easily. However, such method has a higher computational cost than MSD, making it unfeasible when time is a concern.

3.1.1.3 Point-Associated Finite Field

The Point-Associated Finite Field (PAFF) approach works on the partial differential equation system based on continuum mechanics (LIM; DE, 2005; DE et al., 2006; LIM et al., 2006). The object is discretized using a scattered distribution of nodal points. Each node is a center of a specific "region of influence", which supports and divides the model into spherical subdomains.

PAFF is a more recent method, being more efficient than FEM for distribution of forces to the object. While it leads to good results in real time, including haptic feedback, it is difficult to be extended to global deformations.

3.1.2 Numerical Integration

Methods for numerical integration are normally solved using an approximation of an analytic solution using differential equations. These equations must be solved iteratively for the configuration of the system in each time step.

The approximation algorithms can be classified as explicit or implicit methods. The former ones use an equation depending on the current configuration of the system, while the latter, in addition to taking into account the current configuration, also use the computed information of the next configuration in its computation.

As previously described, numerical integration methods need to perform a computation on each time step. The size of the step is proportional to the error of the approximation. Implicit methods are more complex (mathematically), but they provide more stability to the system, allowing the use of greater time steps. On the other hand, explicit methods are simpler, providing an easier implementation and faster performance. However, explicit methods lead to greater errors, implying in an instable system that requires a smaller time step.

We now discuss some methods for solving ordinary differential equations, evaluating them in order to choose the most appropriate for our purpose.

3.1.2.1 Euler

Using the time steps as a discrete interval of integration, this method performs the approximation as follows: given the value of the current approximation y_n, the value of the next iteration y_{n+1} is computed using Taylor series (Eq. (3.1)). This is known as the Euler method, computing the derivative functions for a single point.

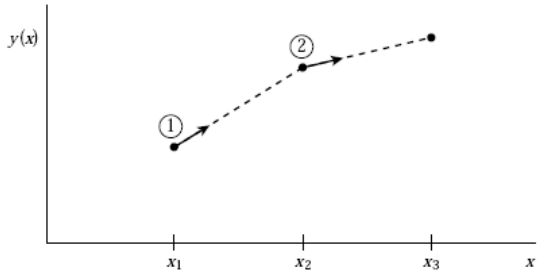
$$y_{n+1} = y_n + \frac{h_n}{1!}y'_n + \frac{h_n^2}{2!}y''_n + ... \tag{3.1}$$

We can truncate the series for faster computations. Eq. (3.2) shows a truncated version, being $f(x_n, y_n)$ the derivative function of y_n. The Figure 3.1 show how the Euler Method computes the value of the next iteration.

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{3.2}$$

The same notation can be used in its implicit form (Eq. (3.3)). In this way, we can solve the equation for y_{n+1} using the derivative of the function $n+1$. This leads to longer

Figure 3.1: Euler method computing the next iteration. The x axis represents the iterations, and the y axis represents the value of the function. When using the Euler method, we can compute the value of x_2 by using just the value and derivatives of x_1.



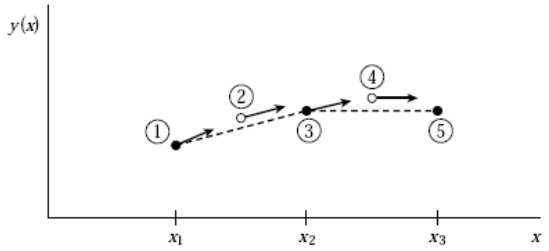
computational time, but it gives more stability to the system, allowing the use of greater time steps.

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \tag{3.3}$$

3.1.2.2 Runge-Kutta

The Runge-Kutta method is an extension of the Euler's method. This method computes approximations in intermediate steps, the number of approximations depends on the order of the method, higher order yields greater precision. Figure 3.2 shows the 2nd-order Runge-Kutta method (RK2). This particular version is also known as the midpoint method, which uses the derivative of the mid point $(1/2(y_n + y_{n+1}))$ to compute the next value.

Figure 3.2: Approximation using Runge-Kutta method. On x axis represents the iterations, and the y axis the value of the function.



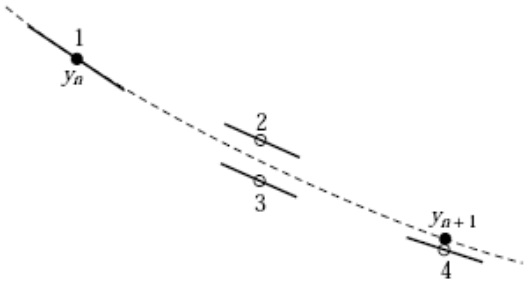
The most common order is the 4th, often referred as RK4, or classical Runge-Kutta method (Figure 3.3). The computation of y_{n+1} is given by the following equations:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3) \end{aligned}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5) \tag{3.4}$$

We can easily note that this computation is more complex than the 2nd order version. In fact, RK4 presents a good trade-off between computation time and precision of the approximation, being widely used in many areas. Moreover, with better approximations, we can use higher step sizes.

Figure 3.3: Approximation using 4th order Runge-Kutta method.



3.1.2.3 Verlet

Another method for numeric integration is the one proposed by Verlet (VERLET, 1967). Using until the 3rd term of the Taylor series, this method sums up all the previous and posterior steps of the system:

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2}y''_n + O(h^3) \tag{3.5}$$

$$y'_{n+1} = y'_n + hy''_n + O(h^2) \tag{3.6}$$

$$y_{n-1} = y_n - hy'_n + \frac{h^2}{2}y''_n + O(h^3) \tag{3.7}$$

The equations Eq. (3.7) and Eq. (3.5) are then used to compute the approximation of the next iteration:

$$y_{n+1} + y_{n-1} = 2y_n + h^2y''_n$$

$$y_{n+1} = 2y_n - y_{n-1} + h^2y''_n + O(h^2) \tag{3.8}$$

This simpler integration computes the next iteration without using the first derivate, or what they call "velocity". There are some variations of this method that computes the (so called) Velocity Verlet to take into account the derivative information.

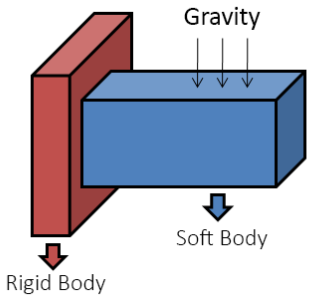
3.2 A study of techniques applied on Mass-Spring Damper

In this section we present results using different techniques to model deformation with MSD. The first test scenario uses a simple MSD implementation. We are going to compare this implementation with different methods for numeric integration.

In the second test scenario, we are going to use a MSD implementation of three physical engines. With this, we can compare the differences between the configurations parameters, understanding the implications of each one in the final result.

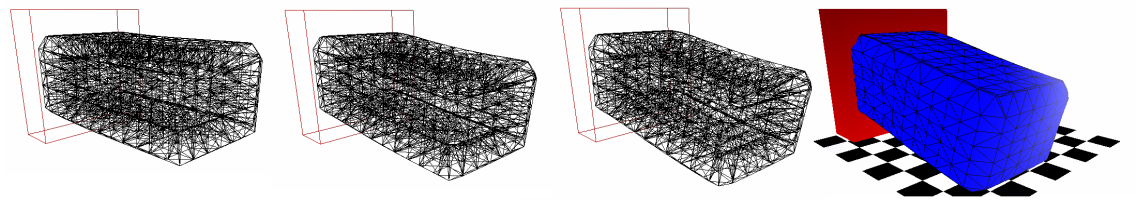
In the tests, we have used a simple beam object. We have used a tool for soft body simulation (PhysX Viewer) to create a mesh with 604 vertices, 1750 tetrahedral, and 728 links.

Figure 3.4: The test scenario is composed by a bar attached to the wall. The gravity is the only external force acting over the body.



The scenarios consist of a deformable object attached to a fixed wall (Figure 3.4); the gravity will deform the object depending on its characteristics (Figure 3.5). We have tested both implementations of the MSD (the simplest and the implementation by physical engines) using different numerical-integration methods. We have also analyzed the deformation of the object depending on the parameters used.

Figure 3.5: Timeline of the bar deformation, using the simplest implementation.



3.2.1 Results

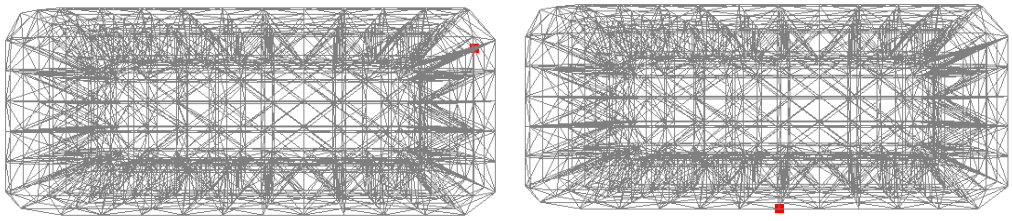
The simulation is very simple, working using the principle of Hooke's law. In the simulation, we have tested three explicit models for numeric integration: Euler, Verlet, and Runge-Kutta 4. In order to evaluate the stability of each numeric-integration method, we have analyzed the characteristics of each simulation while verifying the integrity of the system.

We performed the analysis by evaluating oscillations of position values between two distinct points (Figure 3.6). We also take into account three parameters: stiffness, damping, and size of the time step; evaluating how changing them can affect the deformation. It also provides us information about how the parameters of the MSD can be used to simulate different properties in an object.

3.2.2 Changing of the stiffness parameter

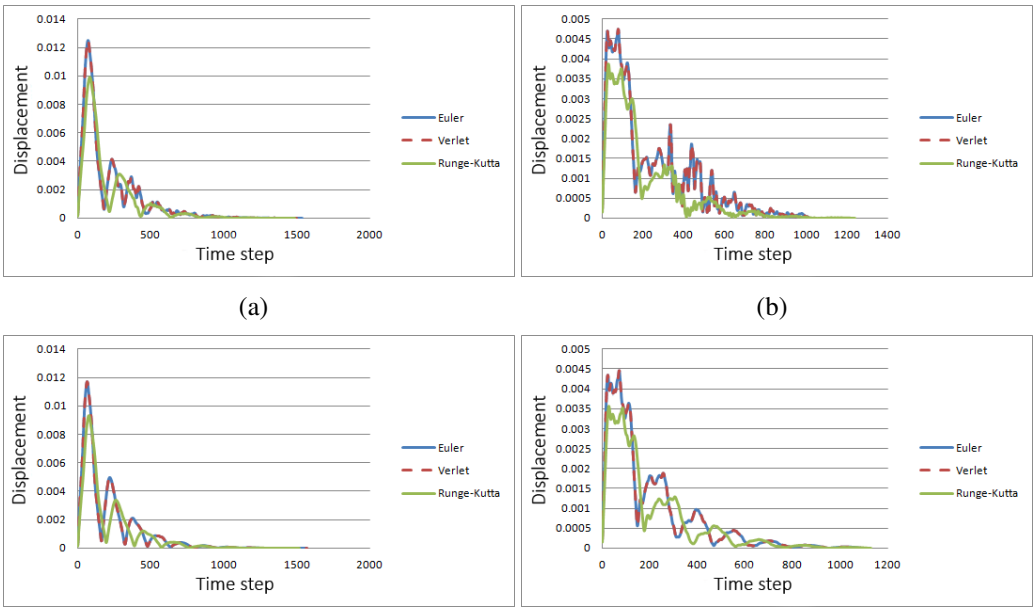
Figure 3.7 shows the results when we change the stiffness of the MSD model. On the left we show the results for the first point (Figure 3.7(a)), while in the right we show the results of the second point (Figure 3.7(b)). In Figure 3.7(top) we have set the stiffness to 250, while in the bottom we have used the value 300.

Figure 3.6: Sampled points used to evaluate the deformation during the test.



(a) (b)

Figure 3.7: Results of different stiffness values. In (left), the displacement of the first point, and in the right, the displacement of the second point. Parameter values: mass= 0.1, damping=0.01, time step=0.0048. In (a) and (b) the stiffness=250, and (c) and (d) stiffness=300.



(a) (b)

(c) (d)

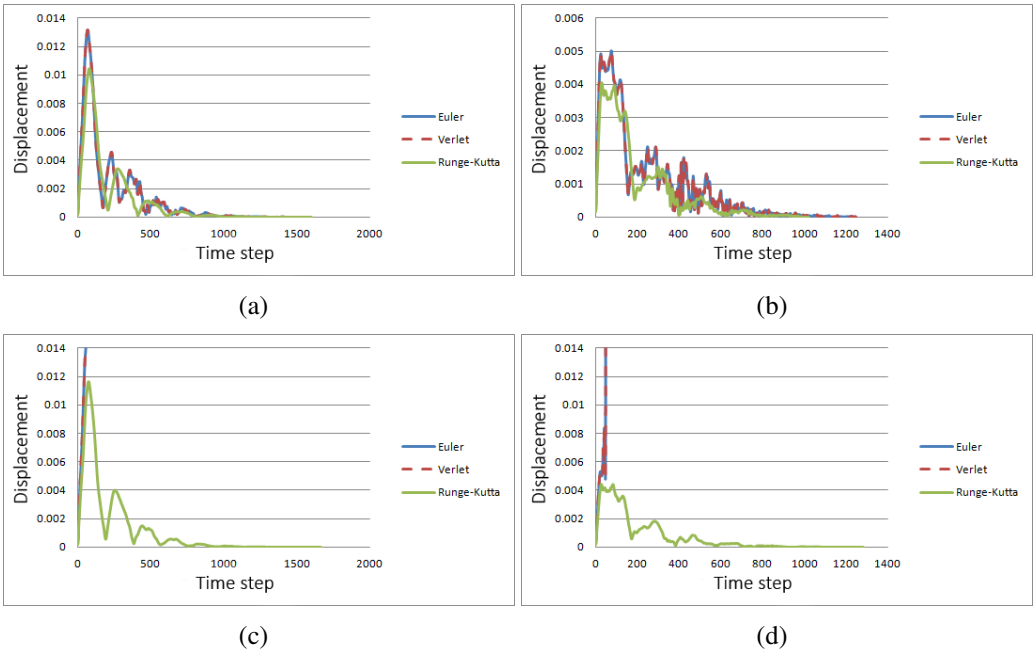
We have noted that increasing the stiffness implies in decreasing the displacement. This was indeed expected, as the stiffness represents the rigidity of an object, i.e., how much it resists deformation is response to a force (WENHAM, 2001). Also, there is a variation in the approximation for the methods Euler and Verlet, which is lower in the RK4 method.

3.2.3 Changing of the size of time step parameter

Figure 3.8 shows the result of changing the size of time step. Again, left is the displacement of the first point, and right the second point. In Figure 3.8(a) and Figure 3.8(b) the time step is set to 0.0050, while in Figure 3.8(c) and Figure 3.8(d) the value is 0.0054.

The size of the time step is greatly related to the speed of convergence. However, greater steps imply into larger approximation errors. This error can compromise the entire system, as shown for the methods of Euler and Verlet. On the other side, the RK4 method can get a good approximation even for big sizes of time steps.

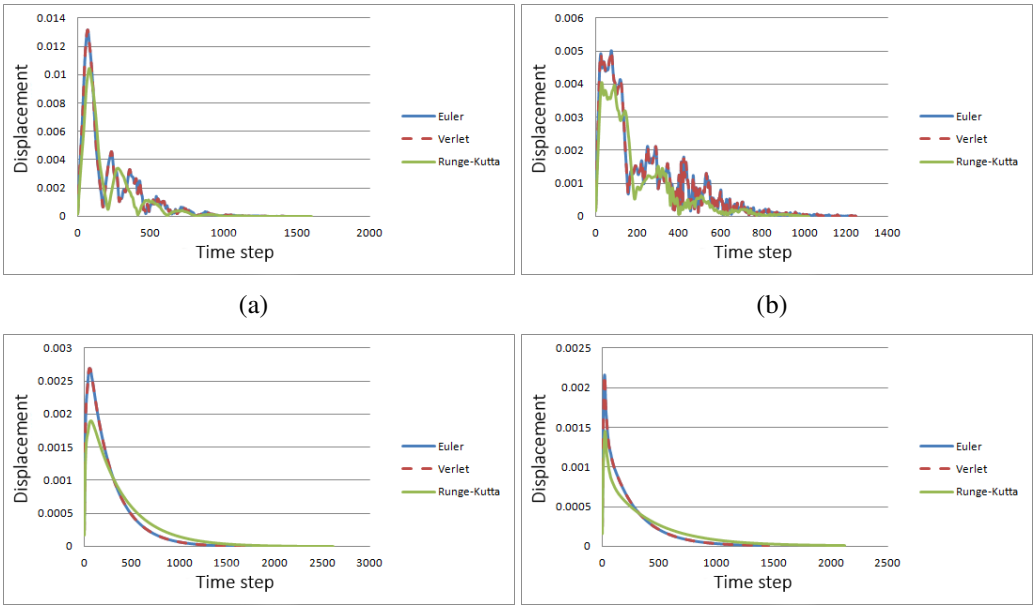
Figure 3.8: Displacements values of the first point (left figures) and second point (right figures) while changing the size of the time step. Parameter values: mass=0.1, stiffness=250, damping=0.01. In (a) and (b) the step=0.005, and (c) and (d) step=0.0054.



