Simulating Cardinal Movements of Human Labour Using Finite

Element Analysis

Zelimkhan Gerikhanov

Supervisor: Dr. Joost Noppen

School of Computing Sciences

University of East Anglia

Upgrade Report

January 2015

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Computer based simulations find numerous applications in medicine. Such applications include training of medical personnel, diagnosing patients based on digital data and scientific research to gain better understanding of physiological phenomena. Bio-mechanical simulations are one of the most challenging uses of computer based simulation in medicine. The underlying principles are very complex and thus require sophisticated theoretical formulations and considerable software development efforts.

One of the areas where bio-mechanical simulations are very important is obstetrics related medical simulations.

The main focus of this project is creating a realistic real-time computer based simulation of human childbirth. Simulations of this kind require modeling biomechanical interactions of high complexity. As mentioned, this implies highly sophisticated. This report will therefore attempt to cover the important aspects of human childbirth.

1.2 Human labour

1.2.1 Cardinal movements

Investigating the mechanisms of labour is an important preliminary step before designing childbirth simulation software. The process of childbirth is complex and involves a multitude of different mechanical processes. Gabbe et al. (1991) describes the cardinal movements as the main mechanisms of labour and lists seven distinct movements: engagement, descent, flexion, internal rotation, extension, external rotation (restitution) and expulsion. Liao et al. (2005) gives the following definitions to the movements and provides figures:

1. Engagement – this step is identified by the passage of the widest diameter of the fetal skull through the pelvic inlet (Fig 1.1).

2. Descent – the stage when the fetal head passes further downwards through the cervix towards the birth canal (Fig 1.4).

Figure 1.1: Engagement

Figure 1.2: Descent and flexion

Figure 1.3: Internal rotation

Figure 1.4: Extension

Figure 1.5: External rotation

Figure 1.6: Expulsion

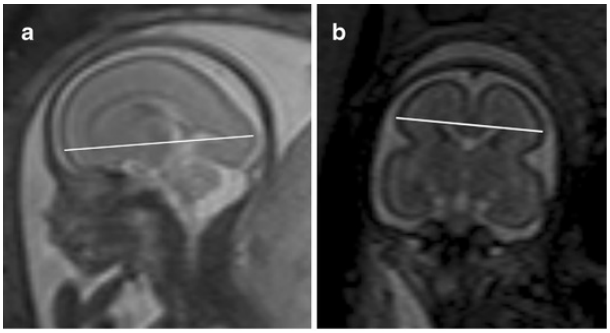

Figure 1.7: Fetal head diameters: a – Fronto-occipital diameter, b – Cerebral biparietal diameter (Canto Moreira et al., 2011).

3. Flexion – is the process when the fetal head flexes with its chin approaching its chest. The shape of the pelvis and the pelvic floor are said to cause flexion. Flexion allows the fetus to pass through the birth canal with a smaller diameter (e.g. the bi-parietal diameter (Fig 1.7) (Fig 1.2).

4. Internal rotation – is also thought to be caused by the shape of the pelvic canal and soft tissues. It is represented by rotation of the fetal head from facing sideways to facing backwards relative to the mother's body. This can be explained by the fact that the pelvic inlet has the widest diameter in the sideways direction, whereas the pelvic outlet is widest in the sagittal axis (Fig 1.3).

5. Extension - happens while the neck of the baby is under the pubic symphysis. During extension the head deflexes and in this stage, chin and head are born facing outwards (Fig 1.4).

6. External rotation (restitution) is represented by external rotation of the head to face sideways as compared to the rest of the body (Fig 1.5).

7. Expulsion - when external rotation is complete and the shoulder has moved under the pubic symphysis the fetus is born (Fig 1.6).

1.2.2 Problematic labour

1.2.3 Childbirth simulators

There are a number of mechanical childbirth simulators that exist already. Such simulators are designed to allow trainee obstetricians and midwifes to interact with a mannequin of a birthing woman. The mannequin is normally made of plastic and/or metal. The section

The motivation behind using a computer based simulation in childbirth modeling is dictated by a number of reasons. While having certain advantages the mechanical childbirth simulators often lack very important features. Primarily, mechanical mannequin will typically be very poorly customizable. Such simulator is typically used for training junior personnel and in many of training cases the simulation scenario is required to be changed based on the type of the case that is being practiced. Using mechanical simulator means that only a specific scenario is available for training with only slight variations. Additional mannequins of a different type will have to be acquired to perform training for alternative scenarios. Contrary to this, computer based simulations allow unrestricted customization. It is also worth mentioning, that there exists a hybrid type of simulators, that combine a computer based underlying bio-mechanical model with an external mechanical mannequin to provide the interface between the trainee and the simulator. Such, simulators combine the advantages of both types of simulators, but also carry the drawbacks of at least one of them.

Computer simulations provide a good tool for representing real world phenomena, but they are only capable of representing the simulated objects to a certain degree of approximation. Better approximations are predominantly much more expensive in terms of computational power. With the increased processing power of modern computers, it is possible to perform simulations with a higher degree of fidelity. However, even the most high-performance machines can struggle with certain types of high-cost simulations. In such cases, we have to utilize the underlying hardware to the highest degree possible. This can be achieved by a number optimization techniques. One of the most effective techniques is using parallel processing in order to speed up the computation.

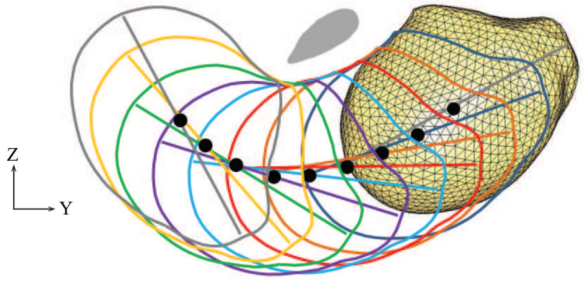

Figure 1.8: An example of an imposed trajectory applied to a fetal head. Image from Dejun Jing, James A. Ashton-Miller (2013).

1.3 Reverse vs Forward engineered approaches

There are a number of existing mechanical and computer based simulators of human childbirth. It is crucially important to contrast the approach chosen for this research project from the existing reverse-engineered simulations.

As will be shown in Section 2.2 there is a large number of simulations using superimposed trajectories on the fetal head. The fetus or just the head is forced to follow a required trajectory without considering the causes of such motions. Even with reverse-engineered trajectories the cardinal movements are in most cases not rendered correctly.

To give an illustrative example of how imposed trajectories are used in such simulations a figure 1.8 from work by Dejun Jing, James A. Ashton-Miller (2013).

1.4 Finite element method in surgical simulations

Finite element method (FEM) is a very useful tool in physical simulations. In particular medical and surgical simulations find a lot of applications for FEM due to its robustness and accuracy.

It is desired to achieve the highest fidelity of the simulations with as little latency as possible. To this end, performance optimizations and GPU utilization for FEM is one of the main focuses of this thesis. Several available application programming interfaces (API's) will be overviewed as the candidate tools for implementation. It is then shown how the chosen API is used to achieve highly

efficient implementation of FEA for soft-tissue simulation.

Another important aspect of creating a computer based simulation of childbirth is acquiring realistic 3D models of the underlying physiological structures. Namely, the fetal body and maternal lower body geometries are required. The possible ways of constructing the required meshes will be covered in this thesis. The approach is not yet decided on.

1.5 Report overview

The organization of the remainder of this report is presented here.

In chapter 2 a literature review is conducted presenting the body of already existing relevant research. The review is split based on relevance to a particular aspect of this research project.

Chapter 3 describes the work that has already been undertaken as of the date of this report.

Three publications have been successfully made in the duration of this PhD course, each of which covered an important topic related to the area of interest of this research. All three papers will be covered in this report. The papers are include in the 7.1.

There is a considerable amount of work still left to complete. The 5 chapter focuses on describing the required steps aimed at achieving the objectives of this research project. An approximate work plan is also presented in this chapter featuring a Gannt chart.

Finally chapter 6 presents a proposed thesis structure for the final write-up.

Chapter 2

Literature Review

2.1 Introduction to Literature Review

The literature review in this report contains an overview of the key literature related to work conducted so far in the area of childbirth simulation.

Literature was primarily found through searches of related keywords in Google scholar and DBLP, along with forwards and backwards citation analysis in addition to targeting specific conference proceedings. It is foreseen that the literature review for the PhD thesis will be significantly more detailed within the sections included in this report as well as containing a number of additional topics.

2.2 Existing childbirth simulation systems

2.2.1 Existing physics based simulations

A number of obstetrics training simulators have been produced over the past decades. They vary in their quality and implementation. By quality we understand fidelity of the simulation presented. The implementation differs in terms of tools used: the first type includes purely software-based simulations and the second includes software simulations that are used in conjunction with real mechanical apparatus.

The work by **Lapeer and Prager (2001)** presents a mechanical model for fetal head moulding. This phenomenon occurs during labour and is essentially fetal head deformation due to labour forces. In another work, Lapeer et al. (2004) present a system developed to provide forceps delivery simulation with augmented reality (AR). They use tracking markers on the forceps to capture input and simulate accordingly. Live video is then combined with rendering of the fetal model to compose an AR simulation. In the consequent paper on the same topic they present a mechanical contact model based on FEA to calculate the effect of the contact forces between the forceps and the skull.

Discussion In both works they present realistic models of the fetal head and its interaction with the forceps. The use of augmented reality brings additional attractiveness to the simulation, as it has more usability in training applications. Moreover, they have managed to solve the double-contact problem, which requires more complex calculations for the FEA. The problem with the simulation is that it fails to achieve real-time refresh rates. However, in the latter work they propose potential solutions to the problem, such as implementing simplified FEA techniques and reducing the model complexity.

Another childbirth simulation using haptic feedback is performed by **Kheddar et al. (2004)**. The system uses imposed trajectories of the fetal decent, but the interaction of the fetus with the pelvic floor is performed based on a physical model. They use a system coupled with a general purpose haptic device to allow the user to interact with the simulated process.

Discussion The virtual hand presents an interest as a mean to provide an

immersive simulation. However, the system in total is limited to simple interactions with the fetus during the descent and also calculating the forces occurring between the pelvic floor and the fetus.

Many childbirth simulators (Moreau et al., 2008), (Buttin et al., 2009), which use a mechanical component, use the BirthSIM system (Silveira et al., 2004). The BirthSIM system allows having realistic haptic feedback using a mechanical part resembling a parturient woman with a mechanically articulated fetus. The trainees are able to interact with the mechanical model and observe the relevant information on the screen of a connected computer.

Work by **Moreau et al. (2008)** uses the BirthSIM system, which allows trainee obstetricians to practice forceps delivery. The software component of the system allows them to observe the process from inside representing the positioning of the forceps relative to the fetus inside the birth canal.

Discussion The simulation system presented in the work provides a suitable solution for forceps delivery simulation purposes. However, the work is more concerned with the labour (uterine) pressures involved in the process. Minimal attention is given to the interaction of the fetal head with the maternal organs.

Dupuis et al. (2009) build upon the simulation system presented in Moreau et al. (2008). The simulation system is successfully used in improving the performance of the trainee obstetricians. The main aim of the discussed paper is to track trajectories of the forceps blades applied by the trainees to identify the problematic points and help improve in educating them.

Discussion The simulation system is as before does not consider the causes of the cardinal movements and is focused away from the topic of this research.

Buttin et al. (2009) present more interesting work where they have managed to model fetal descent during labour without a predefined pathway. They based the model on physical properties of the main bodies in contact with the fetus and the fetus itself. They use FEA for simulating mechanics of the process and couple it with the BirthSIM system.

