NTNU
Norwegian University of Science and Technology

# Modeling and Simulation of the Human Body

**Carl Otto Gjelsvik**

Master of Science in Engineering and ICT
Submission date: January 2016
Supervisor: Ole Ivar Sivertsen, IPM

Norwegian University of Science and Technology
Department of Engineering Design and Materials

THE NORWEGIAN UNIVERSITY
OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF ENGINEERING DESIGN
AND MATERIALS

**MASTER THESIS AUTUMN 2015**
**FOR**
**STUD.TECHN. CARL OTTO GJELSVIK**

**MODELING AND SIMULATION OF THE HUMAN BODY**
***Modellering og simulering av menneskekroppen***

The human body topology is usually equal for all individuals, but sizes and dimensions vary. Some differences are natural from birth, but some are born with handicaps. Variations may come from injuries, but may also be desired by athletes from training over time. The human body is a mechanical system suited for computer simulation.
The partners in this effort have comprehensive and complementary competences for building and simulating the human body:
TechnoSoft Inc. 30 years research experience with automation in design (developing tools and applications for industry worldwide), a key knowledge for modelling variations in the human body.
NTNU 30 years research experience with developing simulation software for mechanical systems, especially mechanisms, based on the Finite Element method and control engineering.
In his project assignment, with the same title, the candidate did a literature study and did some preliminary investigation regarding simulation of the human body.

The assignment includes:

1. Run the simplified AML code for human skeleton modeling and refine the model with more correct details

2. Propose a scanning procedure for the human body as input for simulation and propose a procedure to generate simulation input

3. Propose method for generating main muscles with connection points

4. Extend the AML code to generate simulation input with simplified joint, link and muscle models.

5. Run simulation operations on the simplified model

6. As far as time allow, refine the AML code to generate improved simulation models and rerun simulations

**Formal requirements:**

Three weeks after start of the thesis work, an A3 sheet illustrating the work is to be handed in. A template for this presentation is available on the IPM's web site under the menu "Masteroppgave" (http://www.ntnu.no/ipm/masteroppgave). This sheet should be updated one week before the master's thesis is submitted.

Risk assessment of experimental activities shall always be performed. Experimental work defined in the problem description shall be planed and risk assessed up-front and within 3 weeks after receiving the problem text. Any specific experimental activities which are not properly covered by the general risk assessment shall be particularly assessed before performing the experimental work. Risk assessments should be signed by the supervisor and copies shall be included in the appendix of the thesis.

The thesis should include the signed problem text, and be written as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents, etc. During preparation of the text, the candidate should make efforts to create a well arranged and well written report. To ease the evaluation of the thesis, it is important to cross-reference text, tables and figures. For evaluation of the work a thorough discussion of results is appreciated.

The thesis shall be submitted electronically via DAIM, NTNU's system for Digital Archiving and Submission of Master's theses.

The contact person:
Bjørn Haugen, IPM/NTNU

[illegible]

Torgeir Welo
Head of Division

Ole Ivar Sivertsen

Ole Ivar Sivertsen
Professor/Supervisor

NTNU
Norges teknisk-
naturvitenskapelige universitet
Institutt for produktutvikling
og materialer

## Preface

This is a master's thesis written for the Department of Engineering Design and Materials at the Norwegian University of Science and Technology, as part of the study program Engineering and ICT. The work was carried out during autumn 2015.

It is an extension of a project thesis, conducted by the author during spring 2015. The thesis comprise a literature study on the human musculoskeletal system, and a preliminary investigation regarding simulation of this system during spring 2015.

Oslo, 2016-01-19

Carl Otto Gjelsvik

Carl Otto Gjelsvik

## Acknowledgment

First of all I would like to express a sincere gratitude to supervisor, Ole Ivar Sivertsen, and co-supervisor, Bjørn Haugen, for the continuous support, enthusiasm and guidance they have provided throughout the master's thesis work.

A sincere thanks also goes to Marius Widerøe, for his assistance and knowledge about MRI.

Finally, I would like to thank Lise Bohn Bjøralt and Silje Duley for their great assistance and support.

C.O.G

# Abstract

The theme of this Master Thesis is modeling and simulation of the musculoskeletal system. The purpose is to explore new ways to diagnose the human body, and do so effectively by automation. This Master Thesis investigates the problem of automatic segmentation to obtain geometry of bone, muscle, ligaments and tendons. In addition, it includes a research on how to generate simulation input in a Knowledge Based Engineering System (KBE).

The scope has been to generate simulation input from a medical image. Simulation input includes bone, muscle, ligament and tendon geometry as well as mechanical properties for these tissues. Due to extent and complexity this Master Thesis has focused on extraction of bone data from the medical image and geometry generation as the simulation input.

A segmentation program has been developed in Matlab to obtain bone as the only tissue from medical images. The medical images used in this Master Thesis have been taken with CT. Another program has been developed in a KBE framework, Technosoft's Adaptive Modeling Language.

The segmentation program is able to segment bone tissue and present the result in 3D. However, it requires user input to control the segmentation process. The AML program generates geometry for tibia, femur, pelvis and the spine. It also generates cartilage to represent the intervertebral discs.

The developed programs are platforms for further research and development, and has proven Matlab and AML as viable tools for continued work.

## Sammendrag

Denne Master oppgaven omhandler modellering og simuelring av menneskets muskel-og skjelettsystem. Hensikten er å utforske nye metoder for undersøkelser av menneskroppen, og effektivisere disse ved automatisering. Masteroppgaven utforsker utfordringer ved bruk av segmentering for å oppnå geometriske framstillinger av ben, muskler, leddbånd og sener. Oppgaven inkluderer også undersøkelser av hvordan man kan generere inndata for simulering i et ”Knowledge Based Engineering”-system (KBE).

Omfanget av oppgaven har vært å genere inndata for simulering med utgangspunkt i medisinske bilder. Simuleringsdata innkluderer geometri for ben, muskler, leddbånd og sener samt mekaniske egenskaper for disse vevtyper. På grunn av omfang og kompleksitet er oppgaven avgrenset til å dekke uthenting av bendata fra bildematerialet og generere geometri til inndata for simulering.

Et segmenteringsprogram er utviklet i Matlab for å avgrense uthenting fra det medisinske bildemateriale til kun å være benvev. Det medisinske bildematerialet benyttet i denne Master oppgaven har blitt tatt med CT. I tillegg til nevnte program er det også utviklet et program innen KBE rammeverket, Technosoft’s Adaptive Modeling Language.

Segmenteringsprogrammet gjør det mulig å synliggjøre kun benvev og presentere resultatet i 3D. Imidlertid er segmenteringsprosessen ikke helt automatisert og vil kreve manuell kontroll. AML programmet genererer geometry for leggben, lårben, bekken og ryggrad. Programmet genererer også brusk for å visualisere skiver i ryggraden.

Begge programmene som har blitt utviklet i denne Masteroppgaven er utgangspunkt for videre forskning og utvikling og har bevist at både Matlab og AML er gode verktøy for videre arbeid.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

This master's thesis is part of a coalition of project and master's thesis' at the Department of Engineering Design and Materials, with Professor Ole Ivar Sivertsen as supervisor and coordinator. The objective of the work is to develop a software application in the KBE framework AML, for structural and dynamic analysis of an arbitrary human body. The analysis can be used in simulations where professional athletes, patient rehabilitation, development of prostheses and military applications are the target groups. The first step in achieving this is to generate the simulation input.

The musculoskeletal system is outside the area of expertise for the Department of Engineering Design and Materials. However, the musculoskeletal system is a biomechanism, and as the Department of Engineering Design and Materials expertise on mechanisms this field of study can also be covered. The musculoskeletal system is despite this a complex mechanism, and to model it is an extensive task.

Currently, there are other projects which are working towards similar objectives. The main differences between these projects are the automation of simulation input generation. The processes required in terms of making accurate simulations is embedded in the combined use of medical images, infrared markers to map motion and OpenSim. The processes are mainly manual, and the work needed to produce the subject-specific simulations is quite comprehensive and time consuming.

As a preparation to this master's thesis, a study of the human body, Gjelsvik [2015], was conducted. The study was carried out during spring 2015.

## 1.2 Problem Outline and Scope

This master's thesis is exploring solutions to create a system that automatically generates simulation input of an arbitrary human body.

The scope has been to generate simulation input from a medical image. Simulation input includes bone, muscle, ligament and tendon geometry as well as mechanical properties for these tissues. Due to extent and complexity this Master Thesis has focused on extraction of bone data from the medical image and geometry generation as the simulation input.

Mechanical properties of connective tissues are important for simulation purposes. However, for the work conducted in this master's thesis this has not been prioritized. They can rather be seen as input in class definitions in the program developed, but must be further implemented to have value as simulation input.

During the work with this thesis, the complexity and extent of the subject have been revealed. In consultation with supervisor, it was also agreed upon to prioritize problem 2 and 4 at the expense of problem 3 and 5, of the problem outline. The focus in this master's thesis has therefore been on segmentation of bone and geometry generation based on text input. As a consequence, a proposition for muscle simulation input will not be covered in this thesis.

## 1.3 Structure

Chapter 2 presents related work, theory regarding medical terms in anatomical studies, medical imaging and Knowledge based engineering. Chapter 3 presents the tools used in the research and development of programs during work with this thesis. Chapter 4 describes the development process and Chapter 5 presents the result of the program developed. Chapter 6 discusses the result and Chapter 7 concludes. Chapter 8 presents thoughts on further work.

# Chapter 2

# Theory

The first section of this chapter will review projects in which could be related to the work conducted in this master's thesis. The second section will carry out a short introduction to the topology of the human body and terminology in anatomical studies. This will be useful for later referencing. The third section explores medical images as a method for obtaining data for generating simulation input, and the last section introduce Knowledge-Based Engineering.

## 2.1 Related Projects

There has in recent years been conducted studies and carried out several projects in which could be related to the topic of this thesis. The work done in the different areas presents clear advantages of the development of computational simulation models in terms of understanding the important physiological principles of the human body. However, there are still challenges that have to be overcome before adequate treatment of patients can be facilitated by the use of these models.

The projects to be outlined in this section are all ongoing projects, except the final one, MultiSim, which is still in its developing phase.

### 2.1.1 OpenSim

In 2007 Delp and Thelen [2007] developed a freely available, open–source software system named OpenSim. The reason for developing such a system is to provide a platform where users can develop models of musculoskeletal structures, and where they can create and analyze dynamic simulations of movements. OpenSim can compile and run on regular operating systems, as its software is written in ANSI C++, and the graphical user interface in Java.

The biomechanics community can, by the assistance of OpenSim, build a collection of simulations which can be exchanged, tested, analyzed, and enhanced through a multi-institutional cooperation effort. However, limitations arise when trying to produce a coordinated movement, which seems to be one of the major challenges in this field. Additionally, there are also made many assumptions in the development of these models, where some are based on restricted experimental evidence. Nevertheless, OpenSim clearly provides new opportunities. Testing and reproduction of simulation-based studies offers important research advantages taking place also outside the laboratory, which is crucial if biomechanical simulations are to establish a scientific basis for treatment planning. Development and maintenance of OpenSim is being managed on Simtk.org, which operates as a public storage for data, models, and computational tools, connected to physics-based simulations of biological structures.

#### 2.1.1.1 NMS Builder

NMS Builder which is a project tool in NMS Physiome; a research project funded by the European Commission, is a tool package for developing OpenSim musculoskeletal models (NMSPhysiome [n.d.]). It is a user-friendly tool that develops models from patient-specific biomedical data, utilizing OpenSim to conduct the dynamic simulations of motion (NMSBuilder [n.d.]). Consequently, the NMS Builder makes it possible for researchers to import data containing images, motion analysis data etc., process the data to construct a full scale musculoskeletal model, create OpenSim models and run simulations, and visualize the data.

### 2.1.2 The Virtual Physiological Human

In 2006 a project named STEP – A Strategy for the EuroPhysiome was initiated under support from the European Commission. The aim of the project was to develop an integrated European approach to the multi-scale modelling of the human physiome (defines the functional behavior of the physical state of an individual). The establishment of this project resulted, in early 2007, in a roadmap outlining how European work should be conducted in terms of delivering the Virtual Physiological Human (VPH) (Viceconti and Waters [2007]). VPH represents a framework based on a technological and methodological approach, enabling a collaborative study of the human body. As a result, through VPH one can derive predictive hypotheses from shared observations of the human body, represented as a single compound system. VPH offers the required infrastructure supporting scientists in all different fields to communicate and to share data and technologies. However, there are some challenges that needs to be overcome before the VPH can be fully realized. First of all, the human body is very complex. It comprises multiple interconnections and interactions, and even though sufficient amount of knowledge is becoming available (from genes to whole systems), there are still many physiological functions that have still not been understood. Additionally, the absolute scale of data which is to be generated, processed and exchanged is dependent on a vast storage and software tools that presently are not extensively available. VPH therefore has to define access to resources and computing power accessible from already developed computing centers. Additionally, since the VPH scope is multidisciplinary by definition, and only a few number of journals are capable of accepting physiome-related papers, dissemination is also something that must be handled.

On the road to success the VPH Institute for Integrative Biomedical Research has been developed (VPHInstitute [n.d.]), which is an international non-profit organization. The institute, incorporated in Belgium, has one mission, and that is to ensure that the VPH is being fully implemented, widely adopted and used in an effective manner in research as well as in health centers.

### 2.1.3 Finite Elements for Biomechanics

FEBio – Finite Elements for Biomechanics, a project partially supported by a grant from the U.S. National Institute of Health, is an implicit, non-linear finite element solver that is designed for biomechanics application purposes. The FEBio software was initially developed at the Musculoskeletal Research Laboratories at the University of Utah. Nowadays, the development, distribution and support of the software is a joint effort between the University of Utah and the Musculoskeletal Biomechanics Laboratory at Columbia University (FEBio [n.d.]). The theory behind FEBio is available in a theory manual written by Maas and Ateshian [2015]. In the manual FEBio is described as a software developed to supports two different analysis types, *quasi-static* and *quasi-static poroelastic*. In the first type it is the static response of the system that is wanted (inertial terms are ignored), and in the second type one solves a linked solid-fluid problem. The *quasi-static poroelastic* analysis is therefore useful when modeling tissues containing sufficient amounts of water. In terms of modeling the complex biological tissue behavior there are multiple non-linear constitutive models available, namely isotropic and anisotropic. The isotropic models have a non-linear stress-strain response, while the opposite case are materials showing anisotropic behavior, at least in one direction. The anisotropic models are suitable to use when modeling biological tissues containing fibers, such as muscles and tendons. In addition to these models FEBio comprises a *rigid body* material model. A model in which is useful when modeling structures where deformation is unimportant. FEBio also supports an extensive range of boundary conditions to model interactions between biological tissues, as they can interact in several complex ways. The models contain already preset displacements, pressure forces and nodal forces. Deformable models are possible to connect to rigid bodies, assisting the user to model preset rotations and torques, where the rigid bodies in the next step can be attached with rigid joints. Further information regarding the FEBio software is available in this theory manual, assisting the user to better understand how the program works, and how to utilize it in terms of creating biomechanical simulations.

### 2.1.4 MultiSim

MultiSim is a program aiming to develop a new generation of analytical models (MultiSim [n.d.]). The vision of the program is to establish a modeling framework of the human musculoskeletal system, however, planned to work as a platform to address multiple engineering challenges. The platform should include models skilled to manage compound multi-scale problems, variables and states which are not yet observable, and uncertainties. The main focus of the program is to develop a platform for managing musculoskeletal disorders, which today is not available. However, if this project can be realized it will enable computational simulations to overcome the challenges stated above, and which biological systems faces today.

## 2.2 The Human Musculoskeletal System

*The following section describes the musculoskeletal system, where the source is Neumann [2010] if not otherwise is stated.*

The human musculoskeletal system is a biomechanical system which enables movement in humans. The main elements are bones, muscles, tendons, ligaments and cartilage, which are all different types of connective tissue. The skeleton comprises 206 bones connected in joints. There are two classes of joints, mainly distinguished by their ability to allow considerable motion. The type of joint that allows motion, and thus is an important part of the muscoloskeletal system, is the synovial joint. The bones in the synovial joints are moved and restricted by muscles, tendons and ligaments. For a more detailed visualization of this, see Figure 2.1. There are other elements associated with these joints, but they are outside the scope of this study. Tendons are the connective tissue between muscles and bones, whereas ligaments are the connective tissue between bones. Cartilage is another connective tissue, positioned between bones to prevent excessive tear due to contact interactions.

In the study *Gray's Anatomy for Students*, conducted by Drake and Gray [2005], a system of specified positions in the human anatomy have been defined. A summary of the terminology can be found listed in Table 2.1. Three planes are defined for referencing planes in anatomical studies, and the planes are illustrated in Figure 2.2.

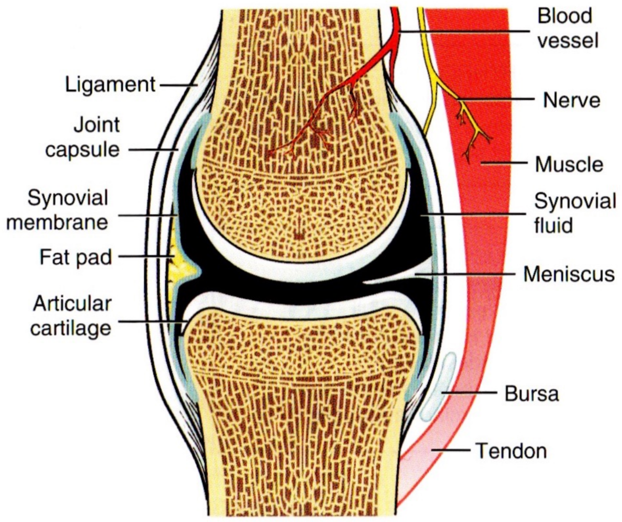


Figure 2.1: A synovial joint and its elements, represented by the knee joint and its two bony members, the femur (top) and the tibia (bottom). (Neumann [2010], p. 30)

Table 2.1: Terminology in anatomical studies. (Drake and Gray [2005])

| **Word** | **Definition** |
|---|---|
| Anterior | Position at the front of the body. |
| Posterior | Position at the back of the body. |
| Superior | Position above another part of the body. |
| Inferior | Position below another part of the body. |
| Proximal | Position closer to the trunk of the body. |
| Distal | Position further away from the trunk of the body. |
| Medial | Position closer to the midline of the body. |
| Lateral | Position further away from the body. |
| Cranial | Position towards the top of the skull. |
| Caudal | Position towards the bottom of the body. |
| Paired | Structures that is present on both sides of the body. |

### 2.2.1 Topology of Bones

Bones are the main structural element in the musculoskeletal system and are the main focus in this master's thesis. In the following chapters, the latin name is being used consequently to describe different kinds of bones. A reference list of the axial human skeleton is illustrated in Table 2.2. Table 2.4 illustrates the lower extremity and Table 2.3 the upper extremity. Paired bones are listed once. Figure 2.3 illustrates the human skeleton, including labeled numbers that corresponds to the numbering represented in the tables.

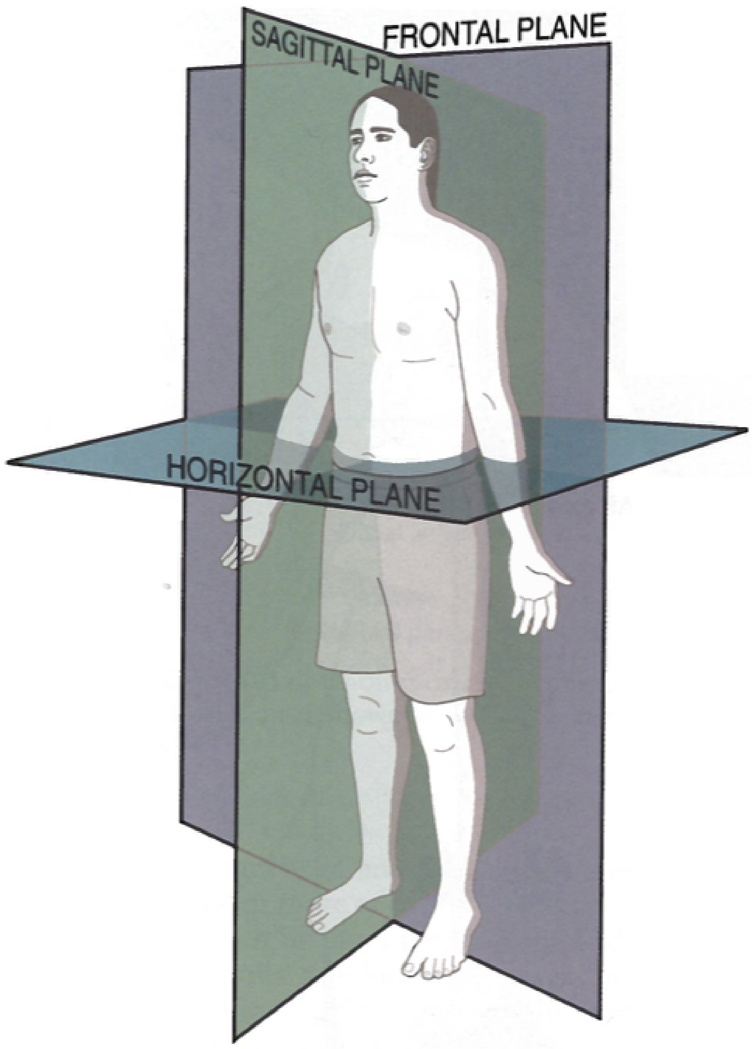


Figure 2.2: The saggital, frontal and horizontal planes as described and illustrated in Neumann [2010].