(a) (b)

(c) (d)

Figure 3.9: Displacements of the first and second points (left and right figures, respectively) while changing the damping value. Parameter values: mass=0.1, stiffness=250, step=0.005. In Figure 3.9(a) and Figure 3.9(b) damping=0.01, and Figure 3.9(c) and Figure 3.9(d) damping=0.1.



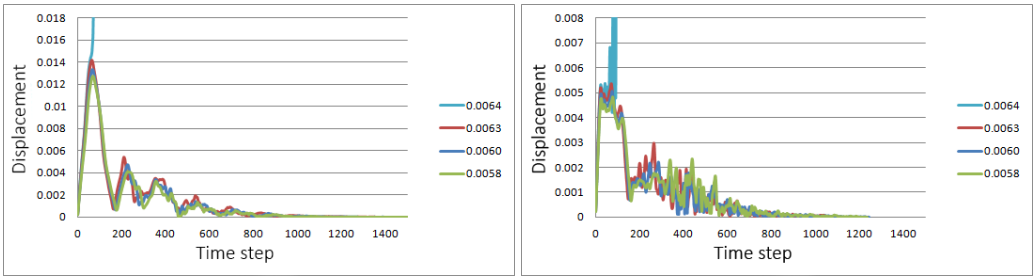
(a) (b)

(c) (d)

3.2.4 Changing of the damping parameter

Figure 3.9 shows the displacement of the first (left) and second (right) points while changing the damping parameter (which is related to how an oscillatory system dissipate

Figure 3.10: Displacements of the first and second points (left and right figures, respectively) while changing the damping value. Parameter values: mass=0.1, stiffness=250, damping=0.01. The time step changes to evaluate the RK4 stability.



(a) (b)

the energy). In Figure 3.9(a) and Figure 3.9(b) we have used the value 0.001, while in Figure 3.9(c) and Figure 3.9(d) the damping was set to 0.1. We can easily note that an increase in the damping will disperse more energy, reducing the number and size of the oscillations.

3.2.5 Analyzing the RK4 method

In Figure 3.10 we have used the same parameters as Figure 3.8, but we have increased the size of time step. Note that even for greater values of time step, the RK4 method still converges, being much more robust than the other integration methods.

3.2.6 Cost of each time step simulation

Figure 3.11 shows the time (in seconds) to compute a single time step of the simulation using each integration method. Note how the times of the methods of Euler and Verlet are very close, being Euler the fastest one. We can note how the RK4 is much slower due to its complex computation.

3.2.7 Comparison between MSD implementations using physical engines

For this second scenario, we have used the same geometric model as the last one. In this trial, we have evaluated the simplest implementation and other engines implementations (PhysX, Bullet, and SOFA). The simulations had different behaviors due to some properties and characteristics of each one.

Figure 3.11: Performance cost (in seconds) for each numerical integration method.

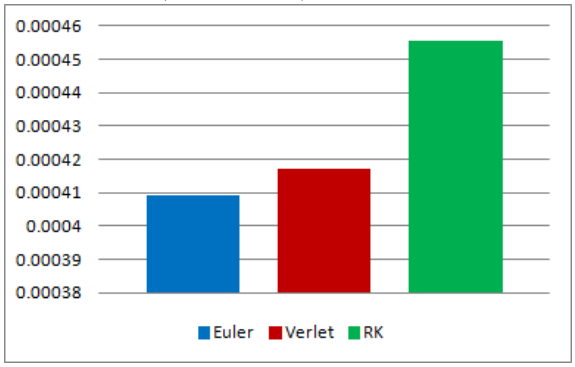
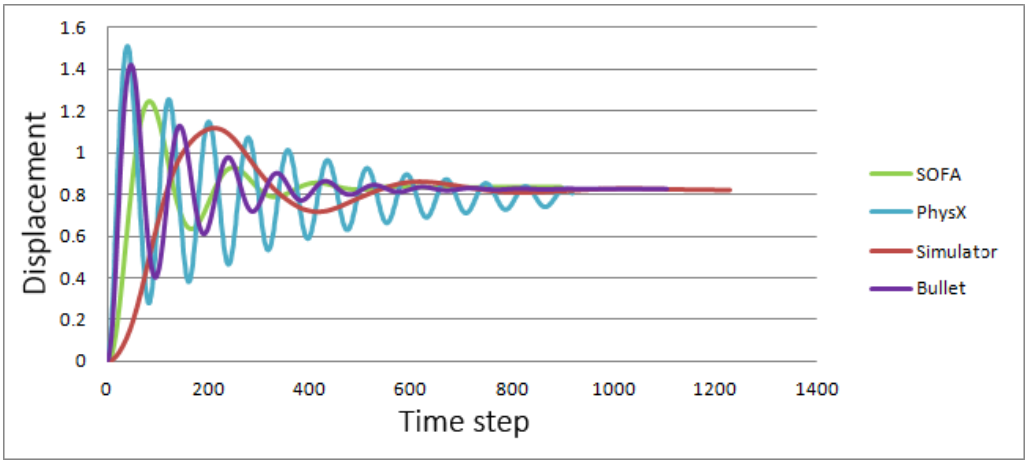


Figure 3.12: Comparison between MSD in different engines. Different values are used for each parameter to submit a similar deformation. The parameters on Figure 3.12(b) shows the deformation behavior on each physical engine (Figure 3.12(a)).



(a)

	Simulador	Bullet	PhysX	SOFA
Mass	0,1	0,1	1	0,3
K	250	0,68	0,975	120
Damping	0,01	0,01	0,01	0,01
Timestep	0,005	1	1	0,1

(b)

The parameters of the model were chosen to make the position of the points to match (as close as possible) at the end of the simulation. This was done by trial and error.

The mass is normally set in kilograms (*Kg*), but in the PhysX, the attribution of the mass is computed by the engine. In PhysX, the user must provide the density of the object, for which the value of mass is computed based on the object volume.

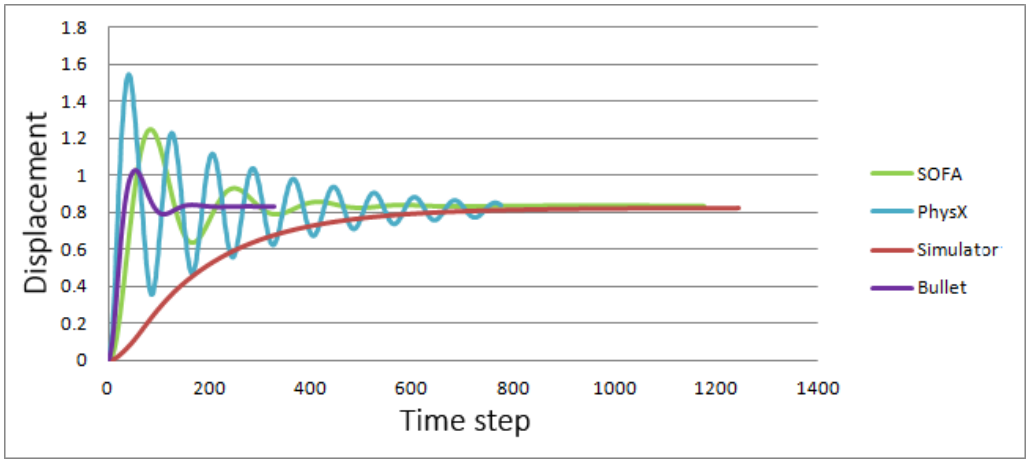
The stiffness, which affects the deformation of the springs, is measured in *N/m* for the simplest and SOFA's implementations. The engines Bullet and PhysX use a mapping from 0, the least rigid, to 1, the most rigid for stiffness values.

The damping, which is responsible for dissipating energy, is normally defined in the interval $[0, 1]$, 0 for total conservation of energy, and 1 for maximum dissipation. The size of time step of the simplest implementation is a fixed value for the entire simulation, while the engines use different scales for the time step during the same simulation.

When evaluating the deformation, all the parameters had greater impact on the result, even for small changes in their values. The stiffness not only encodes if the object is going to have huge or small deformations, but it also affects the verification of when a deformation changes the nodes, as well as the length of the spring when in the rest position. The change in the mass of the model has little influence in the engines PhysX and Bullet, because both take into account the density of the model when specifying the normalized value of the stiffness.

We have two assumptions to explain the oscillations shown in Figures 3.12(a), 3.13(a), and 3.14(a). The first one is that this feature was originated by approximation errors, which bring little peaks of deformation, resulting in a non-realistic behavior of the model. However, physic engines ensure a pleasing realism in the simulation by using sophisti-

Figure 3.13: Comparison between the implementation of MSD in different engines using different damping value. Using the same parameter on Figure 3.12(b), except by the damping. The Figure 3.13(a) shows the deformation behavior from settings on Figure 3.13(b).



(a)

	Simulador	Bullet	PhysX	SOFA
Mass	0,1	0,1	1	0,3
K	250	0,68	0,975	120
Damping	0,05	0,05	0,05	0,05
Timestep	0,005	1	1	0,1

(b)

cated methods of integration and controlling the time steps to circumvent the error approximation. In the second assumption, the oscillation is due deformations generated by other forces that act on the object, such as: pressure, strength, tension, air resistance, and other properties. The described oscillation is dissipated according the amount of damping present in the system.

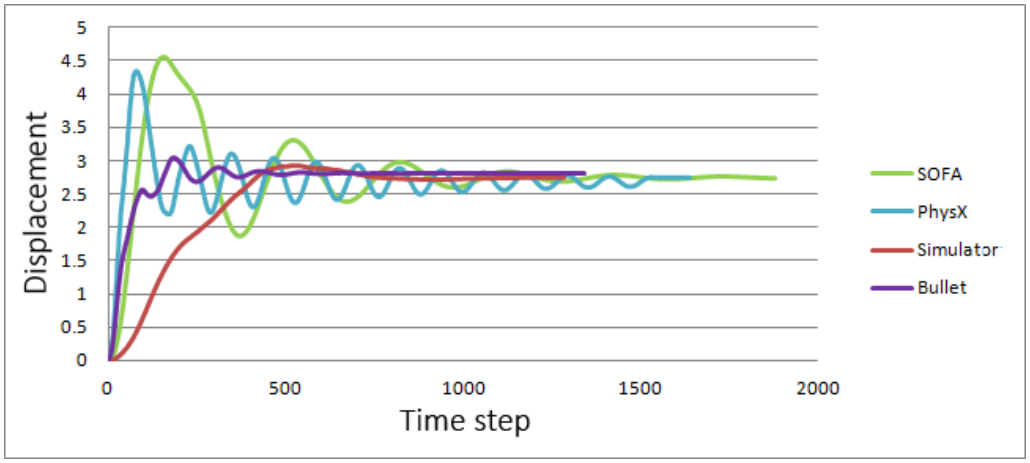
The relation between the numerical integration method and the convergence of deformation is also another point. All parameters are considered: stiffness, damping, mass, and time step. We can observe that the system convergence requires fewer iterations as we increase the size of time step. However, large time steps can generate high approximation error. The use of implicit methods can be the key point for a quick system convergence. Even though they have higher computational cost, implicit methods have better results than explicit ones. The engines use control of time step size to make the simulation plausible, avoiding oscillations caused by approximation errors.

3.3 Summary

The main goal of this chapter was the understanding and implementation of the MSD model, while evaluating and discussing the physical parameters of the model, as well as the methods for numeric integration. We have noticed that the MSD model is very simple and fast, but some disadvantages come with it.

As the MSD does not have complex mathematical computations, it turns that it is not very accurate when compared to other methods (like FEM, for example). When using the

Figure 3.14: Comparison between the implementation of MSD in different engines using different stiffness value. Using the same parameter on Figure 3.12(b), except by the stiffness (K). The Figure 3.14(a) shows the deformation behavior from settings on Figure 3.14(b).



(a)

	Simulador	Bullet	PhysX	SOFA
Mass	0,1	0,1	1	0,3
K	120	0,58	0,858	36,18
Damping	0,01	0,01	0,01	0,01
Timestep	0,005	1	1	0,1

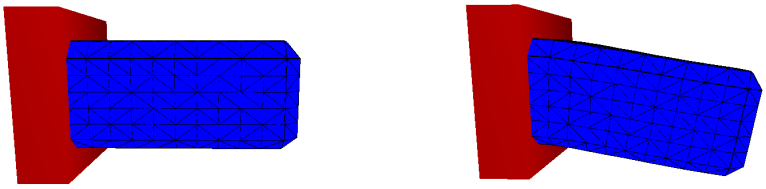
(b)

MSD, the parameters must be exhaustively tested in order to guarantee the stability of the system.

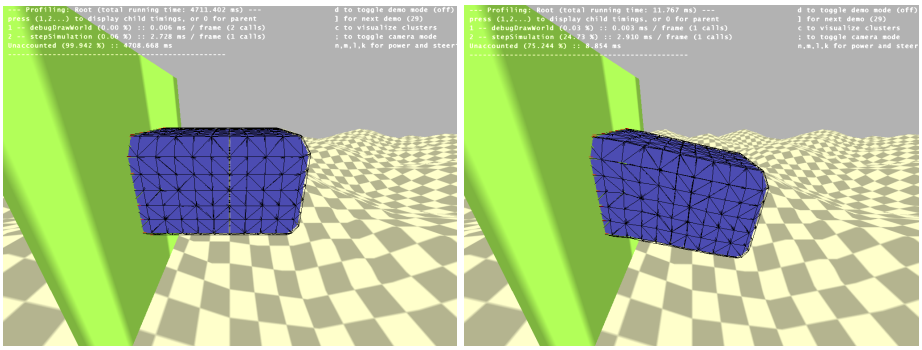
This chapter has presented one of the simplest approach used in non-rigid bodies. The implementation of a system capable of achieving high degree of realism requires more sophisticated mathematics and physical models.

In order to ensure a stable system, we must also choose the integration method carefully. As our scenarios have shown, RK4 was the method that yields better approximations even with large time steps, despite being the one with higher computational cost. Euler and Verlet can achieve better performance, but their robustness is very low when compared with RK4.

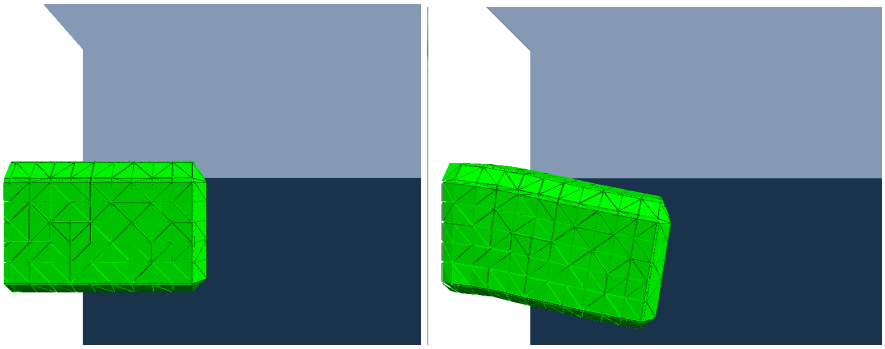
Figure 3.15: Visualization of deformation for each physical engine used. (a) Simple implementation. (b) Bullet. (c) PhysX. (d) SOFA. In left the initialized position, and right the final position, using the settings parameter on Figure 3.12(b).