Discussion The model presented considers the complex interactions of several important organs, which happen to be in contact with the fetus during labour. Additionally, it presents a realistic uterine contraction force model. It is also capable of simulating the whole process of childbirth continuously and performs FEA

based computations of underlying physics. This is a major advantage, which, however, introduces a set of trade-offs. The drawback is the complexity of the computations, which require the system to use simplified representations of the organs and the fetus. Moreover, the bony structures of the bodies are extensively simplified, such that two ellipsoid objects represent the fetal skeleton. In the paper authors do not mention that their simulation shows the cardinal movements. Therefore, it is fair to assume that their approach does not accomplish the task of simulating them.

Hoyte et al. (2008) developed a simulation aimed at identifying the effects of childbirth on levator-ani muscle. Third-party FEA tools were used to perform their analysis. The simulation features a high-quality MRI based pelvic and pelvic floor models.

Discussion The model representing the fetal head in the simulation is extremely rudimentary. A pure sphere is used to represent the head and the volume that it sweeps throughout the simulation is used as the fetal body. Additionally, the path of the sphere is superimposed.

Dejun Jing, James A. Ashton-Miller (2013) present a patient-specific model of the maternal pelvic floor. They also present a model of the fetal head and maternal pelvis. All models were constructed from MRI scans. Constitutative models of visco-hyperelastic materials for FEA are created from data acquire from bi-axial tests on pelvic floor tissues. Abaqus CAE was used to pre-process the FE models and perform the analysis.

Discussion They use superimposed trajectory. A good analysis of the effects of childbirth on the perineal body is presented. The material properties used in the analysis based on actual pelvic floor tissues. They also use a comparatively accurate fetal model. This and the previous work by Hoyte et al. (2008) can serve as a good starting point in designing our simulation contents.

Lien et al. (2004) simulate the levator-ani stretch and its effect on the pelvic floor muscles. Similarly to Hoyte et al. (2008) they use a pure sphere as the fetal head model and apply a superimposed trajectory to simulate fetal descent.

Discussion The difference from other simulations is that the pelvic floor model is not built directly from an MRI or CT, but rather manually created by precisely recreating the underlying fine structures of the pelvic floor and the

vaginal wall. This, however, does not mean that the model is accurate. The model of the pelvic floor is claimed to be extensively customizable. Certain variations corresponding to age, race, etc can be applied. The effects of these variations are then presented.

Parente et al. (2008) used Abaqus to perform analysis on the effects of childbirth on the pelvic floor. The pelvic floor model was created from a MRI scan set, by extracting a point cloud representing the required tissues. The point cloud was then used to build a NURBS surface. The result of this is a very smooth pelvic floor model, as opposed to direct MRI extracted models which very often contain large amounts of imperfections due to bad quality of scans.

Parente et al. (2009) reused the same models and software to study the effects of malpresentation (e.g. occiput-posterior position) of the fetal head on the pelvic floor. The resulting stretch magnitudes are compared between the correct and incorrect presentations. The malpresenting case is shown to have larger stretch values, which indicates higher risks of injury.

Discussion Both works show interesting simulations of childbirth, but they are focused on the pelvic floor muscles and tissues.

Martins et al. (2007) created a simulation that does consider cardinal movements. The models for the fetus and the pelvic floor are reused from Parente et al. (2009). The main focus of the research was in identifying the effects of levator-ani muscle activation during simulated childbirth. Abaqus program is used to perform the analysis.

Discussion The cardinal movements are reverse engineered by imposing rotations on the simulated fetal head. They study the results of cardinal movements on the pelvic floor soft tissues and muscles, rather than the opposite The reverse effects are not considered. It does, however, indicate that the pelvic floor

2.2.2 Mechanical contact problem

The earlier work in the area of computer based childbirth simulations is done by **Geiger (1993)**. In his paper he investigates techniques for building meshes of models based on the Delaunay triangulation approach. The mesh is built from raw MRI data and proves to be sufficiently effective for medical purposes. After the

work concerned with building the meshes he shows an attempt of implementing a childbirth simulation. The simulation is based on the compliance principle. The principle is concerned with optimizing the rotation and orientation of the colliding fetal head so that the contact forces are minimized. The contact forces, in turn, are calculated based on a *surrogate* force, which is based on the depth of the penetration of the contact.

Discussion The author claims that the simulation represents realistic movements of the head. However, as supported by Lapeer (1999), the results seem rather unreliable in terms of fidelity. The fact that arbitrary movement and rotation is applied without proper consideration of contact forces renders the simulation unreliable. The simulation may result in an infinite loop of optimizations, so that the head will continue oscillating. Another important point is that if the 'loop-hole' is not entered this approach will always render the delivery possible. This includes even severe cases of fetal head and maternal pelvis (cephalo-pelvic) disproportion.

Melchert et al. (1995) proposes another FEA based approach to the problem. The contact between the fetal body and the maternal organs is considered. The author performs FEA on the whole body of the fetus and takes into consideration the soft tissues involved in the process.

Discussion The simulation accounts for the whole fetal body and the maternal lower-body. The author claims having a realistic model, however, the paper fails to present reliable validation of the results.

2.3 Conclusion

The main mechanisms of labour, also called the cardinal movements, were shown in this section. Along with the normal process, the section described the problematic cases of labour.

Some of the recent developments in the area of childbirth simulation were presented. It can be concluded that there are promising developments in the field, but more work is required to achieve the high levels of fidelity and comprehensiveness. In particular, none of the reviewed forwards engineered simulations, with the exception of rather unstable simulation of Geiger (1993), managed to

show cardinal movements.

The work by Buttin et al. (2009) presents one of the leading developments in childbirth simulation. However, some of the problems with the work are identified. The most important finding is that such a sophisticated system, but without proper consideration of the bony structures, fails to represent the cardinal movements. It can be argued that this fact supports the proposed idea of the compliance of the bony structures.

Chapter 3

Methodology

3.1 Overview

This chapter covers the work that has been completed to date. The description of the undertaken tasks will be included along the justification for the particular choise of the approach.

It will start by describing the initial developments on creating a forwards-engineered childbirth simulation. Further steps that have been taken on improving the software system are then described. A brief introduction into finite element analysis is given, with particular focus on Total Lagrangian Explicit Dynamic (TLED) scheme. A CPU based implementation of TLED is outlined and then a parallel GPU based implementation is described. Finally, the initial developments in mechanical contact simulation are presented.

3.2 Forwards-engineered Childbirth Simulation

3.2.1 Physics based simulation

Simplified physics models

There are different levels of complexity at which a physics model can be formulated. The same physical concept can be represented with a higher or lower degree of simplified approximation.

Physics based simulations find a great number of applications in various areas. Not all such areas require highest fidelity models. Games and entertainment industry is an example of areas where a simplified model yields sufficiently accurate results with a relatively cheap cost of the computation required.

Simplifying assumptions

The initial attempts on creating a forwards-engineered simulation of human childbirth were focused on using simplified physics model. Due to the requirement on having a childbirth simulation working as soon as possible, a list of simplifying assumptions was adopted.

The list included the following assumptions:

1. The soft tissues of the maternal reproductive system do not need to be considered to observe cardinal movements

2. The trunk of the fetal body does not need to be considered to observe cardinal movements.

3. The labour forces can be replaced by a simple periodical force acting on the atlanto-occipital point (Figure 3.1) of the fetal skull to observe cardinal movements.

The assumptions allowed for a simpler set of requirements for the software implementation aspect of the project and arriving at a working prototype soon.

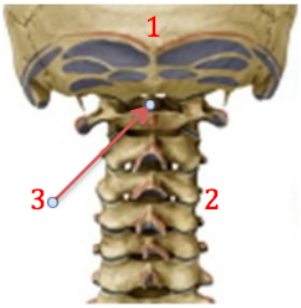

Figure 3.1: The atlanto-occipital point.

3.2.2 IveTrainer childbirth simulation software

Overview

The IveTrainer software was the first attempt on implementing a forward-engineered childbirth simulation. It was limited to the simplified physics model described above.

The resulting software is a simulation tool capable of performing several different simulations. The set of simulations can be extended easily because of the software design pattern utilization. The simulations can be run in the fully forward engineered mode or using imposed trajectories and fetal postures in a reverse engineered way. It provides several input modalities, namely: keyboard, mouse and haptic device.

The use of reverse engineering was justified as the chosen forward engineering approach failed to give sufficiently accurate results. However, as mentioned in the Literature Review 2.2, many of contemporary simulation systems use reverse engineering. Additionally, the two forward and reverse engineered approaches were combined, resulting in a hybrid simulation partially dictated by a predefined trajectory and partially by the underlying physics simulation. Figure 3.2 shows the difference between the fully reverse-engineered and the hybrid approach.

Currently, each simulation consists of three objects and a simulation procedure that defines what interactions between the objects are to be simulated. The three objects are: a fetus, a pelvis and a user controlled hand. In the first simulation, the fetus is represented by its head alone, whereas in the second the full articulated

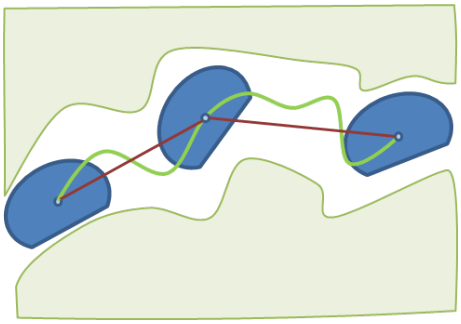

Figure 3.2: The reverse-engineered trajectory in red and the hybrid trajectory in green. It can be seen that the head passes through the specified way-points, but also interacts with passage walls.

fetal body is used. The pelvis is represented by a complex customizable model. The hand is controlled by the user using spatial 3D input from a connected haptic device.

Features

The set of features is currently comprised of means for child delivery simulation for predicting potential outcomes and delivery process visualization for training purposes. The main features of the system in more detail are provided below.

Object Manipulation Object manipulation is a basic, but crucial feature of the system. Considering that the system is meant to be used by medical personal, a user-friendly interface for object manipulation was provided. Thus, the system is incorporated with the means of easy control over the objects' position and orientation using a mouse. This is achieved by implementing an *arcball* input system and a set of keyboard shortcuts.

Cardinal Movements of Fetal Head The first simulation is dedicated to the simulation of cardinal movements of the fetal head only. Provided additional improvement of the underlying physical model and refinement of the set of considered influences, this mode can be used to analyze and investigate the childbirth process and eventually predict the most-likely outcome of a given vaginal delivery.

Customizable Models One of the most important features here is that the system provides a customizable environment. Namely, the main pelvimetric diameters and the diameters of the fetal head can be specified with sub-millimetric accuracy. This may have limited use in predicting real case outcomes, however, it can be used for the purposes of investigation of different child delivery processes.

The coronal, transverse and sagittal extents of the fetal head can be specified, to simulate and analyze how the given diameters affect the outcome of a given delivery. The set of customizable extents for the pelvis consists of the main obstetric diameters, namely: inlet transverse, inlet oblique, inlet anteroposterior, outlet transverse, outlet oblique, outlet anteroposterior and middle anteroposterior diameters. The illustration of all the described diameters will be given in the following sections.

Periodically changing expulsion force The control over expulsion force magnitudes and their temporal change is inspired by the work by Moreau et al. (2008). The authors of this work indicated that the periodical nature of the forces cause the head to progress backwards through the birth canal as well forwards as it generally does. They suggested the potential importance of this phenomenon as an additional factor contributing to a successful delivery.

This idea was addressed while developing the software, by incorporating a sinusoidal signal generator. Given minimum and maximum magnitudes and a direction for the expulsion force, the generator can be used to interpolate between the minimum and maximum force magnitudes. The minimum force magnitude can be set to be negative, which allows the backwards propagation of the fetus.