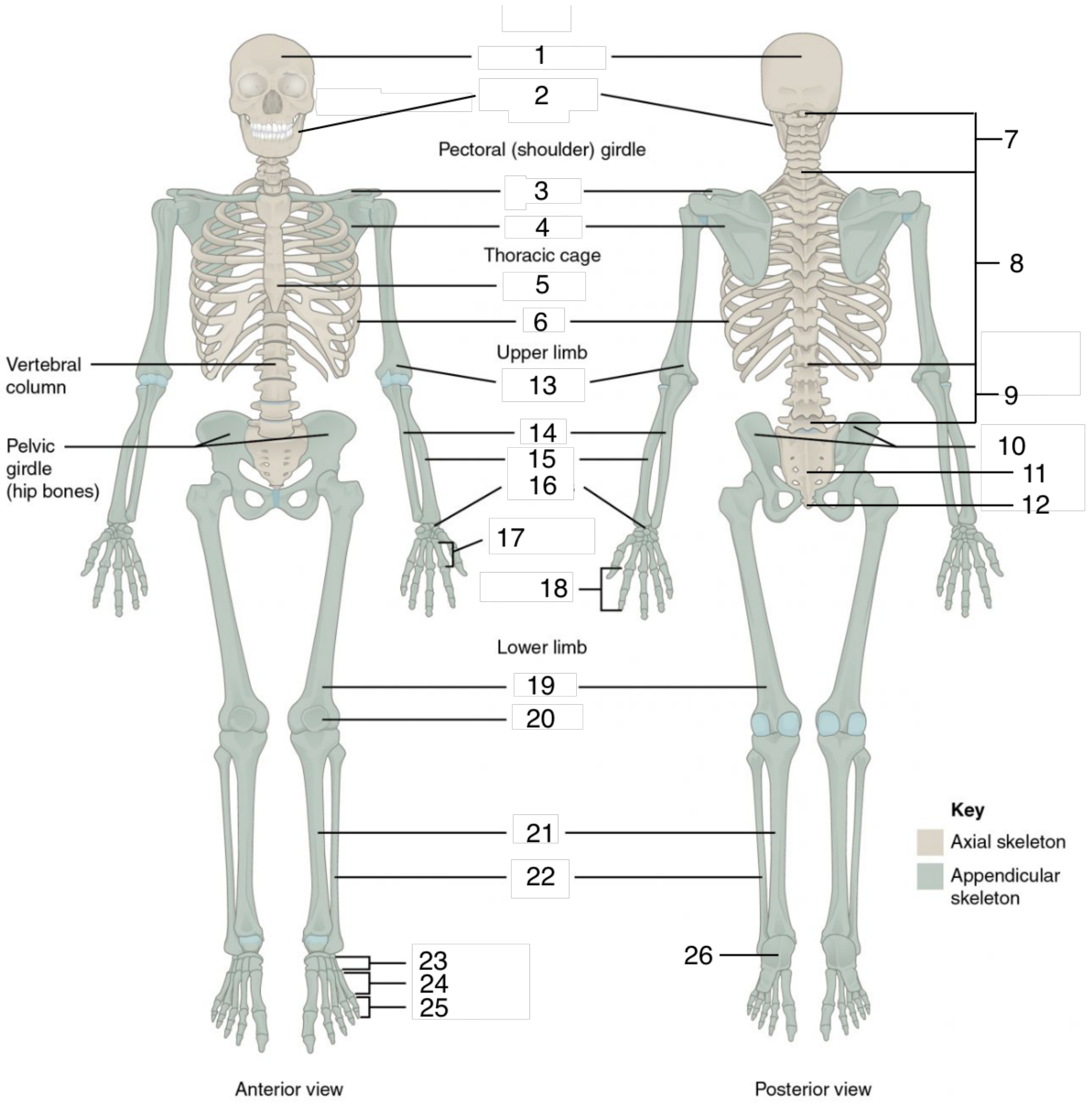


Figure 2.3: Illustration of the human skeleton. Numbers refer to Table 2.2, 2.3 and 2.4. Illustration adapted from (CandelaOpenCourses)

Table 2.2: Topology of axial skeleton. Numbers refer to Figure 2.3.

| No. | Latin name | Comment |
|---|---|---|
| 1 | Cranium | The cranium consists of several bones, but as the head is merely a mass in the context of biomechanical modeling, this report do not elaborate on the cranium features. |
| 2 | Mandibula | No Comment |
| 5 | Sternum | No Comment |
| 6 | Ribs | There are twelve ribs and they are numbered 1-12 from top to bottom. The ribs are attached to the thoracic vertebrae with corresponding numbers. The two lower ribs are not attached to the sternum. |
| 7 | Cervical Vertebrae | There are seven cervical vertebrae and they are arranged from 1-7 from top to bottom. |
| 8 | Thoracic Vertebrae | There are twelve thoracic vertebrae and they are arranged from 1-12 from top to bottom. |
| 9 | Lumbar Vertebrae | There are five lumbar vertebrae and they are arranged from 1-5 from top to bottom. |
| 10 | Pelvis | No Comment |
| 11 | Sacrum | No Comment |
| 12 | Coccyx | No Comment |

Table 2.3: Topology of upper extremity. Numbers refer to Figure 2.3.

| No. | Latin name | Comment |
|---|---|---|
| 3 | Clavicle | No Comment |
| 4 | Scapula | No Comment |
| 13 | Humerus | No Comment |
| 14 | Ulna | No Comment |
| 15 | Radius | No Comment |
| 16 | Carpals | There are five carpals in each hand |
| 17 | Metacarpals | There are five metacarpals in each hand |
| 18 | Phalanges | There are five phalanges in each hand. The phalanges are divided in three bones; the proximal, medial and distal phalanx. |

Table 2.4: Topology of Lower Extremity. Numbers refer to Figure 2.3.

| No. | Latin name | Comment |
|---|---|---|
| 19 | Femur | No Comment |
| 20 | Patella | No Comment |
| 21 | Tibia | No Comment |
| 22 | Fibula | No Comment |
| 23 | Tarsals | The tarsals are a collection of bones in the middle of the foot connecting the legbones (Tibia and Fibula), heel(Calcaneus) and toes(metatarsals and phalanges). |
| 24 | Metatarsals | There are five metatarsals in each foot |
| 25 | Phalanges | There are five phalanges. The four outer phalanges are divided in three bones; the proximal, medial and distal phalanx. The inner phalanx is divided in two; the proximal and medial phalanx |
| 26 | Calcaneus | No Comment |

## 2.3 Medical Imaging

*The following section is based on a dialog with Marius Widerøe, see Appendix A.*

CT and MRI are two methods for capturing medical images of the human body. CT uses electromagnetic radiation to light a subject which creates a shadow on a receiver behind the subject. Bones absorbs more electromagnetic waves than soft tissue, and appears brighter in the image than other tissues, see Figure 2.4.

From a health perspective, MRI is a preferred method to capture medical images due to less radiation. MRI is recognized for being a preferred method for capturing images of soft tissues. The reason is that MRI offers ways of weighting different types of tissues. The Figure 2.5, shows how types of tissues are displayed with different light intensity.

CT-images, on the other hand, is preferred for capturing images of skeleton. This thesis will analyze segmentation of bone structure from medical images, which means “extract bone structure” from the images.

When capturing images of the human body using CT or MRI, the patient will be lying down. In this position the body is less loaded than when standing up. When lying down, the gravity will give less effect on the layout of the patient’s body.

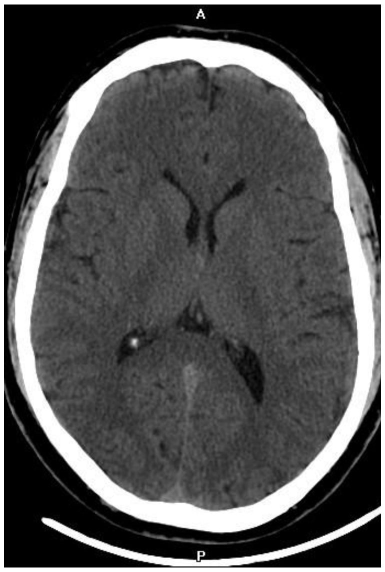


Figure 2.4: CT-image of the brain. The white matter is bone tissue (Neurologica).

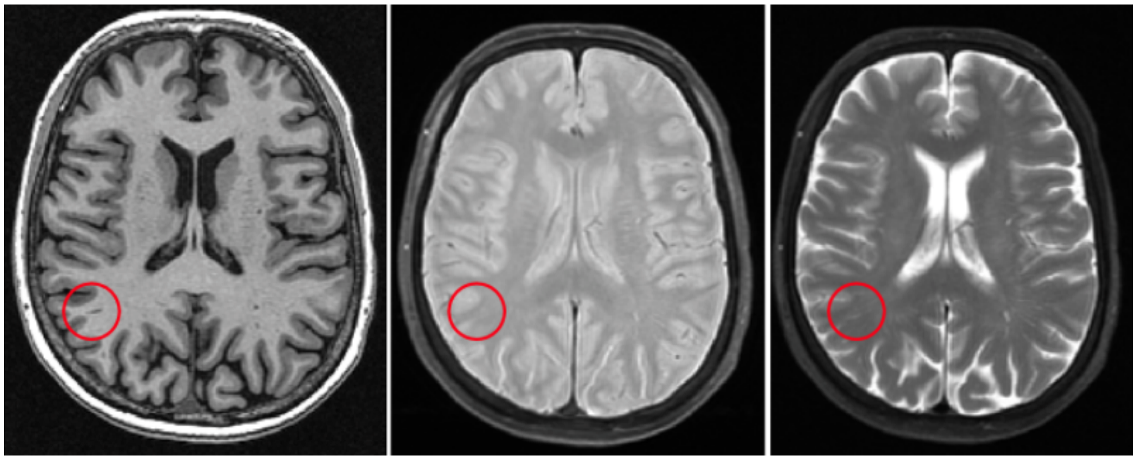


Figure 2.5: Illustration of T1, PD and T2-weighted images of the same section in the brain (BrainLab).

## 2.4 Knowledge Based Engineering

Knowledge based engineering (KBE) is a design philosophy that aims to automate engineering tasks that are repetitive and time consuming. This philosophy has been successfully adapted and implemented in Aker Solutions' KBeDesign Department (Stensvold), and have led to dramatically reduced time for developing offshore platforms and boat hulls. The software framework this department has been utilizing in terms of developing their application is called AML (Adaptive Modelling Language), developed by Technosoft.

There are several software systems that implements KBE. Common for all of them is that they use an object oriented programming language. This is because a KBE system usually represents the real world, which again consists of objects. An object in a KBE system is represented by a class definition. The real world also has events that happens to objects. In a KBE sense, events are called methods and functions.

# Chapter 3

# Methodology

## 3.1 Technosoft's AML

Technosoft's AML (Adaptive Modeling Language) is a KBE framework. The program developed in this master thesis is based on a AML mechanism program developed by Ole Ivar Sivertsen. The Department of Design and Materials has knowledge of AML and user licences for the software. Therefore, it was convenient to continue with AML as the KBE implementation tool. AML has also been proven in the industry as a world class KBE system. As an example, Aker Solutions' KBeDesign has been recognized as an industry leading arena within fast design of offshore platforms using AML (Stensvold).

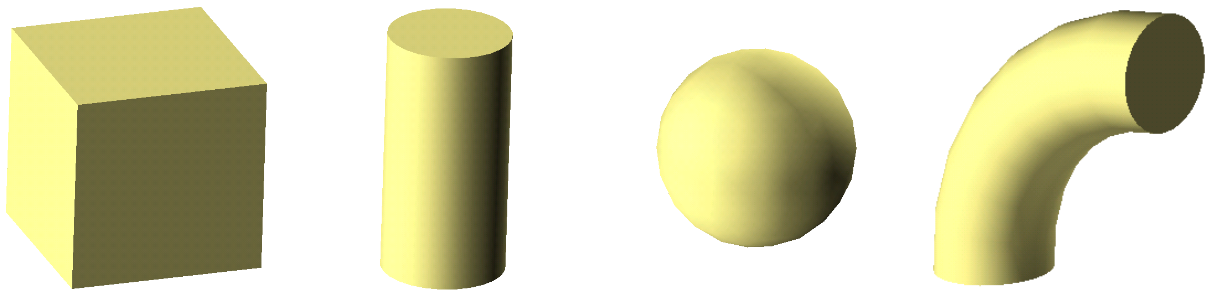
### 3.1.1 Boolean and Native Geometric Classes

AML has classes which can directly be used to create simple objects such as boxes, cylinders, spheres and elbows. See Figure 3.1.

To create complex geometries one can perform operations making it possible to create new objects. These operations are called Booleans and include, difference, intersection and union. See Figure 3.2.

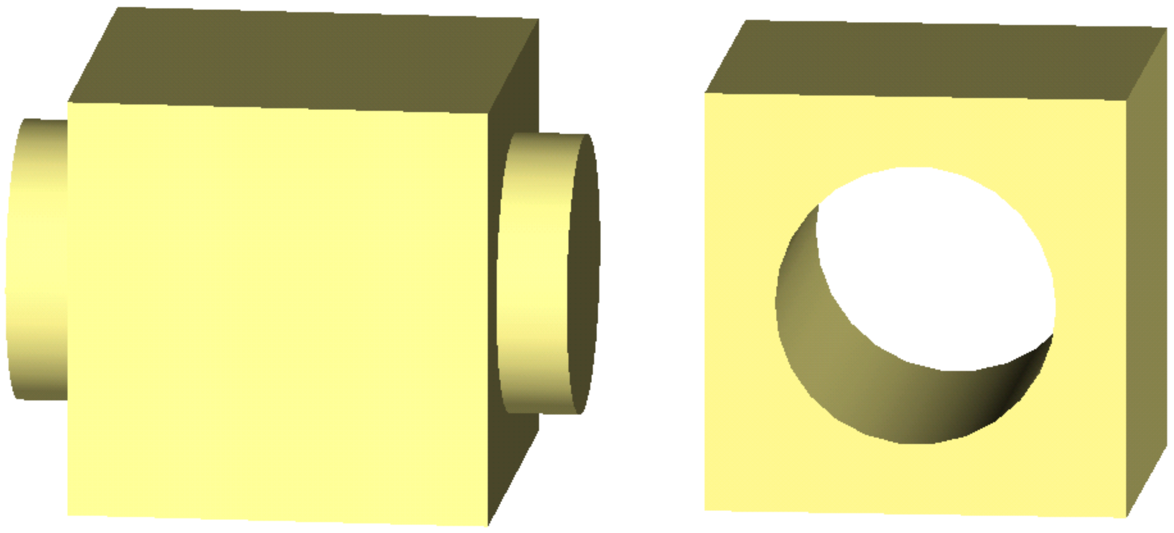
### 3.1.2 Runtime Environment

AML has support for multiple operating systems, but due to availability, Microsoft Windows was chosen. The specific version used for developing and running AML programs was Windows 7 x64 on a 2-core 2.93 ghz



(a) Box-object (b) Cylinder-object (c) Sphere-object (d) Elbow-object

Figure 3.1: Objects made by native classes in AML.



(a) A cylinder positioned through a box. (b) A box with a hole after "difference" Boolean operation.

Figure 3.2: Example of Boolean operation in AML. A difference-object has been created by "differencing" a cylinder from a box.

system with 8gb of RAM. The AML version 5.85 was used with the newest patches installed. In addition, AMSketcher was installed to improve the environment that geometric objects was designed in.

### 3.1.3 Editor

XEmacs is a open source editor based on GNU's Emacs. Technosoft has customized it to work with AML, and it is included when installing AML. The editor supports AML syntax and custom key bindings. The interface has also been edited for easy access to buttons that runs AML, compile and load files, and open AML GUI. When AML is executed and is running, the AML buffer handles commands. Commands can be used for compiling programs, creating models and drawing them. With key bindings, one can perform advanced text operations. However, it is possible to write AML code in any text editor and use XEmacs for compiling only. This can be done by structuring code in the way described in Section 3.1.4, and enter the command:

```
(compile-system :name-of-system)
```

Where name-of-system is the name of the system that has been defined.

### 3.1.4 Source Code Management

For structuring code and simplifying files, a system can be defined. To define a system a system.def file is created and must contain the following outline:

```
(in-package :AML)
(define-system :system-name
   :files '(
   "name-of-file.aml"
   )
)
```

The class definitions and methods are written in .aml-files and included in a folder called "sources". This folder has to be in the same folder as the system.def-file. To compile the system, a line has to be added in the logical.pth file in the AML directory:

```
:system-name C:\\path-to-folder-of-system.def\
```

To compile this system, enter the following in XEmacs:

```
(compile-system :name-of-system)
```

For more details see TechnoSoft [2010]'s Reference Manual.

## 3.2 Segmentation Tool

CT-images were downloaded from `https://mri.radiology.uiowa.edu/visible_human_datasets.html`, in a DICOM-file format. Further information regarding these files is available in Section 3.2.1. CT-images were used due to limited availability of high quality MRI-images. To view these files a DICOM-viewer had to be downloaded. The DICOM-viewer, OsiriX Lite, was chosen as it is freely available to download at `http://www.oririx-viewer.com`. The functionality this program provides is to open DICOM-files and view the medical images the files contains. The images can be scrolled through easily and opened as a 3D image. The 3D image can be segmented in multiple ways. An interesting feature is that the segmented result can be exported to 3D objects, which includes STEP-files. STEP-files are a supported input format of AML. However, a prerequisite for the project was to use a software that automatically generates simulation input. OsiriX does not meet this requirement, and therefore other solutions were investigated.

It was discovered that a toolbox for Matlab (see Section 3.2.2), contained functions for manipulating DICOM-files. This opened up an opportunity to develop a custom program to handle segmentation. Matlab is also known for its fast matrix operations, and since CT-images are distributed to multiple DICOM-files where each image can be imported as a matrix, this functionality was seen as an advantage.

### 3.2.1 DICOM

Digital Imaging and Communication in Medicine (DICOM) is an ISO standard released in 1993 for storing medical images and relevant information. DICOM writes on their website that "DICOM is implemented in almost every radiology, cardiology imaging, and radiotherapy device (X-ray, CT, MRI, ultrasound, etc.)." DICOM is a framework that incorporates different aspects of the medical imaging workflow, such as journal keeping and billing. However, for this thesis, the important information that is available from the files is the image information. This is important for automatic segmentation of the body. Documentation can be downloaded or viewed through the following link: `http://dicom.nema.org/standard.html`. There are not many sample data sets available for download, but the ones used in this thesis can be downloaded from `http://www.osirix-viewer.com/datasets/` and `https://mri.radiology.uiowa.edu/visible_human_datasets.html`.

### 3.2.2 Image Processing Toolbox

To process DICOM-files in Matlab, the Image Processing Toolbox was applied. The toolbox is used for extracting image information and for importation of the medical images. The image information used in this program was:

1. Pixel Width
2. Pixel Height
3. Slice Thickness (The volume of data in a slice)
4. Samples Per Pixel (Data in each pixel)

For more information on the Image Processing Toolbox, see `http://se.mathworks.com/products/image/index.html`.

# Chapter 4

# Development

## 4.1 Segmentation tool

The segmentation tool developed in this master's thesis is a small program written in Matlab R2015B. The program imports a DICOM library, extracts the individual sections, collects the sections in a 3D matrix, and exploits an isosurface method to create a continuous surface. See Figure 4.1 for execution flow. The steps will be described and explained in Section 4.1.1-4.1.6

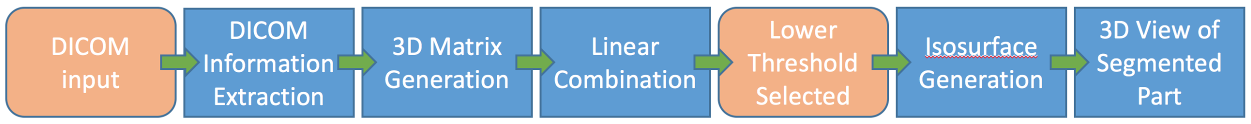
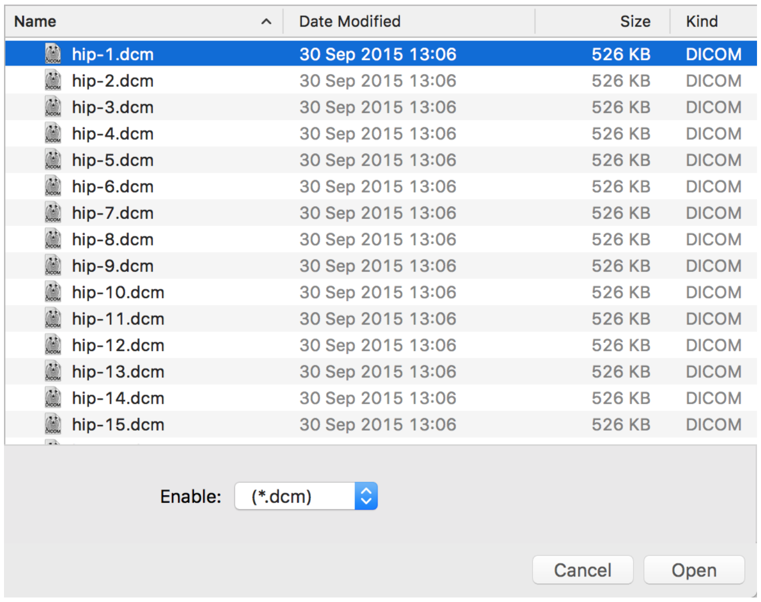


Figure 4.1: Flowchart of Segmentation tool. The orange rounded squares signal user input steps. The blue squares signal automatic steps.

### 4.1.1 DICOM Input

The input is a set of .dcm-files (DICOM-files) obtained from CT-images. The files have to be placed in an empty folder and given names similar to the following example. For a set of 20 DICOM-files, either of the following notations work as a valid naming format:

```
hip-0000.dcm, hip-0001.dcm, ..., hip-0019.dcm
hip.1.dcm, hip.2.dcm, ..., hip.20.dcm
hip40.dc, hip41.dcm, ..., hip60.dcm
```



| Name | Date Modified | Size | Kind |
|---|---|---|---|
| hip-1.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-2.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-3.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-4.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-5.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-6.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-7.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-8.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-9.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-10.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-11.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-12.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-13.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-14.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |
| hip-15.dcm | 30 Sep 2015 13:06 | 526 KB | DICOM |

Enable: (*.dcm)

Figure 4.2: Window prompted to the user for input of a DICOM library to the segmentation program.

This shows flexibility in the naming format, and is valid as long as the files end with a number and have names that increase by one.

To obtain the best results with this program, there are certain prerequisites for how the image is being extracted. If the image has been captured with MRI, the signal weighting is important for how the different tissues are viewed. For instance, T1 signal weighting makes bone the brightest tissue in the image, and also gives good contrast. This is important for a precise segmentation.

When the program is executed, the user will be prompted with an ”open file”-window. The user selects one of the files and press ”open” to import all files within this folder. See Figure 4.2.

### 4.1.2 DICOM Information Extraction

The DICOM format includes metadata about the patient, when and where the image was taken, etc. This information is, however, usually censored in the sample DICOM sets available on the Internet. Regardless of this, the available information is crucial for segmentation, as it contains information on how the image has been captured. Image resolution, slice thickness and color scale are all relevant for manipulating the images, and to obtain the final segmented 3D picture. The following sections will explain why this is important.

### 4.1.3 3D Matrix Generation

The medical images are obtained from a scan where one single DICOM-file has one section of the total image. Therefore, the DICOM-files has to be sorted and assembled in a 3D matrix to obtain a 3D image. A matrix with 3 dimensions is allocated in RAM based on the image resolution and the number of sections. The final matrix is represented in Figure 4.3.

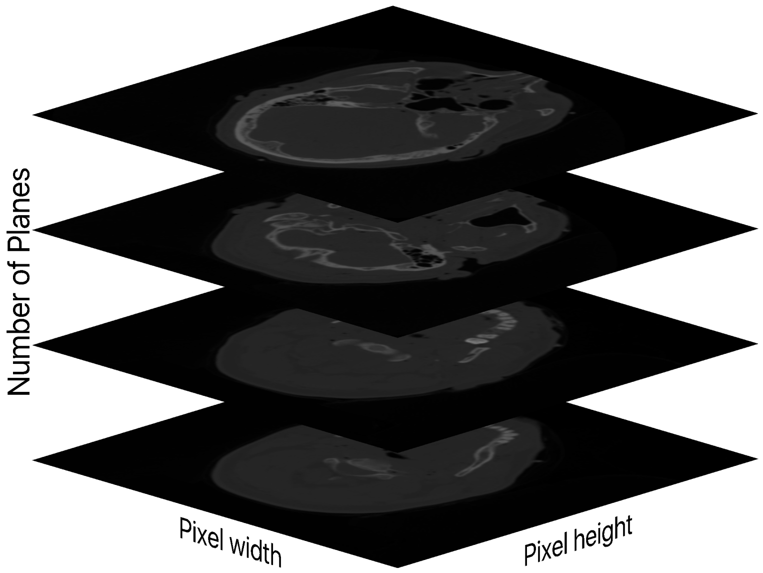


Figure 4.3: Illustration of how the 3D matrix with medical images are structured. Each section is a matrix which is pixel-width X pixel-height. The number of sections is determined by the number of DICOM-files imported.

### 4.1.4 Linear Combination

When some of the DICOM-files are opened, some of the medical images will appear completely black. This is because the pixel values are skewed towards a value of zero with just small differences from minimum value to maximum value. This is adjusted by applying a linear combination function on the 3D matrix. First the minimum and maximum value is found in the 3D matrix, then the values are distributed on a 16bit gray scale from 0 to 65536. An example result of this is shown in Figure 4.4.

### 4.1.5 Select Lower Threshold

An important part of the segmentation tool is the *lower threshold selection.* This section is a manual sequence, where the user is required to enter a number between 0 and 65536. If the image is of the recommended type,

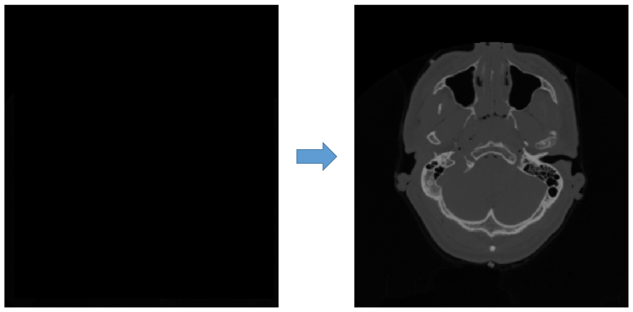


Figure 4.4: Linear combination of pixel values in an image which appears black. The pixel values are distributed on a 16 bit gray scale from 0 to 65536.

namely a T1 MRI-image or CT-image, bones will have the highest pixel value. Therefore, when the lower threshold is increased towards 65536, muscle, skin and other soft tissue will disappear from the image, and bone will be the only visible tissue left. Figure 4.5 illustrated an example of this.