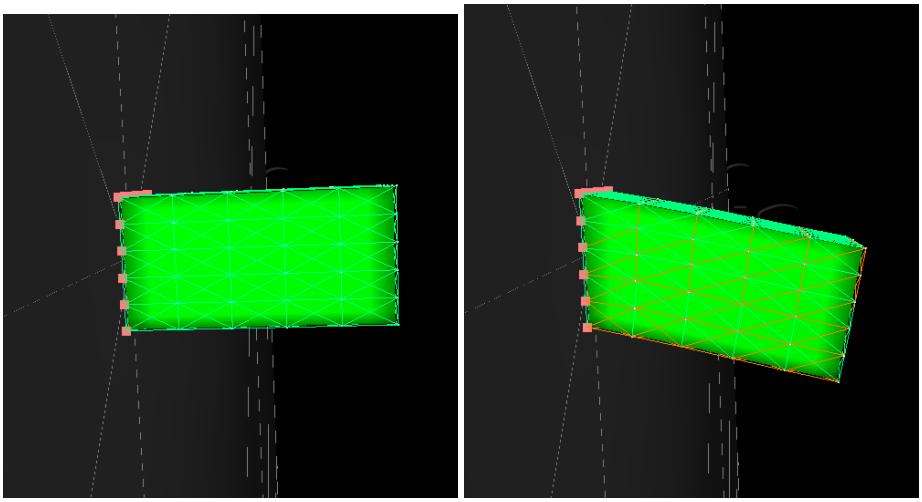
(a)



(b)



(c)



(d)

4 OUR LEVELS-OF-PRECISION DEFORMATION APPROACH

We propose an approach to minimize the computational cost in a surgical-procedure simulation. The organs are modeled using different deformation models, one method that provides greater accuracy and another providing better performance. In this way, when simulating a surgical procedure, we can use a deformation model according to the necessity of accuracy. For example, organs that are more important for the surgery than others will be modeled by a more precise method, while we can use a faster one for the others.

This hybrid system also uses levels of precision considering the user interaction, i.e., if the user is interacting directly with a specific organ, it will be modeled by a more accurate physical system. This model will be used until the deformation converges, and then a simpler method assumes the task.

We first describe each modeling method used. The Mass-Spring Damper (MSD), the one used in the experiments in the previous chapter, is going to be mathematically explained. After that, a more precise method is discussed: The Finite Elements Method (FEM). Third, we present a non-physically deformation method, which we used as an approximation for organs (objects) which are not receiving direct interaction. In the final section we present the main contribution of this work, which is to use different deformation methods for different moments of the simulation according to the type and amount user interaction.

4.1 Mass-Spring

The Mass-Spring Damper (MSD) is a physical model used when one desires good performance. Its mathematical simplicity eases its implementation, in addition to allowing real-time execution given its low-cost computational time.

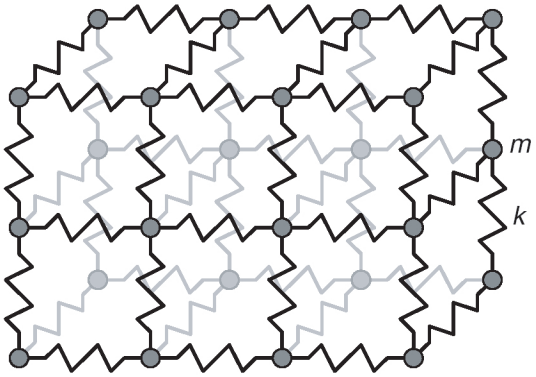
In order to perform the deformation, the MSD model uses an object discretized into vertices and links (Figure 4.1). Each vertex has a mass and stores values of position, velocity, and acceleration. The connection between those vertices are called springs, which in turn have properties as stiffness and standard length (size in the rest state). The idea is to find a solution for the equations, computing the displacement for each vertex during the deformation.

Each spring in the system follows the elasticity concept, using the Hooke's law (Eq. (4.1)) to ensure an equilibrium point between the two masses it connects:

$$F = -kx \tag{4.1}$$

The Newton's Second Law (Eq. (4.2)) is used on each mass of the object for comput-

Figure 4.1: Discretized object, composed by a set of vertices and connected by springs (GIBSON; MIRTICH, 1997).



ing the deformation movement.

$$m_i \ddot{x}_i = -\gamma_i \dot{x}_i + \sum_j g_{ij} + f_i \tag{4.2}$$

In this equation, m_i is the mass of the point i, and x_i is its position. The right side encodes the forces applied to the masses. The first term $-\gamma_i \dot{x}_i$ is the damping force, which depends on the velocity of the point. The second term ($\sum_j g_{ij}$) is the contribution of the forces of the springs that are connected to the point i, and f_i are the external forces.

In order to inhibit the idleness of the springs, a damping modification was added to the system. This adjustment affects directly the velocity of the displacement of the masses, avoiding long time convergences and reducing the amplitude and number of oscillations of the system.

$$M\ddot{x} + D\dot{x} + Kx = f \tag{4.3}$$

Gibson describes this system as a junction of the equations of displacements of all the masses (GIBSON; MIRTICH, 1997). In Eq. (4.3), the matrices M, D, and K are respectively the matrices of mass, damping, and stiffness. This results in a system of first order differentiable equations, with time being the independent variable. We then can use numerical integration methods (such as RK4) to solve the system and compute the displacement values for each mass.

4.2 Finite Elements

We now review other technique that can be used as a deformation model: the finite element method (FEM). The FEM is a numerical technique used for finding approximate solutions for partial differential equations with valued boundary constraints.

The elastic domain on solid mechanics is represented by a variational approach. On calculus of variations continuum, functions are defined as functions of functions, i.e. functionals. To solid mechanics, the functional is expressed by a physical principle, such as potential energy, a complementary energy of Reissner's principle, which models a specific problem as described in Table 4.1.

We now discuss a potential energy principle to model the deformation problem. The potential functional $\Pi(u)$ is described in terms of energy planes. To formulate the finite

Table 4.1: Classification of the finite element method for solid mechanics problems (HUEBNER, 1983).

Model	Variational Principle	Inside Each Element	Along Interelement Boundary	Unknown in Final Equations
Compatible	Minimum potential energy	Continuous displacements	Displacement compatibility	Nodal displacements
Equilibrium	Minimum complementary energy	Continuous and equilibrating stresses	Equilibrium boundary tractions	Stress parameters
Hybrid 1	Modified complementary energy	Continuous and equilibrating stresses	Assumed compatible displacements	Nodal displacement
Hybrid 2	Modified potential energy	Continuous displacements	Assumed equilibrating boundary tractions	Displacement parameters and boundary forces
Hybrid 3	Modified potential energy	Continuous displacement	Assumed boundary tractions for each element and assumed boundary displacements	Nodal displacement
Mixed (Plate-blending problems)	Reissner's principle	Continuous stresses and displacements	Combinations of boundary displacements and tractions	Combination of boundary displacements and tractions

element method, the domain must be subdivided. As we are working in a volumetric three-dimensional case, we have used tetrahedral polygons for the domain discretization. In other words, the element is going to be tetrahedral polygons.

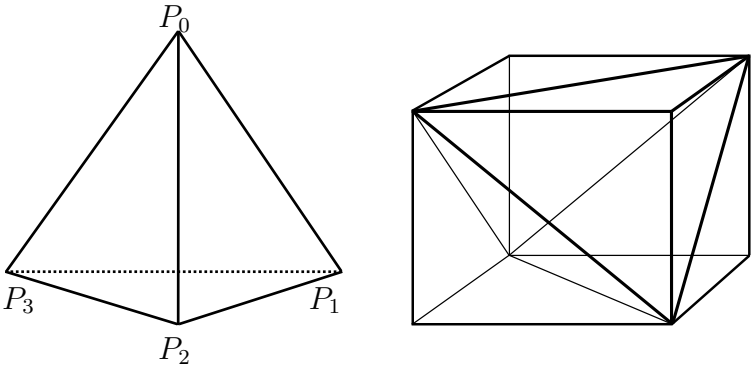
In the finite-element method, each point in the mesh is going to hold the information of three displacement components: $u(x, y, z)$, $v(x, y, z)$ and $w(x, y, z)$. These components define the direction of the coordinates x, y, and z, respectively, and are described as a linear combination of the shape function.

The shape function, or basis function, (N_i) is an interpolation function which describes the behavior between the link connections for each node i on the tetrahedron element. This function is locally defined, and it is assembled to a global function. Thus, for a given tetrahedron, the shape function for each node i is given by Equation 4.4.

$$N_i(x, y, z) = a_i + b_i x + c_i y + d_i z \qquad \text{for} \qquad i = 0, 1, 2, 3 \tag{4.4}$$

Thus, we can compute the coefficients a_i, b_i, c_i and d_i, for the shape function $N_0(x, y, z)$, which is going to take the value 1 for the node $i = 0$, and 0 for all other node. The following equations describe the system for N_0:

Figure 4.2: A tetrahedron primitive is used to discretization of solid elastic domain into finite elements.



$$\begin{aligned} a_0 + b_0 x_0 + c_0 y_0 + d_0 z_0 &= 1 \\ a_0 + b_0 x_1 + c_0 y_1 + d_0 z_1 &= 0 \\ a_0 + b_0 x_2 + c_0 y_2 + d_0 z_2 &= 0 \\ a_0 + b_0 x_3 + c_0 y_3 + d_0 z_3 &= 0 \end{aligned} \tag{4.5}$$

By solving this equation system, we can compute the four unknown coefficients a_0, b_0, c_0 and d_0 for the shape function $N_0(x, y, z)$. Analogously, we can compute a_1, b_1, c_1, and d_1 by solving the system equation of $N_1(x, y, z)$, which will take the value 1 for the node $i = 1$, and 0 for all others. Those coefficients are used to compute the strain energy, which is described as Equation 4.6.

$$\Pi(u) = \frac{1}{2} \iiint_{\Omega} \delta^T \sigma \mathrm{d}x \tag{4.6}$$

As stated earlier, the finite element method is modeled on a variational principle which uses potential energy. The energy functional is composed by all the energies that act over the modeled domain. In our case, the functional is defined by the energies of strain δ and stress σ as shown in Equation 4.7.

$$\overrightarrow{\delta} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial w}{\partial z} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \end{bmatrix} \tag{4.7}$$

The strain components are the first derivative values of the displacement vector. This can be written as $\delta = Bu$, where u is the displacement vector and B is the derivative matrix. The coefficients of shape functions computed in Equation 4.4 are used to define matrix B showed in Equation 4.8, where $i = 0, 1, 2, 3$.

$$B = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} = \begin{bmatrix} b_i & 0 & 0 \\ 0 & c_i & 0 \\ 0 & 0 & d_i \\ c_i & b_i & 0 \\ 0 & d_i & c_i \\ d_i & 0 & b_i \end{bmatrix} \tag{4.8}$$

By the Hooke's law, the stress energy is defined by the strain vector as $\sigma = C\delta$, where C is a symmetric material stiffness matrix (Eq. (4.9)). The constants on Eq. (4.9) refer to lamé material and they are computed by known measures of the material elasticity, the Young's modulus E, and Poisson's ratio σ as in Equations 4.10

$$C = \begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{4.9}$$

$$\lambda = \frac{Ev}{(1+v)+(1-2v)} \qquad \mu = \frac{E}{2(1+v)} \tag{4.10}$$

Thus, the functional energy, described by strain and stress energies, is rewritten by adding the contribution of external forces f (Eq. (4.11)).

$$\Pi(u) = \frac{1}{2} \iiint_{\Omega} u^T B^T C B \mathrm{d}x - \iint_{\Gamma_c} f T u \mathrm{d}V \tag{4.11}$$

$$\Pi(u) = \sum_{e=1}^{M} \Pi^e(u) \tag{4.12}$$

To achieve the equilibrium of the system, the potential energy takes the minimum value. This happens when the first variation of the functional vanishes:

$$\delta\Pi(u) = \sum_{e=1}^{M} \delta\Pi^e(u) = 0 \tag{4.13}$$

In this case, the equation can be written as Eq. (4.14), and the local stiffness matrix K^e can be computed Eq. (4.15).

$$0 = \iiint_{\Omega^e} B^{eT} C B^e u^{eT} \mathrm{d}V - f^e \tag{4.14}$$

$$K^e = B^{eT} C B^e V^e \tag{4.15}$$

After obtaining the stiffness matrix of each element, we must assemble those local matrices in a global stiffness matrix, as showed in Eq. (4.16), this results in our linear system (Eq. (4.17)) . The K matrix is symmetric, sparse, diagonally dominant, and singular. Due to those characteristics, by applying the minimum boundary conditions to the matrix (solving the singularity problem), our linear system returns a single solution.

$$K = \sum_{e=0}^{i} K^e \tag{4.16}$$

$$Ku = f \tag{4.17}$$

The Dirichlet boundary conditions, also referred as fixed boundary conditions, are used to define absolute values on the boundaries of the domain. To apply those conditions in the linear system, we take the global stiffness matrix assembled before, along with the K matrix, and follow the steps below:

1. For each node i, find the nodes that belong to that boundary
2. Set all values in the i^{th} row to 0
3. Set the (i, i) value to 1

Each node belonging to the boundary on which Dirichlet conditions are applied does not receive contributions from adjacent nodes. Thus, the amount of displacement is equal to the energy applied at the node.

Now we obtain a non-singular matrix for our linear system. Finally, to integrate in time, we must describe this system in a motion equation, adding mass and damping matrices and its respective derivatives (Eq. (4.18)).

$$M\ddot{u} + D\dot{u} + Ku = f \tag{4.18}$$

To calculate those matrices we use a density scalar ρ and a scaling factor α.

$$M^e_{ii} = \frac{1}{3}\rho V^e \qquad D^e_{ii} = \alpha M^e_{ii} \tag{4.19}$$

After that, we can obtain the diagonal mass and damping matrices. We applied finite differences method to approximate derivatives, and our Eq. (4.18) becomes Equation 4.20

$$\frac{M}{\Delta t^2}(u_{t+\Delta t} - 2u_t + u_{t-\Delta t}) + \frac{D}{2\Delta t}(u_{t+\Delta t} - u_{t-\Delta t}) + Ku_{t+\Delta t} = f_{t+\Delta t} \tag{4.20}$$

Reducing this equation to our previous system, we can rewrite:

$$\widetilde{K} = \frac{M}{\Delta t^2} + \frac{D}{2\Delta t} + K \qquad \widetilde{f}_{t+\Delta t} = \frac{M}{\Delta t^2}u_t - (\frac{M}{\Delta t^2} - \frac{D}{2\Delta t})u_{t+\Delta t} + f_{t+\Delta t} \tag{4.21}$$

And we obtain time-varying to our linear system Eq. (4.22)

$$\widetilde{K}u = \widetilde{f}_{t+\Delta t} \tag{4.22}$$

4.3 Green Coordinates

We have discussed two widely spread physical modeling methods. But we want to use a simpler approach when an object is only slightly deformed, and that does not interact (or is being interacted) with the user. Therefore, we have chosen a method that is not based on physics but yields high performance and plausible deformation. In this way, we opt to use a generalization of barycentric coordinates. This method uses a cage that encloses the object and, by deforming the cage, the vertices of the object are also deformed. In the face of many options of generalized barycentric-coordinates methods, we have chosen the Green Coordinates (GC) (LIPMAN; LEVIN; COHEN-OR, 2008), which we briefly explain here.

The modeling of this approach is defined as a cage-based technique. Points inside of a cage C are described as affine combinations of the cage vertices v_i, showed in Eq. (4.23).

$$p = F(p, C) = \sum \phi(p) v_i \tag{4.23}$$

In our case, p is each vertex of the object, and each one will be written up as a combination of the cage vertices. In addition to that, the GC method preserves the object shape by considering the $n(t_j)$ normals of the triangles of the cage. It also considers a scaling factor S_j in order to be scale invariant, ensuring the shape-preserving deformation. The full equation of the GC is described in Eq. (4.24).

$$p = F(p, C) = \sum \phi(p) v_i + \sum \psi_j(p) S_j n(t_j) \tag{4.24}$$

Figure 4.3: A bird and its deformation applying a generalized barycentric coordinate (Green Coordinates).