Control over physical quantities The system allows the user to specify the essential physical quantities involved in the physics simulation. The most important ones include: minimal and maximal expulsion magnitudes, expulsion force change period and the friction coefficient. Each of the listed quantities can be set as required.

Imposed trajectory management The system is capable of forward-engineered simulation that extrapolates from the initial position based on the physical model

and avoids any imposed characteristics. Along with this mode, the system can utilize predefined trajectories of the fetus, which specify the position, as well as the orientation, of the fetus for an arbitrary number of delivery stages.

The trajectory of the fetus consists of a set of stages. Each stage is a momentary snapshot of the fetal position and orientation at any given stage of the delivery. The snapshots are called *way points* and are arranged into a list that forms a *trajectory*.

The system allows convenient management of the trajectories by providing means to create, delete, edit, save and load them. Given a *trajectory* consisting of several *way points*, the user is able to create new and remove, edit or rearrange existing ones.

Hybrid Simulation A compromise solution for the forward-backward engineered simulation was achieved. This implies that the simulation will proceed in the forward engineered mode, unless the simulated fetus deviates from the specified trajectory in terms of position and orientation. This allows current position and orientation to be optimized. Thus, using this approach the system can avoid cases when simulation leads to unrealistic outcomes.

Shoulder Dystocia Visualization It shows how the case occurs starting from the initial stages of labour till the state when the shoulder is pressed against the *symphysis pubis*. The limitation of it is that the body of the baby is completely rigid.

Face Presentation One of the predefined trajectories demonstrates face presentation delivery. As expected the head will pass through the pelvis fully extended, but will be arrested near the outlet.

Manipulating Articulated Fetus The third simulation is a result of further development of the shoulder dystocia visualization described above. The main improvement here is that the simulation has become more interactive rather than purely demonstrative. This is due to the development of an articulated fetus that the user can interact by using a haptic device. This feature required implementing

haptic input management and inverse kinematics. The user is able to use a haptic device to select the head, a hand or a foot of the fetus and move it as required. A set of constraints can be enabled to modulate the IK result. The set can be customized for each joint.

3.2.3 Simulation pipeline

The simulation system adopts a typical update loop architecture. The diagram in Figure 3.3. shows the simplest representation of the simulation.

1. The first step includes all the required initialization and pre-processing. This step is only performed once. It should be mentioned that during this stage the spring object is instantiated and is assigned with two attachments. The attachments are represented by the attached dynamic objects `[U+FFFD]` references.

2. Step two involves complex processing of the geometries and collision detection. This step consists of a loop going through all the simulation objects and pairwise collision detection. The calculated contact information is used to compute the resulting contact force for each contact.

3. Steps 3 and 5 represent the same process of force and moment accumulation. For the linear component the application is a simple addition. It should be noted that the same process is used for contact and spring force application, which contributes towards a more generic physics engine and conforms with the Don't Repeat Yourself (DRY) principle.

$$F_{all} = \sum_{i=0}^{n} F_n \tag{3.1}$$

$$M_{all} = \sum_{i=0}^{n} F_n \times r_n \tag{3.2}$$

4. The fourth step involves calculating the spring force acting on the attached objects. The force is calculated according to the Hooke's law.

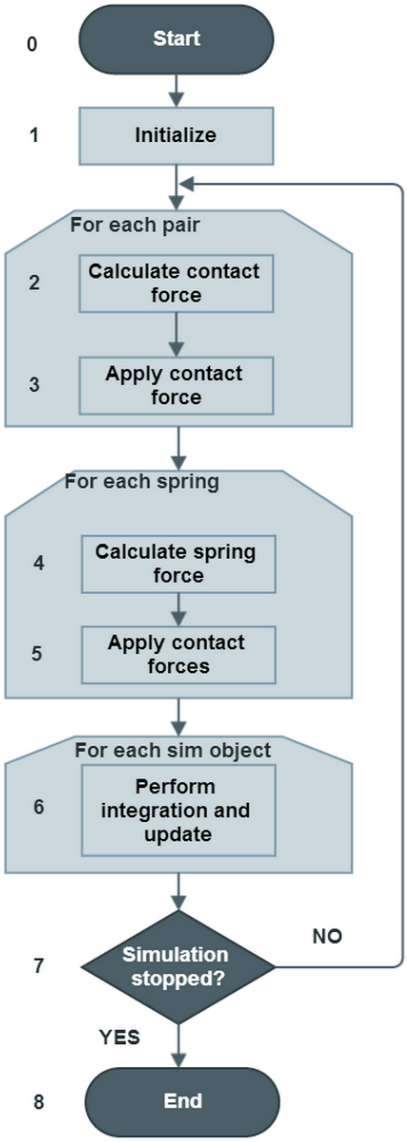
0
Start
1
Initialize
For each pair
2
Calculate contact force
3
Apply contact force
For each spring
4
Calculate spring force
5
Apply contact forces
For each sim object
6
Perform integration and update
7
Simulation stopped?
NO
YES
8
End

Figure 3.3

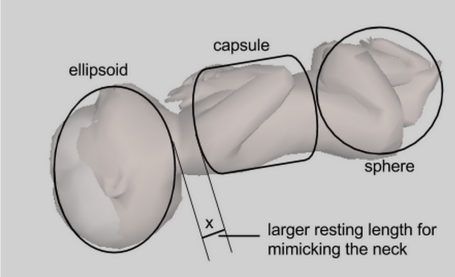

Figure 3.4

5. The resulting spring force is applied to the object in the exact same way as described in item 3. Additionally, according to Newton's third law of motion, the inverted force is also applied to the second object.

Full Fetal Body Simulation

The next step in developing a childbirth simulation system was including the full articulated fetal body. Following the simplified physics modeling approach described before, we developed a spring mass model capable of representing the fetal body to some degree accuracy. The spring mass model consists of the primitives comprising the main sections of the fetal body and a number of springs that connect them. Figure 3.4 demonstrates the sample assembly of the fetal body components.

The mass-spring model is implemented using simple Hookean springs. The forces exerted by the spring onto the bodies that it is connected to is calculated according to Hook's law 3.3.

$$F = -k * x \tag{3.3}$$

Figure 3.5 demonstrates in detail how the forces are calculated. Two bodies A and B are connected by a spring with resting (relaxed) length L_r. The attachment points A'_1 and A'_2 are specified in the local coordinates of the objects that the attachment is on. The world vectors A_1 and A_2 are calculated by transforming the local attachment vectors by the respective transformation matrices of the attached objects. The current length of the spring L_s is defined as the length of the vector S, which connects the attachment points in the world space and is

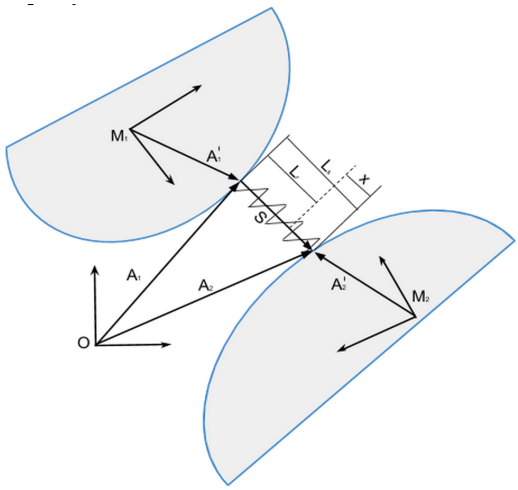

Figure 3.5

equal to $A_2 - A_1$.Thus elongation (compression) x is defined by $L_s - L_r$.

Results and Conclusions

The results of the experiments performed using the described simulation system are described in the published paper Gerikhanov et al. (2013).

The simulation system was successful on displaying the first three cardinal movements. The Engagement, Descent and Flexion were observed in all experiments. As reported by the paper, the Internal Rotation was observed occasionally in cases when the flexion was manually imposed onto the fetal head. When using the hybrid approach, the simulation demonstrated all of the cardinal movements in cases when the number of way-points was sufficient to impose the required trajectory.

Using the simplified full fetal body model did not improve the simulation results. No additional cardinal movements were observed when using the simplified model, but the originally displayed movements were slightly emphasized.

The table 3.1 summarizes the described experimental results.

This simplified forwards-engineered childbirth simulation system is incomplete and requires further development and refinement. This was indeed anticipated as developing a medical simulation software with full functionality can take several years. The resulting system can be seen as an initial step towards a fully

Cardinal movement	Waypoints			
	0	1	2	3
Descent	+	+	+	+
Engagement	+	+	+	+
Flexion	+	+	+	+
Internal rotation	?	+	+	+
Extension			+	+
External rotation				+
Expulsion			+	+

Table 3.1: Initial experimental results

functional system for childbirth simulation and obstetrician training. The following sections describe the steps taken in order to improve upon the described simulation system.

3.3 Simulation System Engineering

3.3.1 Requirements

Based on the results and findings of the experiments presented in section 3.2, we have come with a set of requirements for the next iteration of our simulation system.

It was decided to drop the older IveTrainer software in favor of developing a new system from ground up. IveTrainer was aimed to be a well scalable and extendable simulation system, but during the development many of the features were constructed in a poorly extendable way.

The updated software needs to adopt industry standard software design techniques. The software developed during this project also adheres to the 'good' programming practices. Improved modularity and high coherence leads to unconstrained testing and functionality extension.

The new version of the system is required to be much more scalable and portable. The system is also required to include support for more sophisticated physical models.

3.3.2 BirthEngine with .NET/Mono

The .NET framework for the first iteration of reimplementing the simulation software. The new version of the software was named *BirthEngine*.

Cross-platform runtime

The Mono Framework (**?**) is a cross-platform implementation of Microsoft's .Net framework that allows building and deploying .Net applications on platforms other than Microsoft Windows. By utilizing the framework the BirthEngine simulation system is capable of being run on multiple platforms.

The cross-platform capability of the framework come mainly from the fact that it has an intermediate run-time. As seen in Figure 3.6 the source code is compiled into another form that is then interpreted by the run-time of the target platform.

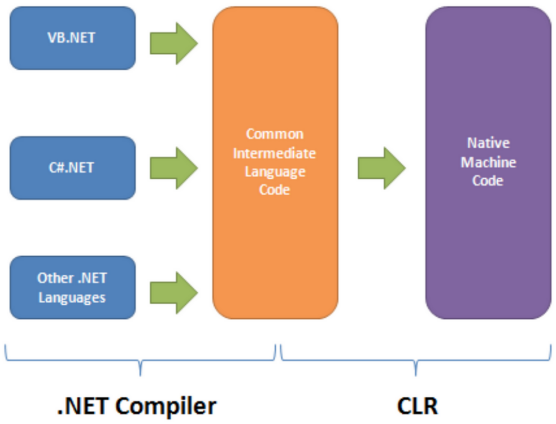

Figure 3.6: .NET/Mono run-time architecture.

Rendering capabilities

Simulation systems often lack the appealing rendering capability of the more graphics oriented applications, such as games. However, we believe that having a realistic graphical representation of the simulation environment carries a great role in immersiveness of the scene, which in turn affects how effective the training is. Therefore, a considerable attention is given the graphical engine of the simulations system.

OpenTK (**?**) is an open-source toolkit that wraps around a number of low level API's. It provides access to the OpenGL, OpenAL and OpenCL API's of the target platform. OpenTK was used in developing the graphical component of the simulation system.

The IveTrainer rendering engine was based on the less efficient OpenGL 2.0 API. The OpenGL 3.0 is used. The choice of using the third version of the API instead of the latest OpenGL 4.0 is justified by the requirement of backwards capability. The simulation system is intended to work even on a less modern hardware, which will not necessarily support the latest OpenGL API.