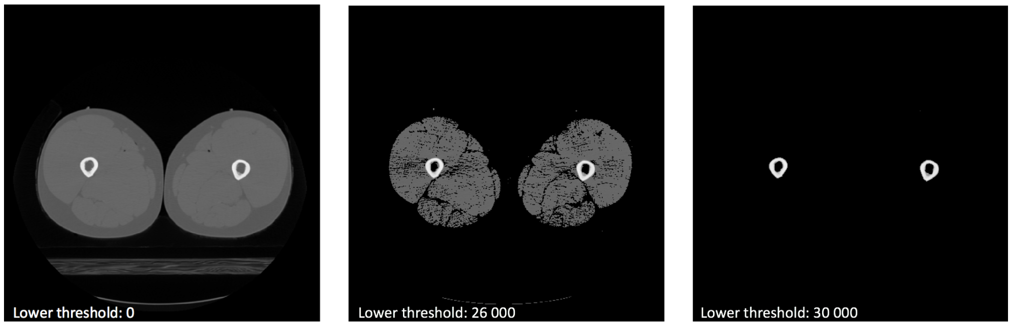


Figure 4.5: When the lower threshold is increased in a T1 weighted MRI-image or CT-image, soft tissue will disappear, while bone tissue will remain visible.

### 4.1.6 Isosurface Generation

The matrix one will end up with consists of sections where bone is the only visible tissue. There is, however, no connections between the sections. The sections are 2D planes, where the last phase of the program is to create a face between the sections. This has been conducted with *isosurface generation* and *isocaps*. Matlab includes functions for *isosurface generation* and for adding of *isocaps*, to make a surface on the end of geometry. The geometry is then visualized in 3D, and represented in the Matlab figure viewer. See Figure 4.6.

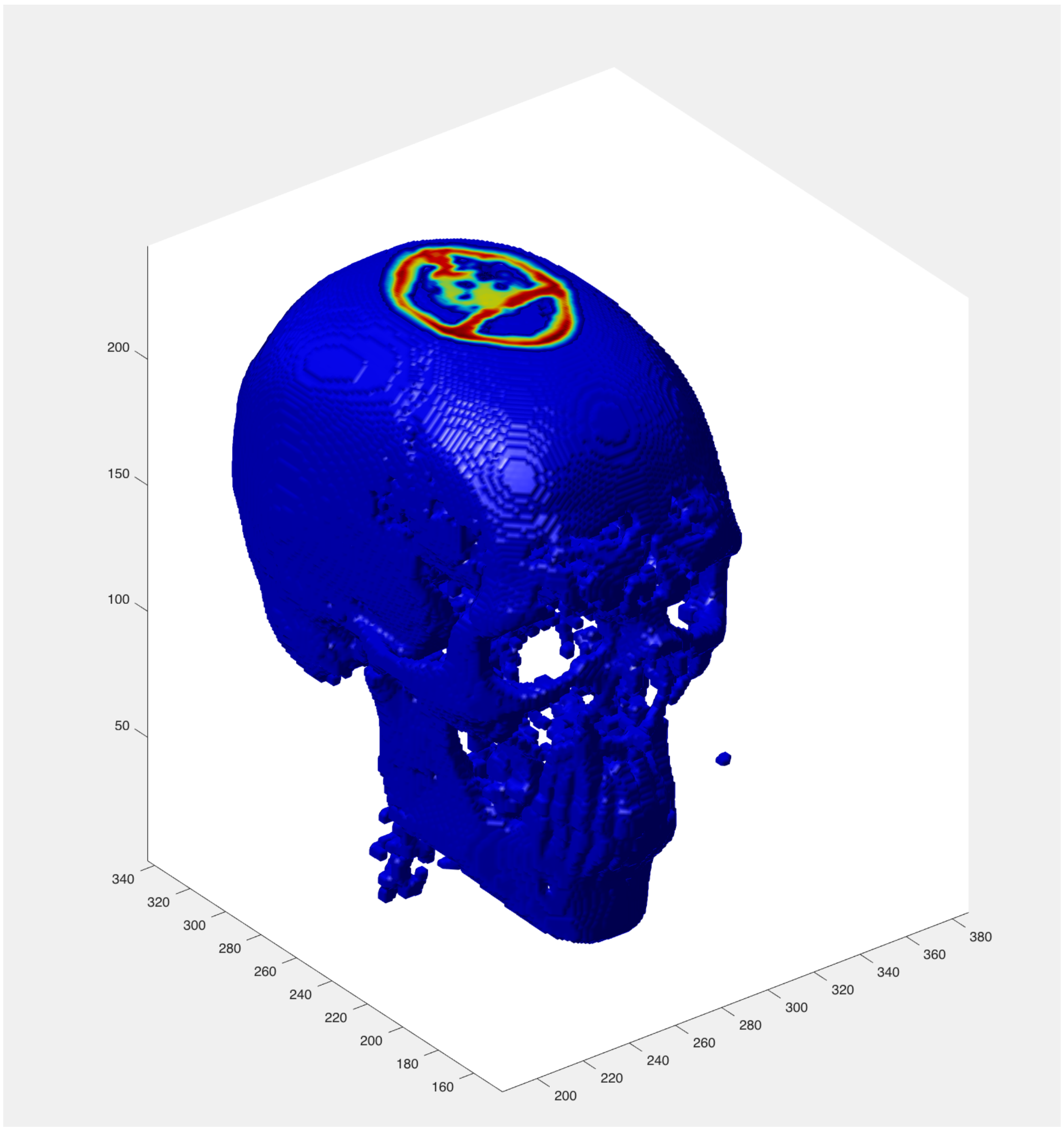


Figure 4.6: Segmented skull with applied isosurface and isocaps, viewed in the Matlab figure viewer.

## 4.2 Musculoskeletal Program - AML Development

### 4.2.1 Intention

This master's thesis has been working towards software that automatically generates simulation input from medical images as input data. Simulation input is geometry, mechanical properties and muscle power output. The objective has been to develop a KBE application in Technosoft's AML program that automatically generates:

- Geometry of an arbitrary human skeleton.
- Geometry of muscles, tendons, ligaments and cartilage, and positioning of these tissues on the human skeleton geometry.
- Mechanical properties for the tissues based on medical images as input data.
- Muscle power output based on available information and medical images as input data.

The program developed and described in this section will propose an approach on how to automatically generate skeleton and cartilage geometry.

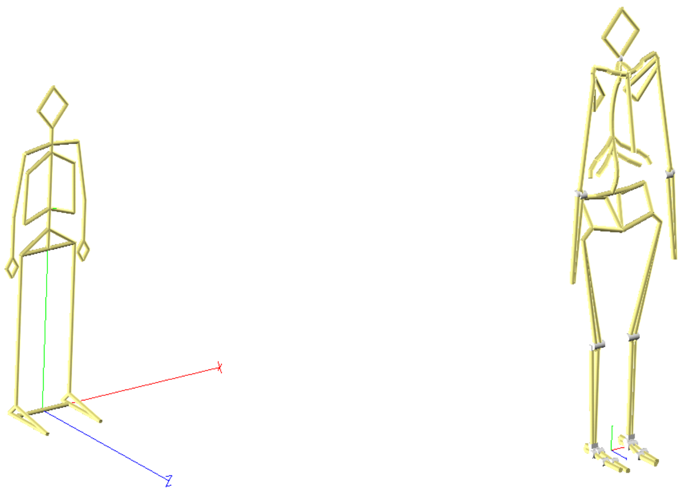
### 4.2.2 Starting point

The program is inspired and partially derived from Ole Ivar Sivertsen's Mechanism Program. The program comprised a model of the human skeleton, mainly two dimensional, except for the feet, which was two dimensional in a plane orthogonal to the rest of the body. An attempt was therefore made to improve the topology and geometry of this model. Examples of conducted improvements on the model:

- Topology is closer to reality:
  - Clavicle is now connected to the sternum
  - Feet have more bones
  - Shoulder blades are added
  - Spine has been divided into 24 sections
- Bones is modeled closer to reality:
  - Added neck to femur bone

- The clavicle has an arc
- Spine has added depth

See Figure 4.7 for the results illustrating these improvements.



(a) Model of human skeleton. The model is mainly in 2D and there are some topology issues. For instance, the clavicle is directly connected to the spine.

(b) A refined model of the human skeleton. The details of the model have increased. There is now an individual cylinder for every vertebrae in the spine. The clavicle now originates from the sternum.

Figure 4.7: Models in Ole Ivar Sivertsen's Mechanism Program

Despite these improvements, the deviation between the rendered model and reality was concluded to be too high. This is because the original program was made for mechanical mechanisms. Human joints have mechanical joint analogies, but is not similar in composition. It was decided that a standalone program, focusing on the human musculoskeletal system, was more fitted for the task. However, the Mechanism Program and the Musculoskeletal Program share the same basic structure and similar input methods. Sections 4.2.3 and 4.2.4 describes the Musculoskeletal Program in detail. For an overview of the program, see Figure 4.8.

### 4.2.3 Program Input

The program creates models based on input that has been written prior to execution. The models are defined in text files with a specific format and they are described in Section 4.2.3.1-4.2.3.3. The text files are stored in a folder called "models" within the system folder. See Figure 4.10 for an example directory. For the pelvis, femur and tibia there are properties in the program during runtime that can alter different parameters, such as femur shaft radius and pelvis height. See Figure 4.9.

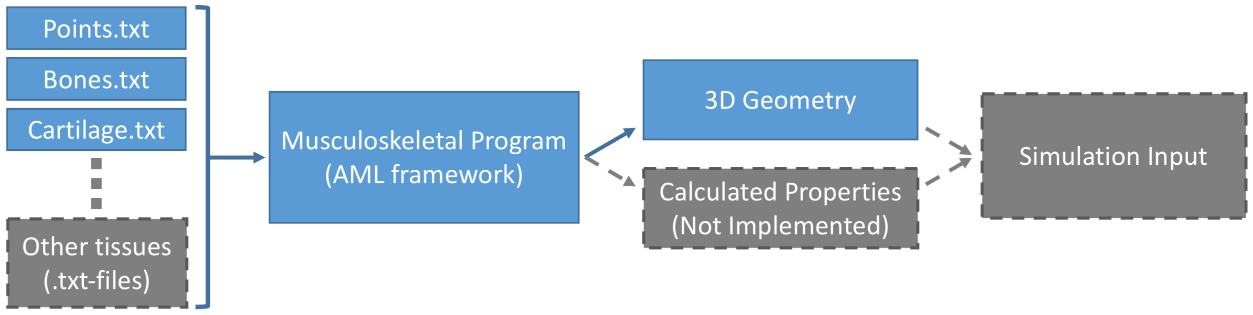
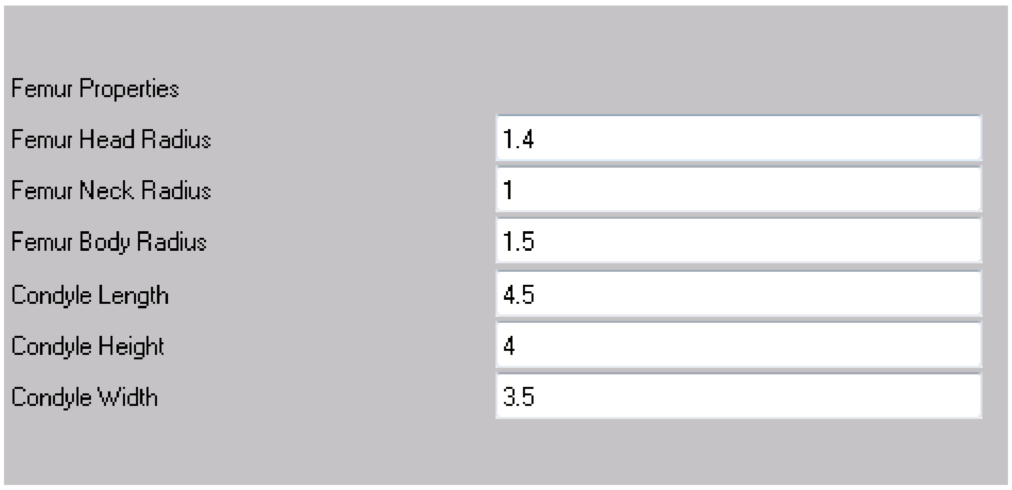

Points.txt
Bones.txt
Cartilage.txt
Other tissues (.txt-files)
Musculoskeletal Program (AML framework)
3D Geometry
Calculated Properties (Not Implemented)
Simulation Input

Figure 4.8: Flowchart of Musculoskeletal Program. Input is created as text files. The program generates geometry and calculates properties which are the basis for simulation input. The dashed arrows and boxes represent features that are not implemented.



Femur Properties

| | |
|---|---|
| Femur Head Radius | 1.4 |
| Femur Neck Radius | 1 |
| Femur Body Radius | 1.5 |
| Condyle Length | 4.5 |
| Condyle Height | 4 |
| Condyle Width | 3.5 |

Figure 4.9: Runtime properties for the femur. Similar property fields exists for tibia and pelvis.

#### 4.2.3.1 Point List

The input file *points.txt* contains a list of coordinates. The coordinates are cartesian and in the following order:

```
(x-coordinate y-coordinate z-coordinate)
```

The points are referred to in the other input files and can be used to define heights, widths and radii for geometric objects. Additionallt, the points can be used to create help objects like vectors and coordinate systems. A list of points could be written: (See Figure 4.12 for result)

```
-1.0   0.0   -1.0
 1.0   1.0    0.0
 0.0  -1.0   -1.0
 0.0   2.0   -1.0
 1.0   1.0    1.0
```

The point list can also be modified during runtime by opening ”Point list” in Main Properties.

#### 4.2.3.2 Bone List

The file *bones.txt* defines bones in the model. How the bones are generated is described in Section 4.2.4. One line in the input file defines one bone in the model. The format of the definition is on the form:

```
(Bone name) (Final element class) (List of points) (Bone type)
```

The *bone name* is visible in the tree hierarchy in the program. The *final element class* is not implemented in this version of the program, but is included for future use. A given point in *list of points* refers to a specific point in the *points.txt* input file. The numbering convention is *zero-indexed* which means that 0 refers to the first point in *points.txt*. The last element is the *bone type* which decides how the points are used to generate geometry. For instance, the first three vertebrae in the lumbar region of the spine could be written as:

```
"Lumbar5" "FE-vertebral" "(0 2)" "vertebral"
"Lumbar4" "FE-vertebral" "(3 5)" "vertebral"
"Lumbar3" "FE-vertebral" "(6 8)" "vertebral"
```

In the first line, ”0” refers to the first element i points.txt and ”2” to the third point.

#### 4.2.3.3 Cartilage List

The file *cartilage.txt* is similar to *bones.txt* and defines cartilage in the model. The format is:

```
(Cartilage name) (Final element class) (List of points) (Cartilage type)
```

The *cartilage name* is visible in the tree hierarchy in the program. The *final element class* is not implemented in this version of the program, but is included for future use. A given point in *list of points* refers to a specific point in the *points.txt* input file. The numbering convention is *zero-indexed* which means that 0 refers to the first point in *points.txt*. The last element is the *cartilage type* which decides how the points are used to generate geometry. The first two intervertebral discs in the lumbar region of the spine could be written as:

```
"L5-L4" "FE-intervertebral-disc" "(0 2 3 5)" "intervertebral-disc"
"L4-L3" "FE-intervertebral-disc" "(3 5 6 8)" "intervertebral-disc"
```

#### 4.2.3.4 Models Library

Models are made by creating points.txt, bones.txt and cartilage.txt files. The files are stored in a folder named with the model name. The folder is then placed in the ”models” folder in the directory where the system is located. See Figure 4.10. A system can have multiple models, which can be selected in a drop down menu when the program is executed. See Figure 4.11 for an illustration of this.

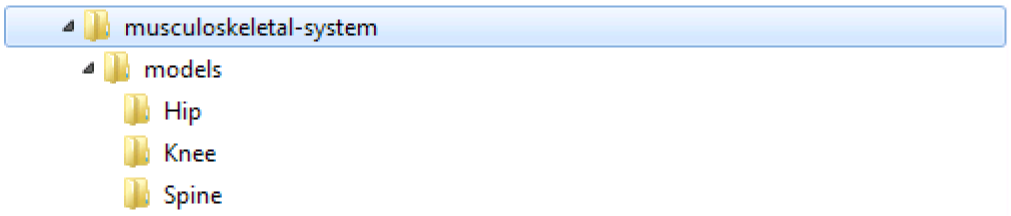


Figure 4.10: Example of file directory. The ”models” folder contain folders which in turn contain input files.

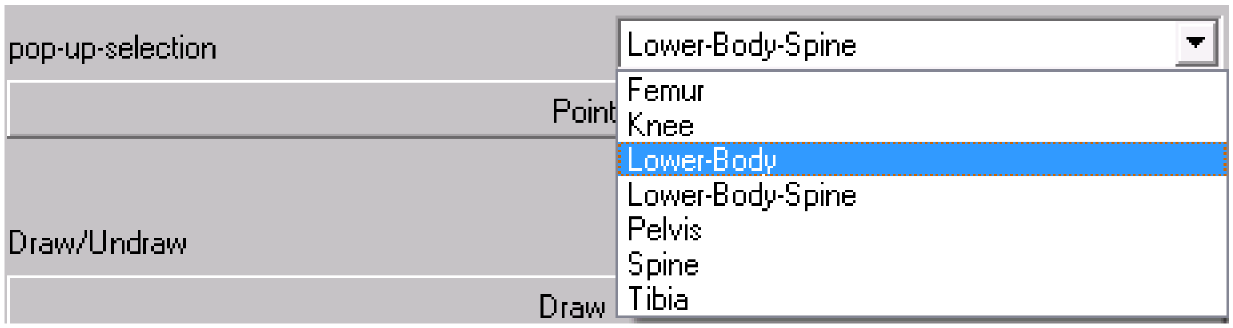


Figure 4.11: Dropdown Menu in Program. When the program is executed and the user edits the ”start-ui”-object, a list of the available models are shown.

### 4.2.4 Geometric Rules

#### 4.2.4.1 Geometry Collection and Part

The methods for generating bone and cartilage geometry are similar, but divided in separate *class definitions* in the file *"geometry.aml"*. In the remaining part of this section the word "tissue" represents either "bone" or "cartilage". Both tissues will be found represented in one class called "tissue-collection", and in another class called "tissue-part". The collection class holds a list of every element in the tissue input file. The class inherits from the class *series-object*. Series-object enables multiple objects to be initialised based on a quantity. The quantity is the sum of elements in the tissues list. The object class initialised is set in *class-expression*. The class-expression is either *bone-part* or *cartilage-part*. The tissue-part is where the specific tissue is instantiated. Consequently, if a bone in the input list is specified as a vertebral, the class bone-part will instantiate an object of class vertebral.

#### 4.2.4.2 Triads

The program reads a *point.txt* file to *points-list* in the class *musculoskeletal-program*. This list is then used to create *coordinate-system-objects*. The coordinate-system-objects are called "triads" and are oriented in $x$, $y$ and $z$-direction relative to the global coordinate system, which have origin in x=0, y=0, z=0. See Figure 4.12 for an illustration of how the nodes are represented in the "AML Main Modeling Form".

#### 4.2.4.3 vertebral.aml

The bones comprising the spine is called "vertebrae". In reality, all vertebrae have different geometries, but most of them have similar topology. See Figure 4.13. In this program, the vertebrae are modeled identically. However, the scale of the models differs. This is done by connecting the distance between the two points that defines a vertebral, with the size of it. The vertebral is aligned to the first point so that the point is in the center of the bottom face. Then the vertebral is rotated, so that the axial of the main body follows a vector in which has been drawn from point 1 to point 2. See Figure 4.14. A vertebral declaration in bones.txt have the form:

```
"Lumbar5" "FE-vertebral" "(0 1)" "vertebral"
```

Vertebral objects are modeled by applying Boolean operations on native geometric objects. They are built as a *union-object* of 4 cylinders and a *difference-object*. See Section 3.1.1 for definitions. The *difference-object* is a result of two cylinders with different radii that are subtracted from each other. All parts are then united

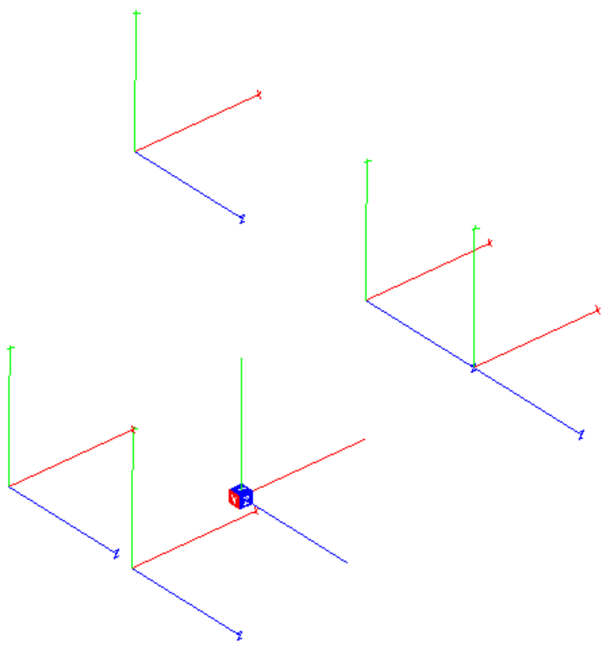


Figure 4.12: Triads defined by *points.txt*. The triads are used by classes to define geometry. The triad with a box in the center is representing origin in the global coordinate system.

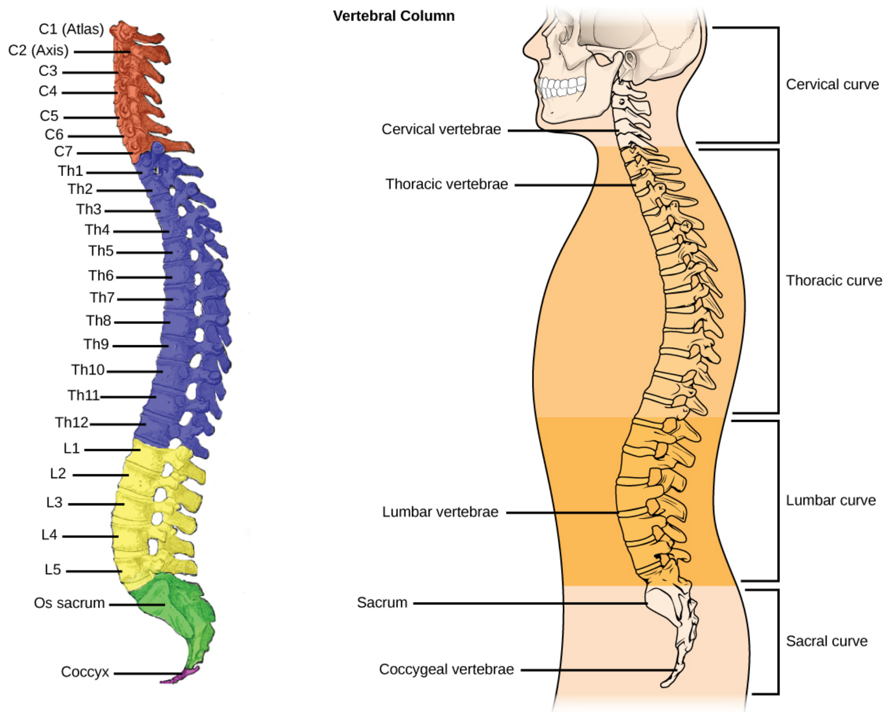


Figure 4.13: The human spine, with 24 vertebrae and sacrum and coccyx. The vertebrae are divided in three classes, the lumbar, thoracic and cervival vertebrae. Each vertebral has different geometry, but similar topology.