Figure 4.3 shows the deformation of a 2D image using GC method. By changing the position of the vertices in the cage, all the point (in this case pixels) inside the cage can be deformed accordingly.

In our case, we are going to use the 3D version of the GC. The cage are going to be a simple rectangular prism that encloses all the object. Once that the coefficients $\phi(p)$, $\psi_j(p)$, S_j, and $n(t_j)$ are computed for each vertex of the model, the resulting deformable object can be computed just by applying the multiplications and summations of the equation.

In order to apply external forces (gravity) and displacements due to collisions we have used the MSD method using the vertices of the cage. This allows our system to approximate the desired behavior using a small model.

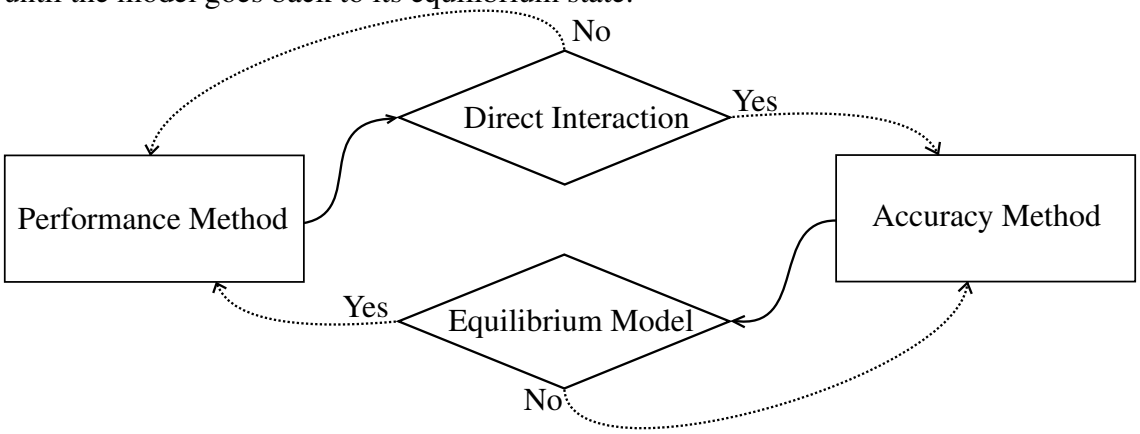
4.4 Dynamic Environment

We have discussed three deformation methods. While two of them are physically based (one being more precise than another), the last one does not take into account physical characteristics. However, as we have shown, these three models offer different levels of manipulation, when considering the characteristics *precision vs performance*. We now present our dynamic environment, which adapts according to the need of each model.

We want to make it clear that our environment is much more than choosing the method for deformation, it also detects interaction, changing the current model in real time. The deformation method used to simulate each tissue changes during the simulation depending on the organs (objects) that the user is interacting with. Initially, all object are simulated using the fastest method: the green coordinates. This ensures greater performance as it is less resource intensive (computationally). When user contact occurs on any object, the system triggers a state change, and this specific object assumes the most accurate method. Even if the user stops interacting with the body, our system will still use the most precise method until the deforming displacements converge to zero to reach the equilibrium of the object.

Figure 4.4 shows a diagram of how out system changes over the different methods. In the diagram, the "performance method" is the green coordinate method, while the accurate method can be either the MSD or the FEM, depending on the importance of the object. The user (or the application) specifies which organs are the most important, these are going to be modeled using FEM (when under interaction), while the others will use the MSD model.

Figure 4.4: Diagram for changing state of the model. When interaction happens via user interaction, there is a context switch, making the system to use the more precise method until the model goes back to its equilibrium state.



5 ADDITIONAL COMPONENTS

In a surgery procedure, several tools are used to manipulate the organs inside the patient's body, and in a simulation we need a similar structure to manipulate the virtual objects. Also in the surgical context, the use of electrocautery is indispensable in most procedures, so we also need to simulate the same process for our surgical training system. In this chapter we describe these additional components that our environment provides in order to deliver a realistic simulation.

5.1 Haptic interaction

The sense of touch when interacting with virtual objects is very important for the realism of a simulation. Therefore, we must consider when using an haptic device along with a graphical display. Haptical rendering is analogous to the graphic rendering, despite the difference that the virtual objects are sensed through the touch, instead of vision. Tactile interfaces have been widely used in several applications, including surgical training, investigation of human touch sensitivity, and others (STANNEY, 2002).

To manipulate virtual objects in a 3D environment we have opted for the Phantom Omni haptic device (TECHNOLOGIES, 2015), distributed by Sensable. This device provides 6 DOF tracking, allowing translations over the axis x, y, and z, and the rotations of pitch, yaw, and roll. Considering a surgery, we can simulate the movements of a real electrosurgical scalpel. The Phantom device also gives force feedback in the three axis. This allows the user to feel resistance by the Phantom; this is very important when we want to simulate the interaction of the virtual tool with the virtual object, as it provides greater immersion to the user.

5.2 Collision Detection

Our system can simulate many objects simultaneously, making it possible to represent different internal organs of the human body. Whenever the scalpel touches any part of the object's mesh, the collision must be properly detected and treated. Collision detection has been a complex problem in computer graphics (JIMéNEZ; THOMAS; TORRAS, 2000; MAULE; MACIEL; NEDEL, 2010), and we present a simple and pleasing solution for our particular detection problem.

Our collision-detection algorithm starts by finding the vertex closest to the virtual scalpel P_b. We are assuming that the closest vertex will always belong to the closest triangle (which is a reasonable assumption). In order to find the closest vertex, we have to perform an optimized search in our graph structure. By starting in a vertex, we greedily

search for the neighbor connection that is closest to the scalpel, iterating until we find the closest vertex of the mesh. In the first search, this method starts with a random vertex, but the following searches are going to depart from the last returned vertex.

Algorithm 1 Find the nearest vertex from cursor device.

```
Let p_0 be the cursor point
index ⇐ −1
while index! = vertex do
  index ⇐ vertex
  dist ⇐ distance(vertex, p_0)
  for n neighbors do
    d ⇐ distance(n, p_0)
    if d < dist then
      dist ⇐ d
      vertex ⇐ n
    end if
  end for
end while
```

After finding the closest vertex, we search for the closest triangle. It can be easily found by computing the distances between the virtual scalpel position and all the triangles that contain the closest vertex. Then, the following steps are used to find the closest point.

1. Calculate the projection of the virtual scalpel on the plane of the triangle
2. Verify if the projected point is inside the triangle
3. If it is inside, return the computed distance from the point to the plane, otherwise continues
4. Project the virtual scalpel in edges of the triangle
5. Return the distance from the virtual scalpel to the nearest edge

The collision detection verifies if the tip of the scalpel is inside or outside the mesh. We do so by computing the dot product of the vectors V_n and V_b, where V_n is the normal of the closest triangle, and V_b is the vector from closest point to the scalpel position.

If the dot product is greater than zero, the scalpel is inside the mesh and a collision is detected, otherwise it is outside the mesh. In practice, we are just verifying if two vectors are pointing out in the same direction or not. If the Phantom cursor is inside the mesh, V_b will always be pointing out for a direction that is close to V_n, making the angle of these two vector lower than $90\,^{\circ}$. On the other hand, when the cursor is outside the mesh, the direction of V_b is going to be mostly like opposed to V_n, meaning that the angle between them is greater than $90\,^{\circ}$.

5.2.1 God-object

There are many problems when an haptic object penetrates a virtual object. The first one is the force feedback: we cannot force the user to stop penetrating the virtual object, but we can arbitrarily set the virtual location of the haptical interface. This representation is known in the literature as *god-object* (TOMASI, 2013).

When there is no collision, the position of the Phantom cursor and the god-object are the same. However, when there is a collision, the god-object is maintained on the mesh surface, while the Phantom cursor continues to penetrate. We must compute the position of the god-object when a collision is detected. This can be easily done using the information when we searched for the closest triangle, the position of the god-object G_o is given by:

$$G_o = P_h + d_p * N \tag{5.1}$$

where P_h is the Phantom-cursor position, d_p is the distance between P_h and the closest triangle, and N is the normal of this closest triangle.

5.2.2 Force feedback

The most important feature provided by the haptic device is the force feedback. This feature is very important when trying to create a realistic simulation, as it can impose the reaction when we apply force to a virtual object. In our case, when a specialist touches a biological tissue with a scalpel, an opposed forced is exerted by the tissue (following the third Newton law: action and reaction). The haptical device is capable of giving this feedback, but we must compute the intensity and direction of this force.

It is important to note that this opposed force only exists if there is a collision, or, when the virtual scalpel is penetrating the mesh. In this case, we can compute this force using the information we already have. The direction of this force will be the same as the normal of the triangle (which the scalpel is penetrating). In practice, this force will try to push the scalpel outside the mesh. The intensity of this force is given by the distance between the god-object position (position of the scalpel over the mesh surface) and the Phantom-cursor position. It will ensure that the more the user penetrates the mesh, the greater will be the force feedback. In this way, the force feedback can be computed as Eq. (5.2).

$$F = k * d^2 * N \tag{5.2}$$

Where d is the distance between the phantom cursor position and the god-object position, N is the normal of the closest triangle and k is the elasticity constant. This k dictates how rigid the object is.

5.2.3 Picking

Our environment also handles the picking of a specific area. Analogous to the previous interaction, when the scalpel collides with the model, the user can grab the area and drag it. The force feedback is computed in a similar way as Eq. (5.2), taking the distance between the god-object and cursor position, which is placed over the mesh, and the phantom-cursor position.

5.3 Heat Diffusion

Electrosurgery is based on applying heat on the tissues, creating an effect that depends on the power applied. This implies directly on the velocity that the heat is generated. Thus, a model of heat distribution is essential if we want an electrosurgery simulation. The heat equation is a well known partial differential equation, it stands as:

$$\frac{\partial u}{\partial t} = (\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) \tag{5.3}$$

By adding the positive constant α to model the thermal diffusivity of the material (in the right-hand size of the equation), we can generally simplify the equation to:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u = \alpha \Delta u \tag{5.4}$$

Where Δ or ∇^2 is the Laplace operator (given by the divergence of the gradient of a function). This equation can be solved in many ways, by using convolution with a specific kernel over initial conditions, or by solving a linear system with boundary conditions. We have opted by the first one due to simplicity and ease of use. Therefore, whenever the user decides to heat a point in the mesh (by pushing a button while touching the object), the temperature of this point will increase and it will diffuse to neighbors based on the following equation (TOMASI, 2013):

$$heat_i = (1 - \kappa * distances_i) * heat_i + \sum_{k \in neighborhood_i} heat_k * (|node_i - node_k|_2)^2 \tag{5.5}$$

$$distances_i = \sum_{k \in neighborhood_i} (|node_i - node_k|_2)^2 \tag{5.6}$$

where $heat_i$ is the temperature of the node i, $neighborhood_i$ are the nodes that share a link with node i, and $|...|_2$ is L-2 norm of the vector. κ is the *thermal diffusivity*, which can be seen as the measure of thermal inertial (VENKANNA, 2010); the higher the *thermal diffusivity*, faster the heat will move through the material. If the node i is at the surface of the model, the $neighborhood_i$ also includes the air, making the node takes into account the temperature of the air, along with its *thermal diffusivity*. Additionally, the neighborhood of a node also stores links of nodes from different models (when they are touching each other).

6 RESULTS: AN APPLICATION IN MINIMALLY INVASIVE SURGERY SIMULATION

In this Chapter we show our environment, evaluating its efficiency and effectiveness. As it is hard to show the environment running using only images, we strongly suggest the reader to watch the videos in our supplementary material. Nonetheless, the information discussed in this Chapter are valuable for the analysis and validation of our system.

We have implemented our system in C++, using OpenGL with the assist of the SOIL library for textures (BARRETT, 2008). For creating the GUI (graphical user interface) we have used the AntTweakBar (ANTTWEAKBAR, 2014). For performing arithmetic operations, we have used the Armadillo library (ARMADILLO, 2014), which in turn performs matrix decompositions through the integration with LAPACK (LAPACK, 2013). For the interaction with the system we have used the Phantom Omni device (TECHNOLOGIES, 2015), using the library provided by the manufacturer to incorporate it in our code.

The collision between objects is based on the PQP (*Proximity Query Package*) library (PQP, 1999). Our system can import mesh files of a specific format, but nothing precludes the incorporation of libraries to import other file formats. For the format of the meshes to be imported, we have used the files generated by Tetgen (SI, 2013), with an additional file specifying the boundary constraints. The deformation parameters of the models (which are different depending if the object is modeled in MSD or FEM) can be interactively set in our GUI (see Figure 6.1).

We have coded all the deformation methods: MSD, FEM, and GC (see previous Chapters), and all the phantom interactions allowed in the system (such as collision, picking, and heating). The deformation computations are performed in a dedicated thread, which is asynchronous with the one of the environment rendering. This ensures that our system does not suffer from low FPS, but the models may have low update rates for high-cost deformation computations (in the case of high cost models using the more complex deformation method, FEM).

As explained above, our thread that computes the deformation of each model is not the same as the one that renders our environment. Thus, measuring the FPS of our application will have no meaning when comparing the performance and efficiency of each deformation method. Therefore, in the next sections we are going to evaluate the times for computing each iteration, as well as the number of iterations, needed for each deformation method until it achieves convergence. It is important to note that, as the time to compute each iteration is proportional to the complexity of the method, the number of iterations needed for convergence depends upon the parameters used for the object (mass, rigidity, and others), so we cannot relate the time complexity of a method by just adding up all the iteration times. Also, it is very important to analyze the iterative computation times, as

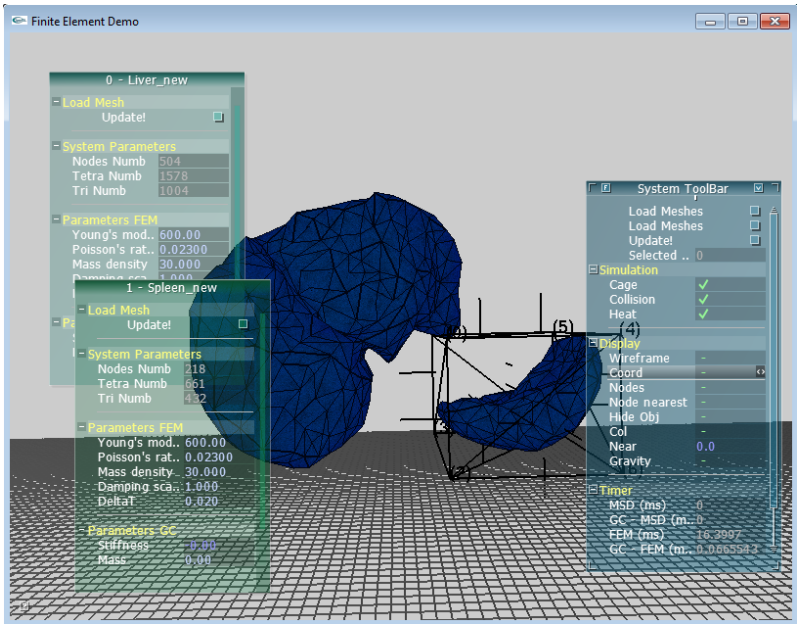


Figure 6.1: Graphical User Interface of our system. The user can change the parameters of each deformation method associated with each object.

these dictate the update rate of the model, which is indispensable for smooth transitions in the rendering of the model, as well as for the force feedback by the interactive device. Next, we present 2 test scenarios, one using simple shapes, and another using more complex models. All the measures were taken on a 3.3 GHz i5 CPU with 8 GB of RAM. For each measure shown in the graphs, we have computed a robustness average (identifying and removing the outliers) over five evaluations.