The chosen API is sufficiently modern to allow more realistic and efficient rendering capabilities. The list of advanced rendering techniques available in software are list here:

- Shaders based rendering for a highly customizable rendering capabilities
- Deferred shading for efficient multi-light scenes
- Volumetric rendering of voxel data using ray-tracing in shaders
- Vertex Buffer Object based mesh rendering to minimize memory transfer times for increased rendering speeds

The rendering engine of BirthEngine is build around the low level API to allow easier and faster creation of graphical representation of simulations. Object-oriented abstractions are built on top of the C API.

Limitations

BirthEngine software is a generic, functional and extendable simulation system. It has been used by a Master student for his dissertation and performed very well in being a generic foundation for his simulation. The language and tooling kit simplify and improve productivity making it possible to develop large amounts functionality in short periods of time.

However, due to the nature of .NET and Mono run-times, as seen in Figure 3.6, the performance of the application degrades quickly with increased computational loads. Particularly, collision detection, which involves a large amount floating point numerical calculations, slows down considerably with increased mesh sizes. We have not performed detailed benchmarking, however we observed that the difference between a native C++ version 3.3.3 can reach 2x timesteps.

3.3.3 BirthView with C++/SDL2

The disadvantages of the BirthEngine simulation system required the next iteration of development be based on lower level *native* technologies. A run-time that has no overhead of an intermediate virtual machine needs to be used.

The next version of our simulation system was named BirthView and was based on low-level C++ language and API's. Precompiled and highly optimized C++ applications perform the best as compared to any other language.

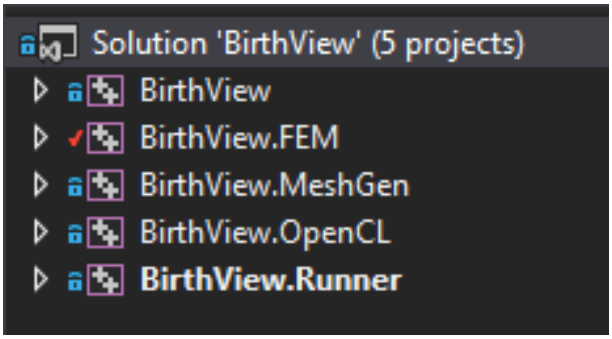

Figure 3.7: BirthView solution's list of plugin projects.

Cross-platform deployment

The Simple DirectMedia Layer (SDL) is a cross-platform graphical application development library Layer (2015). It provides access to low level API's of the underlying operating system in a platform-agnostic way. The same code-base can be used for deploying the application on Windows, Linux, Mac, Android and Ios.

The choice of SDL over other alternatives was mainly due to the fact that it has a wider cross-platform support. Additionally, it is possible to deploy onto mobile devices. In a distant future it is possible that certain aspects of our simulation software maybe ported onto mobile devices.

Modular architecture

BirthView was built with modularity and scalability in mind. The main approach towards a modular structure is plugin based extendability. The additional functionality of the system, which is not its generic core part, is contained in a separate project. As seen in Figure 3.7 the BirthView solution contains several projects each of which represent a plugin extending the generic simulation system.

Currently there are three plugins for BirthView simulation system created:

OpenCL

Contains all of the functionality related to specializing BirthView to use OpenCL for parallel processing. Can be easily replaced by an equivalent plugin bringing in support for alternative parallelization API's like CUDA.

FEM

Contains all of the TLED FEA related routines and data-structures. The justification of separating this functionality into a separate plugin is due to the fact that not all simulations require FEA.

MeshGen

Contains the initial implementation of the intended FE mesh generation functionality. Currently contains rudimentary mesh generation routines.

The same modular architecture is implemented in BirthEngine as well.

Entity Component System

Entity component system (ECS) is a popular framework primarily used in high-performance oriented graphical applications development, particularly in game development. The framework is an example of a broader approach to software engineering known as *Composition Over Inheritance*. The approach is a well known design pattern and can be found in the book (Freeman et al., 2004).

ECS is chosen as the core approach to describing the simulation elements and their behaviors. The main idea is that each simulation object is represented by an *entity*. Typically, all scene elements are entities directly and not through inheritance. However, in cases when certain behavior is commonly present in many entities. Example of such a subclass is *Model* that has the rendering capabilities built-in, as opposed to manually attaching components responsible for rendering.

The class diagram in Figure 3.8 provides an example of how ECS is used to represent a part of a simulation scene. The entity representing the pelvic floor is a subclass of the *Entity* class. It contains several *Component* subclasses each specifying a particular behavior. The behavior of the entity can be easily modified by adding, removing or replacing the attached components. For example, removing *RenderableComponent* instance will make the entity invisible. Similarly, removing the Finite Element

Note that the pelvic floor entity does not not need to be a subclass, as it could simply be an instance of the *Entity* class with an appropriate name and a collection of components.

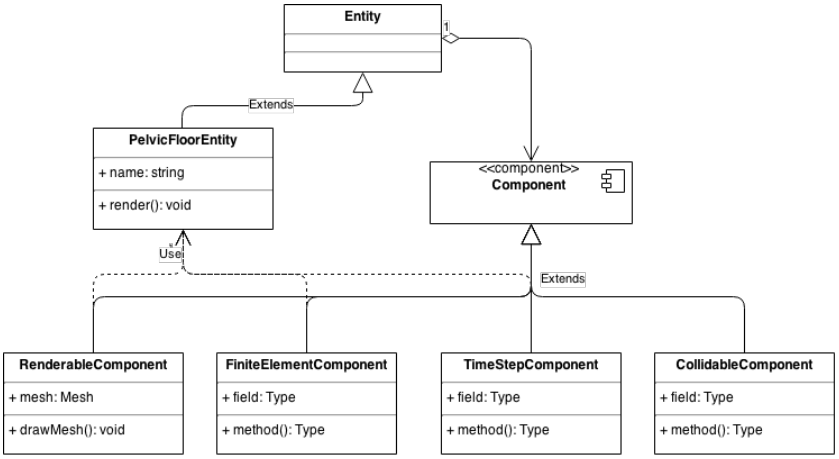

Figure 3.8: Entity component system use in BirthEngine.

3.4 Soft Tissue Simulation using Finite Elements

3.4.1 FEA concepts

Finite element method (FEM) is a numerical method extremely useful in a wide variety of physical simulations. It finds applications in mechanical structural analysis, electrodynamics, hydraulics and many others. Such a wide range of applications arises from the fact that FEM can be used in approximately solving any *field* problem Hutton (2004). A field problem, also known as a *boundary value* problem, is a mathematical problem in which a set of dependent variables must satisfy a differential equation everywhere in the specified domain and satisfy the boundary conditions. *Boundary conditions* are the predefined set of values that are applied to the boundary of the analyzed domain.

The main focus of this research is on mechanical structural analysis and particular in soft tissue simulation. Oden (2010) identifies two key attributes of FEM:

Partitioning

The analyzed domain is partitioned into a set of small non-overlapping domains over which complex function can approximated by local polynomial function.

Weak-formulation

The local functions are not required to satisfy the analytical solution locally, as long as the overall domain integral satisfies the boundary conditions.

The finite sub-domains are also known as *elements*. The subdivision principle is illustrated in Figure 3.9. For a detailed introduction to FEM refer to Hutton (2004).

In the case of surgical simulations FE model needs to predict the deformation *field* within a particular organ that the surgery is aimed at. The *boundary conditions* are the attachment tissues that connect and fix the organ to the body. The *external loads* are the forces and deformations that are applied from the virtual surgical tools when they are in contact with the organ. The governing equation that the method is aimed to solve is:

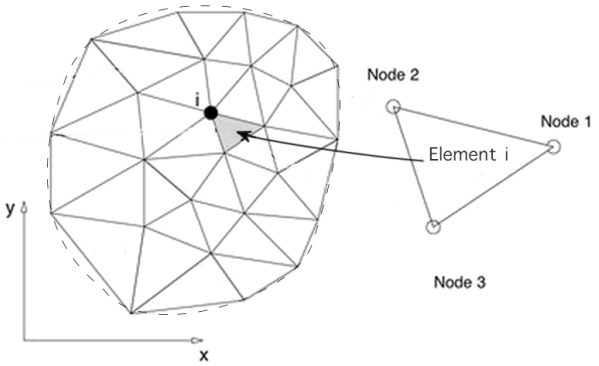

Figure 3.9

$$\mathrm{M}\ddot{U} + \mathrm{D}\dot{U} + K(U)U = \mathrm{R} \tag{3.4}$$

where M is the mass matrix, U are the deformations of the FE nodes, D is the damping matrix, R are the externally applied loads. The $K(U)$ component is a stiffness matrix represented by a function of the deformations.

3.4.2 Total lagrangian explicit dynamic FE

Total lagrangian explicit dynamic (TLED) is a variation on the Lagrangian formulations of FEM. It is initially proposed by Miller et al. (2007) as a means for efficient computer based finite element analysis. The alternative approach is known as the Updated lagrangian explicit dynamic approach is based on using the previously calculated configuration of the deformed body in order to calculate the stresses occurring in the finite elements. In contrast to this technique TLED uses the original reference configuration to perform the calculations. TLED has the advantage of allowing precomputations of a large chunk of data as compared to the approach using the updated lagrangian.

TLED is a very efficient explicit FEA algorithm that is very useful in surgical simulations. The details of the algorithm and the derivation of the formulation are presented in paper Miller et al. (2007)

Discussion The TLED scheme provides very accurate and efficient simulations for soft-tissue simulation. However, as the authors described it, it is more suitable for simulating "very soft" tissues. In cases when the stiffness of the material is higher the minimal time-stem decreases exponentially leading to dramatically decreased simulation rates.

This makes it impossible to use the approach for simulating bony structures present in childbirth. The fetal head and maternal pelvis deformations cannot be performed using TLED approach. Additionally, it is possible that certain tissues in the pelvic floor may have sufficiently high Young's moduli making it impractical to simulate them using TLED.

Basic algorithm

The algorithm can be divided into 3 distinct steps:

1. Precomputation
2. Initialization
3. Time stepping

Note how the precomputation step is a separate entry which allows excluding the computational effort from the main update loop. The reason why the values can be precomputed becomes clear when the notation used in Bathe (1995) is applied. The 0 on left size of the symbols indicate that the value is for the initial reference configuration which stay constant throughout the simulation.