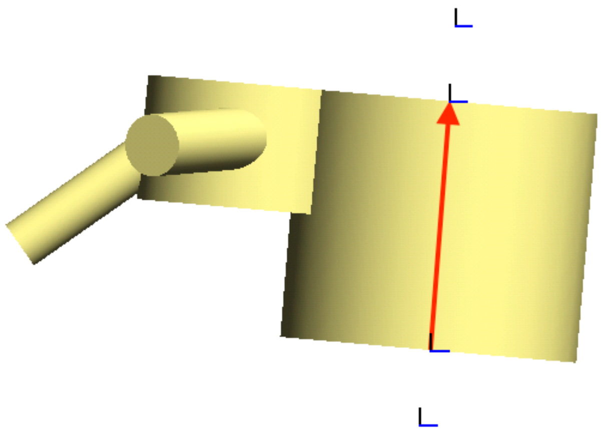


Figure 4.14: Vertebral objects are defined by two points. The orientation and alignment is found by calculating a vector from the first to the last point, which is declared in the bone definition.

to one final part, as indicated in Figure 4.15.

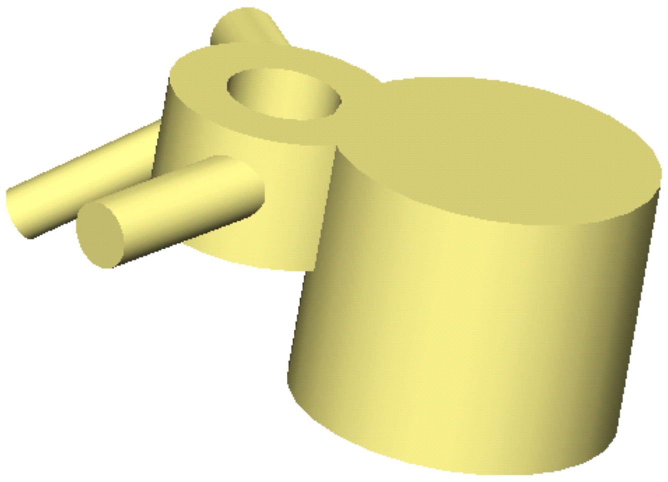


Figure 4.15: Vertebral geometry from the AML program.

#### 4.2.4.4 intervertebral-disc.aml

Intervertebral discs are the cartilage between vertebrae in the spine. They are modeled as elbow-objects, as represented in Figure 3.1d, and are defined by four points. Two points determines the start and end point. The center of the bottom face will be aligned with the start point, and the center of the top face with the end point. Vectors are calculated from the second point to the first point and from the third point to the fourth point. These vectors are normals to the bottom and top faces. See Figure 4.16. A vertebral declaration in bones.txt has the form:

```
"L5-L4" "FE-intervertebral-disc" "(0 1)" "intervertebral-disc"
```

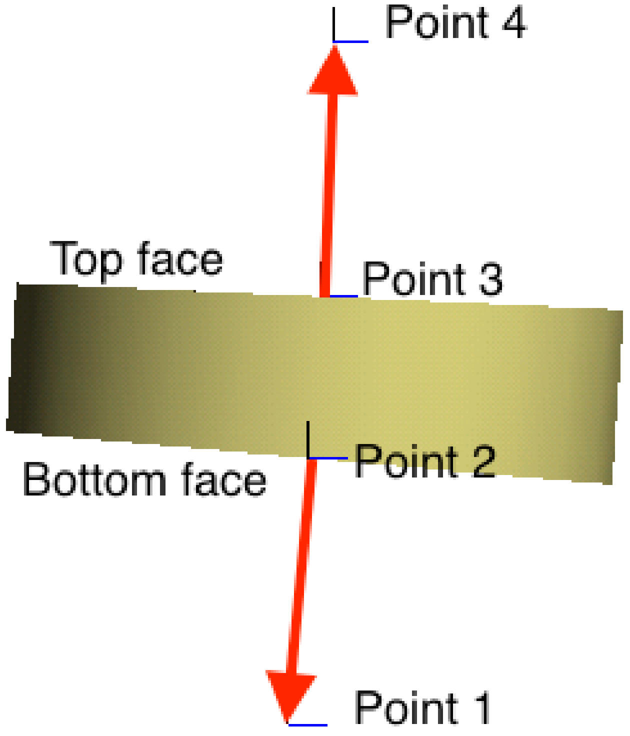
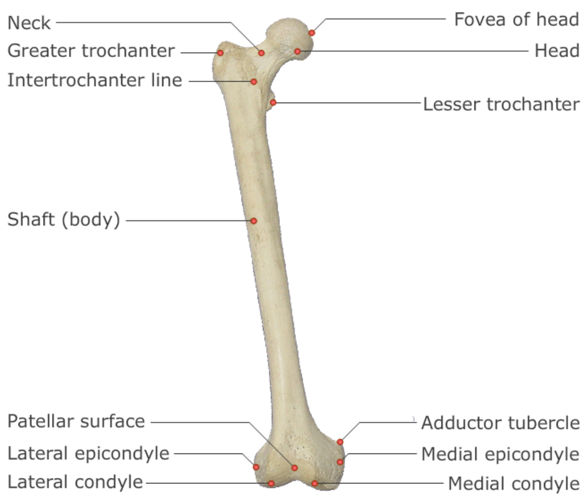


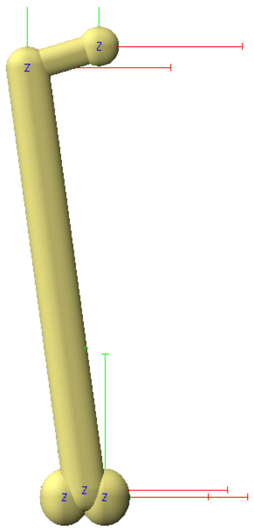
Figure 4.16: Intervertebral disc geometry. Top and bottom face are aligned with point 2 and 3 respectively. Vectors from point 3 to 4 and from point 2 to 1 are normals to the top and bottom faces respectively.

#### 4.2.4.5 femur.aml

The femur is the thigh bone and is defined by five points. The points are represented in Figure 4.17b as yellow and red lines. The two lower points are center points for ellipsoids which represents the femur condyles. The point between the condyles is a start point for the femur shaft which ends in the leftmost top point. The femur shaft is represented with a *cylinder-object*. The leftmost top point is the center for the "greater trochanter", and also the start point for the femur neck. The end point for the femur neck is in the rightmost top point, which is also the center for the femur head. See Figure 4.17a for femur landmark names.



(a) Femur bone with landmark names. Illustration by GetBodySmart.com.



(b) Femur bone and triads that define the bone.

Figure 4.17: Illustration of similarities between femur in AML geometry and "real" geometry.

#### 4.2.4.6 pelvis.aml

The pelvis is a complex structure, which can be found illustrated in Figure 4.18. The *pelvis-object* class in the Musculoskeletal Program is merely a representation. The process of creating the pelvis in AML was as follows: (See Figure 4.19 for visualization of the steps)

1. Add *box-object*
2. Add *ellipsoid-object*
3. Add *difference-object* and add (1) and (2) to the *object-list*
4. Add *ellipsoid-object* that surrounds the hollow part of the box.
5. Add *intersection-object* and add (3) and (4) to the *object-list*

6. Add *cylinder-object*, *truncated-cone-object* and two *ellipsoid-objects*
7. Add *difference-object* and add (5) and (6) (in that order) to the *object-list*

The size of the pelvis model is based on two points. One point is center for the left femur head, and the other point is center for the right femur head. See Figure 4.20.

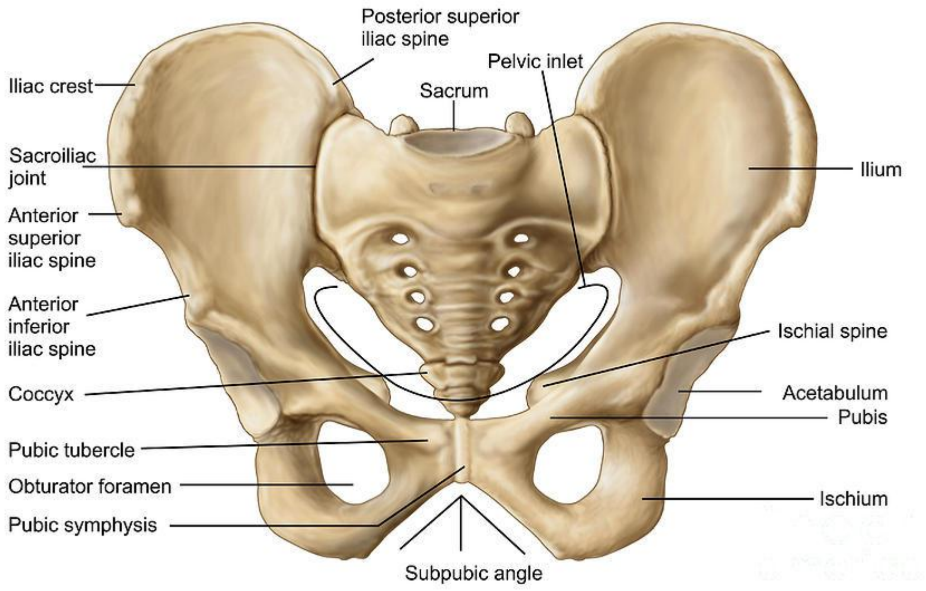


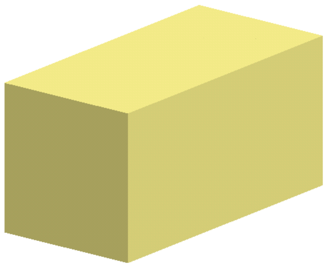
Figure 4.18: Illustration of the pelvis by Singapore.

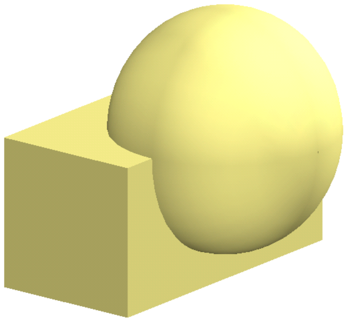
#### 4.2.4.7 tibia.aml

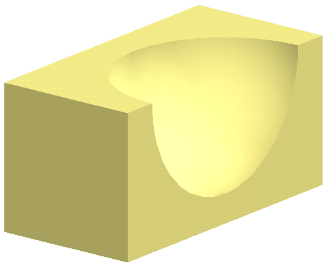
The tibia, represented in Figure 4.21a, comprises a cylinder as the tibia shaft and spheres in both ends. The top sphere, which represents the head of the tibia, is made of a *difference-object* of a sphere that is hollowed by another sphere. The hollow part can be adjusted during runtime to fit a femur object. The tibia is defined by two points, and a visualization of it can be found in Figure 4.21b.

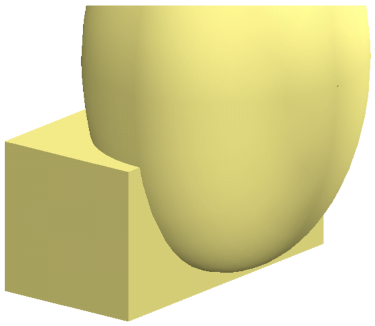
### 4.2.5 Spine Point Generator

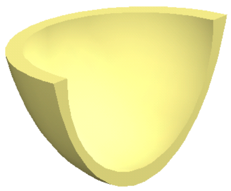
A spine point generator was programmed in Matlab to easier make changes to the spine model in the Musculoskeletal Program. During development of the spine model, it became apparent that it was easier to have a script when creating point lists, instead of writing them manually. The output is a *points.txt* file

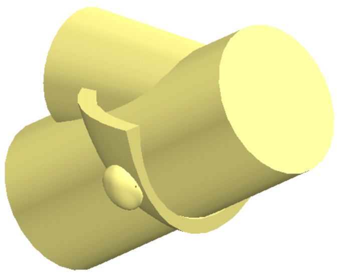

(a) 1: Box-object


(b) 2: Ellipsoid-object


(c) 3: Difference-object of 1 and 2


(d) 4: Ellipsoid-object


(e) 5: Intersection-object of 3 and 4


(f) 6: Ellipsoid-objects to make hollows for femur head and cylinder and cone objects to make pelvis inlet and connection point for the sacrum.

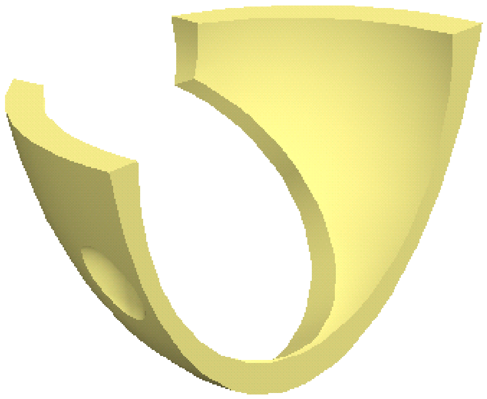

(g) 7: Difference-object of 6 and 5. This is the final representation of the pelvis.

Figure 4.19: The native geometric classes and Boolean operations to create a representation of the pelvis.

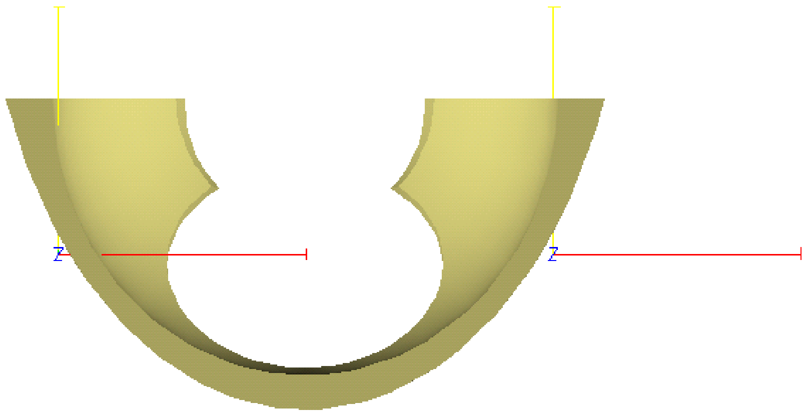
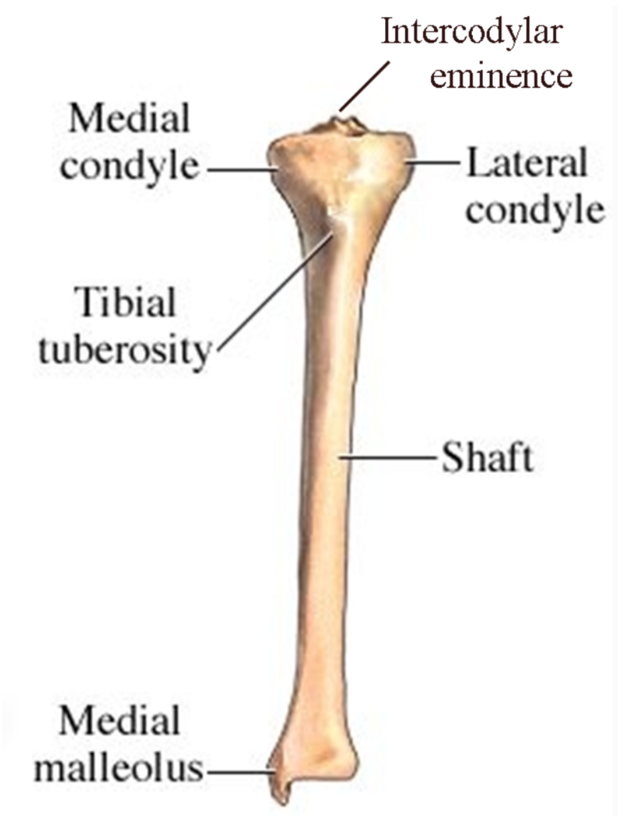
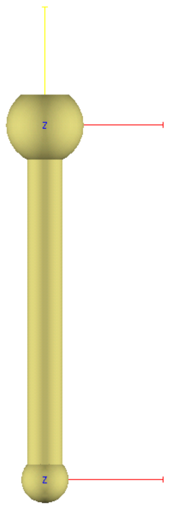


Figure 4.20: The two points in the illustration represents the center of the femur head. the pelvis size and position is determined by these points.



(a) Tibia with labels. Illustration by Aireurbano



(b) Tibia generated in AML from two points

Figure 4.21: A comparison between tibia in ”real” geometry and in the AML representation.

containing 74 points, corresponding to a *bones.txt* file (see complete file in Appendix B.2) previously written. The input parameters are height, lumbar angle, thoracic angle and cervical angle. See Figure 4.22 for an illustration of how the angles are defined. The program is able to generate a range of spine layouts, for some examples see Figure 4.23. However they are restricted to angles in the sagittal plane. See Figure 2.2. Therefore, the disease scoliosis is not in the domain of this program, whereas kyphosis and lordosis are. See Figure 4.24 for illustration of the forementioned diseases.

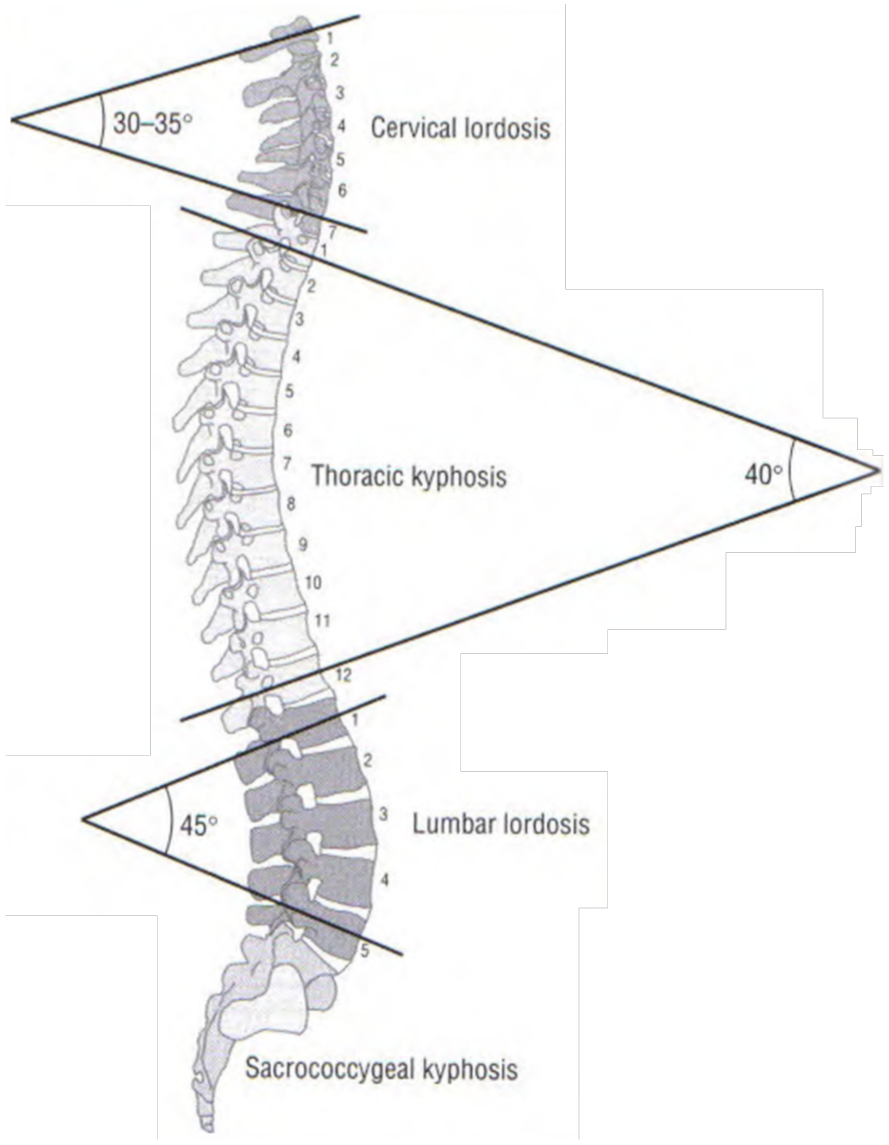
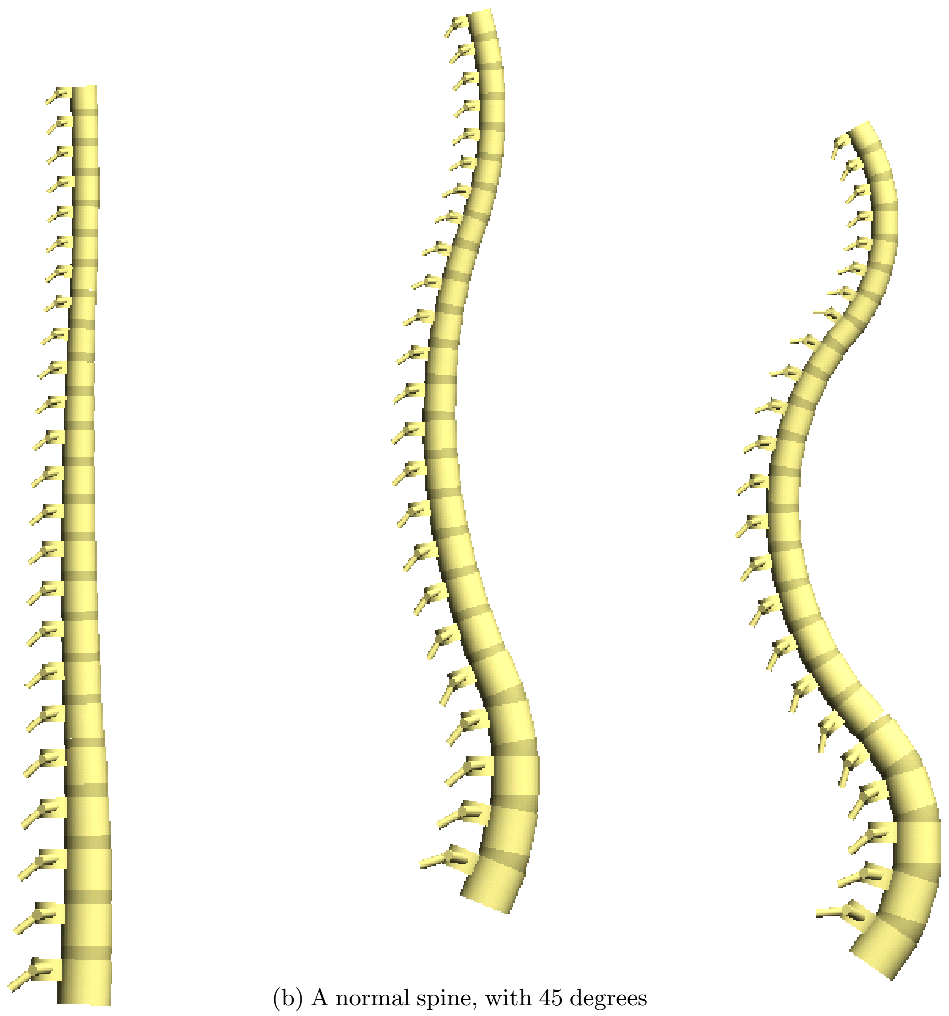


Figure 4.22: According to Neumann [2010], the spine with perfect cervical, thoracic and lumbar angles. Illustration by Neumann [2010].



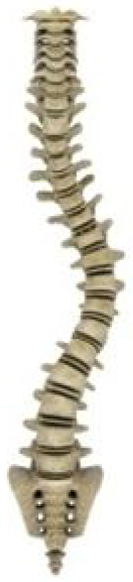
(a) An unlikely spine, with close to 0 degrees in all vertebrae sections.

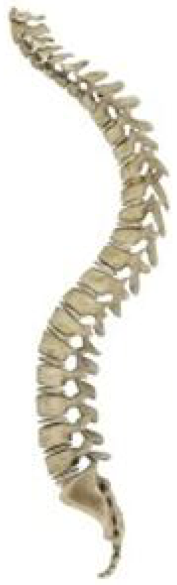
(b) A normal spine, with 45 degrees in lumbar region, 40 degrees in thoracic region and 30 degrees in cervical region.