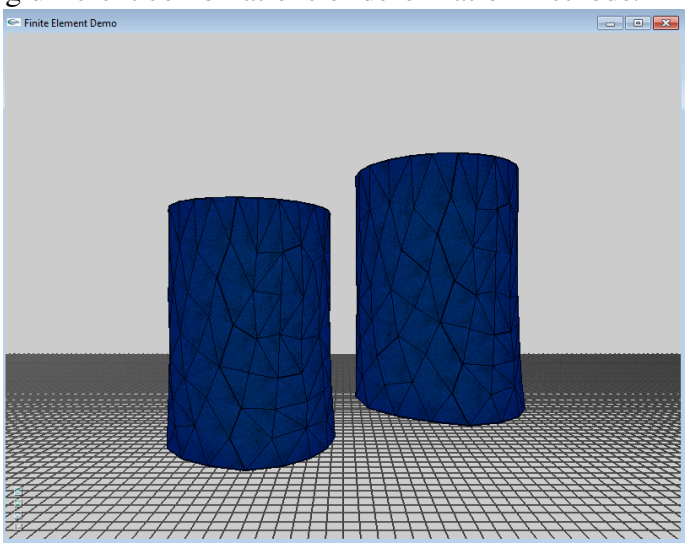
6.1 First Scenario

In the first scenario (Figure 6.2), we evaluate the deformation behavior of two cylinders, which we are going to name as **Model A** and **Model B**, respectively. These cylinders will be placed side by side, and we are going to perform evaluations by modeling each object using different combinations of deformation methods. Each cylinder has 252 nodes, 336 triangles, and 939 tetrahedrons. Table 6.1 shows the parameters used for this scenario.

Table 6.1: In the first scenario was setup different parameter to each method.

Parameters	FEM	MSD	GC
Young's modulus	600	-	-
Poisson's ratio	0.0023	-	-
Stiffness	-	700	50
Density	30	120	20
Damping	1.0	0.4	0.3
Time-step	0.02	0.02	0.02

Figure 6.2: The first scenario consist of two cylinders placed side by side. Each object will modeled using different combinations of deformation methods.



6.2 Second Scenario

In the second scenario (Figure 6.3), we loaded four complex objects: a liver, a spleen, a stomach and a pancreas. Technical information, such as number of nodes, can be found in Table 6.3. These objects meshes were created using the Tetgen in order to simulate specific organs, and all the parameters for the deformation models were carefully chosen to simulate an approximate behavior from real organs. For this scenario, we have disabled the gravity of the environment, as this would hamper the expect behavior of the simulation (due to the lack of surrounding organs and connective tissues).

Table 6.2: It was setup the parameters only to FEM and GC, used in this scenario.

Parameters	FEM	GC
Yong's modulus	140	-
Poisson's ratio	0.023	-
Stiffness	-	50
Density	10	20
Damping	10	0.3
Time-step	0.01	0.02

For this scenario, we used specific parameters to approximate the behavior of the tissues, showed in Table 6.2.

6.3 Efficiency Evaluation

To evaluate the efficiency of our deformation methods, we tested our system in our two scenarios described previously. In this section, we are going to evaluate the times of one iterative step computation of each method, analyzing how they behave when we apply an external force (gravity), user interaction, and other situations.

The feature we are concerned in those tests is the time taken by each method to compute a single iteration (as this can tell us the update rate of each model). In the following

Figure 6.3: The second scenario consist of multiple objects to simulate the behavior of real tissue organs. In this scenario we used shapes to describe the organs tissues, simulating a thorax surgical procedure.

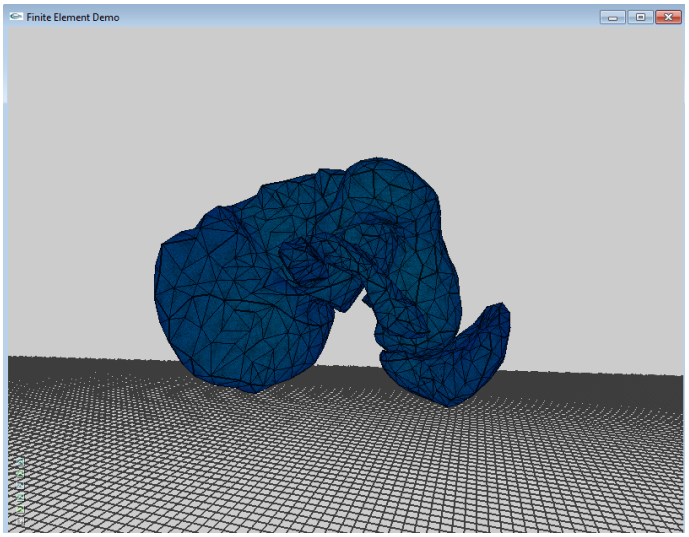


Table 6.3: Second scenario: simulate a surgery environment, providing a realistic interaction and visualization of the organs.

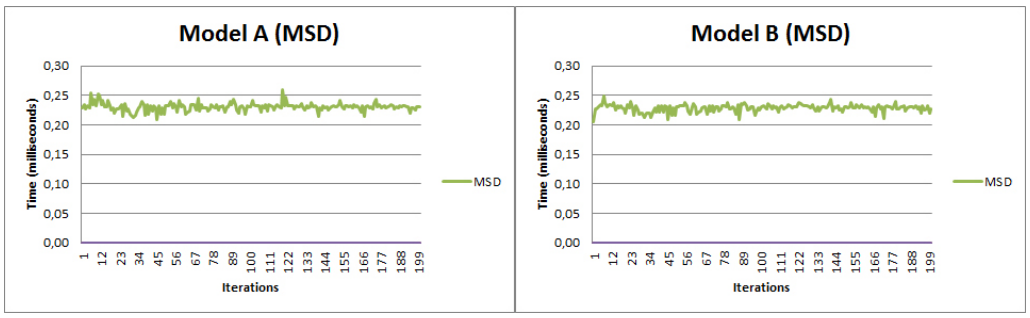
Shape	Vertex	Tetrahedrons	Triangle surface
Liver	504	1578	1004
Spleen	218	661	432
Stomach	291	890	578
Pancreas	229	657	454

figures, we will be showing graphs where the y-axis represents the time (in milliseconds) of the x^{th} iteration, represented by the x-axis. To help understanding the graphs, we used colors for each deformation method: blue for the FEM, red for a FEM model being approximated by GC, green for the MSD method, and purple for MSD being approximated by GC. We are going to evaluate the times for computing each iteration.

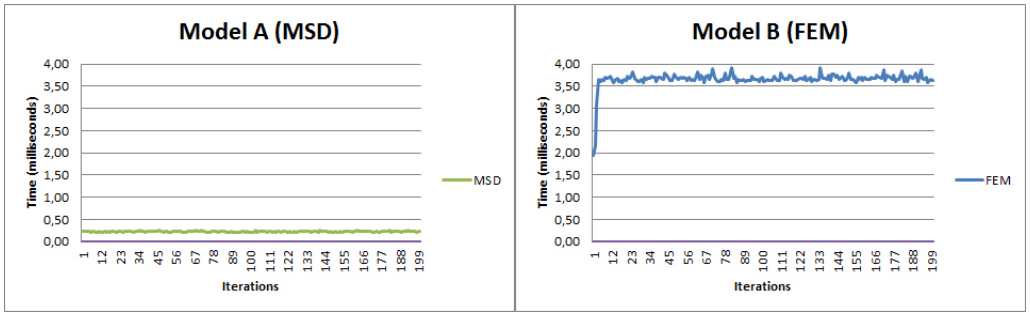
6.3.1 Interaction with an external force

In this first trial, we just let an external force (gravity) interacting with the objects, recording the times for each iterative computation. In Figure 6.4, we have modeled the objects using different combinations of deformation methods (MSD/FEM) in order to evaluate how they behave. It is known that, despite FEM being much slower than MSD, it converges much faster (in number of iterations). However, this does not mean that one is better than another, just that by using these parameters, the number of iterations for convergence are different (see in the graphs). As Figure 6.4 shows times without using GC, we present times for when it is used in Figure 6.5. Note how the GC method outperforms both MSD and FEM in time per iteration, being around 2 times faster than MSD, and more than 20 times faster than FEM. Also, the times for computing each iteration of FEM is much higher than MSD and GC (see Figure 6.5 (b) where we have used the same axis scales).

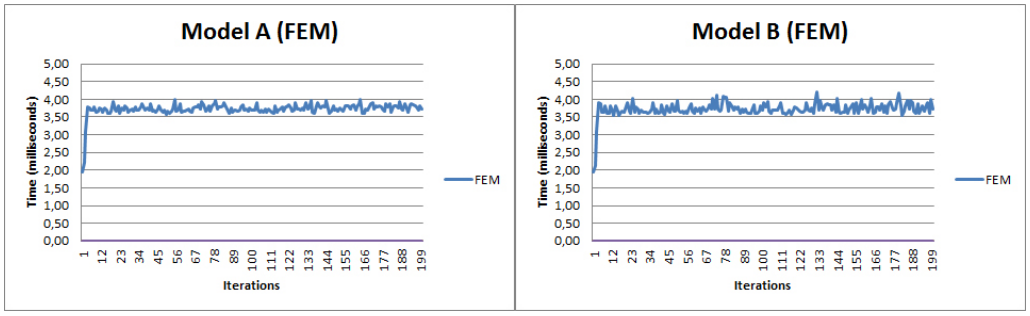
Figure 6.4: Efficiency evaluation of first scenario (two cylinders) just applying the gravity. In (a), both cylinders are modeled using MSD. In (b), the cylinders are modeled using MSD and FEM, respectively; while in (c), both are using FEM. Note the differences in the times for computing each iteration, as well as the number of iterations needed for convergence.



(a) MSD+MSD



(b) MSD+FEM



(c) FEM+FEM

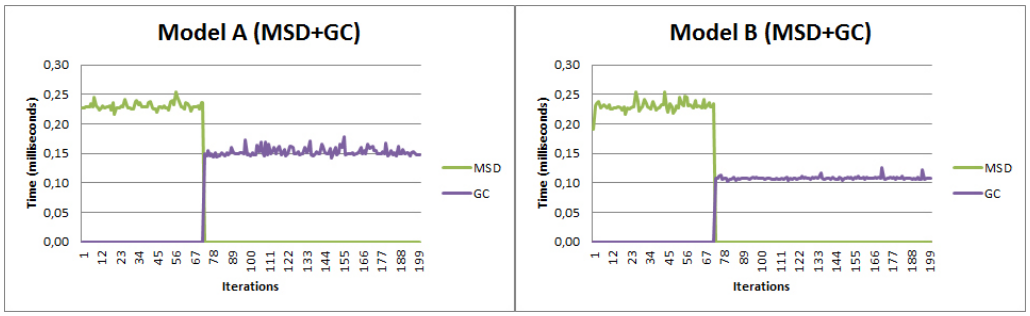
6.3.2 Simple user interaction

In a second trial, we evaluate how each combination of deformation methods behave under user interaction. We have recorded a simple interaction, where all models start in their rest states with GC. After that, the user interacts with Model A, pushing the object for some time, and then, removing the scalpel from the model.

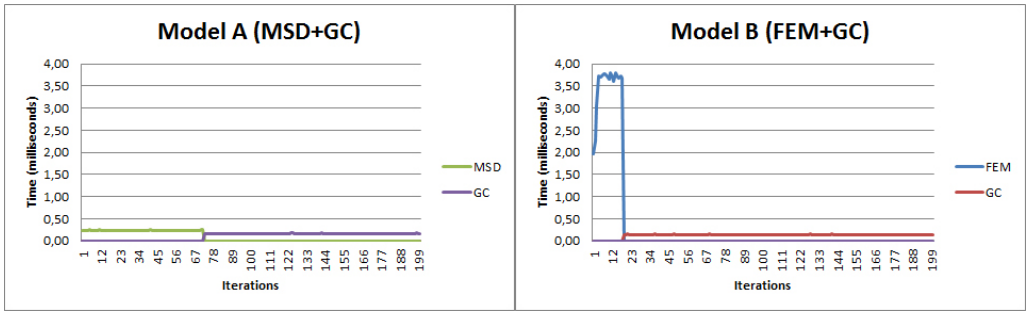
Figure 6.6 shows time results for this trial. These times only account for measuring the computation of collision with the virtual scalpel and for the deformation method, not including the collision detection between objects or heating diffusion.

In Figure 6.6, we can easily note that the times taken by the Model B for computing its deformation is irrelevant when compared with Model A (we show each pair with the same scale factor). The steps seen in Model A graphs are due to the scalpel touching the object,

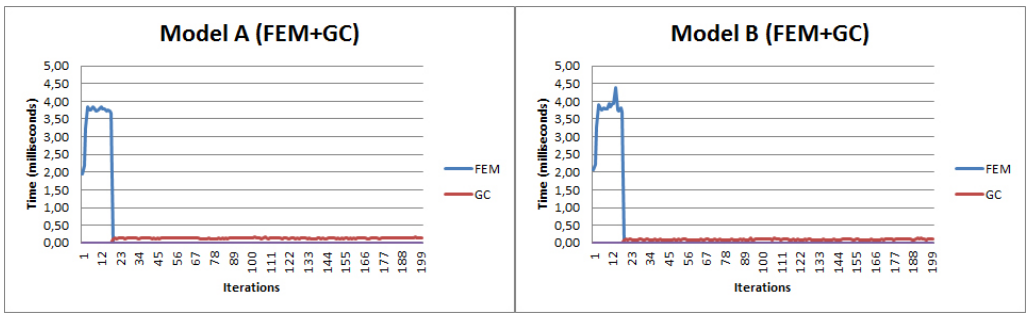
Figure 6.5: Efficiency evaluation of first scenario (two cylinders) just applying the gravity. In (a), both cylinders are modeled using MSD and, after convergence, start using GC. In (b), the cylinders are modeled using MSD and FEM, respectively; while in (c), both are using FEM. Note the times saved when each model reaches the convergence.



(a) MSD+MSD



(b) MSD+FEM



(c) FEM+FEM

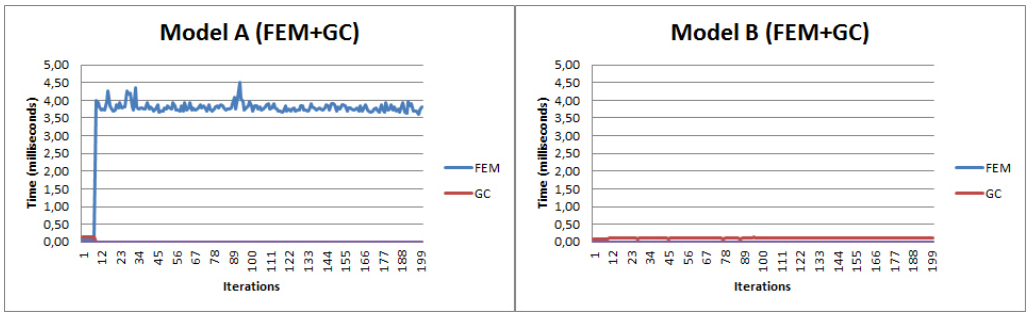
forcing a state change from the faster method (GC) to the precise method (FEM/MSD).

6.3.3 User interaction with objects collision

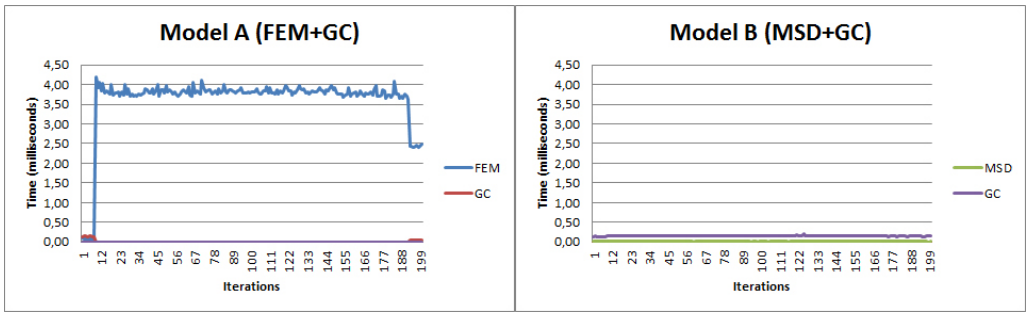
For the third trial, we recorded an interaction with the Model A in a way that it collides with Model B, i.e., we perform an operation that affects both objects (one directly and other indirectly).

Figure 6.7 shows the time results for this trial using the same setups used on the last subsection. In those times, we only considered the time for detecting the collision of the scalpel, and the time for computing the deformation. In the end of the subsection, we present times taken for detecting the collision.