Here we provide a more detailed break-down of the steps:

1. Preprocessing
   (a) Read the input file and load the mesh along with other assembly information (constraints, external loads, etc).
   (b) For each element of the mesh compute the following quantities
      - Jacobian determinant $det(\mathbf{J})$
      - Spatial derivatives of the shape functions $\partial\mathbf{h}$

- Strain-displacement matrices ${}^t_0\mathbf{B}_{L0}$

(c) Compute the diagonalized (lumped) mass matrix ${}^0M$ representing the mass of the whole mesh

2. Initialization

- Set the nodal displacements $\mathbf{u}$ to the prescribed values and add prescribed external force loads into $\mathbf{R}$

3. Time-stepping

- For each element:

(a) Compute deformation gradient ${}^t_0\mathbf{X}$ based on the previous nodal deformations

(b) Compute the strain-displacement matrix for the current configuration using the transpose of the new deformation gradient ${}^t_0\mathbf{X}$ and the precomputed strain-displacement for the reference configuration ${}^t_0\mathbf{B}_{L0}$:

$$ {}^t_0\mathbf{B}_L^{(k)} = {}^t_0\ \mathbf{B}_{L0}^{(k)\,t}{}_0\mathbf{X}^T \tag{3.5} $$

(c) Compute the Second Piola-Kirchoff stress ${}^t_0\mathbf{S}$ according to:

$$ {}^t_0\mathbf{S} = det({}^t_0\mathbf{X})\,{}^0_t\mathbf{X}\,{}^t\tau\,{}^0_t\mathbf{X}^T \tag{3.6} $$

(d) Compute reaction forces for the nodes of the current element. An efficient way of evaluating the integral in the equation is using a Gaussian quadrature.

$$ {}^t\mathrm{F}^{(m)} = \int_{0_V} {}^t_0\mathbf{B}_L^T\ {}^t_0\mathbf{S}\ d^0V \tag{3.7} $$

- For each node:

(a) Update the deformation of the current node using a time-integration method (e.g. Central Difference Method)

(b) Apply the imposed values to nodal displacements $\mathbf{u}$ and add current external force loads into $\mathbf{R}$

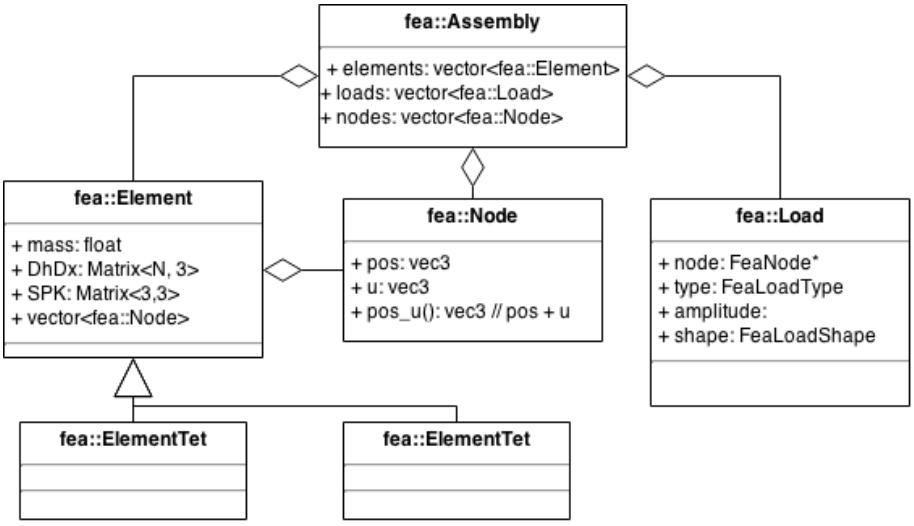

Figure 3.10

3.4.3 CPU based implementation

Our CPU based implementation of the algorithm is essentially a direct translation of the above algorithm into C++. By using appropriate data-structures and custom built matrix routines we were able to achieve basic TLED functionality.

Data-structures

Figure 3.10 gives a basic overview of the data-structure framework used in our TLED implementation.

Traditional FEA simulation

Traditionally FE analyses are performed on a particular setup with fixed parameters. Time frame is one the most important aspects of the simulation. To perform an analysis a starting time and a finishing time are normally chosen and the simulation is commenced. The natural and essential external loads are varied throughout the timeframe according to well defined laws and functions. This kind of simulations are done in the modern FEM packages like Dassault Systems Abaqus or Siemens

The developed BrithView TLED simulation system is capable of performing this type of a simulation. The dedicated component class called *FixedTimeSimulationComponent* can be used to perform a fixed-time simulation.

The advantage of such type of simulations is apparent when a highly controlled analysis environment is required. When testing the stability and strength of mechanical structures such simulations allow greater control and repeatability of the analyses.

However, there are cases when the analysis is required to flow continuously throughout the simulation. For such cases we have developed an alternative simulation framework described in section 3.4.3.

Interactive real-time simulation

The typical scenario of a training session will involve user input. Due to the lack of control over what the input is, various aspects of the simulation remain completely unpredictable. User input can be erroneous and unexpected, especially in the case of a novice trainee. The non-interactivity makes the traditional simulations unusable in cases of real-time training sessions.

In contrast to the previously described type of FE simulation, an interactive simulation is not limited to a fixed timeframe. The simulation is run continuously throughout the training session and the input from the user is continuously transformed into external loads acting on the assembly.

Experiments

A simple test scene was created where a single soft cube is stretch in the positive x axis by an essential load. An essential load is applied directly onto the nodes as a displacement, without involving integrating forces.

Figure 3.11 shows the experimental results reported in Miller et al. (2007). A simple 1000 element (10 by 10 by 10) cube is fixed on one side. The opposite side is stretched by 30% of cube's dimension along the direction of the stretch. We replicated the identical experiment so to simplify validation by having identical models.

Figure 3.11

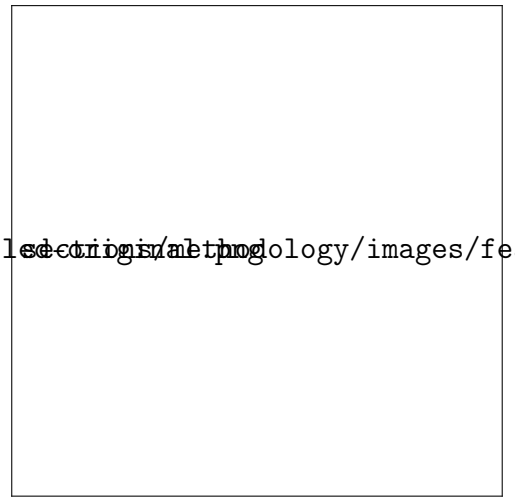

Figure 3.12

More detailed benchmarking and validation is required to be done in the future.

3.5 Parallel FEA Simulations Using GPU

3.5.1 General purpose computation with GPU's

General-purpose computing on graphics processing units (GPGPU) is the technology that allows utilization of the Graphical Processing Units (GPU's) non-graphical computational tasks. Traditionally, GPU's are only used for rasterization and per-vertex transformation in graphical applications. However, the high-throughput hardware comprising the GPU's is useful not only for rendering purposes. Assuming that the computations can can be subdivided into considerably small chunks the processing can be parallelized using the many-core GPU architecture.

An algorithm is run in parallel on a set of object. In the context of GPGPU such algorithm is named a *kernel*. Each of the instances of the kernel is being run in a single *thread*.

Finite elements is an example of a task that requires large amounts of processing. In the case of a real-time interactive simulation, the quality of the resulting simulations may depend on the maximum rate at which a simulation can be performed. Therefore, considering that explicit FEM schemes and especially TLED is remarkably parallelizable GPGPU implementation presents great potential on accelerating the simulation rates.

The power GPGPU can be accessed through a number of available API's. This section is aimed at discussing the available options and justifying the final choice of API for parallel implementation of the TLED scheme. The details of the implementation are provided further.

3.5.2 C++ AMP, OpenCL and CUDA

C++ AMP

C++ AMP is an abstraction layer over CUDA, OpenCL and traditional operating system threading API's. It allows simplifying the computation set-up procedures. It can be used to parallelize code regardless of the underlying architecture or hardware, which means that both multi-core CPU's and many-core GPU's are

accessible.

Another important aspect of C++ AMP is that the parallel kernels are written within platform C++ code with minimal changes to the language. This is another advantage for the choice of this framework as the parallelization library for the TLED GPU implementation.

Discussion It has been reported Wong et al. (2010) that the abstraction layer carries a performance overhead. Also, we were unable to find any existing C++ AMP finite element implementations. This may indicate that the abstraction layer actually complicates FEM implementation, as the developer requires low-level access to the hardware.

CUDA

CUDA is a proprietary NVidia **?** GPGPU parallelization API. It is very popular in many areas including scientific research. This is due the fact that it provides a usable interface. Like C++ AMP it also provides a much simpler integration with host machine code. The parallel kernels can be written inside C++ code using the $\_\_global\_\_$ keyword. This makes developing parallel applications much simpler.

Discussion High accuracy and performance make the it appealing as the target parallelization library. However, the proprietary nature of CUDA makes it less attractive as it implies non-optional requirement of using an NVidia GPU. Also, worth mentioning that only GPU parallelization is supported. Another important fact is that CUDA has already been used to implement TLED as presented in Johnsen et al. (2014). We believe that an alternative implementation may present more scientific merit.

OpenCL

OpenCL is an open-source cross-platform parallel processing API supervised by Khorons Group **?**. The API is similar to OpenGL and provides low-level access to the hardware. It is different from the other API's discussed above. It does not allow embedding kernel code inside. This, however, can be fixed by using third-

party tools Lawlor (2003) that extract and compile OpenCL kernels from the host code. It is however similar to C++ AMP in terms of following a heterogeneous parallel computation model Stone et al. (2010).

Discussion Due to the fact that it is a lower-level heterogenous API, OpenCL has certain drawbacks. The fact that the parallel kernels need run on the wide range of supported hardware, implies that all OpenCL code needs to be loaded dynamically and precompiled on application start-up. However, this is a minor complication, compared to its advantages. The favor is given to OpenCL to its cross-platformity and high-performance. Additionally, there are yet to be created any parallel implementations of TLED scheme using OpenCL.

3.5.3 TLED using OpenCL

The implementation by Johnsen et al. (2014) uses CUDA to parallelize their FE simulation routines and offload them to the GPU. It has been decided that OpenCL presents a more appealing alternative for the API of choice.

Adapting the TLED algorithm

Splitting work The algorithm presented in section 3.4.2 does not account for a parallel implementation. The whole of the algorithm cannot be run in parallel in a single OpenCL kernel. As seen on Figure 3.13 The internal force calculation needs to be in a separate kernel and be completed before the *deformation update* kernel can be run. The important difference and ultimately the reason for the separation is in the fact that the two kernels are run on different objects. The internal forces are calculated per element and thus the kernel is run in n threads, where n is the number of elements in the finite element mesh. The deformation update kernel however uses the accumulated force to integrate over time and update the deformation of every node. The latter kernel is run n times, where n is the number of nodes in the finite element mesh.

Pure C kernels The chosen OpenCL API is most suitable in parallelizing pure C kernels. The CPU version of our TLED implementation is C++ based. This

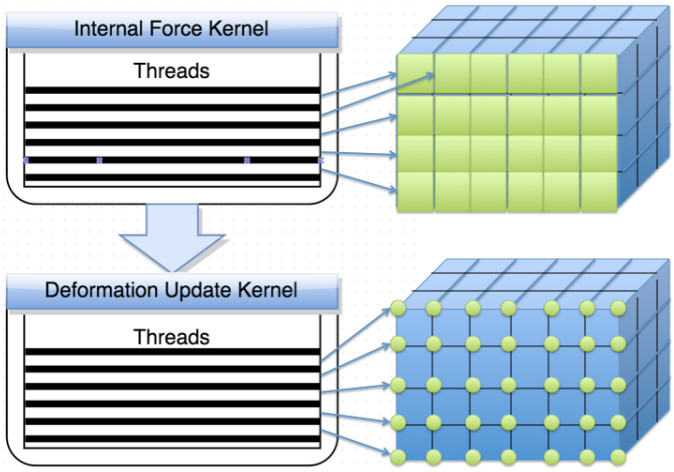

Figure 3.13: GPU based implementation algorithm flow. Element-wise internal force calculation needs to be run in a separate thread, before performing per-node deformation update.

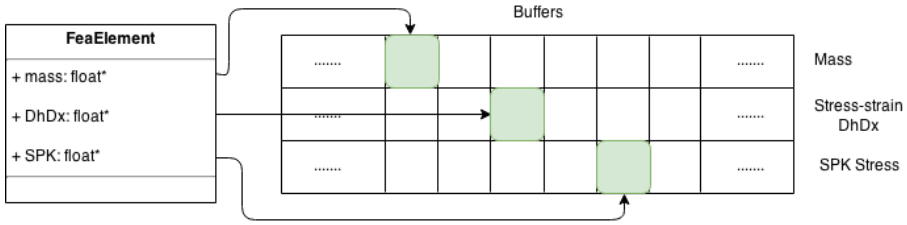

Figure 3.14: Modified data-structure layout for the GPU based implementation following *structs of arrays* principle.

required converting the internal force calculation part of our algorithm into a pure C version. This version was then included into an OpenCL kernel.