(c) A bad spine, with symptoms of exaggerated lordosis and kyphosis.

Figure 4.23: Examples of spines generated from points made by the Spine Point Generator

(a) Lateral and anterior view of a normal spine.


(b) Scoliosis


(c) Kyphosis

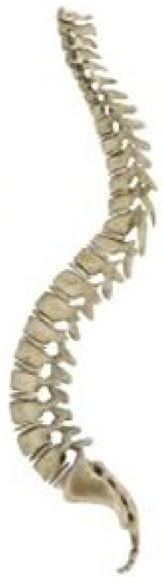

(d) Lordosis

Figure 4.24: Figure 4.24a shows a normal spine and Figure 4.24b, 4.24c and 4.24d shows different spinal diseases. The Spinal Point Generator can generate input for Kyphosis and lordosis, but are unable to generate angles in the frontal plane. Illustrations by PT.

# Chapter 5

# Results

This chapter reviews the results obtained from the segmentation tool written in Matlab and the Musculoskeletal Program written in Technosoft's AML.

## 5.1 Segmentation Tool

The Segmentation Tool was written to segment bone structure from medical images and visualize the results in 3D. This section reviews the results of segmenting four sets of CT-images from four parts of the body; hip, pelvis, torso and head. The objective of the segmentation was to obtain a visualization of the bone structure in each CT-image. Each segmentation is introduced with image information and execution time. Subsequently, the user interaction and the final results will be reviewed and described. All CT-images used in this section are obtained from `https://mri.radiology.uiowa.edu/visible_human_datasets.html`. They are images of a human male positioned in *HFS*. HFS is the term for "head-first supine", meaning that the person is on his back with the head first into the machine (NEMA [2015], section C.7.3.1.1.2).

### 5.1.1 Segmentation Data and Image Information

Table 5.1 summarizes the DICOM data, utilized to generate images in the Segmentation Tool. Slice thickness is the width of a section. It adds the volume dimension to an image. This is used to space each section in the final 3D visualization. Color type is the way the image data is represented. The program only supports gray scale. The color depth is the number of bits available to represent pixel values. A higher number can represent more colors and therefore represent data with higher precision. Pixels gives a value of how many individual data samples are captured during imaging and is given as ”width x height”. Patient positioning is the way the patient is positioned during image capture.

Table 5.2 summarizes the values used in the Segmentation Tool to generate a result. The number of sections is found by counting the number of files given as input. All values below the selected threshold will be set to 0 and effectively be deleted from the matrix. The Execution time is a measure of performance in seconds for the program. The ratio $\frac{executiontime}{numberofsections}$ gives an indication of performance and varies from 0.13 for the head to 0.2 for the torso segmentation.

Table 5.1: Data extracted from the DICOM files of the hip, head, pelvis and torso. The extraction is made in the Segmentation Tool.

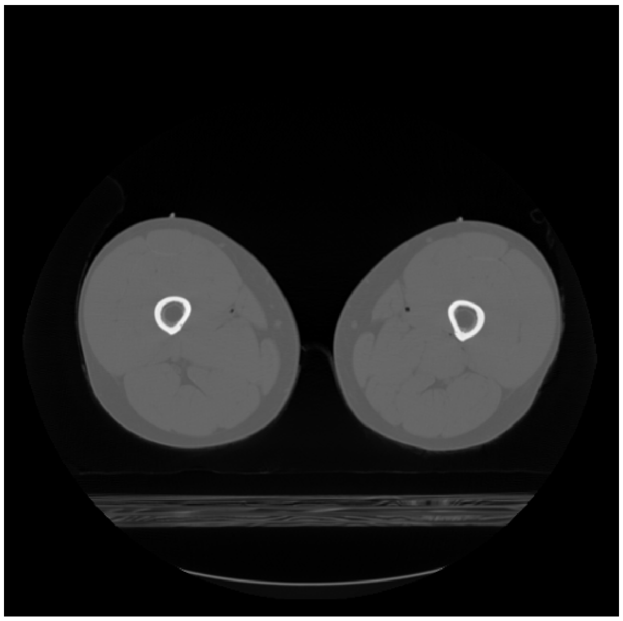
| **DICOM information** | **Hip** | **Head** | **Pelvis** | **Torso** |
|---|---|---|---|---|
| Slice Thickness | 1mm | 1mm | 1mm | 1mm |
| Color Depth | 16bit | 16bit | 16bit | 16bit |
| Color Type | Grayscale | Grayscale | Grayscale | Grayscale |
| Pixels | 512px X 512px | 512px x 512px | 512px x 512px | 512px x 512px |
| Patient Position | HFS | HFS | HFS | HFS |

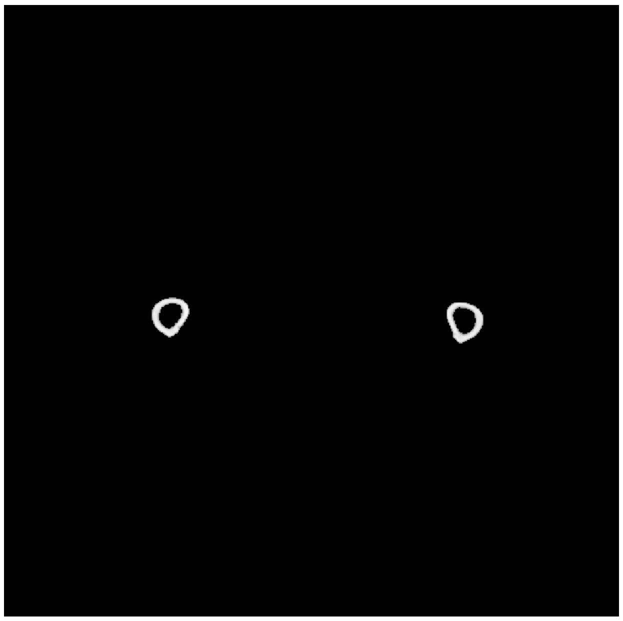
Table 5.2: Data from segmentation of hip, head, pelvis and torso in the Segmentation Tool.

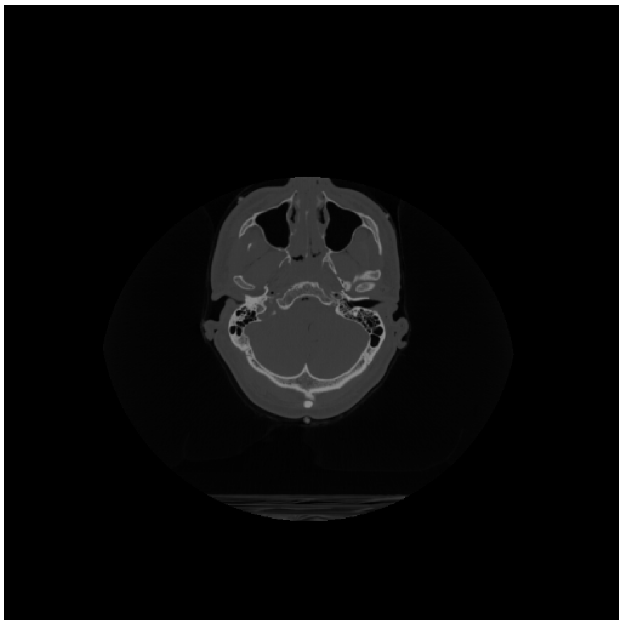
| **Segmentation Data** | **Hip** | **Head** | **Pelvis** | **Torso** |
|---|---|---|---|---|
| Number of sections | 400 | 245 | 140 | 460 |
| Selected Threshold | 35000 | 27000 | 30000 | 27000 |
| Execution Time | 80s | 32s | 25s | 93s |

### 5.1.2 Threshold Selection

The desired result of the segmentation is a 3D visualization of bone tissue only. During execution of the program, the user is prompted to input a threshold value which determines what values to erase from the result. All values below the threshold is deleted and a 2D image of the result is displayed. See Figure 5.1 and 5.2 for examples of threshold selection. Figure 5.1b shows an example of a clean segmentation, where the two white rings, which are femur bones, are the only visible objects. Figure 5.1d, 5.2b and 5.2d are slightly more difficult to classify as successful segmentations. However, when compared to their unsegmented counterparts in Figure 5.1c, 5.2a and 5.2c respectively, the selected threshold was approved to use in the isosurface generation. In Figure 5.2d there is an anomaly present along the edge of the scan area. This is analysed to be the soft tissue of the patient’s arm crossing the edge, and this is clearly an issue as this is not bone tissue.


(a) Selected threshold value at 0


(b) Selected threshold value at 35000


(c) Selected threshold value at 0

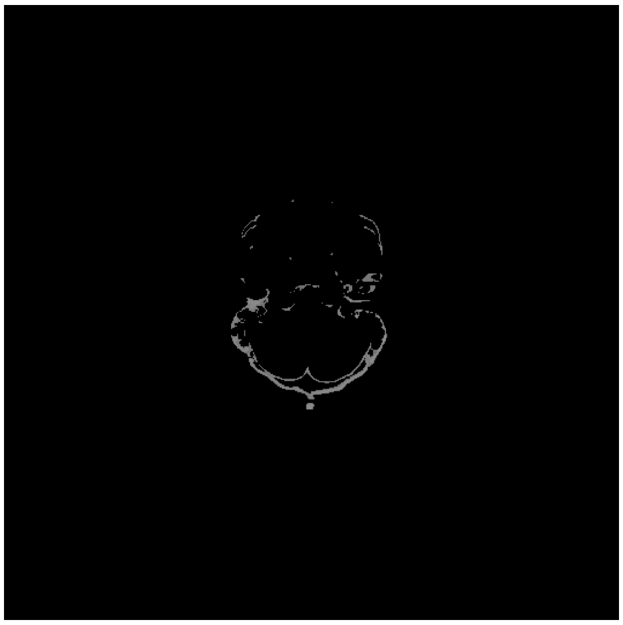
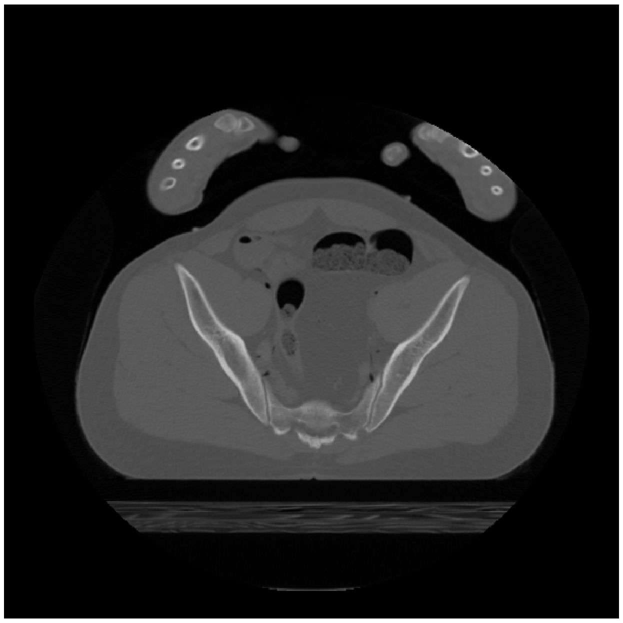
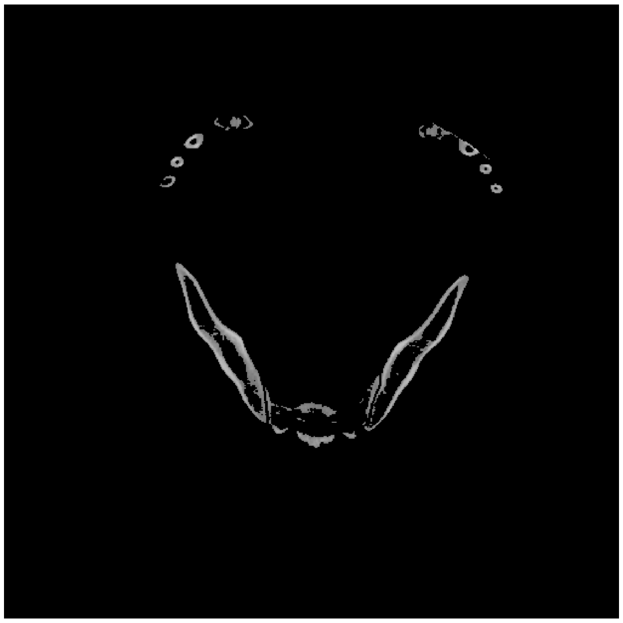
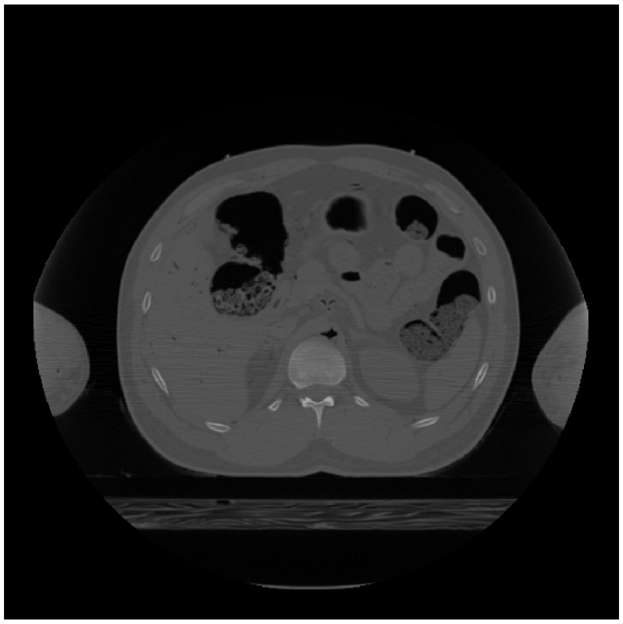

(d) Selected threshold value at 27000

Figure 5.1: 2D images of sections where a threshold value has been applied. Figure 5.1a and 5.1b represents how the hip looks and Figure 5.1c and 5.1d represents how the head looks in this process. Figure 5.1a and 5.1c are unsegmented, while Figure 5.1b and 5.1d are segmented. The images are from the horizontal plane.


(a) Selected threshold value at 0


(b) Selected threshold value at 30000


(c) Selected threshold value at 0

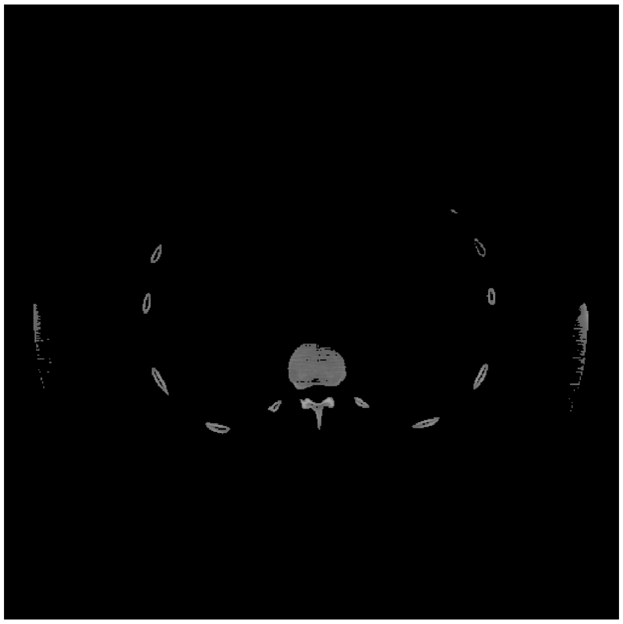

(d) Selected threshold value at 27000.

Figure 5.2: 2D images of sections where a threshold value has been applied. Figure 5.2a and 5.2b represents how the pelvis looks and Figure 5.2c and 5.2d represents how the shoulder looks in this process. Figure 5.2a and 5.2c are unsegmented, while Figure 5.2b and 5.2d are segmented. The images are from the horizontal plane. Along the edge of the scan area in 5.2d, there is an anomaly which cause unwanted geometry in the final model. The anomaly is caused by arms crossing the border of the scan area.

### 5.1.3 Isosurface Results

After the user has set the threshold value, an isosurface is applied on the matrix and the result is visualized as a 3D matrix in Matlab. The result of the isosurface generation of the segmented hip, head, pelvis and shoulder is represented in Figure 5.3, 5.5, 5.7 and 5.8 respectively. In the figures, the horizontal plane has unit pixels, and the vertical axis has unit sections, where one section represent one image. The blue geometry represents bone and the colorful fields at the edge of the images are the isocaps. This represents where solid bone exits the scan-area.

In Figure 5.3, the femur and lower part of the pelvis are displayed. The coccyx (tailbone) and fingertips can be shown posterior and anterior to the pelvis, respectively.

In Figure 5.4 the same fields are observed. However, they form circles with a hollow inside. This is where the bone marrow has been deleted from the matrix due to lower pixel value than cortical (hard) bone.

The result of segmenting a CT-scan of the head is represented in Figure 5.5. The surface has some discontinuous areas at the forehead. The image resolution is 512px x 512px, see Table 5.1. The low resolution may be a source for this error. However, this could also be a result of setting the threshold value too high.

A blue spot is observed next to the jaw to the right, as can be observed in Figure 5.6. This is an anomaly, and it is unknown what it might be, but an initial guess is that it is a part of the clavicle.

Figure 5.7 illustrates the result of segmentation of pelvis. The hands of the patient is visible. This is because the scan procedure is performed in a tight tube so the arms are place along side the body.

In Figure 5.8, a segmented torso is viewed. The ribs are observed as "hanging" in mid-air. This is due to the cartilage between the rib cage and sternum having a non-boney composition. Disturbance at the border of both sides is observed and is highlighted in Figure 5.9. It can be observed that the humerus is exiting the scan-area close to the disturbance, and the disturbance is analysed to be the soft tissue of the over arm exiting the scan-area.

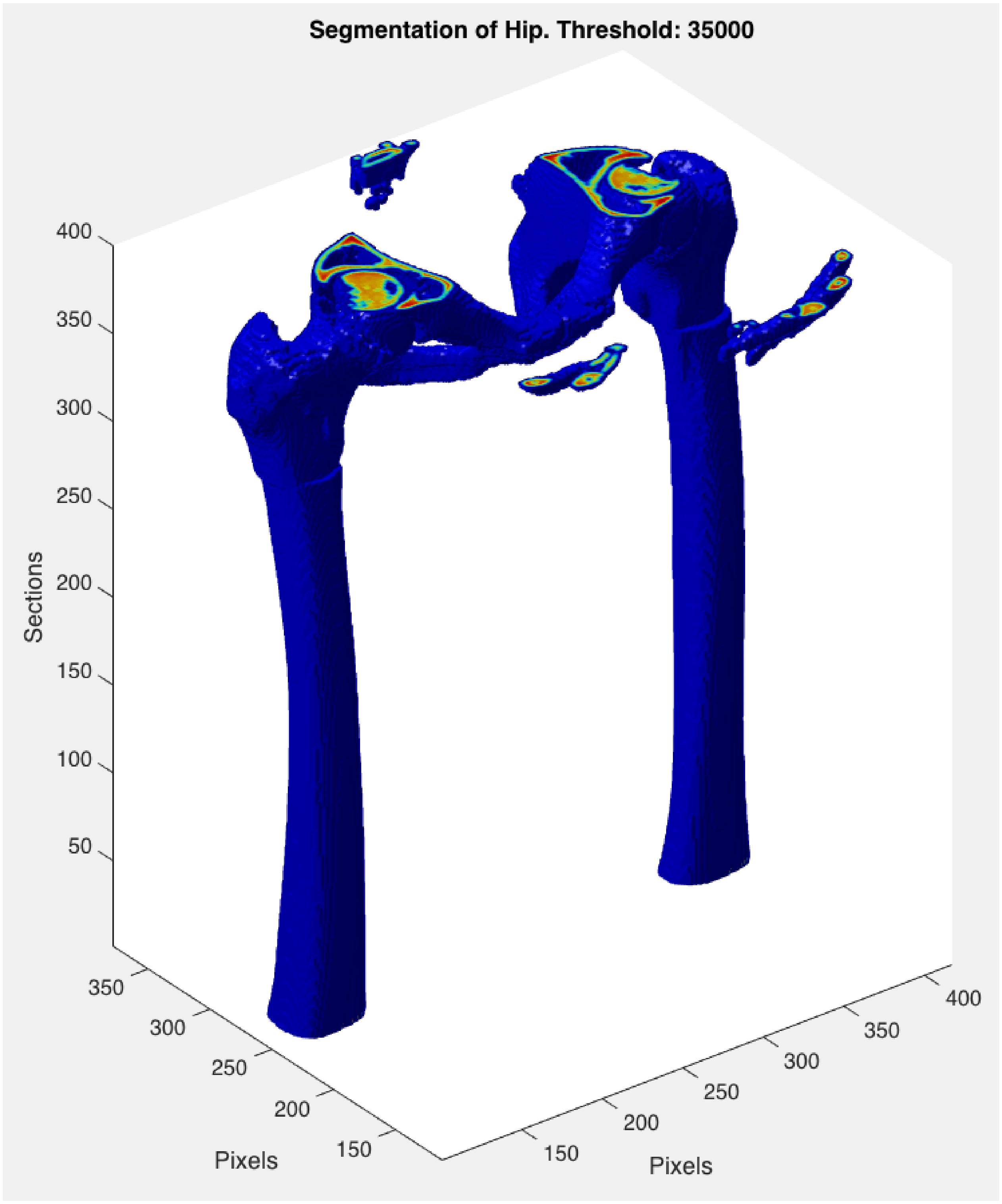


Figure 5.3: Isometric view of the generated isosurface of a segmented hip. Fingertips and tail bone is also present in the illustration.

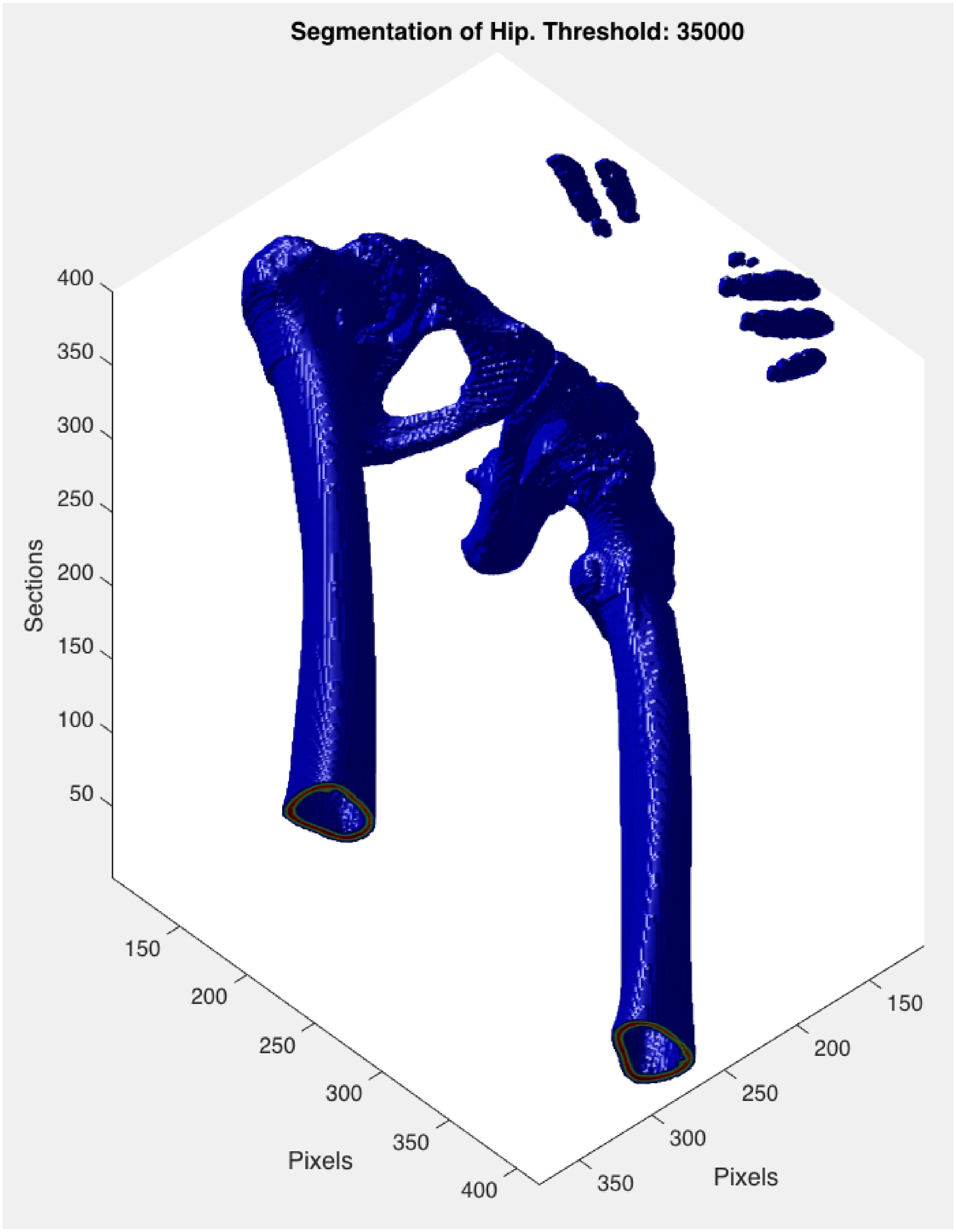


Figure 5.4: Isometric bottom-up view of the generated isosurface of a segmented hip. A hollow bone is the result of segmentation since bone marrow has lower pixel value than cortical bone.