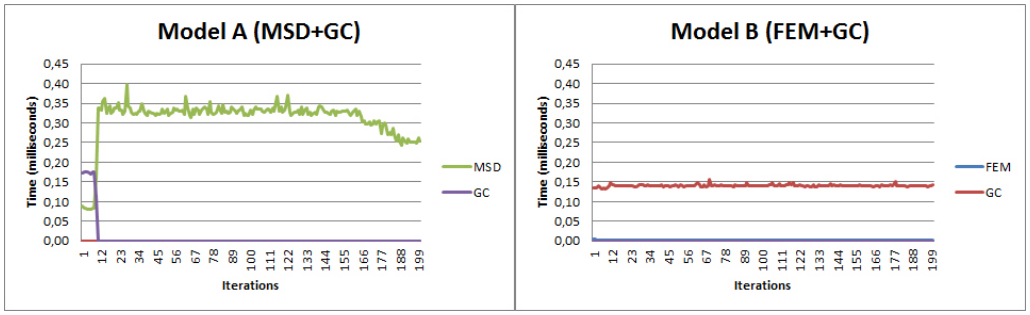
Figure 6.6: Efficiency evaluation of first scenario under simple user interaction. In (a), both cylinders are modeled using FEM. In (b) the model A is modeled using FEM and model B using MSD. In (c), model A uses MSD and model B uses FEM. In (d), both models are modeled using MSD. Note the differences in the times for computing each iteration for the object that is being interacted (model A) and the one that remains rested (model B) and using GC.



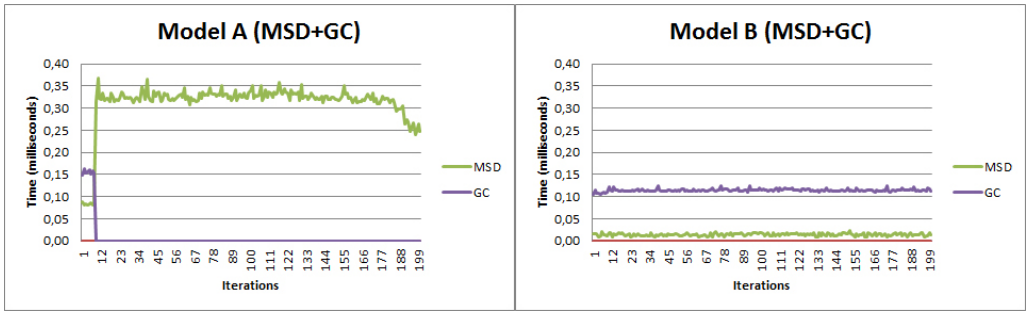
(a) FEM+FEM



(b) FEM+MSD

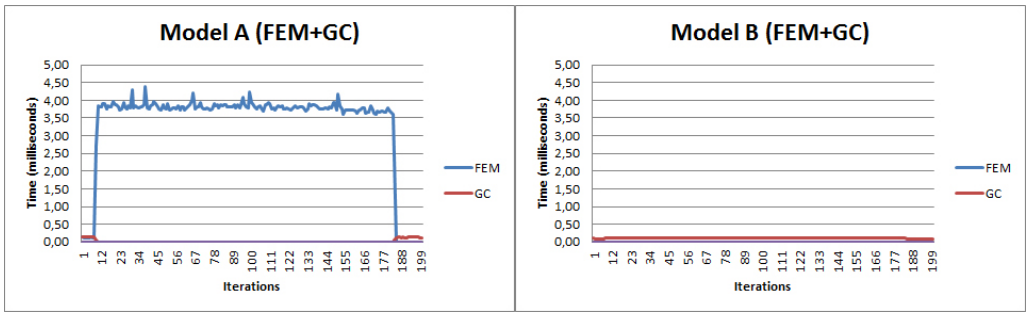


(c) MSD+FEM

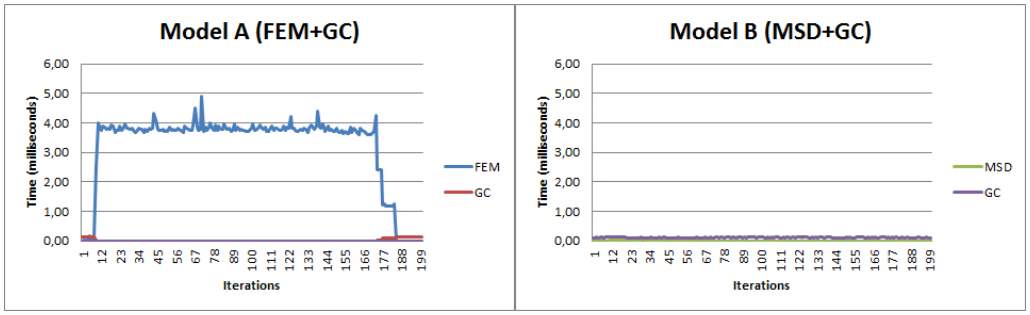


(d) MSD+MSD

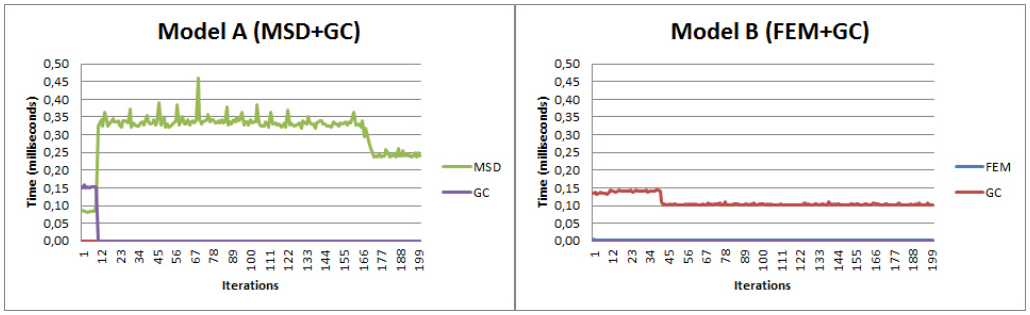
Figure 6.7: Efficiency evaluation of first scenario under user interaction affecting both objects. In (a), both cylinders are modeled using FEM. In (b) the model A is modeled using FEM and model B using MSD. In (c), model A uses MSD and model B uses FEM. In (d), both models are modeled using MSD.



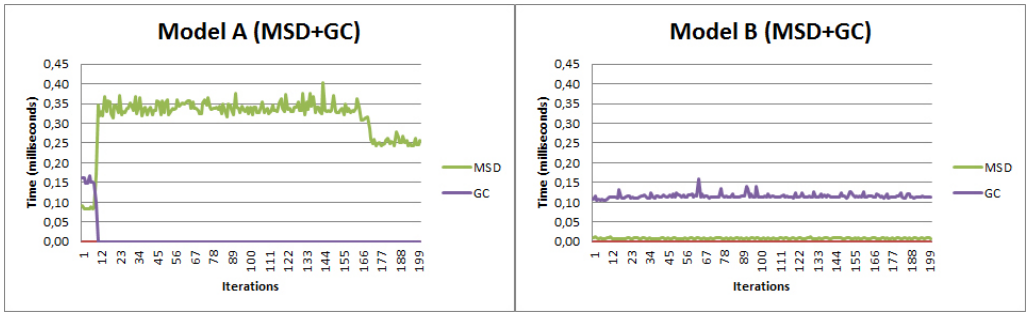
(a) FEM+FEM



(b) FEM+MSD



(c) MSD+FEM



(d) MSD+MSD

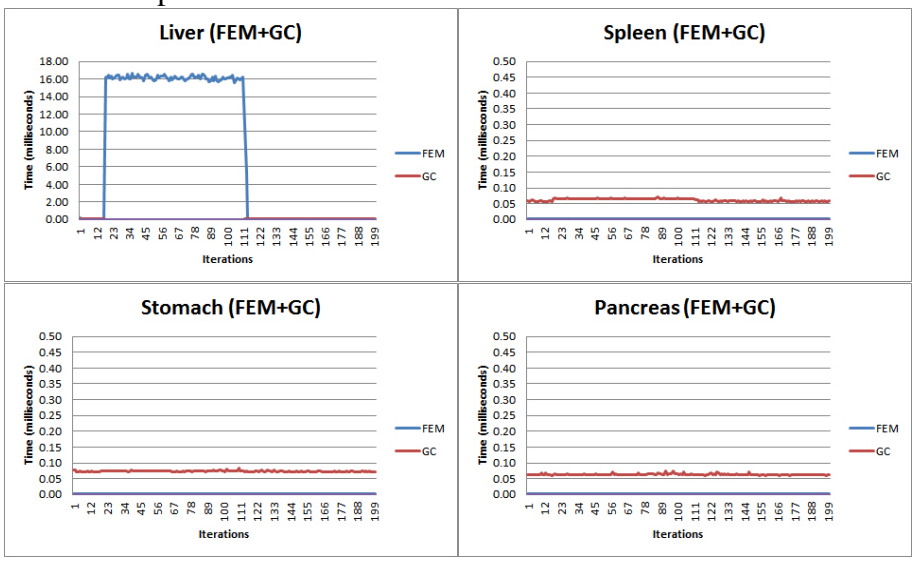
6.3.4 Additional Evaluation

After seeing how our environment behaves when using different combinations of deformation methods for modeling simple objects, we present times for the most complex

scenario. The parameter specification for the organs was done only for the FEM model, so we are only presenting times using this deformation method.

As the local disposition of the organs causes them to touch, it is impossible to perform a trial where an interaction does not result in a collision. Thus, this trial is based on the user interacting with the liver, and this action propagated to other organs. We are only showing the times taken by using the higher-computational-cost deformation method (FEM), switching to GC when there is no direct interaction. Figure 6.8 shows the times for computing the iterations of this scenario (pay attention to the y-axis scale). Note that for larger meshes, the gap between the FEM and the GC becomes even higher, which makes unfeasible to keep modeling all the objects using FEM and justifies the use of GC.

Figure 6.8: Efficiency evaluation of second scenario (organs) under user interaction affecting all objects. All the organs are modeled using FEM, but only the liver interacts directly with the scalpel.



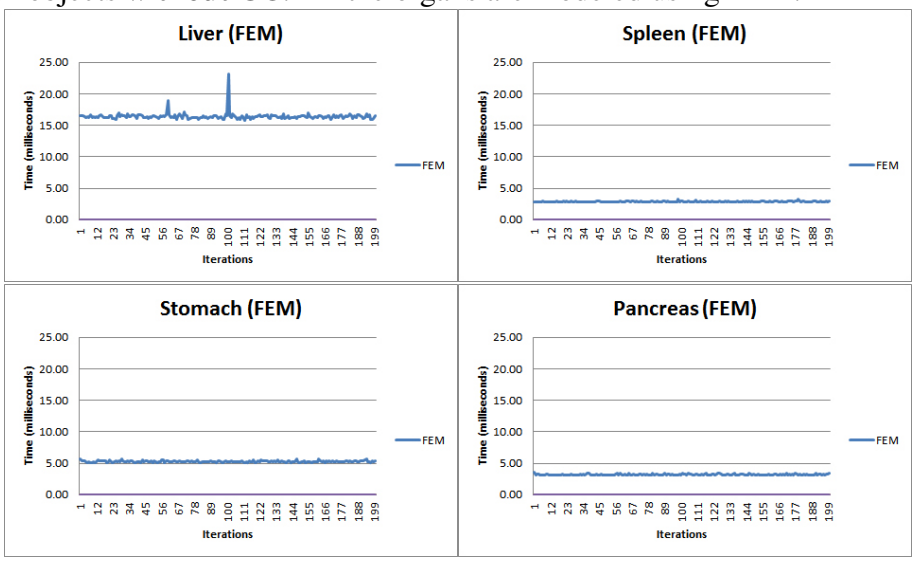
For comparison, we also performed the same test without GC. Figure 6.9 show the times for this case. Note how the times for the spleen, stomach, and pancreas are more than 50 times faster when using GC.

6.3.4.1 Collision Evaluation

Our collision-detection algorithm is an implementation based on the PQP method (PQP, 1999). This technique takes surface nodes of each mesh and returns the ones that are colliding. Therefore, all the deformation methods yield the same amount of processing for the detection of collisions.

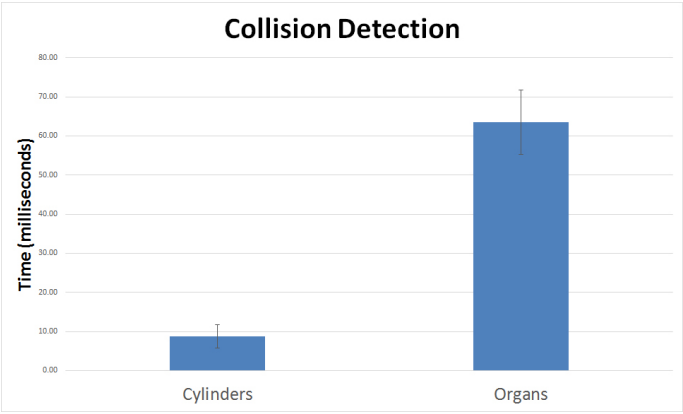
However, the way the collision is treated varies depending on the deformation method of the model. When a collision is detected, our system computes the displacements that are going to be applied to the nodes. In the case of FEM and MSD, the displacements computed are applied directly on the vertices of the mesh, while in the GC, we compute the contribution of each displacement value for each cage vertex. This coarse approximation of GC achieves a good trade-off between computational time and accuracy, largely improving on the former with a small reduction on the latter. Figure 6.10 shows times for detecting the collisions and computing the displacements for the first and second scenarios. As the parameters of each deformation method have a huge influence on when

Figure 6.9: Efficiency evaluation of second scenario (organs) under user interaction affecting all objects **without** GC. All the organs are modeled using FEM.



and how the collision is going to happen, these times measures were taken using FEM to model both cylinders.

Figure 6.10: Collision detection times. Times taken for our PQP method to detect collision and compute displacements.



6.3.4.2 Heat Diffusion Evaluation

We have evaluated the times for heat diffusion. Our diffusion algorithm is independent of deformation method, as it only runs through the vertices computing the contribution of their neighborhood. For this trial, we have recorded a simple task consisting of the user applying heat in a particular area, making our system to compute the diffusion as described in Section 5.3. Figure 6.11 shows how the times climb when we increase the complexity of the mesh and the number of links between other objects. Figure 6.11 (left) shows the times taken by the cylinders to perform the heat diffusion, while in the (right) we can see the times taken for each organ of our second scenario. Note that, although the liver is the model with higher number of nodes, the stomach has the slowest heat diffusion. This happens because the stomach is near all the other organs, making the neighborhood of its surface nodes account for nodes of the spleen, pancreas, and liver.

Figure 6.12 compares the total heat-diffusion time taken for each scenario. The measured times do not include the collision detection of the scalpel with the organs.

Figure 6.11: Heat-diffusion times. Times taken for performing the heat diffusion on first scenario (left), and second scenario (right).

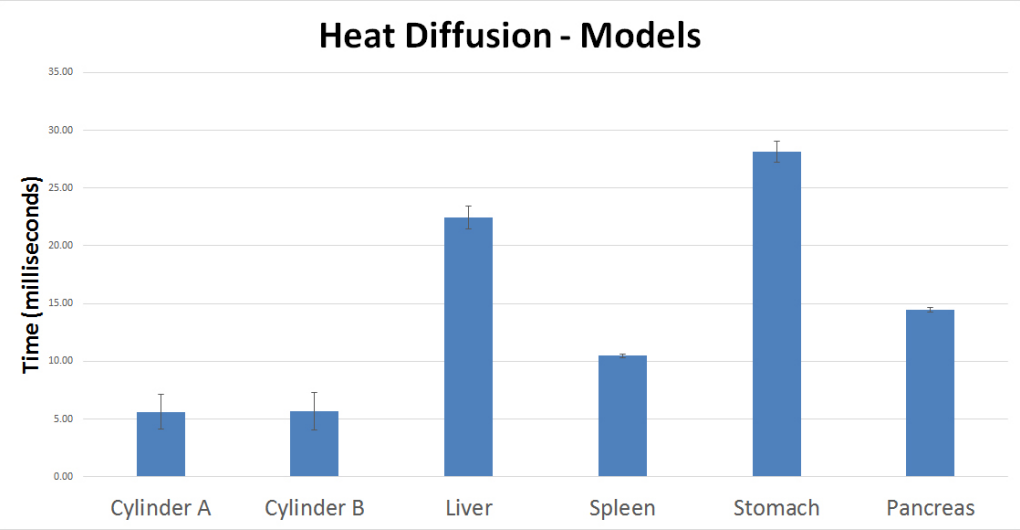
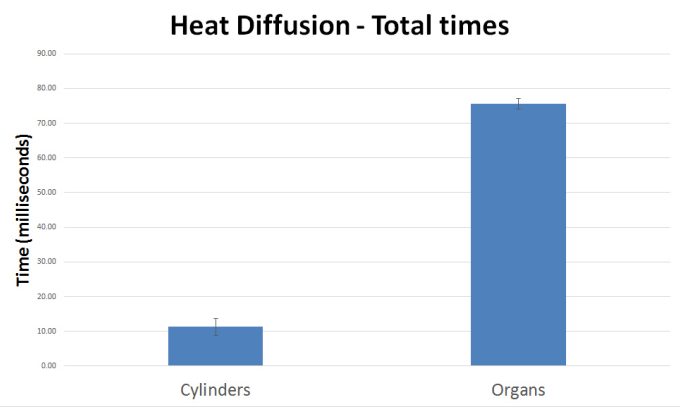


Figure 6.12: Total heat-diffusion times. Total times for heat diffusion of each scenario.