Flat data-structures It is considered best-practice to avoid using standard data-structures used in CPU programs when performing GPGPU computations Munshi et al. (2011). Therefore, all data-structures have been redesigned to incorporate buffered data, also known as *structs of arrays* paradigm. The implementation of the principle in our system is illustrated on Figure 3.14. The class representing a finite element does not contain the data, but only points to the location in the buffer containing all the data of a particular class field.

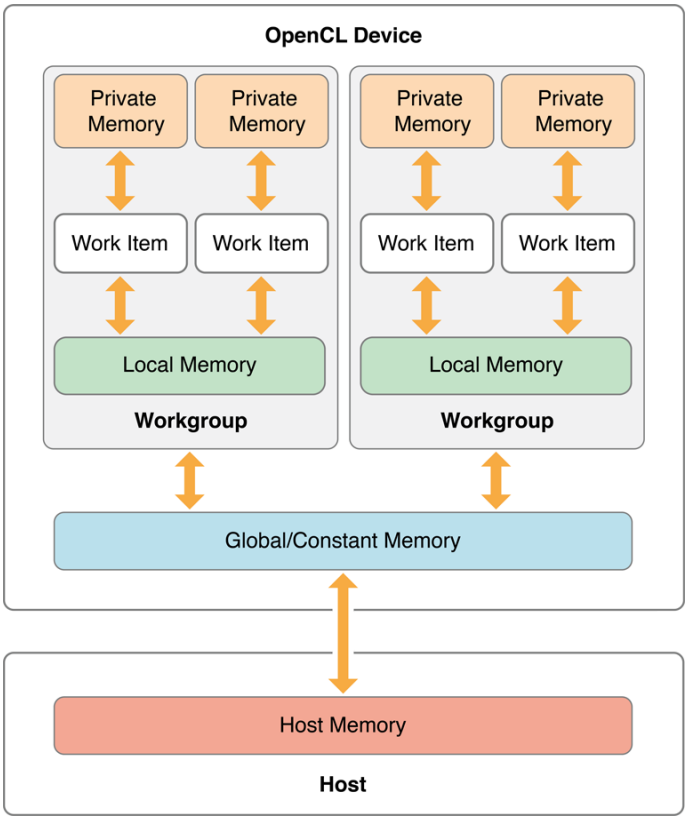

Figure 3.15: OpenCL memory model. The private memory provides small by very fast memory. The global memory is accessible from all threads, but carries large access time overhead.

Efficient memory utilization

The GPU implementation of the TLED force calculation in the OpenCL kernel is not memory bound but computation bound. However, it is important to minimize the memory access overhead, as otherwise performance may suffer considerably.

All modern processing units have a memory hierarchy. Typically the lowest level of memory hierarchy is the global omni-accessible shared memory. The highest in the hierarchy is the private memory accessibly only to the core that it belongs to. OpenCL API provides a specific way of using the GPU memory hierarchy Munshi et al. (2011) as seen in Figure 3.15. The private memory provides small by very fast memory. The global memory is accessible from all threads, but carries large access time overhead.

Our implementation utilizes this hierarchy by avoiding constant requests to the global memory. The *internal forces* kernel has several buffer inputs in global memory. Certain chunks of the buffers are accessed repeatedly in the kernel. To avoid memory access overhead we copy the whole chunk into the private memory of the thread for further access in the thread.

Data-race conditions

Any parallelized algorithm that requires writing to a global location in memory suffers from data-race problem. The issue occurs in the case of two or more threads accessing the same location in memory, when at least one of them is modifying the said memory stored at the location. The non-deterministic result of the operation is also known as *undefined behavior*. It should be noted that reading from a common location does not carry any of the overhead.

Figure 3.16 illustrates the problem. The threads representing parallel processes perform read and write operations. However, the operations are performed on different occasions and no issues arise. However, in the unpredictable even when two threads modify the same memory location based on the current value (consider C++'s $+=$ operator) the result of the operation in both cases depends on which of the threads was first. The access order information is deliberately inaccessible.

In the particular case of the the OpenCL TLED kernel the only place when the data-race condition occurs is when the calculated forces are stored for each node. Note, that this problem arises from the fact that some of the elements share nodes. This means that when two or more element threads will try to modify the force value for a particular node.

In order to overcome this issue use a similar approach to the one used in Johnsen et al. (2014). The approach is to replicate the nodal force information for each element. This means that the amount of data required is proportional to the number of nodes in the type of finite element used to represent simulated mesh.

We adopt a simpler memory layout for replicated nodal forces buffer as compared to the one used in Johnsen et al. (2014). The paper states that they use

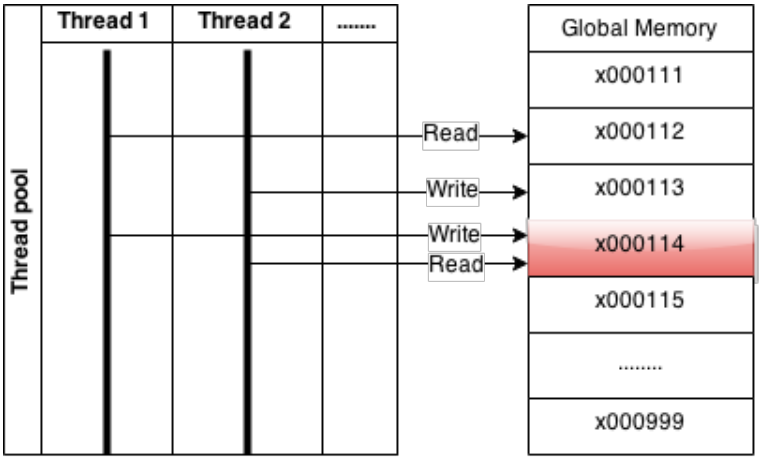

Figure 3.16: Data-race condition in a parallel application when two or more threads are attempting to access a memory location with one of the threads making a modification to memory.

a lookup table to find the addresses of the nodal forces. The memory layout adopted in BirthView is continuous and the nodes of a single element are adjacent in memory. Lets consider the instance of a hexahedral mesh, where each element consists of a eight nodes. This results in an eight fold increase in the amount of memory required to store the nodal force values. Fortunately, the modern hardware is capable of storing considerable amount of data. As seen in Figure 3.17 each element corresponds to a set of eight nodal forces.

Figure 3.17: The memory layout of the nodal forces buffer.

Chapter 4

Mechanical Contact

4.1 Collision Detection

4.1.1 Overview

This section is dedicated to collision detection and other relevant concepts. Starting with the general description of the process it identifies the main problem that occurs when dealing with collision detection. It also provides description of the various solutions to the problem, such as: utilizing spatial decomposition techniques and using parallel processing.

4.1.2 Description

Collision detection is essentially the process of identifying if two given objects are colliding. The term collision determination refers to the process of identifying the exact point of collision and potentially the interpenetration of the two objects. However, generally, collision detection includes the process of collision determination as well.

In case of this particular project, we consider collision detection between polygonal meshes representing the main acting bodies during the childbirth process. The polygonal meshes consist of a number of (generally triangular) polygons arranged to represent an approximation of a given object. It is also worth mentioning that, this representation only gives the description of the surface of the

object omitting the internal volumetric structure.

A naïve approach to collision detection involves testing each pair of triangles for intersections. The computational complexity of this process will be $O(n^2)$ (Harris et al., 2007), which, in case of models with a high polygon count n, can yield reduced performance hence low frame rates. The most popular solution to this problem is spatial partitioning using Bounding Volume Hierarchies (BVH). This approach subdivides collision detection into two consequent phases. The first process, often called the broad-phase, involves pruning of the more distant (unlikely to collide) polygons from the list of all polygons to undergo the intersection tests, leading to reduced complexity $O(n\ log\ n)$ (**?**). After a small number of potentially colliding polygons is identified, the narrow-phase collision detection takes place. This phase involves pairwise intersection tests on the set of identified triangles. A more detailed description of the whole process is given in the following sections.

4.1.3 Broad Phase

As mentioned above, medical simulations deal with high polygonal meshes and require implementing a BVH. There are several types of BVH's, which differ in terms of bounding volume shapes and degrees. Here we will give descriptions for the different types of BVH's and provide justification for the option chosen to be implemented.

Bounding Volume

The term Bounding Volume (BV) refers to the simpler volume, which can be used to approximate the position and shape of a more complex triangular mesh. There are several options for a BV shape: sphere, axis-aligned bounding box (AABB), oriented bounding box (OBB), discrete orientation polytope (k-dop), line sweep sphere or convex hull (Nguyen, 2006). Figure 4.1 illustrates the listed bounding volumes. A Bounding Volume Hierarchy (BVH) is a hierarchical collection of BV's, which can be represented as a tree.

The bounding sphere has equal extents in all possible directions from its centre, which are equal to the radius of the sphere. AABB is represented by a box,

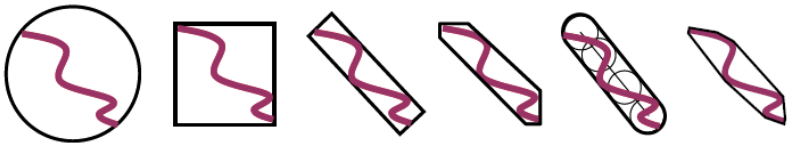

Figure 4.1: Different bounding volumes enclosing the same shape. From left to right: bounding sphere, AABB, OBB, k-DOP, line sweep sphere, convex hull from (Nguyen, 2006).

which has three extents: width, length and height. The box is always oriented to match the axes of the world coordinate system. The difference between OBB and AABB is that OBB can have any arbitrary orientation. The k-dop's can have k number of vertices (in 2D) and represent a BV with arbitrary complexity, thus providing an adaptive solution (Ericson, 2004). Convex hulls provide accurate approximations for the objects, which are always convex (Chazelle, 1991).

Each of the BV shape options have advantages and disadvantages in terms of intersection testing and tightness of fitting. The bounding sphere and AABB allow simpler intersection tests, but suffer from poor fitting. OBB's, k-dop's and convex hulls provide tighter bounding, however require complex intersection tests. Figure 4.1 demonstrates the difference in fitting for the different BV shapes.

Degrees of the Tree

Another aspect of the BVH is the maximum number of children that any given node in the tree can have. This number is often called the degree of the tree. There are several popular types of BVH's that are represented by trees with different degrees: binary tree, quadtree and octree. A binary tree has the degree of two allowing a maximum of two children for each node; four and eight children for the quadtree and octree respectively.

In our simulation system we built two types of BVH collision detectors. The two differ in the degree of the tree that they represent. The Bitree and Octree

Intersection Detection

The main characteristic of a BVH is that the parent node encapsulates all of its children, therefore if a BV is disjoint with the parent it is also disjoint with all

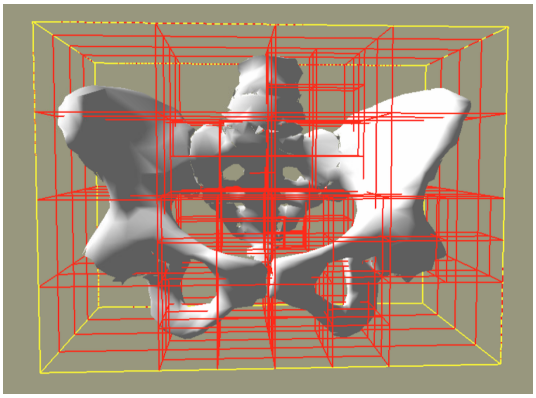

Figure 4.2: A model of a pelvis subdivided our Octree subdivision algorithm.