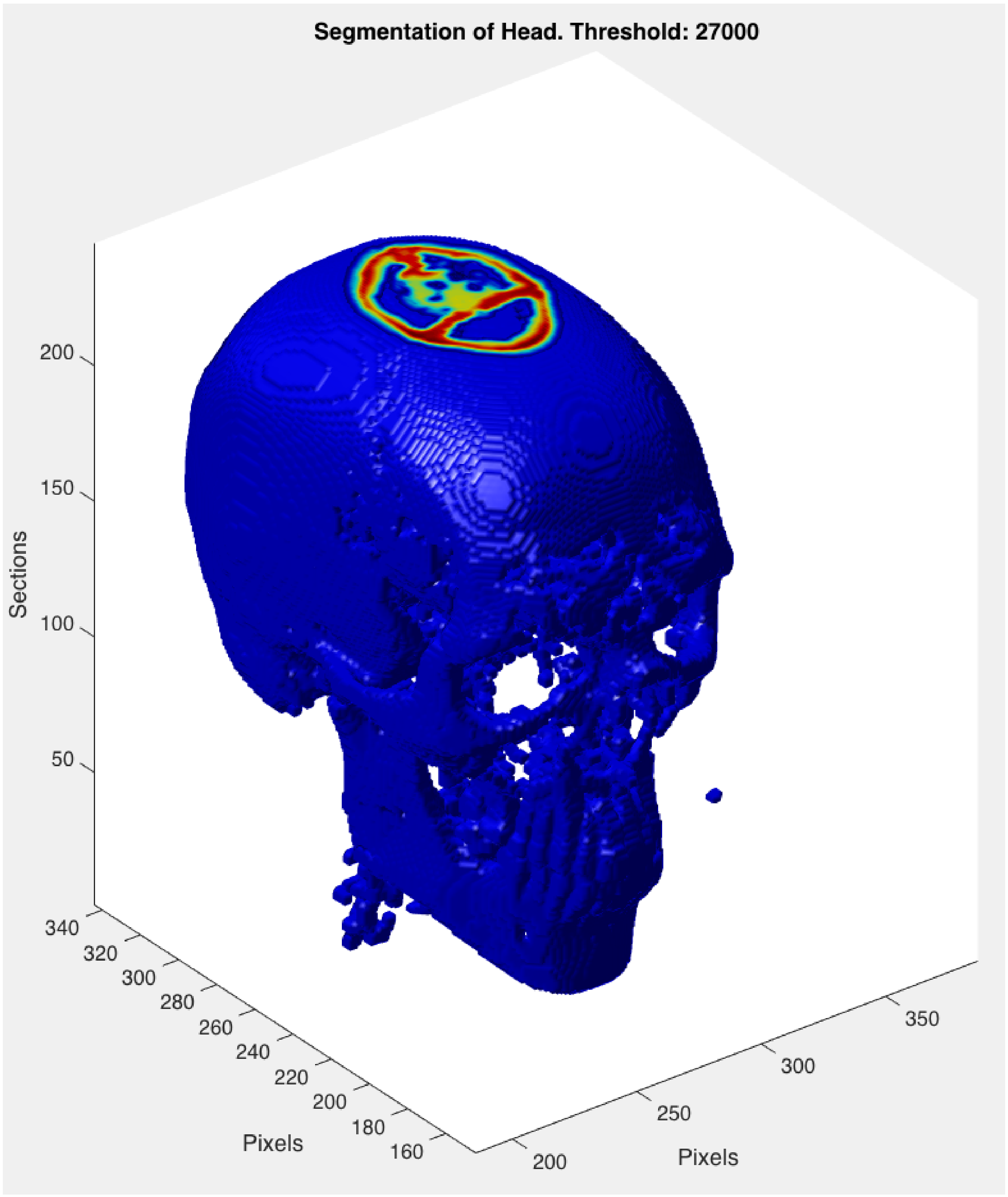


Figure 5.5: Isometric view of the generated isosurface of a segmented head.

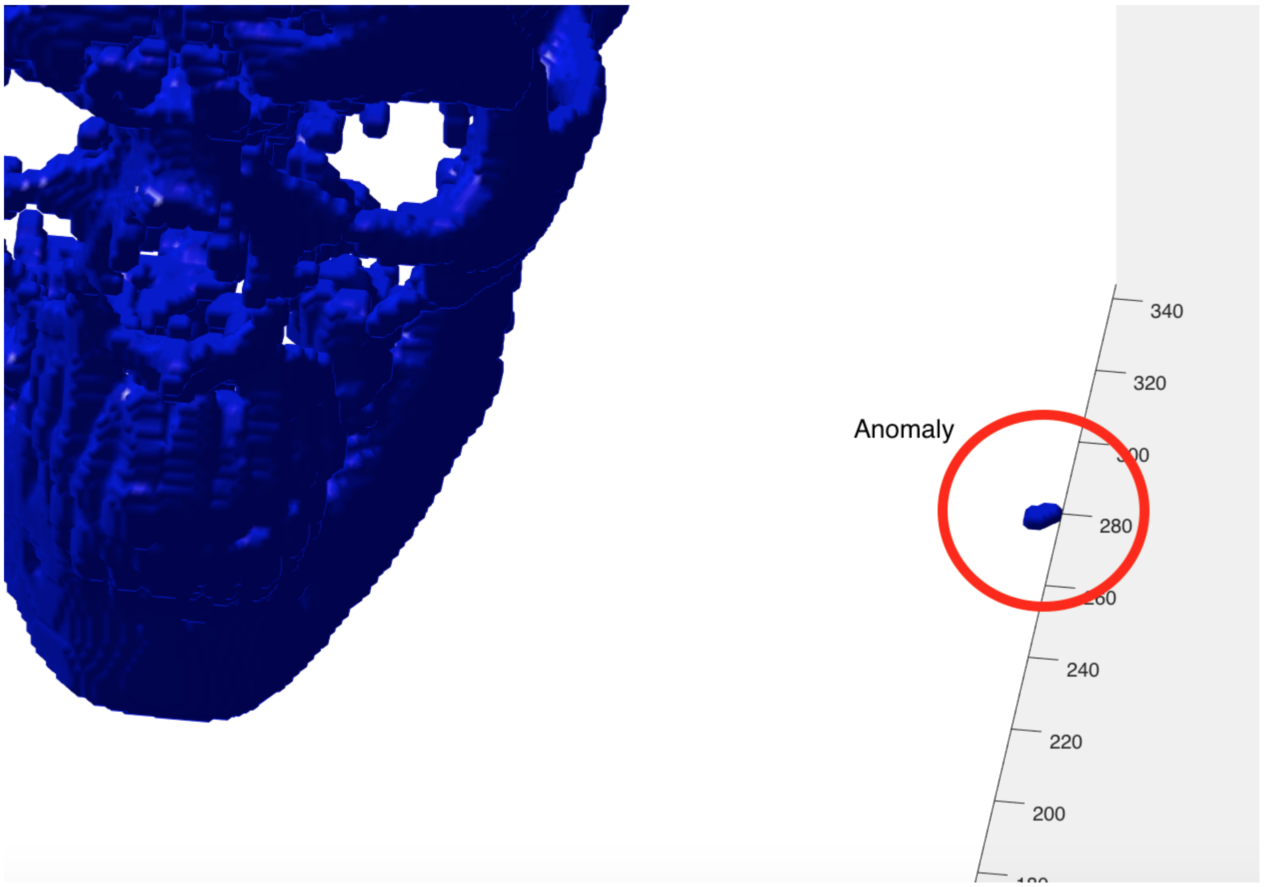


Figure 5.6: An anomaly is observed and highlighted in the illustration. It might be an object present during scan or bone from the clavicle.

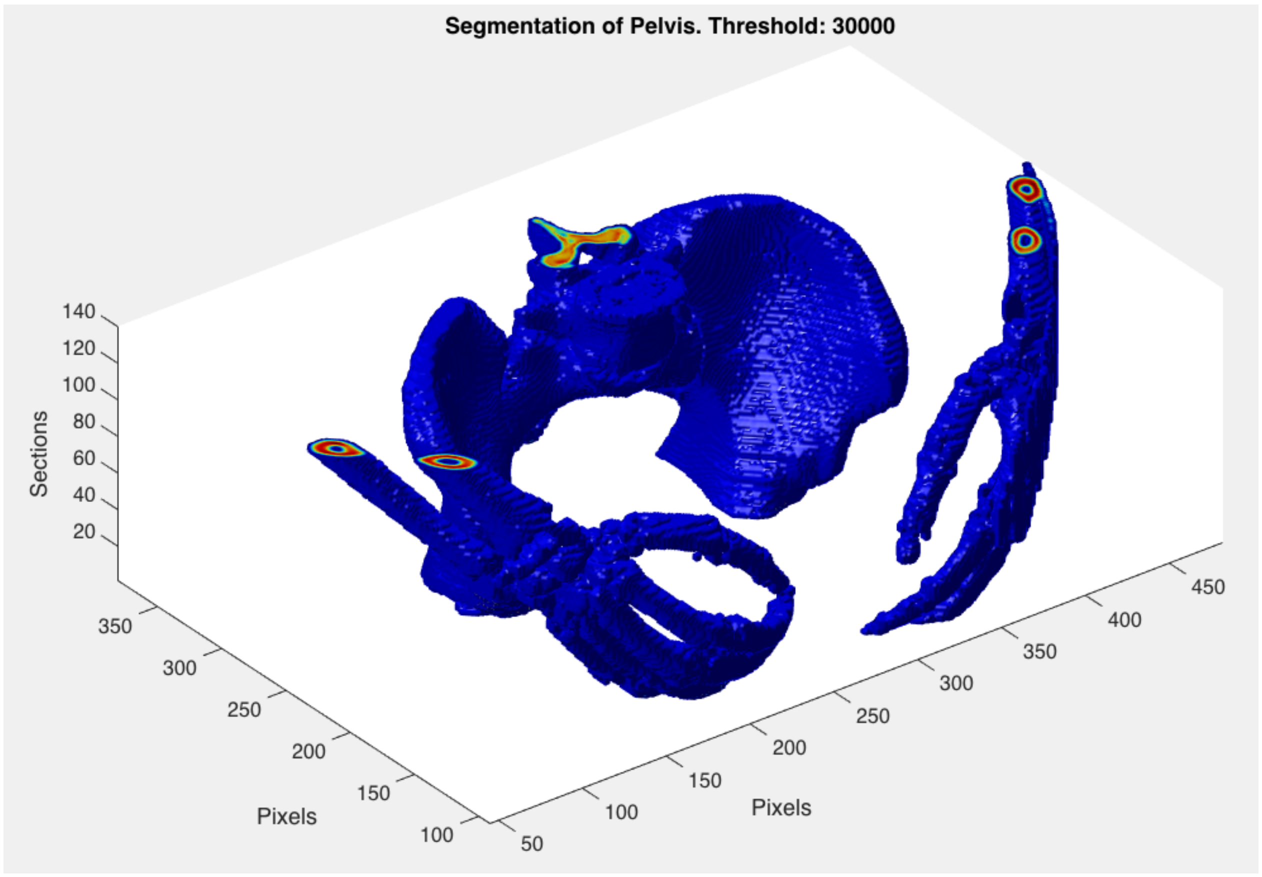


Figure 5.7: Isometric view of the result of segmentation of pelvis and hands.

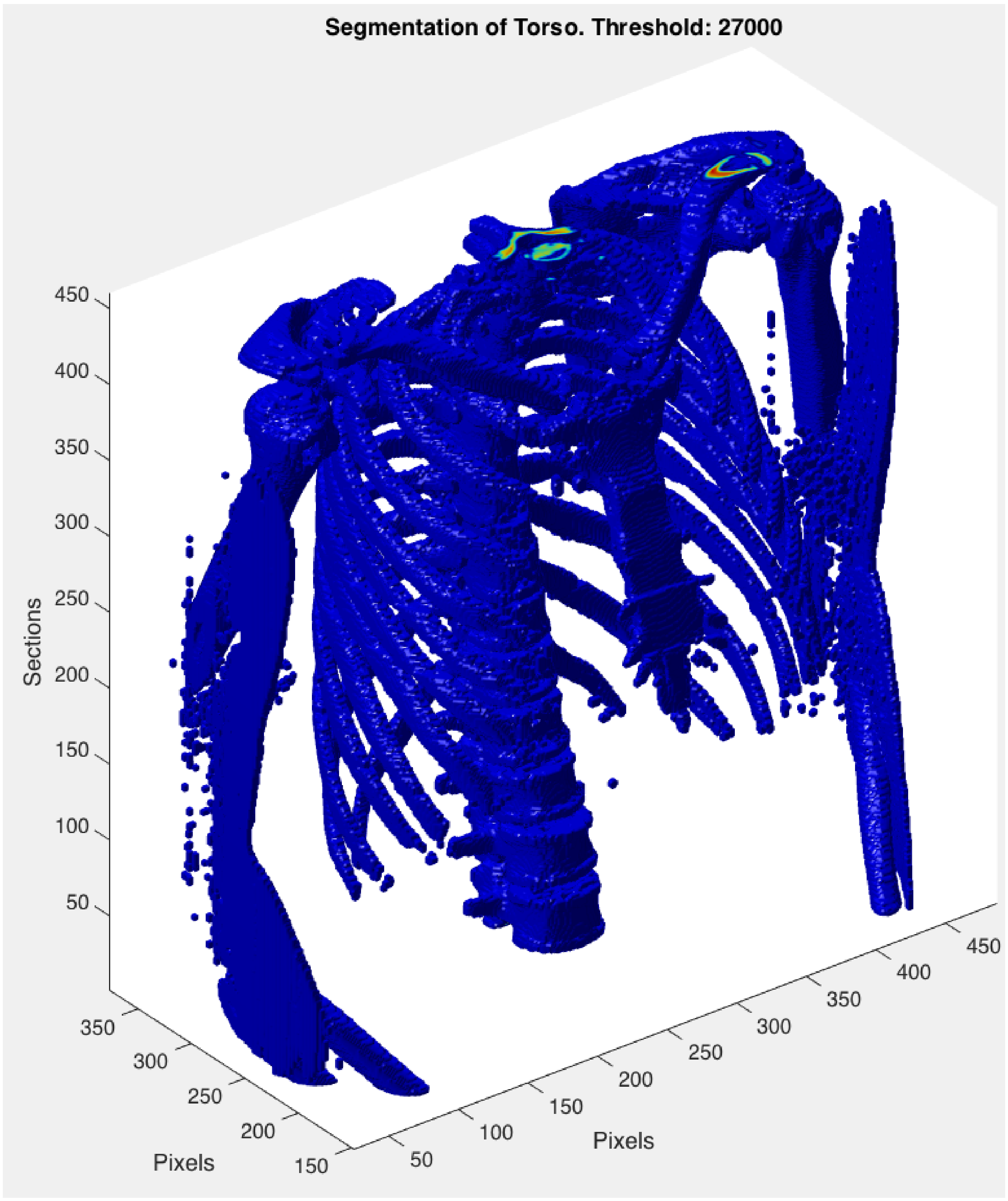


Figure 5.8: Isometric view of the result of segmentation of a torso. Disturbances are present in the result and are highlighted in Figure 5.9.

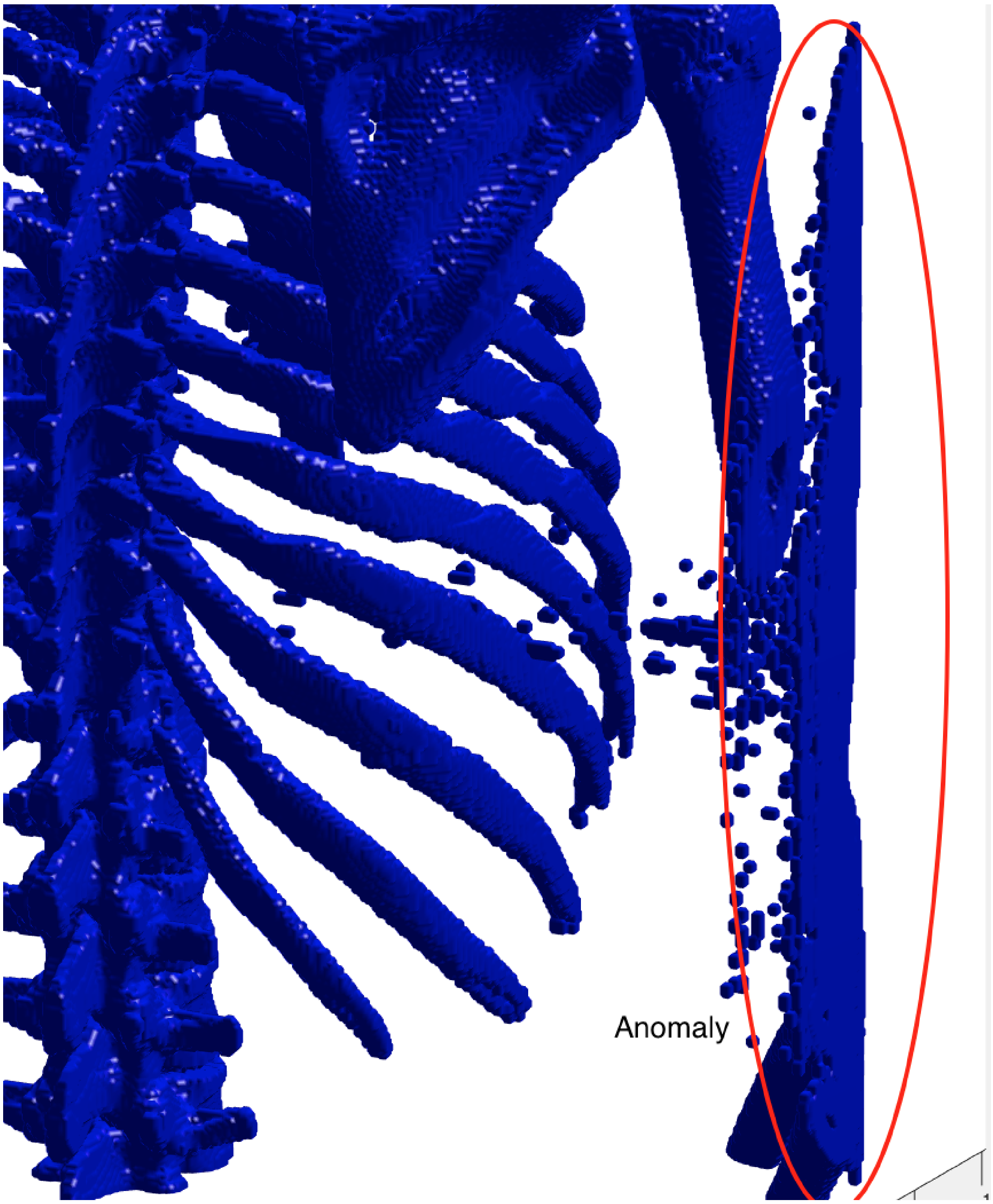


Figure 5.9: Close up of right shoulder of the patient. The red ring highlights an anomaly that is analysed to arise when soft tissue exits the scan-area.

## 5.2 Musculoskeletal Program

The AML Musculoskeletal Program was developed to create bone geometry from text files. The text files can either be written manually or generated by software such as the ”Spine Point Generator”. The following sections will review the results of different input to the program, and outline what can be done to optimize the result.

A model is instantiated by selecting from a drop down menu, illustrated in Figure 5.10. The menu is displayed by editing the ”start-ui” object in the program tree. See Figure 5.11. Models that have been added to the *Model Library*, will be available in the drop down menu. When a model is selected, the mechanism-object will be populated with new objects. These objects are determined by the input files for that distinct model. The object tree in Figure 5.11a shows a model that is comprised of tibia, femur, pelvis and spine. Figure 5.11b shows the accompanying object tree of cartilage. Cartilage is only available for the spine.