6.4 Effectiveness Evaluation

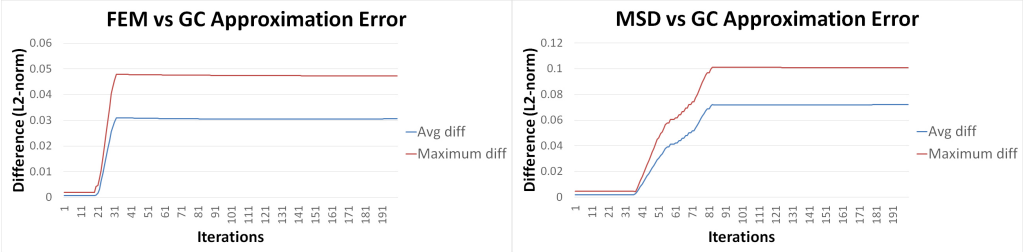
It is well known that the methods do not have a similar behavior, and it is unfeasible to find parameter settings that will make a FEM and a MSD behave in the exactly the same. Thus, our goal is not to compare each other. We have evaluated how good our GC-MSD approximates these physical approaches. As explained, the GC are only used in objects that are not under direct interaction, so we are willing to tolerate some errors in those approximations.

In order to measure the effectiveness of our system, we have recorded the mean difference (based on all the nodes) of the object being modeled by a physical deformation method with the same object using GC, both under the same interaction. As mean squared error (MSE) and other error measures are only used for comparing two or more approximation methods (and we are just interested in evaluating how plausible is a single approximation), we thought this mean difference would be more useful. In order to obtain

a glimpse of intuition about the errors, we took notes of the dimensions of the cylinders (after applying the gravity): *width=1.24*, *height=1.72*, and *depth=1.24*.

Figure 6.13 (left) shows the average and maximum errors for each iteration, using FEM as the ground truth, and GC as the approximation, while (right) shows these errors when using GC for approximating MSD. We decided to also record the maximum error, so when using a deformation like FEM (which is known for performing local deformations), we can evaluate the worst node approximation on each iteration. By considering the dimensions of the cylinders, we could say that the maximum error when approximating FEM was 4% of the size of the smallest dimension (width/depth), while when approximating MSD the maximum error was 8.1% of the smallest dimension. In our opinion, this is a low error compared to the performance improvement of using this approximation.

Figure 6.13: Effectiveness evaluation of first scenario. On the left, comparing the error by using GC to approximate FEM, and on the right the error by using GC to approximate MSD. The dimensions of the cylinders (after applying the gravity) are: *width=1.24*, *height=1.72*, and *depth=1.24*. We have recorded the average and maximum errors for each iteration.



6.5 Discussion and Limitations

The implementation of all the presented features (collision, interaction, deformation computations, heating diffusion, and others) does not make use of any parallel optimization. As many of the calculations for solving these problems could be computed concurrently, we believe that the performance of our system could be greatly improved by parallelizing some steps.

There are a lot of generalized barycentric coordinates, such as the Mean Value Coordinates (MVC) (FLOATER, 2003), Harmonic Coordinates (JOSHI et al., 2007), and others (MEYER et al., 2002; LANGER; BELYAEV; SEIDEL, 2006). What motivates the choice of GC was the straightforward extension of it to 3D space, and specially its shape-preserving property (LIPMAN; LEVIN; COHEN-OR, 2008). As mentioned earlier, the model cage is defined as a rectangular prism based on the minimum and maximum values of x, y, and z. Although we have achieved good approximations using this simple cage, we believe that generating a more refined cage could improve the accuracy of the approximations. Some methods are capable of performing automatic cage generation (XIAN; LIN; GAO, 2009, 2012), but it is still uncertain if it is worth to trade performance for precision in this case, where the GC intention is to be as fast as possible.

The MSD parameters for the cage were chosen empirically. We tried to find a correlation between the physical deformation model (FEM/MSD) parameters and the ones used in the MSD-GC cage, but this had proven to be a hard problem, even for simple objects like a cylinder. Thus, whenever we want to approximate different behaviors (achieved

by particular objects and different FEM/MSD parameters), it is necessary to perform a search for new parameters that obtain better approximations.

Another problem is the exchange of forces between the models, used mainly in the collision feature. We need to perform interaction among distinct behavior models (MSD, FEM, and GC) by using displacement and force values. To circumvent this problem, we applied specific constrains on each deformation method to receive an appropriate force/displacement, depending on the method input.

7 CONCLUSION

This thesis presented a hybrid and adaptive environment to simulate deformable objects in real-time. Our method approaches the gap between performance and accuracy combining different (physical and non-physical) deformation methods, providing a realistic scenario with pleasing immersion for the user. The physically-based methods are expressed by the theory of elasticity referred to as Hooke's law, providing a high precision behavior on simulated objects. Furthermore, the non-physically-based method is represented by a generalization of barycentric coordinates, used to obtain high performance while ensure a plausible deformation.

Our system models each object of the environment based on its relevance to the simulation. Therefore, it allows the user to choose the deformation method (FEM or MSD) arbitrarily for each object. Furthermore, during the simulation, our environment approximates objects with no interaction using a faster method, saving computational resources.

The collision detection is performed independently of deformation method, but its outcome works differently depending on the method. While the collision affects FEM and MSD modeled objects by applying computed displacements directly on the mesh vertices, for the case of GC, it computes the contribution of each displacement value for each cage vertex. This rough approximation can be further improved using a cage that better fits the object.

The surgery environment simulated by our system can help students and specialists, providing a safe and practical way to experiment complex surgical procedures. Our case studies do not cover a complete surgery simulation system. Nevertheless, they implement typical surgical interactions, such as picking, burning, and haptic feedback. Such interactions are greatly enhanced by faster systems with high update rates.

7.1 Future Work

The use of a cage to approximate the behavior of physical models could be further improved. Finding the appropriate parameters for the MSD cage is a hard and cumbersome task, without any assurance of reliability. The use of an interpolation mapping among these parameters is an interesting direction for future exploration. The capability of creating more refined cages automatically is also an attractive feature that, potentially, could improve the accuracy of the GC.

Another typical task in surgeries is the cutting of organs and tissues. This was set aside because of the complexity involved in this task. When we want to simulate such function, it is necessary to treat complex problems like creating new geometry and assigning new physical connections for a realistic behavior. This is a challenge we would like to solve in the future.

REFERENCES

ANTTWEAKBAR. anttweakbar.sourceforge.net/. AntTweakBar GUI library v1.16 (2014).

ARIKATLA, V. S. et al. Towards virtual FLS: development of a peg transfer simulator. **The International Journal of Medical Robotics and Computer Assisted Surgery**, Hong Kong, China, 2013.

ARMADILLO. http://arma.sourceforge.net/. Armadillo C++ Library v4.3 (2014). Open Source License (MPL), supported by the NICTA research center in Australia.

BARRETT, S. **Simple OpenGL Image Library - SOIL**. http://www.lonesock.net/soil.html. Version 1.16 (2008).

BASDOGAN, C. et al. Haptics in minimally invasive surgical simulation and training. **Computer Graphics and Applications, IEEE**, Los Alamitos, CA, USA, v.24, n.2, p.56–64, March 2004.

BATHE, K. **Finite Element Procedures in Engineering Analysis**. New Jersey, United States: Prentice-Hall, 1982. (Prentice-Hall Civil Engineering and Engineering Mechanics Series).

BECKER, M.; IHMSEN, M.; TESCHNER, M. Corotated SPH for Deformable Solids. In: FIFTH EUROGRAPHICS CONFERENCE ON NATURAL PHENOMENA, 2009, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2009. p.27–34. (NPH'09).

BOURGUIGNON, D.; CANI, M.-P. Controlling Anisotropy in Mass-Spring Systems. In: EUROGRAPHICS WORKSHOP ON COMPUTER ANIMATION AND SIMULATION (EGCAS), 2000. **...** Springer-Verlag, 2000. p.113–123. (Springer Computer Science). Proceedings of the 11th Eurographics Workshop, Interlaken, Switzerland, August 21–22, 2000.

BRO-NIELSEN, M.; COTIN, S. Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation. **Computer Graphics Forum**, Hoboken, New Jersey, United States, v.15, n.3, p.57–66, 1996.

CHEN, D. T.; ZELTZER, D. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 19., 1992, New York, NY, USA. **Proceedings...** ACM, 1992. p.89–98. (SIGGRAPH '92).

CHOY, I.; OKRAINEC, A. Simulation in Surgery: perfecting the practice. **Surgical Clinics of North America**, USA, v.90, n.3, p.457 – 473, 2010. Simulation and Surgical Competency.

CIRIO, G. et al. "Tap, squeeze and stir" the virtual world: touching the different states of matter through 6dof haptic interaction. In: IEEE VIRTUAL REALITY CONFERENCE (VR), 2011, 2011, Singapore. **Proceedings...** [S.l.: s.n.], 2011. p.123 –126.

DE, S. et al. Physically realistic virtual surgery using the point-associated finite field (PAFF) approach. **Presence**: teleoper. virtual environ., Cambridge, MA, USA, v.15, p.294–308, June 2006.

DELINGETTE, H.; AYACHE, N. Hepatic Surgery Simulation. **Commun. ACM**, New York, NY, USA, v.48, n.2, p.31–36, Feb. 2005.

DUAN, Y. et al. Synchronous simulation for deformation of liver and gallbladder with stretch and compression compensation. In: ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY (EMBC), 35TH ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE, 2013. **Proceedings...** [S.l.: s.n.], 2013. p.4941–4944.

FLOATER, M. S. Mean Value Coordinates. **Comput. Aided Geom. Des.**, Amsterdam, The Netherlands, The Netherlands, v.20, n.1, p.19–27, Mar. 2003.

GEORGII, J.; WESTERMANN, R. Mass-Spring Systems on the GPU. **Simulation Modelling Practice and Theory**, Amsterdam, Netherlands, v.13, n.8, p.693–702, 2005.

GIBSON, S. F. F.; MIRTICH, B. **A Survey of Deformable Modeling in Computer Graphics**. Cambridge, MA, USA: Mitsubishi Electric Research Laboratories, 1997.

HUEBNER, K. H. **The Finite Element Method for Engineers**. Nova Jersey, EUA: John Wiley & Sons, 1983.

INTUITIVE SURGICAL, I. **da Vinci**. `http://www.intuitivesurgical.com/products/`. daVinci 2014.

JIMéNEZ, P.; THOMAS, F.; TORRAS, C. 3D Collision Detection: a survey. **Computers and Graphics**, Amsterdam, Netherlands, v.25, p.269–285, 2000.

JOHNSON, C. **Numerical Solution of Partial Differential Equations by the Finite Element Method**. Cambridge, England, United Kingdom: Cambridge University Press, 1987.

JOSHI, P. et al. Harmonic Coordinates for Character Articulation. **ACM Trans. Graph.**, New York, NY, USA, v.26, n.3, July 2007.

KAUFMANN, P. et al. Flexible simulation of deformable models using discontinuous Galerkin FEM. **Graph. Models**, San Diego, CA, USA, v.71, n.4, p.153–167, July 2009a.

KAUFMANN, P. et al. Enrichment textures for detailed cutting of shells. **ACM Trans. Graph.**, New York, NY, USA, v.28, n.3, p.50:1–50:10, July 2009b.

LANGER, T.; BELYAEV, A.; SEIDEL, H.-P. Spherical Barycentric Coordinates. In: FOURTH EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING, 2006, Aire-la-Ville, Switzerland, Switzerland. **Proceedings...** Eurographics Association, 2006. p.81–88. (SGP '06).

LAPACK. `http://www.netlib.org/lapack/`. Linear Algebra PACKage v3.5.0 (2013).

LIM, Y.-J.; DE, S. Nonlinear tissue response modeling for physically realistic virtual surgery using PAFF. In: EUROHAPTICS CONFERENCE, AND SYMPOSIUM ON HAPTIC INTERFACES FOR VIRTUAL ENVIRONMENT AND TELEOPERATOR SYSTEMS. WORLD HAPTICS 2005. FIRST JOINT, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.479 – 480.

LIM, Y.-J. et al. Soft Tissue Deformation and Cutting Simulation for the Multimodal Surgery Training. In: IEEE INTERNATIONAL SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS. CBMS., 19., 2006. **Proceedings...** [S.l.: s.n.], 2006. p.635–640.

LIPMAN, Y.; LEVIN, D.; COHEN-OR, D. Green Coordinates. **ACM Trans. Graph.**, Los Angeles, California, v.27, n.3, 2008.

MARSHALL, P.; PAYANDEH, S.; DILL, J. Suturing for surface meshes. In: IEEE CONFERENCE ON CONTROL APPLICATIONS. CCA., 2005. **Proceedings...** [S.l.: s.n.], 2005. p.31–36.

MAULE, M.; MACIEL, A.; NEDEL, L. Efficient Collision Detection and Physics-Based Deformation for Haptic Simulation with Local Spherical Hash. In: PROCEEDING ON THE 23RD SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES (SIBGRAPI), 2010. **...** [S.l.: s.n.], 2010. v.23, p.9–16.

MESIT, J.; GUHA, R.; FURLONG, W. Simulation of Lung Respiration Function Using Soft Body Model. In: FOURTH UKSIM EUROPEAN SYMPOSIUM ON COMPUTER MODELING AND SIMULATION (EMS), 2010. **Proceedings...** [S.l.: s.n.], 2010. p.102 –107.

MEYER, M. et al. Generalized Barycentric Coordinates on Irregular Polygons. **J. Graph. Tools**, Natick, MA, USA, v.7, n.1, p.13–22, Nov. 2002.

MOROOKA, K. et al. Navigation system with real-time finite element analysis for minimally invasive surgery. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY (EMBC), 35., 2013. **Proceedings...** [S.l.: s.n.], 2013. p.2996–2999.

OKRAINEC, A.; SMITH, L.; AZZIE, G. Surgical simulation in Africa: the feasibility and impact of a 3-day fundamentals of laparoscopic surgery course. **Surgical Endoscopy**, Germany, v.23, n.11, p.2493–2498, 2009.

PARK, C. H.; WILSON, K.; HOWARD, A. Examining the learning effects of a low-cost haptic-based virtual reality simulator on laparoscopic cholecystectomy. In: IEEE 26TH INTERNATIONAL SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS (CBMS), 2013. **Proceedings...** [S.l.: s.n.], 2013. p.233–238.

PQP. **PQP - A Proximity Query Package**. http://gamma.cs.unc.edu/SSV/. Version 1.3 (1999).

PROVOT, X. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In: IN GRAPHICS INTERFACE, 1996. **...** [S.l.: s.n.], 1996. p.147–154.

QUEIROS, S. et al. A laparoscopic surgery training interface. In: IEEE 1ST INTERNATIONAL CONFERENCE ON SERIOUS GAMES AND APPLICATIONS FOR HEALTH (SEGAH), 2011. **Proceedings...** [S.l.: s.n.], 2011. p.1–7.

SCIENCE, S. **LapSim**. http://www.surgical-science.com/lapsim-the-proven-training-system/. LapSim®: The Proven Training System, 2014.