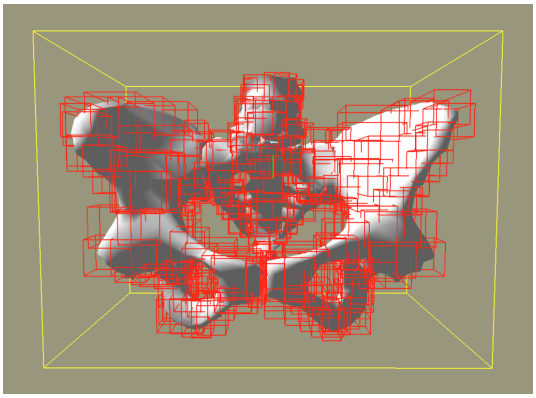

Figure 4.3: The same model subdivided using our Bitree algorithm.

its descendants. This property can be used to prune a large number of polygons that cannot intersect. Thus the process of collision detection between two BVH's is performed by traversing both trees to identify intersecting leafs. The following pseudo code can be used for finding intersecting nodes of the two BVH's. The *broad* phase ends when two intersecting leafs are found.

Analysis

The AABB approach was implemented initially, as it is widely used in interactive graphical applications like games and simulations (Tu and Yu, 2009). This approach works well for the simulation where there are no rotations involved. The nature of the childbirth simulator requires the objects to be rotated in 3D space. As stated before AABB's are always aligned with the world coordinate system and when the bounded object is rotated they can fail to encapsulate the object properly and need to be rebuilt.

Using OBB's allows arbitrary rotations of the objects, by changing their orientations. However, as already mentioned, using OBB's instead of AABB's requires more complex intersection tests. The AABB's intersection tests requires only a three axis test, whereas OBB's require 15 axis tests for the axes of each OBB and their cross products (Ericson, 2004).

The orientations of the tested OBB's are usually represented by matrices as

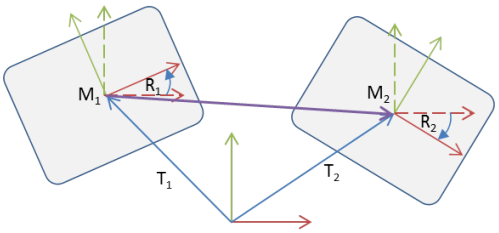

Figure 4.4: Global and local coordinate frames of OBB's are shown with dotted and solid arrows accordingly.

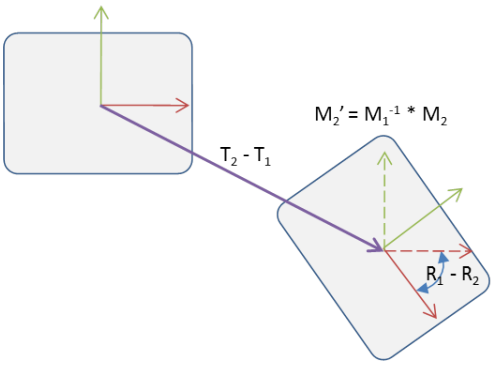

Figure 4.5: Representing the right OBB in the coordinate frame of the left.

suggested in (Ericson, 2004). One way to perform the OBB intersection tests is by modifying the extent vectors of each of the boxes by their orientation matrix. However, one of the vector by matrix transformations can be avoided if the tests are performed locally to one of the OBB's. This can be achieved by calculating the transformation matrix that represents the first OBB in the coordinate frame of the second and performing the transformation only on the second. Figure 4.5 illustrates the idea. In the cases when the BVH consists of many OBB's this can improve the performance. Equation 4.1 can be used to calculate the localization matrix.

$$M_2' = M_1^{-1} \cdot M_2, \tag{4.1}$$

where M_2 - *from* matrix, M_1 - *to* matrix, M_2' - localization matrix from M_2 to M_1.

4.1.4 Narrow Phase

The *narrow* phase of collision detection involves determining the exact intersections between polygons identified during the *broad* phase. The generally accepted approach for testing polygon-polygon intersection uses the Separating Axis Theorem (SAT). The SAT states that, if two polygonal objects are not colliding, there exists an axis that separates the projections of the objects onto that axis. Thus,

if an axis on which the projections of the objects do not overlap is found, the objects are not intersecting. The important point here is that the number of test axes is fixed and the number of required tests is calculated as in 4.2. For triangles the number of axes is 2 face normals and 9 edge vectors and their cross products.

$$N_t = f_1 + f_2 + e_1 \cdot e_2, \tag{4.2}$$

where N_t - number of required tests, f_n - number of faces of the n-th object, e_n - number of edges in the n-th object.

There are several implementations of the SAT for determining intersection between two triangles and the one presented in (Miller, 1997) presents a suitable solution.

4.2 Basic Mechanical Contact Model

4.2.1 Basic Contact Model

The contact model used in our childbirth simulation system is described in terms of calculating the crucial physical values. The entities are: friction between the fetal head and the pelvis, forces and rotational moments arising from the contact. The Figure 4.6 describes the model in more detail.

Friction

One of the most often used models for calculating friction between two rigid bodies in contact is Coulomb's law of friction (Coulomb, 1785). As illustrated on the Figure 4.7, the law states that the friction force is defined by the normal contact force N times the coefficient of friction (μ). In case of a sliding contact, the friction force is exactly equal to μN by magnitude. The direction is defined to be opposite to the tangential component of the acting force.

The physical model to be used in the childbirth simulation will replace the gravitational force with the uterine contraction force. However, there is a need to specify a suitable friction coefficient for the head-pelvis contact. During the literature review no resources were found where the exact coefficients for this

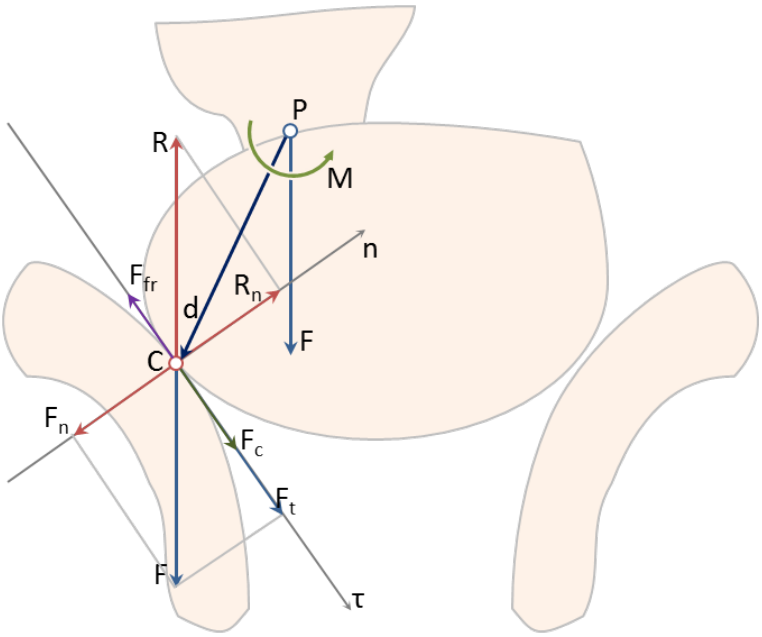

Figure 4.6: A contact between two objects: P – pivot point, C – contact point, F — external (uterine) force, R — reaction (contact) force, F_τ — tangential component, F_n — normal component, n -– contact normal, τ -– tangent plane, d - radius vector, F_{fr} – friction force, F_c – contact force, M – the resulting rotation moment.

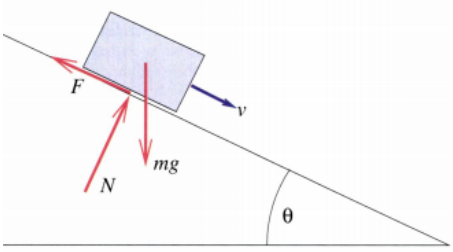

Figure 4.7: Simple rigid body friction problem: F – friction force, N – normal contact force, mg - gravitational force, v - velocity, θ - angle between the surface and horizontal plane (**?**)

particular case are specified. The work by Shockey et al. (1985) performed series of experiments and identified friction coefficients between a muscle and a bone. These values vary under different loads between 0.29 and 0.36. Considering that it is not the bare bones interacting, the friction coefficients of lubricated skin were examined. The works by Asserin et al. (2000) and Sivamani et al. (2003) provide the coefficients, but the values vary widely according to the normal loads and the used lubricants. As it is difficult to choose a single value an average of 0.35 was chosen as the starting option. The value can be changed during further experiments.

Rotational moments

In order to calculate rotational moments caused by collisions it is necessary first to identify a rotational pivot. For the rotational pivot of the fetal head the atlanto-occipital point (Fig 3.1) will be chosen. The rotational moment M is found by calculating the cross product of the contact force F_c vector and vector (d), being the position vector from the rotational point to the point of contact. Equation 4.3 shows the relation:

$$\vec{M} = \vec{F_c} \times \vec{d} \tag{4.3}$$

However, the contact force F_c needs to be calculated first. The force is the resulting force acting on the fetal head during contact. It comprises of the normal, tangential and frictional components.

The normal component is trivial to find by projecting the inverted external force onto the contact normal as in equation 4.4.

$$\vec{F_n} = (\vec{F} \cdot \vec{n})\widehat{n} \tag{4.4}$$

The tangential component F_τ of the reaction force is more ambiguous to find. It is known that the normal component is collinear with the normal and the tangential component will be inside the tangent plane as in Figure 4.8, but the plane contains potentially an infinite number of such vectors. However, considering the intersection of the tangent plane with the plane formed by the normal and the force F it is possible to find the tangent vector. Thus, the direction of the tan-

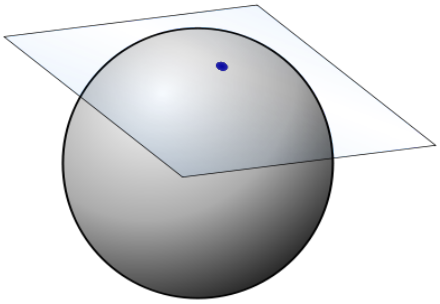

Figure 4.8: Tangent plane of a contact

gential component is calculated from the contact normal and the external force F using equations 4.4, 4.5 and 4.6.

$$\vec{R} = -\vec{F} \quad \wedge \quad \vec{F_\tau} = \vec{F} - \vec{F_n} \tag{4.5}$$

$$\vec{F_\tau} = \vec{F} - (\vec{F} \cdot \vec{n})\vec{n} \tag{4.6}$$

Thus, after identifying all the necessary forces for the contact the resulting contact force F_c is defined as in equation 4.7. Equation 4.3 can be worked out now. The rotation of the object is then calculated based on the combined rotational moments as seen in equation 4.8. It should be noted that each of the moments are calculated for each contact point separately.

$$F_c = F_\tau - F_{fr} \tag{4.7}$$

$$M_{comb} = \sum_{i}^{n} M_i \tag{4.8}$$

4.3 FE Based Mechanical Contact

4.3.1 Principles

Explicit FEA techniques allow access to the forces resulting from the deformations of the finite element mesh. The forces are then used to update the deformations

U with an appropriate time integration technique. This feature results in a useful way to apply external force loads to the mesh, but also provides a way to extract force information at a particular contact point.

The developed TLED framework can be represented as a pipeline as described in 3.4.2. The first occurrence of the nodal forces in 3b is useful to apply external nodes. By the time the execution reaches the end of the time step, the buffer of forces is updated with the contributions from the deformed elements. The simulation module responsible for mechanical contact interaction can access the force data to perform contact calculations.