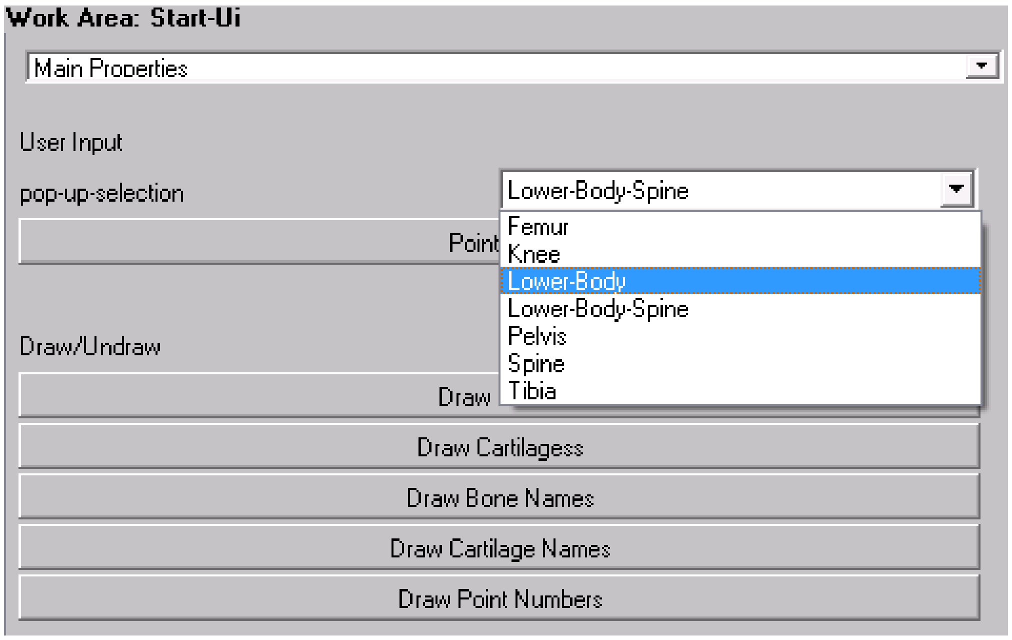
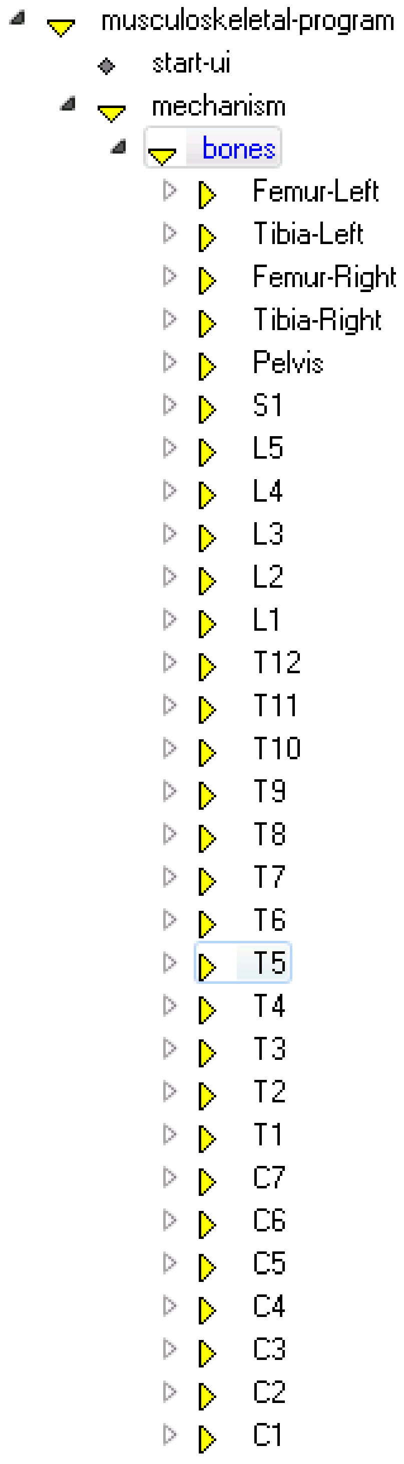
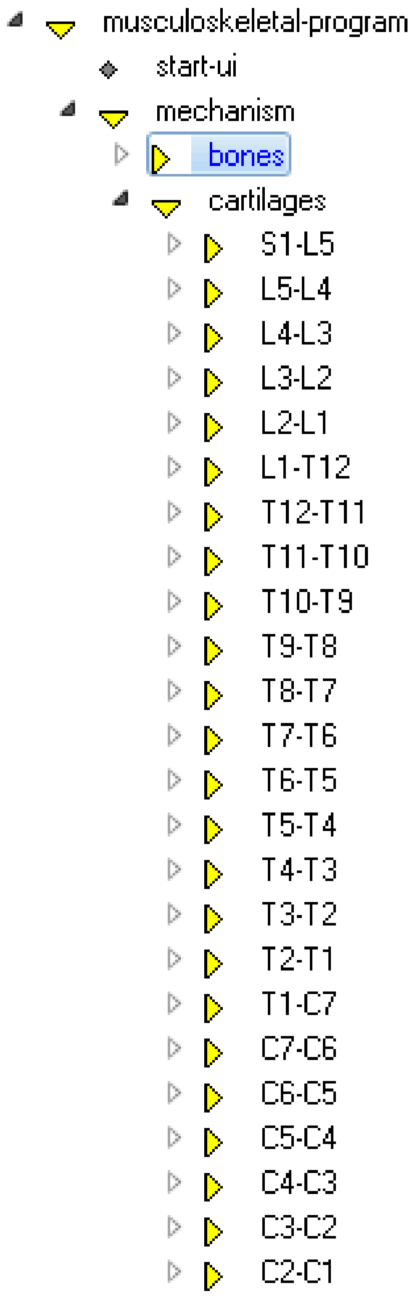


Figure 5.10: Main Properties are shown when right clicking ”start-ui” and selecting ”edit”. The drop down menu which is active in the image holds the different models that are available.



```
musculoskeletal-program
  start-ui
  mechanism
    bones
      Femur-Left
      Tibia-Left
      Femur-Right
      Tibia-Right
      Pelvis
      S1
      L5
      L4
      L3
      L2
      L1
      T12
      T11
      T10
      T9
      T8
      T7
      T6
      T5
      T4
      T3
      T2
      T1
      C7
      C6
      C5
      C4
      C3
      C2
      C1
```

(a) Object tree showing a list of bones in the selected model.



```
musculoskeletal-program
  start-ui
  mechanism
    bones
    cartilages
      S1-L5
      L5-L4
      L4-L3
      L3-L2
      L2-L1
      L1-T12
      T12-T11
      T11-T10
      T10-T9
      T9-T8
      T8-T7
      T7-T6
      T6-T5
      T5-T4
      T4-T3
      T3-T2
      T2-T1
      T1-C7
      C7-C6
      C6-C5
      C5-C4
      C4-C3
      C3-C2
      C2-C1
```

(b) Object tree showing a list of cartilages in the model.

Figure 5.11: Illustrations of object trees when a model has been selected. The bones and cartilage list contains one object per row in the bonest.txt and cartilage.txt input files.

### 5.2.1 Input and properties

Two models of the femur bone is illustrated in Figure 5.12. The models share the same *bones.txt*-file:

```
"Femur-Left"  "FE-Femur" "(0 1 2 3 4)"  "femur"
```

The *points.txt* file is the reason for the difference in geometry. The file that resulted in Figure 5.12a is listed below:

```
-1.4   0.0    0.0
 1.4   0.0    0.0
 0.0   0.5    1.0
-4.0   30.0  -1.0
 1.0   31.5  -1.0
```

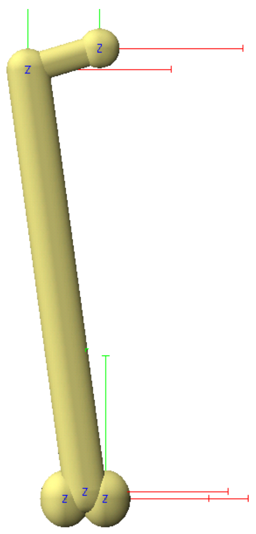
Where the coloumns are x, y and z coordinates respectively. The file that resulted in Figure 5.12b is listed below:

```
-1.4   0.0    0.0
 1.4   0.0    0.0
 0.0   0.5    1.0
-6.0   20.0  -1.0
 3.0   22.5  -1.0
```

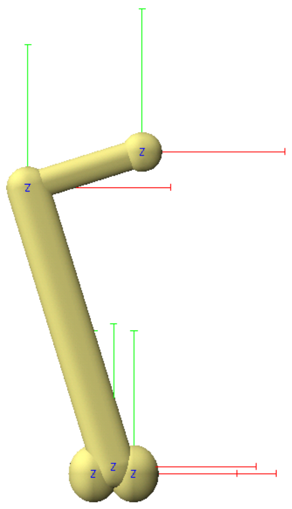
The first two rows are identical. In the fourth row, the x and y coordinates are changed. The result is that the model in Figure 5.12a is higher and narrower than the model in Figure 5.12b. The input file *points.txt* determines the centers of condyles, greater trochanter and femur head (see Figure 4.17a for femur landmarks). The femur shaft and neck are extruded between these points.

Figure 5.13 illustrates the properties that are available at runtime to further adjust the model. Femur head, neck and body radius can be adjusted and the condyles can be made longer, wider or heigher. The result of increasing the femur head and neck during runtime is illustrated in Figure 5.14.

The runtime adjustments can be exploited to fit joint elements together. The process of adjusting the knee joint is illustrated in Figure 5.15. The pelvis also have options for runtime adjustments, but these are limited to height and depth. See Figure 5.16. In Figure 5.16a the height of the pelvis is 14 cm and in Figure 5.16b the height is 18 cm. However, the center of the hip bowl and the width is unchanged and is set by the *points.txt*-file.

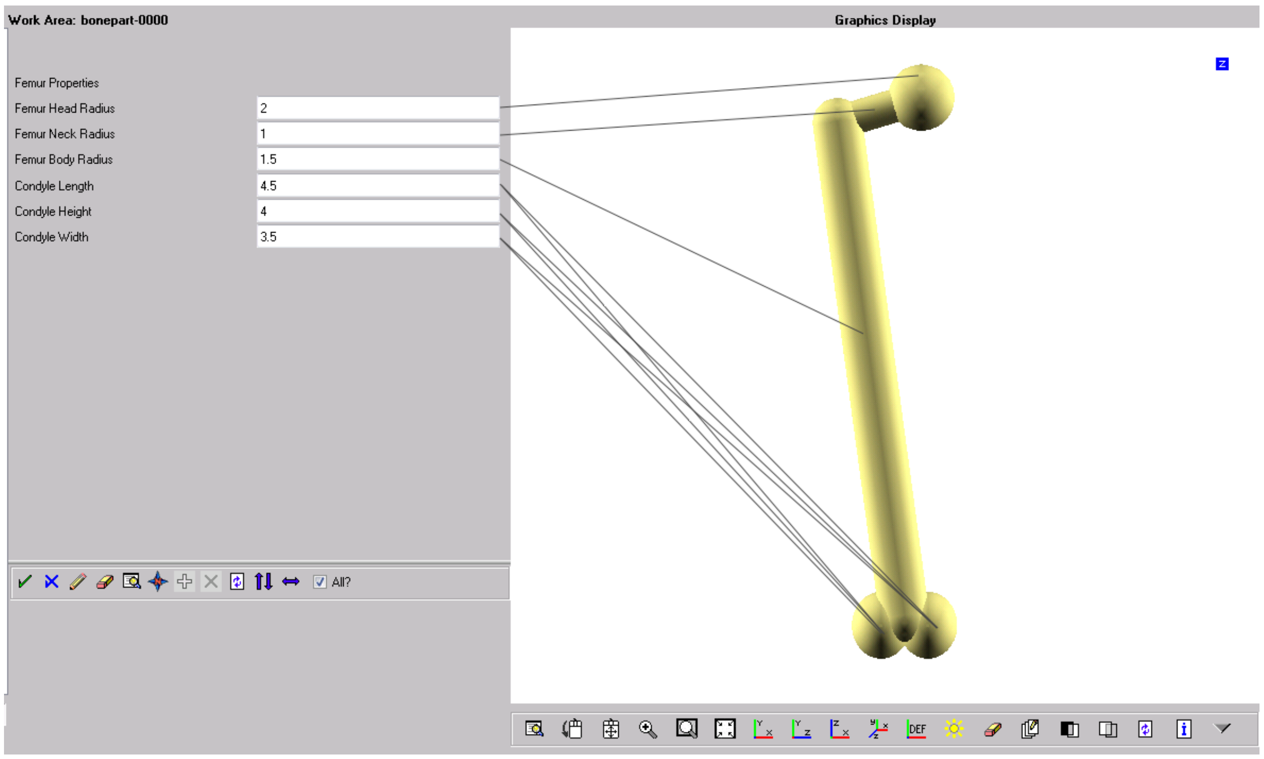


(a) Femur bone model. Height: 32 cm



(b) Femur bone model. Height: 23 cm

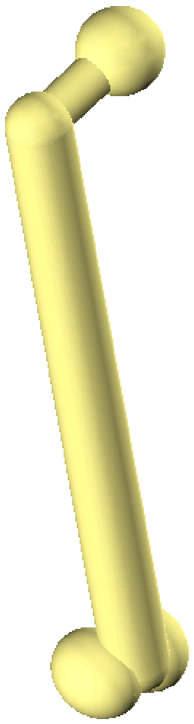
Figure 5.12: Illustration showing that different point.txt files result in different femur bone geometry.



Femur Properties

| | |
|---|---|
| Femur Head Radius | 2 |
| Femur Neck Radius | 1 |
| Femur Body Radius | 1.5 |
| Condyle Length | 4.5 |
| Condyle Height | 4 |
| Condyle Width | 3.5 |

Figure 5.13: Geometry in the Musculoskeletal Program have properties that can be adjusted during runtime. The properties for the femur geometry is illustrated above.


(a) Femur head radius: 1.4 cm
Femur neck radius: 1 cm

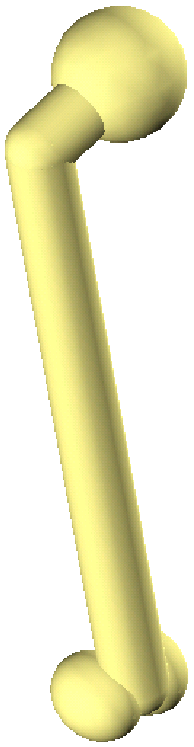
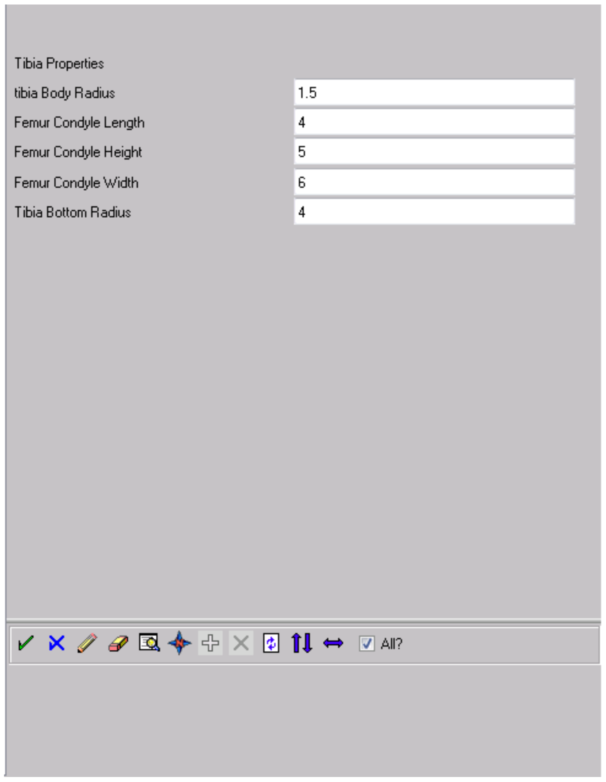

(b) Femur head radius: 2 cm
Femur neck radius: 1.5 cm

Figure 5.14: Femur geometries with different neck and head radii. The radii were adjusted during runtime.



Tibia Properties

| | |
|---|---|
| tibia Body Radius | 1.5 |
| Femur Condyle Length | 4 |
| Femur Condyle Height | 5 |
| Femur Condyle Width | 6 |
| Tibia Bottom Radius | 4 |

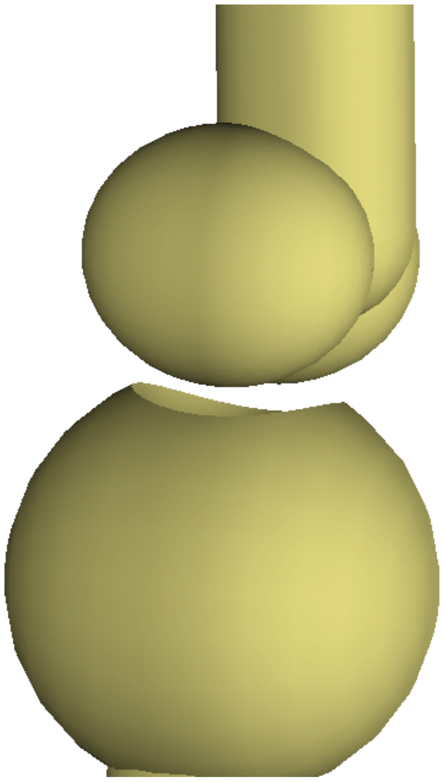
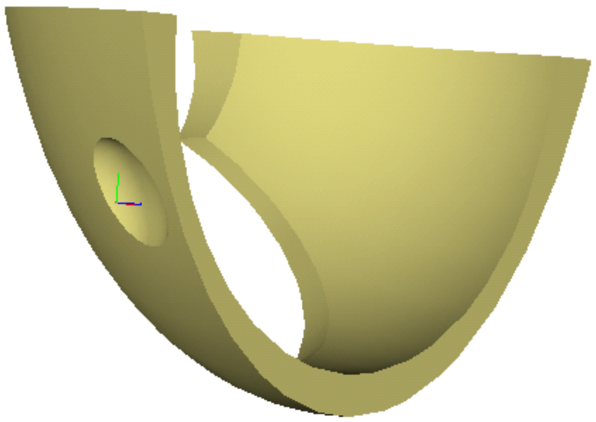
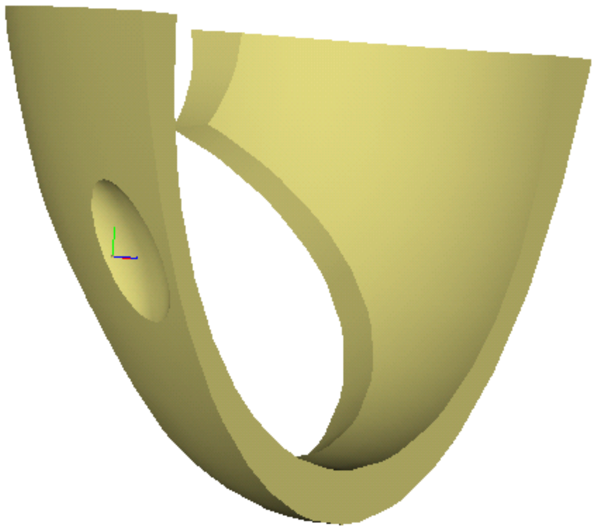


Figure 5.15: Adjustments of the knee during runtime. The condyle of the tibia can be adjusted to match that of the femur condyles.



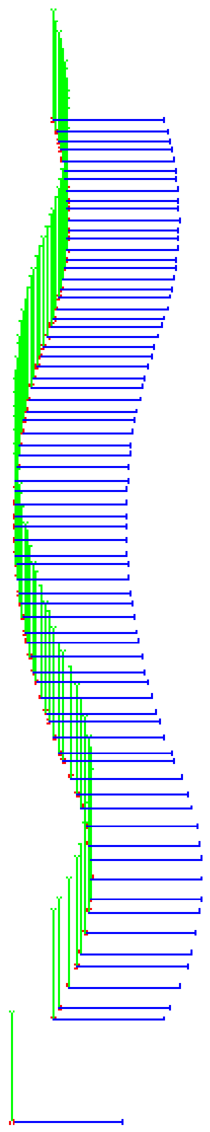
(a) Pelvis bone model. Height: 14 cm

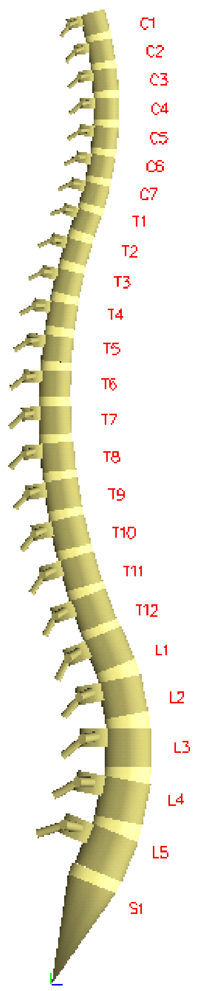


(b) Pelvis bone model. Height: 18 cm

Figure 5.16: Pelvis geometries with different heights. The height where adjusted during runtime.

The Spine Point Generator was developed to shorten development time of the geometry for vertebrae and intervertebral discs. Points generated by the Spine Point Generator is illustrated in Figure 5.17a. The *points.txt*-file is listed in Appendix B with title "points.txt-file for a spine". The geometry created from the input is shown in Figure 5.17b. The labels corresponds to the bone-names defined in the input file *bones.txt*-file.


(a) Illustration of the points that define the bone and cartilage geometries in Figure 5.17b.



(b) Model of the spine and sacrum. The bright yellow geometries represent intervertebral discs.

Figure 5.17: Model of the spine. The *points.txt*-file is made in Spine Point Generator.

A model comprised of the femur, tibia, pelvis, sacrum and spine is shown in Figure 5.18. The rest of the body is not yet defined in the program. The left and right femur are instances of the same class definition. The same principle applies to the tibia. To make the joints fit together, the knee and hip joint were adjusted during runtime. The tibia condyle was adjusted to fit the femur condyles. The femur head radius was decreased to fit the hip bowl of the pelvis.

The Spine Point Generator generated the points for the spine in this model. To make the spine originate from the pelvis, a point that was offset 5 cm posterior and 1 cm above the center of the pelvis was set as the start point in the Spine Point Generator.

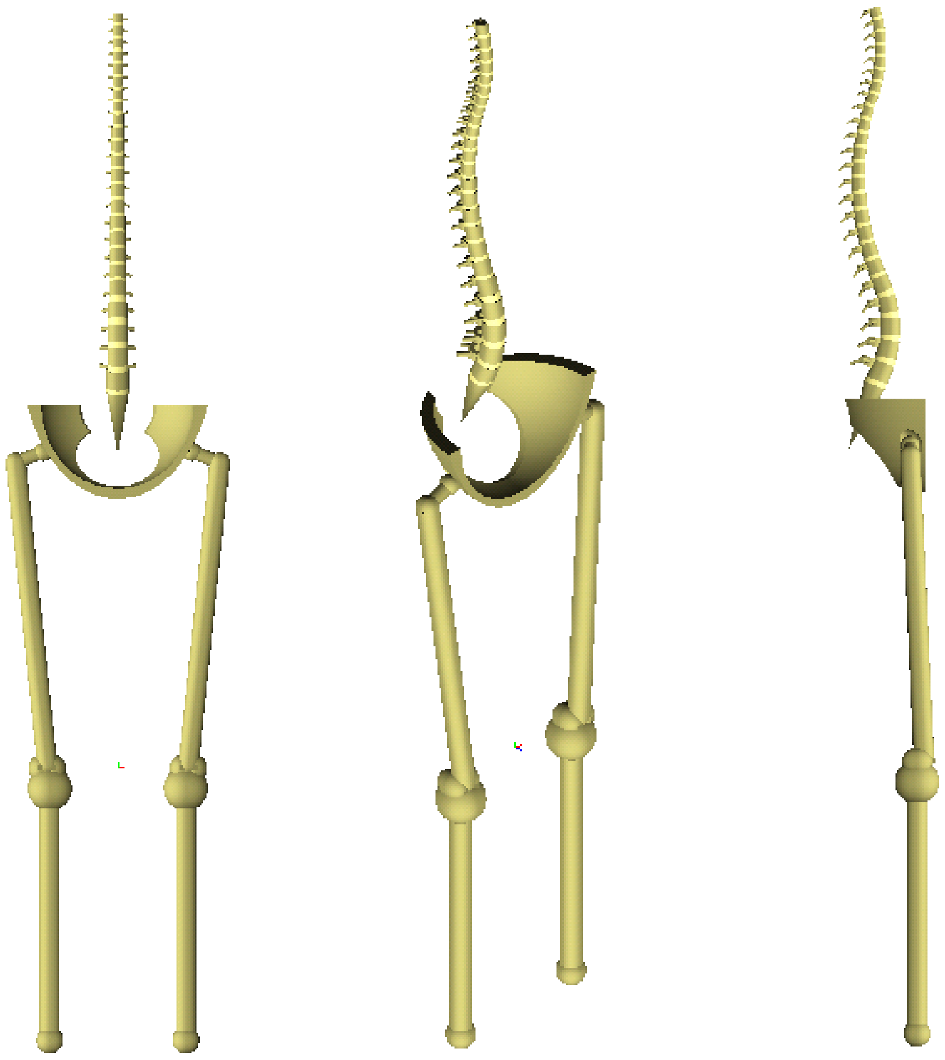

(a) Posterior view (b) Isometric view (c) Lateral view

Figure 5.18: A model comprised of the femur, tibia, pelvis, sacrum and spine. The knee joint was adjusted during runtime to fit the femur condyle with the tibia condyle. The femur head was adjusted to fit the hip bowl on the pelvis.

# Chapter 6

# Discussion

## 6.1 Segmentation Tool

The objective of the desired software was to automatically segment medical images and create input, either as 3D objects or as text input, to a KBE modeling environment. As of today, the segmentation tool is able to segment bones from CT-images and visualize the bones as isosurfaces within Matlab. The segmentation is however, not automatic, and manual input of a lower threshold value is required. The output of the program is pure visual and can not be directly exported to the Musculoskeletal Program for further manipulation. During development, a function was found online (see URL at Sandberg) that exports to .obj, which is a 3D file format. However, this format is not supported by AML and further investigation was put on hold. Nevertheless, this function can be a starting point for development of a new function that exports to either IGES, STEP, DXF or STL. These formats are supported by AML.

In the CT-images that were tested, bone marrow had lower pixel value than cortical (hard) bone. The result is that bone marrow is invisible after segmentation. If the result is to be used to generate a surface geometry, this is not a problem. However, for generating a solid geometry, the difference between mechanical properties in cortical bone and bone marrow will influence the results.

There were several types of disturbances in the results. In the segmented torso, an anomaly on the edge of the scan-area was observed. The humerus can be seen exiting the scan-area in the proximity of the anomaly and the anomaly was analysed to be the soft tissue of the overarm causing an increase in signal strength at the scan-area border.