SCOTT, D. J. et al. Certification pass rate of 100proficiency-based training. **Surgical Endoscopy**, [S.l.], v.22, p.1887–1893, 2008.

SELLE, A.; LENTINE, M.; FEDKIW, R. A mass spring model for hair simulation. **ACM Trans. Graph.**, New York, NY, USA, v.27, n.3, p.64:1–64:11, Aug. 2008.

SHAH, J.; DARZI, A. Simulation and skills assessment. In: INTERNATIONAL WORKSHOP ON MEDICAL IMAGING AND AUGMENTED REALITY., 2001. **Proceedings...** IEEE, 2001. p.5–9.

SI, H. **TetGen - A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator**. http://wias-berlin.de/software/tetgen/. TetGen v1.5.0 (2013).

SIMBIONIX. **LAP Mentor**. http://simbionix.com/simulators/lap-mentor/. LAP MentorTM 2014.

STANNEY, K. M. **Handbook of Virtual Environments**: design, implementation, and applications. [S.l.]: Taylor & Francis, 2002. (Handbook of Virtual Environments: Design, Implementation, and Applications).

STOYANOV, D. et al. Current issues of photorealistic rendering for virtual and augmented reality in minimally invasive surgery. In: INFORMATION VISUALIZATION, 2003. IV 2003. PROCEEDINGS. SEVENTH INTERNATIONAL CONFERENCE ON, 2003. **...** IEEE, 2003. p.350–358.

SULAIMAN, S. et al. Optimizing time step size in modeling liver deformation. In: IEEE 3RD INTERNATIONAL CONFERENCE ON SYSTEM ENGINEERING AND TECHNOLOGY (ICSET), 2013. **Proceedings...** [S.l.: s.n.], 2013. p.209–214.

TECHNOLOGIES, S. **PHANTOM OMNI HAPTIC DEVICE**. http://www.dentsable.com/haptic-phantom-omni.htm.

TOMASI, D. L. **Virtual electrosurgery based on physical**. Trabalho de conclusão(graduação) - Universidade Federal do Rio Grande do Sul. Instituto de Informática. Bacharelado em Ciência da Computação: Ênfase em Ciência da Computação.

VECCHIO, R.; MACFAYDEN, B. V.; PALAZZO, F. History of laparoscopic surgery. **Panminerva Med**, [S.l.], v.42, n.1, p.87–90, 2000.

VENKANNA, B. K. **Fundamentals of Heat and Mass Transfer**. [S.l.]: New Delhi: PHI Learning, 2010.

VERLET, L. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. **Phys. Rev.**, [S.l.], v.159, p.98–103, Jul 1967.

WENHAM, M. **200 science investigations for young students**. [S.l.]: SAGE Publications Ltd, 2001.

WICKHAM, J. E. The new surgery. **BMJ**, [S.l.], v.295, n.6613, p.1581–1582, 1987.

XIAN, C.; LIN, H.; GAO, S. Automatic generation of coarse bounding cages from dense meshes. In: IEEE INTERNATIONAL CONFERENCE ON SHAPE MODELING AND APPLICATIONS., 2009. **Proceedings...** [S.l.: s.n.], 2009. p.21–27.

XIAN, C.; LIN, H.; GAO, S. Automatic cage generation by improved OBBs for mesh deformation. **The Visual Computer**, [S.l.], v.28, n.1, p.21–33, 2012.

YIN, G. et al. Soft Tissue Modeling Using Tetrahedron Finite Element Method in Surgery Simulation. In: FIRST IEEE INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND ENGINEERING, 2009, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2009. p.3705–3708. (ICISE '09).

APÊNDICE A UMA ABORDAGEM DE NÍVEIS DE PRECISÃO PARA MODELAGEM DE TECIDOS MOLES FISICAMENTE BASEADOS

Resumo da Dissertação em Português

Cirurgia minimamente invasiva vem tornando-se cada vez mais comum em hospitais. Este procedimento tem como objetivo a preservação máxima da anatomia do organismo, bem como mínima agressão a ele (WICKHAM, 1987; VECCHIO; MACFAYDEN; PALAZZO, 2000). Esta técnica proporcionam muitos benefícios ao paciente, tais como: cicatrizes menores, pouca dor, níveis menores de complicações, rápida recuperação, menor estadia no hospital e maior conforto.

Ambientes computacionais permitem aos profissionais e estudantes da área a familiarizarem com procedimentos reais, e auxiliam em treinamentos (STOYANOV et al., 2003). Entretanto, aplicações com este intuito requerem um elevado grau de complexidade a fim de obter uma aparência realista e agradável.

Em aplicações médicas, órgãos internos do paciente são simulados como estruturas rígidas e não-rígidas. Simulações para treinamento e planejamento são compostos de um ambiente virtual onde a visualização e interação necessitam ser o mais próximo possível de situações reais (SHAH; DARZI, 2001).

Nesta dissertação, apresentamos um ambiente híbrido e adaptativo capaz de simular procedimentos cirúrgicos. Nosso sistema trata a questão *desempenho vs acurácia* através do uso de diferentes modelos de deformação. Desta forma, nosso sistema é capaz de poupar recursos computacionais através do tratamento dos modelos utilizados. Ainda mais, nosso ambiente fornece retorno háptico, e recursos comuns a cirurgias, como *picking* e aquecimento. Mostramos como nosso método é capaz de obter melhores taxas de atualização, perdendo pouca precisão para isto.

Destacamos as seguintes contribuições desta dissertação:

- *um ambiente híbrido e adaptativo para simulação precisa e em tempo-real*. O ambiente utiliza uma combinação dos métodos de deformação Massa-Mola, Método dos Elementos Finitos e *Green Coordinates*, para reduzir o *trade-off* entre performance por precisão;

- *uma técnica capaz de realizar troca entre os métodos de deformação para cada objeto renderizado*. A técnica realiza troca automática (em tempo real) dos métodos de deformação de acordo com condições do ambiente.

- *um meio para que diferentes métodos de deformação possam interagir entre si, trocando forças*. Modelamos os métodos de deformação para que pudessem se comunicar, realizando troca de forças de acordo com suas *constraints*;

Para obter as contribuições citadas, foi necessário realizar um estudo anterior para melhor entendimento de seu funcionamento. Tal estudo leva em conta fatores como: performance, convergência e estabilidade. Dentre as contribuições secundárias incluem-se:

- *um estudo de caso de técnicas utilizadas em simulações de corpos deformáveis*. Implementamos o método de deformação Massa-Mola, e aplicamos as técnicas de *Euler*, *Verlet* e *Runge-Kutta 4* para integração numérica. Neste estudo foram avaliados o desempenho, erro numérico e estabilidade do modelo Massa-Mola com base em diferentes configurações dos objetos;

- *um estudo de caso de diferentes engines físicas*. Avaliamos as seguintes *engines*: PhysX, Bullet, SOFA e nossa implementação de Massa-Mola. Buscamos entender como cada *engine* contorna o problema de estabilidade do modelo.

A.1 Modelos de Deformação

Modelos baseados em deformações físicas são importantes para criar representações virtuais de estruturas sólidas deformáveis. Em muitas aplicações, onde o usuário interage com o mundo virtual, é necessário que estes objetos comportem-se com um elevado grau de similaridade do mundo real. Discutiremos os métodos mais usados para modelagem de deformações.

A.1.1 Massa-Mola

O Massa-Mola é descrito como um sistema regido por uma equação diferencial ordinária que usa métodos matemáticos para resolver aproximações numéricas (PROVOT, 1996; GIBSON; MIRTICH, 1997; BOURGUIGNON; CANI, 2000; GEORGII; WESTERMANN, 2005; SELLE; LENTINE; FEDKIW, 2008; MESIT; GUHA; FURLONG, 2010). O método usa um objeto discretizado e, a fim de simular o comportamento de deformação, usa equações de movimento aplicadas sobre as massa e molas do modelo. Neste modelo, uma massa é a informação de cada nodo do objeto, sendo armazenado valores de posição, velocidade e aceleração. As molas são conexões entre os nodos vizinhos, cada mola também contém suas características, como valores de rigidez e da distância nominal entre os nodos.

A.1.2 Método dos Elementos Finitos

Diferente do método Massa-Mola, o Método dos Elementos Finitos funciona sobre a mecânica contínua, aproximando funções de interpolação (BATHE, 1982; JOHNSON, 1987; CHEN; ZELTZER, 1992; BRO-NIELSEN; COTIN, 1996; YIN et al., 2009; KAUFMANN et al., 2009a; BECKER; IHMSEN; TESCHNER, 2009; KAUFMANN et al., 2009b). O método utiliza de um *grid* irregular para simulação de objetos deformáveis, definindo o objeto como um conjunto de elementos finitos. Este método computa uma função para cada elemento, a qual satisfaz uma equação de equilíbrio. Contudo, tal método tem maior custo computacional comparado ao Massa-Mola, tornando-se pouco viável em situações onde o tempo é uma preocupação.

A vantagem do Método dos Elementos Finitos é a facilidade em tratar casos como: objetos com geometria complexa, condições de contorno gerais, ou propriedades de materiais não-lineares.

A.1.3 *Green Coordinates*

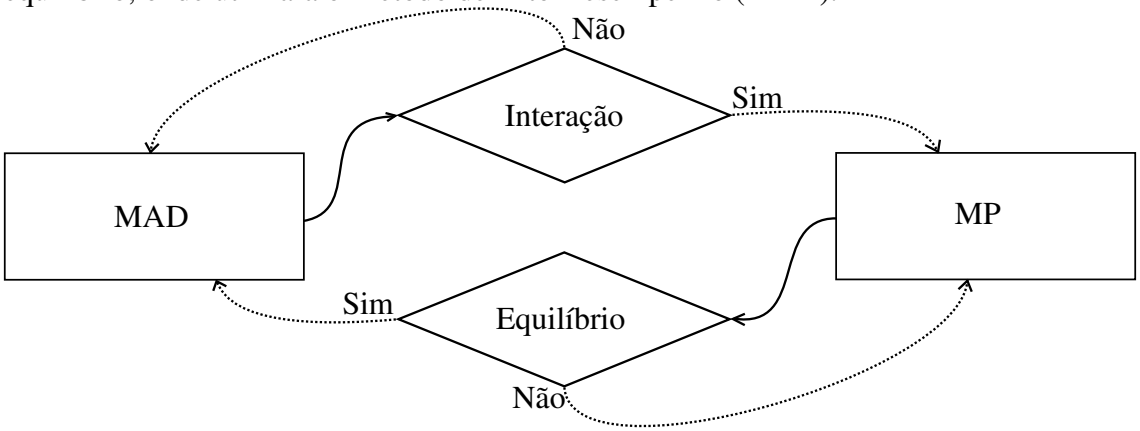
Green Coordinates é um método não-baseado em física (LIPMAN; LEVIN; COHEN-OR, 2008). Sua modelagem baseia-se em uma *cage* que, ao sofrer mudança na sua estrutura, deforma o objeto envolto. A *cage* é uma espécie de malha que envolve o objeto, a interação de forças são aplicadas sobre ela deformando o objeto em questão. O método utiliza de coordenadas baricêntricas generalizada, computando diferentes pesos para cada nodo do objeto. Desta forma, o *Green Coordinates* é capaz de obter um alto desempenho, resultando em uma deformação realisticamente plausível ao objeto.

A.2 Uma Aplicação de Simulação Cirúrgica Minimamente Invasiva

Os três métodos de deformação abordados, dois baseados em física e o último não-baseado em física, oferecem diferentes níveis de manipulação. Apresentaremos nosso ambiente dinâmico, que se adapta de acordo com a necessidade de cada modelo.

A Figura A.1 mostra um diagrama dos possíveis estados de acordo com os diferentes métodos de deformação. No diagrama, o "Método de Alto Desempenho"(MAD) é o método *Green Coordinates*, enquanto que o "Método Preciso"(MP) pode ser o Massa-Mola ou o Método dos Elementos Finitos (de maior precisão e custo computacional), dependendo da importância do objeto. O usuário (ou aplicação) especifica quais órgãos são os mais relevantes, estes serão modelados usando o Método dos Elementos Finitos (quando sob interação), enquanto os demais serão modelados utilizando Massa-Mola.

Figura A.1: Diagrama para mudança de estado do modelo. Quando ocorre interação via interação do usuário, há uma mudança de contexto, fazendo com que o sistema use um Método Preciso (MP) até que a interação termine e o modelo volte ao seu estado de equilíbrio, onde utilizará o Método de Alto Desempenho (MAD).



A.3 Resultados

Submetemos o ambiente proposto a dois cenários, onde avaliamos sua eficiência e eficácia. O primeiro cenário consiste na deformação de dois cilindros dispostos lado a

lado. Já no segundo cenário, selecionamos alguns órgãos (fígado, estômago, pâncreas e baço) para simular uma situação próxima a um procedimento cirúrgico.

Para cada um dos cenários foram avaliados o tempo de cada iteração, bem como o número de iterações necessárias para convergência do método. Também foram analisados os tempos de computação para cada iteração. Estes tempos são indispensáveis para mensurar taxas de atualização, tanto no *rendering* do modelo, bem como para retorno de força.

Os resultados obtidos neste trabalho não fizeram uso de qualquer otimização paralela. Além disso, funcionalidades adicionais foram implementadas ao sistema, tais como: colisão, interação, deformação, difusão de calor, e outras.

Foram obtidos resultados expressivos no ganho de eficiência, com uma perda aceitável de precisão. Dentre os testes realizados, foi observado que o uso do método *Green Coordinates* proporcionou um ganho de aproximadamente trinta vezes maior se comparado com o Método dos Elementos Finitos, e de 2 vezes quando comparado com o Massa-Mola. Avaliamos também a eficácia do método *Green Coordinates*, utilizando a média do erro quadrático para análise de sua aproximação. Neste caso, obtivemos um erro tolerável de 5%.

A.4 Conclusão

Nosso método aborda o *gap* entre desempenho e precisão combinando diferentes métodos de deformação (físicos e não-físicos). Desta forma, é possível proporcionar um cenário realista com grande imersão ao usuário. Os métodos baseados em física são regidos pela teoria da elasticidade referida como a lei de Hooke, proporcionando um comportamento de alta precisão em objetos simulados. O método não-baseado em física é representado por uma generalização de coordenadas baricêntricas, e é utilizado para a alcançar um maior desempenho enquanto assegura uma deformação plausível.

Nosso sistema modela cada objeto do ambiente baseado na sua relevância durante a aplicação. Na simulação, objetos que não sofrem interação são modelados utilizando o Método de Alto Desempenho, economizando recursos computacionais.

A detecção de colisão é independente dos métodos de deformação, mas funciona de forma diferente, dependendo do método utilizado para modelagem do objeto. Quando a colisão afeta diretamente objetos modelados pelo Massa-Mola ou Método dos Elementos Finitos, aplica-se deslocamento aos nodos na forma de *constraints*.Já para o modelo *Green Coordinates*, é calculada a contribuição dos deslocamentos realizados na *cage* para cada um dos nodos do objeto.

O ambiente de cirurgia simulado pelo nosso sistema pode ajudar alunos e especialistas, fornecendo uma forma segura e prática para realizar experimentos de procedimentos cirúrgicos complexos. Nosso estudo de caso não cobre um sistema de simulação de cirurgia completa, mas implementa interações cirúrgicas típicas, como *picking*, queima de tecido e retorno háptico.

A.4.0.1 Trabalhos Futuros

O uso de uma *cage* para aproximar o comportamento de modelos físicos pode ser ainda melhorado, Encontrar parâmetros apropriados para o Massa-Mola da *cage* é uma tarefa difícil e trabalhosa, sem nenhuma garantia de confiabilidade. O uso de um mapeamento entre estes parâmetros é uma direção interessante para se explorar. A criação automática de *cages* mais refinadas é também um aspecto interessante que potencialmente

irá melhorar a aproximação do *Green Coordinates*.

Outra tarefa típica em cirurgias é o corte arbitrário de tecidos. Este recurso não foi tratado devido a alta complexidade envolvida. Quando deseja-se simular tal tarefa, é necessário tratar problemas como a criação de nova geometria, bem como a atribuição de novas conexões físicas, para um comportamento realístico. Este é um desafio que gostaríamos de resolver no futuro.