4.4 Experiment

4.4.1 Solid Sphere and Soft Cube Interactive Simulation

A simple scene was created as an initial testing setup for the TLED finite element simulation. As seen on Figure 4.9 a solid sphere is moved into contact with a cube of a soft material.

Figure 4.9

Chapter 5

Future Work

This chapter outlines future work to be performed as part of the project, split into different areas of focus.

5.1 Experiments

The main target of this research is establishing all seven cardinal movements of human labour in a simulated environment without imposed trajectories. Purely physics based simulation is to be used to accomplish this, as opposed to the hybrid method described in the previous section.

Unfortunately we do not possess any quantitative data on the trajectory of fetal descent that can be used to meaningfully compare with the observed trajectory during our simulation.

Bony structures with a basic pelvic floor model

A scenario where the fetal skull is descending through the maternal pelvis with a simple pelvic floor model attached. This scenario is a potential starting point for simulation experiments. This experiment will be directed at validating the simulation system. The work presented in Parente et al. (2009) represents a very similar scenario, although more simplified. We believe that having a realistic skull model will improve fidelity of the observed experimental values.

5.2 Effects of muscle activation

Work by Martins et al. (2007) bring to attention effects of simulating the contracting muscles while performing the rest of the simulation. It may be of interest to look into how active pelvic floor and uterine muscles affect the propagation of the fetal head through the birth canal.

5.3 Volumetric mesh generation

Mesh generation is designated as one of the higher priority tasks to accomplish. Having acquired or generated high quality tetrahedral or hexahedral meshes is a

crucial requirement in being able to perform the research experiments.

5.4 Improved contact model

Mesh generation is designated as one of the higher priority tasks to accomplish. Having acquired or generated high quality tetrahedral or hexahedral meshes is a crucial requirement in being able to perform the research experiments.

5.4.1 GPU based TLED improvements

Parallelize deformation updating

Currently only the internal force calculation part of the algorithm is performed in parallel. As described in in section 3.5.3, the two parts of the algorithm cannot be run continuously in parallel. A separate kernel and its invocation is required.

Hour-glass artifact compensation

OpenGL and OpenCL shared context

5.4.2 More validation and benchmarking

The accuracy of the proposed implementation needs to be thoroughly analyzed. For that purpose, a similar experimental setups will be created to other implementations and same experiments will be performed. This allows greater accuracy in the results of validation.

Additionally, we need to establish the exact difference in performance. It also important to identify the bottle-necks and slowest points in our implementation of the algorithm.

Chapter 6

Proposed Thesis Structure

1. Introduction
2. Literature Review
    (a) Introduction
    (b) Reverse-engineered Simulators
    (c) Physics Based Simulators
    (d) Finite Element Analysis
    (e) Mechanical Contact Modeling
3. Methodology
    (a) Forward-engineered simulation of childbirth
    (b) Finite Element Method based simulation of childbirth
        i. Theoretical principles
        ii. Explicit Dynamic approaches
        iii. Total Lagrangian Dynamic Explicit FEM
    (c) Parallel implementation of FEM on GPU
        i. GPGPU
        ii. Optimization techniques
        iii. GPU based FEM
    (d) Mechanical contact
        i. Collision detection
        ii. Contact model
4. Experiments and Results
    (a) Validation
        i. Comparison with other FEM packages
            A. Abaqus CAE
            B. NiftySim
    (b) Experiments

i. Head, Pelvis and Pelvic Floor

ii. Full body, Pelvis and Pelvic Floor

iii. Full body, Pelvis, Vaginal Wall and Pelvic Floor

5. Conclusion

Bibliography

Asserin, J., Zahouani, H., Humbert, P., Couturaud, V., and Mougin, D. (2000). Measurement of the friction coefficient of the human skin in vivo: Quantification of the cutaneous smoothness. *Colloids and Surfaces B: Biointerfaces*, 19(1):1 – 12. 56

Bathe, K.-J. (1995). *Finite Element Procedures*. Prentice Hall, 1st edition. 36

Buttin, R., Zara, F., Shariat, B., and Redarce, T. (2009). A biomechanical model of the female reproductive system and the fetus for the realization of a childbirth virtual simulator. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 5263 –5266. 11, 15

Canto Moreira, N., Teixeira, J., Themudo, R., Amini, H., Axelsson, O., Raininko, R., and Wikstrom, J. (2011). Measurements of the normal fetal brain at gestation weeks 17 to 23: a MRI study. *Neuroradiology*, 53:43–48. 10.1007/s00234-010-0772-8. iv, 4

Chazelle, B. (1991). An optimal convex hull algorithm and new results on cuttings. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 29 –38. 51

Coulomb, C. A. (1785). Théorie des machines simples en ayant égard an frottement et à la roideur des cordages. *Mémoires des Savantes étrangers*. 54

Dejun Jing, James A. Ashton-Miller, J. O. D. (2013). A subject-specific anisotropic visco-hyperelastic finite element model of the female pelvic floor

stress and strain during the second stage of labor. *Journal of Biomechanics*, 29(3):997–1003. iv, 6, 12

Dupuis, O., Moreau, R., Pham, M. T., and Redarce, T. (2009). Assessment of forceps blade orientations during their placement using an instrumented childbirth simulator. *BJOG : an international journal of obstetrics and gynaecology*, 116(2):327–32; discussion 332–3. 11

Ericson, C. (2004). *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 51, 52, 53

Freeman, E., Freeman, E., Bates, B., and Sierra, K. (2004). *Head First Design Patterns*. O' Reilly & Associates, Inc. 32

Gabbe, S., Niebyl, J., Simpson, J., and Anderson, G. (1991). *Obstetrics: normal and problem pregnancies*. Churchill Livingstone. 2

Geiger, B. (1993). *Three-Dimensional Modeling of Human Organs and Its Application to Diagnosis and Surgical Planning*. PhD thesis, Sophia Antropolis, France. AAI3235306. 13, 14

Gerikhanov, Z., Audinis, V., and Lapeer, R. (2013). Towards a forward engineered simulation of the cardinal movements of human childbirth. In *E-Health and Bioengineering Conference (EHB), 2013*, pages 1–4. 26

Harris, M., Sengupta, S., and Owens, J. D. (2007). Parallel prefix sum (scan) with CUDA. In Nguyen, H., editor, *GPU Gems 3*, chapter 39, pages 851–876. Addison Wesley. 50

Hoyte, L., Damaser, M. S., Warfield, S. K., Chukkapalli, G., Majumdar, A., Choi, D. J., Trivedi, A., and Krysl, P. (2008). Quantity and distribution of levator ani stretch during simulated vaginal childbirth. *American Journal of Obstetrics and Gynecology*, 199(August). 12

Hutton, D. V. (2004). *Fundamentals of Finite Elements*. Elizabeth A. Johnes, New York, 1 edition. 34

Johnsen, S. F., Taylor, Z. A., Clarkson, M. J., Hipwell, J., Modat, M., Eiben, B., Han, L., Hu, Y., Mertzanidou, T., Hawkes, D. J., and Ourselin, S. (2014). NiftySim: A GPU-based nonlinear finite element package for simulation of soft tissue biomechanics. *International journal of computer assisted radiology and surgery*. 42, 43, 46

Kheddar, A., Devine, C., Brunel, M., Duriez, C., and Sibony, O. (2004). Preliminary design of a childbirth simulator haptic feedback. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3270 – 3275 vol.4. 10

Lapeer, R. (1999). *A biomechanical model of foetal head moulding*. PhD thesis, Cambridge, UK. 14

Lapeer, R., Chen, M. S., and Villagrana, J. (2004). An augmented reality based simulation of obstetric forceps delivery. *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, (ISMAR):274–275. 10

Lapeer, R. J. and Prager, R. W. (2001). Fetal head moulding: finite element analysis of a fetal skull subjected to uterine pressures during the first stage of labour. *Journal of Biomechanics*, 34(9):1125–1133. 10

Lawlor, O. S. (2003). Embedding OpenCL in C++ for Expressive GPU Programming. *First International Workshop on DomainSpecific Languages and HighLevel Frameworks for High Performance Computing WOLFHPC 2011*. 43

Layer, S. D. (2015). Simple directmedia layer. 31

Liao, J., Buhimschi, C., and Norwitz, E. (2005). Normal labor: mechanism and duration. *Obstet Gynecol Clin North Am*, 32(2):145–64, vii. 2

Lien, K.-C., Mooney, B., DeLancey, J. O. L., and Ashton-Miller, J. a. (2004). Levator ani muscle stretch induced by simulated vaginal birth. *Obstetrics and gynecology*, 103(1):31–40. 12

Martins, J. a. a. C., Pato, M. P. M., Pires, E. B., Natal Jorge, R. M., Parente, M., and Mascarenhas, T. (2007). Finite element studies of the deformation of

the pelvic floor. *Annals of the New York Academy of Sciences*, 1101:316–334. 13, 61

Melchert, F., Wischnik, A., and Nalepa, E. (1995). The prevention of mechanical birth trauma by means of computer aided simulation of delivery by means of nuclear magnetic resonance imaging and finite element analysis. *Asia-Oceania Journal of Obstetrics and Gynaecology*, 21(2):195–207. 14

Miller, K., Joldes, G., Lance, D., and Wittek, A. (2007). Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. *Communications in Numerical Methods in Engineering*, 23:121–134. iv, 35, 39

Miller, T. (1997). A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2:25–30. 54

Moreau, R., Pham, M., Redarce, T., and Dupuis, O. (2008). Simulation of forceps extraction on a childbirth simulator. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1100 –1105. 11, 21

Munshi, A., Gaster, B., Mattson, T. G., Fung, J., and Ginsburg, D. (2011). *OpenCL Programming Guide*. Addison-Wesley Professional, 1 edition. 44, 45

Nguyen, A. (2006). *Implicit bounding volumes and bounding volume hierarchies*. PhD thesis, Stanford, CA, USA. AAI3235306. 50, 51

Oden, J. T. (2010). Finite element method. 5(5):9836. revision 122201. 34

Parente, M. P. L., Jorge, R. M. N., Mascarenhas, T., Fernandes, a. a., and Martins, J. a. C. (2008). Deformation of the pelvic floor muscles during a vaginal delivery. *International Urogynecology Journal and Pelvic Floor Dysfunction*, 19:65–71. 13

Parente, M. P. L., Jorge, R. M. N., Mascarenhas, T., Fernandes, a. a., and Martins, J. a. C. (2009). The influence of an occipito-posterior malposition on the biomechanical behavior of the pelvic floor. *European Journal of Obstetrics Gynecology and Reproductive Biology*, 144:166–169. 13, 61

Shockey, J., von Fraunhofer, J., and Seligson, D. (1985). A measurement of the coefficient of static friction of human long bones. *Surface Technology*, 25(2):167 – 173. 56

Silveira, R., Pham, M., Redarce, T., Betemps, M., and Dupuis, O. (2004). A new mechanical birth simulator: Birthsim. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3948 – 353. vol.4. 11

Sivamani, R. K., Goodman, J., Gitis, N. V., and Maibach, H. I. (2003). Friction coefficient of skin in real-time. *Skin Research and Technology*, 9(3):235–239. 56

Stone, J. E., Gohara, D., and Shi, G. (2010). OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *Computing in science & engineering*, 12(3):66–72. 43

Tu, C. and Yu, L. (2009). Research on collision detection algorithm based on AABB-OBB bounding volume. In *Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on*, volume 1, pages 331 –333. 52

Wong, H., Papadopoulou, M.-M., Sadooghi-Alvandi, M., and Moshovos, A. (2010). Demystifying gpu microarchitecture through microbenchmarking. In *Performance Analysis of Systems Software (ISPASS), 2010 IEEE International Symposium on*, pages 235–246. 42

Chapter 7

Appendices

7.1 BirthView application screenshots