Another kind of disturbance was observed in the head segmentation, but an analysis of what might be causing it has not been successful. The last type of disturbance is other body parts entering the scan-volume. This was observed in the pelvis and hip segmentation. If the desired result was geometry of the lower body, the finger tips would have to be removed.

## 6.2 Musculoskeletal Program

The Musculoskeletal Program can generate complex geometry that is based on points in 3D space. A single class definition can be used to generate infinitely results by changing the points that is used as input.

The presented bone-objects has similar topology to that of their human counterpart, but the geometry of their structures can be enhanced. However, for simulation, this level of accuracy might be preferable as simulation time increases with complexity.

The tibia, femur and pelvis can be modified during runtime to fit better together. In the properties of the tibia, the condyle can be adjusted to fit the femur condyle counterpart. The femur head radius can also be adjusted to fit the acetabulum (hip bowl).

The Musculoskeletal Program is lacking support for muscles, tendons and ligaments, neither as geometry nor as representation in another form. Simulation is dependent on input that have relations between the geometric entities. These relations are muscles, tendons and ligaments that enables motion and prevents joints from disassembling.

The program has two interfaces for input: Either by writing text files that is used to generate geometry or refining of the geometry during runtime. The objective of the final program was to generate geometry that did not need further refining. Therefore, the adjustments needed, such as adjusting femur head radius and pelvis height, which were shown in Section 5.2, should be moved from runtime adjustments to text file input.

# Chapter 7

# Conclusion

The objective of this master's thesis was to investigate available software programs for use in segmentation of medical images, and use the results as input to generate geometry for simulation input.

In this work, OsiriX was found useful as a segmentation program. However, by exploiting Matlab's Image Processing Toolbox, a program was developed to segment medical images. This program can be further developed to automatically perform segmentations.

Technosoft's AML was chosen as the framework to develop a program that generates 3D geometry. This was a natural choice of KBE framework due to experience in the Department of Engineering Design and Materials. AML has also been successfully used by Aker Solutions to generate rigs and tankers Steensen [2008].

As discussed, there are still improvements that needs to be made, and therefore there is a potential for further development. However, Matlab has proven to be a useful tool when manipulating DICOM-files. The Musculoskeletal Program, developed in AML, also shows the possibilities of the platform. With more resources the possibilities could be further explored and exploited.

Although, the segmentation results show anomalies, these are not caused by deviations due to malfunction, but can be traced back to the way the images were captured.

The conclusion is that the developed programs are platforms for further research and development, and the thesis has proven Matlab and AML as viable tools for continued work.

# Chapter 8

# Further work

## 8.1 Segmentation Tool

As discussed, a method for exporting a 3D object can be implemented in the Segmentation Tool. Sandberg has developed a method for exporting .obj, and either by modification or further conversion, this might lead to a method for exporting to IGES, STEP, DXF or STL.

To prevent bone marrow disappearing when bone is segmented, an alternative method for segmenting must be implemented. As discussed, this is not necessary for surface geometry.

For creating geometry of other tissues, MRI with other modulation might be used. With P2 and PD modulation, soft tissue are represented with a brighter color than bone.

## 8.2 Musculoskeletal Program

As of today, the program only supports input of femur, tibia, pelvis, vertebrae and intervertebral discs. This limits the domain of the program to visualization, and is therefore not sufficient for simulation input. For further work, the geometry classes have to be supplemented with geometry rules for the rest of the human bones and cartilage. Classes and methods for muscle, tendons and ligaments must also be implemented.

To meet the objective of a program that automatically generates geometry, all inputs have to be given as text files; not by refining during runtime. It is recommended that the segmentation program is developed further so that it can generate input files for the Musculoskeletal Program. In this way, there will be no need for refinement in the program and both systems can be tailor made for each other.

To increase the detail level in the geometry generated by the program, additional points could be exploited and a shape file could be implemented. The function of a shape file is to make detailed adjustments to the geometry. By adding such functionality, geometric accuracy could be increased to enhance the simulation accuracy, or it could be decreased to reduce the computation time.

AML supports *nurbs* and *splines*, which could be useful when generating more detailed geometry. Skaare [2015] uses these classes, which are used for generating cylinders with bends and other ”organic” forms.

# Bibliography

Aireurbano. Medial bone of the leg. URL http://www.aireurbano.com/medial-bone-of-the-leg/. Last visited 2016-01-16.

BrainLab. Divisions of the skeletal system. URL http://sabre.brainlab.ca/docs/processing/stage7.html. Last visited 2016-01-18.

CandelaOpenCourses. Divisions of the skeletal system. URL https://courses.candelalearning.com/ap2x1/chapter/divisions-of-the-skeletal-system/. Last visited 2016-01-18.

Anderson F.C. Arnold A.S. Loan P. Habib A. John C.T. Guendelman E. Delp, S.L. and Darryl G. Thelen. Opensim: Open-source software to create and analyze dynamic simulations of movement. *11*, 54:1940–1946, 2007.

DICOM. About the dicom format. URL http://dicom.nema.org/Dicom/about-DICOM.html. Last visited 2016-01-10.

Vogl W. Mitchell A. Drake, R. and H. Gray. *Gray's anatomy for students*. Philadelphia: Elsevier/Churchill Livingstone, 2005.

FEBio. Febio software suite, n.d. URL http://febio.org. Last visited 2016-01-16.

GetBodySmart.com. Anterior femur bone. URL http://www.getbodysmart.com/ap/skeletalsystem/skeleton/appendicular/lowerlimbs/femur1/print4.html. Last visited 2016-01-16.

Carl Otto Gjelsvik. Modeling and simulation of the human body. 2015.

Rawlins D. Weiss Dr.J. Maas, S. and Dr.G. Ateshian. Febio - finite elements for biomechanics, theory manual. pages 1–7, 2015.

MultiSim. Multisim explained, n.d. URL http://multisim-insigneo.org/multisim-explained/. Last visited 2016-01-16.

NEMA. *DICOM PS3.1 2015c - Information Object Definitions*. NEMA, 2015.

Donald A. Neumann. *Kinesiology of The Musculoskeletal System*. Mosby/Elsevier, 2010.

Neurologica. Ceretom. URL http://www.neurologica.com/products/ceretom. Last visited 2016-01-18.

NMSBuilder. Nms builder beta release, n.d. URL https://www.biomedtown.org/nmsphysiome/reception/nmsbuilder_beta/. Last visited 2016-01-16.

NMSPhysiome. Personalised models of the neuromusculoskeletal system, n.d. URL http://www.nmsphysiome.eu/index_html. Last visited 2016-01-16.

Osteoporosis PT. Posture & bone health. URL http://www.osteoporosispt.com/Posture---Bone-Health.html. Last visited 2016-01-16.

Anders Sandberg. Matlab and bryce. URL http://www.aleph.se/Nada/Ray/matlabobj.html#2. Last visited 2016-01-10.

Osteopathy Singapore. Anatomy of the hip. URL http://singaporeosteopathy.com/category/hip/. Last visited 2016-01-16.

Rasmus Korvald Skaare. Mechanism parametrization, modeling and fe-meshing. 2015.

Anders J. Steensen. Semi på en dag. 2008.

Tore Stensvold. Tegner nye offshorekonstruksjoner på rekordtid. URL http://www.tu.no/petroleum/2013/01/09/tegner-nye-offshorekonstruksjoner-pa-rekordtid. Last visited 2016-01-15.

TechnoSoft. *AML 5.0B5 Reference Manual*. TechnoSoft Inc., 2010.

Jan S.V.S. Garny A. Brook B.S. Viceconti, M. and S.L. Waters. Seeding the europhysiome: A roadmap to the virtual physiological human. 6-41, 2007.

VPHInstitute. What is the vph institute, n.d. URL http://www.vph-institute.org/what-is-vph-institute.html. Last visited 2016-01-16.

Appendix A

# Mail converation with Marius Widerøe

The following mail dialog is in norwegian. The purple text is the part where questions are answered.

Hei,

Beklager at det tok alt for lang tid å svare. Ble syk og du havnet litt langt ned i mailbunken.
Jeg har lagt inn svar under hvert spørsmål under.
Jeg kopierer inn Toril Sjøbak i denne mailen. Hvis dere ønsker å gå videre med MR så kan hun kanskje hjelpe dere med det praktiske. Det er mulig hun også har noen eksempler dere kan se.

Marius Widerøe
MD, PhD
Associate Professor
Manager of MR Core Facility

MR-Center
Department of Circulation and Medical Imaging
Norwegian University of Science and Technology
Postboks 8905, N-7491 Trondheim, Norway

Phone: +47 73 55 13 54 Mobile: +47 40 23 19 23
e-mail: marius.wideroe@ntnu.no

---

Fra: Carl Otto Gjelsvik
Sendt: 21. september 2015 12:47
Til: Marius Widerøe
Emne: VS: Søker informasjon om MR til masteroppgave

Hei!

Har du fått sett på spørsmålene?

--

Med vennlig hilsen

Carl Otto Gjelsvik

---

Fra: Carl Otto Gjelsvik
Sendt: 9. september 2015 11:31
Til: Marius Widerøe
Emne: SV: Søker informasjon om MR til masteroppgave

Hei,

Takk for raskt svar!

For å utføre analysen er vi interessert i å ta utgangspunkt i en detaljert modell av skjelett, muskler og muskelfester.

1. Er MR riktig teknologi for å oppnå dette?
Den kan være et godt utgangspunkt for det. Røntgen og CT er bedre på skjelett enn MR, men MR er best på bløtvev.

2. Er det mulig å ta en full kropp-scan med MR og eventuelt hvor lang tid tar det
Det er mulig å skanne en hel kropp, men ikke på enn gang. Man må skanne endel av gangen. Litt avhengig av hva dere er ute etter så vil det også være en fordel å bruke forskjellige innstillinger for forskjellige deler av kroppen for å optimalisere mengden informasjon dere får fra bildene.
Hvor lang tid det tar avhenger av hvor detaljerte bilder dere trenger, hva slags informasjon dere er ute etter i bildene. Jeg vil tippe at vi fort snakker om en time eller to.

3. Hva slags bilde-oppløsning er mulig?
Det kommer an på hvilken kroppsdel det skal tas bilde av, hva slags informasjon dere vil ha og ikke minst tidsbruk. Høyere oppløsning tar lengre tid. Normale bilder har oppløsning på 1x1mm med snittykkelse på, ca 5mm. Vi kan komme ned i 1x1x1mm men det krever mye tid og er ikke mulig alle deler av kroppen.

For å få et nøyaktig utgangspunkt for simulasjonene tenker vi at det hadde vært optimalt hvis scannen ble gjort stående.

4. Er stående MR mulig?
nei. Den som skannes må ligge

5. Hvor forskjellig er posisjonen på et skjelett som ligger fra et som står?
Det vil være noe forskjell i hvordan musklene "henger" på kroppen - du kan tenke på hvordan tyngdekraften virker på musklene i stående versus liggende stilling. Det vil også være litt forskjell på hvordan leddene er belastet, spes i beina og i ryggen. Det trenger imidlertid ikke ha så mye å si for simuleringen vi jeg tro.

For å utføre analyser og simuleringer på dataen må vi først oversette den til noe vi kan jobbe med.

6. Hvordan er prosessen fra scan til dataskjerm? Kan man velge hva slags vev man er interessert i og utelukke andre ting?
Bildene lagres normalt i et dataformat som heter dicom. Fra det kan det lages 3D rekonstruksjoner. Det finnes mye software for bildeanalyse i etterkant, men jeg vet ikke om noe software for automatisk gjenkjenning av muskler og skjelett, men det finnes sikker tilgjenglige algoritmer for dette. Det går på segmentering av bildene. For optimal automatisk segmentering er det viktig at bildene er tatt opp med inntillinger

som gjør at det blir mest mulig kontrast mellom de vevene dere ønsker å skille fra hverandre.

7. Vet du hva slags format man får dataen på? Er det eventuelt mulig å få ut rådata i tillegg?
Se svar over. Det er mulig å få ut rådata også men da kreves det rekonstruksjon av disse. For ditt formål så vil jeg nok betrakte dicom bildene som råe nok..
8. Hva slags informasjon har man tilgang på fra en MR-maskin? Kan man se tettheten i vevet?
Signal og kontrast i bildene gjenspeiler i hovedsak tettheten av protoner samt det kjemiske miljøet protonene befinner seg i. Det er ikke mulig å måle tetthet i vevet direkte, men man kan få noe indirekte mål på celletetthet. Ved å justere på innstillingene ved opptaket av bildene kan måan optimalisere kontrasten mellom vev med ulike egenskaper.
Det er også mulig å få ut annen informasjon som vanndiffusjon og perfusjon i vevet samt molekylær (hovedsaklig metabolitter) informasjon,  men da men mye  lavere oppløsning.

Takk!

Med vennlig hilsen

Carl Otto Gjelsvik

---

Fra: Marius Widerøe <marius.wideroe@ntnu.no>
Sendt: 9. september 2015 10:15
Til: Carl Otto Gjelsvik
Emne: Sv: Søker informasjon om MR til masteroppgave

Hei,

Jeg kan godt svare på noen spørsmål, men jeg har ikke tid til å møtes før i neste uke. Hvis du beskriver litt mer hva du er ute etter og evt de spørsmål du har per mail så kan jeg gi deg raskt svar på det. Så kan vi evt møtes senere for mer utdyping.

mvh

Marius Widerøe
MD, PhD
Associate Professor
Manager of MR Core Facility

MR-Center
Department of Circulation and Medical Imaging
Norwegian University of Science and Technology
Postboks 8905, N-7491 Trondheim, Norway

Phone: +47 73 55 13 54 Mobile: +47 40 23 19 23
e-mail: marius.wideroe@ntnu.no

---

Fra: Carl Otto Gjelsvik
Sendt: 9. september 2015 10:07
Til: Marius Widerøe
Emne: Søker informasjon om MR til masteroppgave

Hei!

Mitt navn er Carl Otto Gjelsvik og jeg skriver masteroppgave ved Institutt for produktutvikling og materialer i høst.

Prosjektet har som mål å finne en metode for å utføre analyser og simuleringer på en vilkårlig kropp.

En av oppgavene er å foreslå en metode for å scanne en kropp og lage data til bruk i simulasjonen.

Jeg har en del spørsmål rundt MR-prosedyren og hva slags data man får ut.

Har du tid og mulighet til å svare på noen spørsmål, eller eventuelt videresende til noen som kan hjelpe meg?

Med vennlig hilsen

Carl Otto Gjelsvik

# Appendix B

# Musculoskeletal Program - Files and Input

## B.1 Source code

Listing B.1: system.def

```
;;;--------------------------------------------------------
;;; System: Musculoskeletal-system
;;; Purpose: Generate geometry of the human body from input
;;;
;;; Author : Carl Otto Gjelsvik
;;;--------------------------------------------------------

(in-package :AML)

(defvar #MODEL-LIBRARY# "")
(setf #MODEL-LIBRARY#
  (logical-path :mp "models")
  )

(define-system :mp
  :files '(
    "bone.aml"
    "cartilage.aml"
    "femur.aml"
    "pelvis.aml"
    "sacrum.aml"
    "tibia.aml"
    "intervertebral-disc.aml"
    "vertebrae.aml"
    "geometry.aml"
    "draw.aml"
    "io.aml"
          )
  )
```

Listing B.2: bone.aml

```
(define-class bone-object
  :inherit-from (coordinate-system-class)
  :properties(
    part-topology-list nil
    color 'khaki4
    inner-bone-density nil
    outer-bone-density nil
    average-stiffness nil
    )
  )

(define-class default-bone-object
  :inherit-from (bone-object cylinder-object)
  :properties (
    part-topology-list nil
    start-point (nth
      (nth 0 ^part-topology-list) ^points-list)
    end-point (nth
      (nth 1 ^part-topology-list) ^points-list)
    height (vector-length
      (subtract-vectors ^end-point ^start-point)
      )
    diameter (* 3 ^height)
    orientation (list (align
      (center-of-face (the superior) 0)
      (normal-to-face (the superior) 0)
      nil
      (^start-point)
      (subtract-vectors ^end-point ^start-point)
      nil
      :move? t :align? t :orient? nil
      )
    )
    reference-coordinate-system
    (the superior superior self)
    )
    )
  )

(define-class cylinder-bone
  :inherit-from
  (bone-object cylinder-object coordinate-system-class)
  :properties(
    draw-axes? nil
    )
  )

(define-class ellipsoid-bone
  :inherit-from
  (bone-object ellipsoid-object coordinate-system-class)
  :properties(
    draw-axes? nil
    )
  )
(define-class sphere-bone
  :inherit-from
  (bone-object sphere-object coordinate-system-class)
  :properties(
    draw-axes? nil
    )
  )
```

Listing B.3: cartilage.aml

```
(define-class cartilage-object
  :inherit-from (object)
  :properties(
    part-topology-list nil
    color 'khaki3
    density nil
    stiffness nil
    )
  )

(define-class default-cartilage-object
  :inherit-from (cartilage-object cylinder-object)
  :properties(
    part-topology-list nil
    start-point (nth
      (nth 0 ^part-topology-list)
      ^points-list
      )
    end-point (nth
      (nth 1 ^part-topology-list)
      ^points-list
      )
    height (vector-length
      (subtract-vectors ^end-point ^start-point)
      )
    diameter (* 3 ^height)
    orientation (list (align
      (center-of-face (the superior) 0)
      (normal-to-face (the superior) 0)
      nil
      (^start-point)
      (subtract-vectors ^end-point ^start-point)
      nil
      :move? t :align? t :orient? nil
      )
    )
    reference-coordinate-system
    (the superior superior self)
    )
  )
```

Listing B.4: femur.aml

```
(define-class femur-object
  :inherit-from (bone-object)
  :properties(
    property-objects-list
    (list
      "Femur Properties"
      (the superior femur-head-radius self)
      (the superior femur-neck-radius self)
      (the superior femur-body-radius self)
      (the superior femur-condyle-length self)
      (the superior femur-condyle-height self)
      (the superior femur-condyle-width self)
      )
    (femur-head-radius :class
      'editable-data-property-class
      label "Femur Head Radius"
      formula (default 1.4)
    )
    (femur-neck-radius :class
      'editable-data-property-class
      label "Femur Neck Radius"
      formula (default 1)
    )
    (femur-body-radius :class
      'editable-data-property-class
      label "Femur Body Radius"
      formula (default 1.5)
    )
    (femur-condyle-length :class
      'editable-data-property-class
      label "Condyle Length"
      formula (default 4.5)
    )
    (femur-condyle-height :class
      'editable-data-property-class
      label "Condyle Height"
      formula (default 4)
    )
    (femur-condyle-width :class
      'editable-data-property-class
      label "Condyle Width"
      formula (default 3.5)
    )
    femur-condyle1-position
    (nth (nth 0 ^part-topology-list) ^points-list)
    femur-condyle2-position
    (nth (nth 1 ^part-topology-list) ^points-list)
    femur-body-bottom
    (nth (nth 2 ^part-topology-list) ^points-list)
    knuckle-position
    (nth (nth 3 ^part-topology-list) ^points-list)
    femur-head-position
    (nth (nth 4 ^part-topology-list) ^points-list)
    )
  :subobjects(
    (femur-neck :class 'cylinder-bone
      height (vector-length
        (subtract-vectors
          ^^femur-head-position
          ^^knuckle-position))
      diameter (* 2 ^^femur-neck-radius)
      orientation (list (align
        (center-of-face (the superior) 0)
        (normal-to-face (the superior) 0)
        nil
        (^knuckle-position)
        (subtract-vectors
          ^femur-head-position
          ^knuckle-position)
        nil
        :move? t :align? t :orient? t
        )
      )
      reference-coordinate-system
      (the superior superior self)
      )
    (femur-body :class 'cylinder-bone
      height (vector-length
        (subtract-vectors
          ^^knuckle-position
          ^^femur-body-bottom))
      diameter (* 2 ^^femur-body-radius)
      orientation (list (align
        (center-of-face (the superior) 0)
        (normal-to-face (the superior) 0)
        nil
        (^femur-body-bottom)
        (subtract-vectors
          ^knuckle-position
          ^femur-body-bottom)
        nil
        :move? t :align? t :orient? t
        )
      )
      reference-coordinate-system
      (the superior superior self)
      )
    (condyle1 :class 'ellipsoid-bone
      x-diameter ^^femur-condyle-width
      y-diameter ^^femur-condyle-height
      z-diameter ^^femur-condyle-length
      orientation (list (translate
        ^^femur-condyle1-position))
      )
    (condyle2 :class 'ellipsoid-bone
      x-diameter ^^femur-condyle-width
      y-diameter ^^femur-condyle-height
      z-diameter ^^femur-condyle-length
      orientation (list (translate
        ^^femur-condyle2-position))
      )
    (femur-head :class 'sphere-bone
      diameter (* 2 ^^femur-head-radius)
      orientation (list (translate
        ^^femur-head-position))
      )
    (bend :class 'sphere-bone
      diameter (* 2 ^^femur-body-radius)
      orientation (list (translate
        ^^knuckle-position))
      )
    (bottom :class 'sphere-bone
      diameter (* 2 ^^femur-body-radius)
      orientation (list (translate
        ^^femur-body-bottom))
      )
    )
  )
```

Listing B.5: pelvis.aml

```
(define-class pelvis-object
  :inherit-from (bone-object union-object)
  :properties(
    property-objects-list
    (list
      "Pelvis Properties"
      (the superior pelvis-height self)
      (the superior pelvis-depth self)
      )
    (pelvis-height :class 'editable-data-property-class
      label "Height"
      formula (default ^size)
      )
    (pelvis-depth :class 'editable-data-property-class
      label "Depth"
      formula (default ^size)
      )
    left-femur (nth
      (nth 0 ^part-topology-list) ^points-list)
    right-femur (nth
      (nth 1 ^part-topology-list) ^points-list)
    size (* 0.7 (vector-length
      (subtract-vectors
        ^left-femur
        ^right-femur)))
    pelvis-width (* 1.95 ^size)
    object-list (list (the ))
    )
  :subobjects (
    (difference-object1 :class difference-object
      display? nil
      object-list (list
        ^^box-object1
        ^^ellipsoid-object1))
    (box-object1 :class box-object
      display? nil
      width ^^pelvis-width
      height ^^pelvis-height
      depth ^^pelvis-depth
      )
    (ellipsoid-object1 :class ellipsoid-object
      display? nil
      x-diameter (* ^^pelvis-width 0.75)
      y-diameter (* ^^pelvis-height 1.6)
      z-diameter (* ^^pelvis-depth 1.5)
      orientation (list
        (translate
          (list 0
            (* 0.5
              ^^pelvis-height)
            (* 0.25
              ^^pelvis-width))))
      )
    (intersection-object1 :class intersection-object
      display? nil
      object-list (list
        ^^difference-object1
        ^^ellipsoid-object2)
      )
    (ellipsoid-object2 :class ellipsoid-object
      display? nil
      x-diameter (* ^^pelvis-width 0.95)
      y-diameter (* ^^pelvis-height 2.8)
      z-diameter (* ^^pelvis-depth 1.8)
      orientation (list
        (translate
          (list 0
            (* 1 ^^pelvis-height)
            (* 0.5 ^^pelvis-depth))))
      )
    (cylinder-object1 :class cylinder-object
      display? nil
      diameter (* 0.8 ^^pelvis-height)
      height (* 2 ^^pelvis-depth)
      orientation (list
        (translate
          (list 0 (* 0.1 ^^pelvis-height) 0))
        (rotate -30 :x-axis)
        (translate
          (list 0 0 (* 0.1 ^^pelvis-width))))
      )
    (difference-object2 :class difference-object
      shade? 't
      render 'shaded
      object-list (list
        ^^intersection-object1
        ^^cylinder-object1
        ^^truncated-cone1
        ^^ellipsoid-object3
        ^^ellipsoid-object4)
      orientation (list
        (align
          (center-of-object ^^ellipsoid-object3)
          '(0 1 0)
          '(0 1 0)
          (^^left-femur)
          '(0 1 0)
          '(0 1 0)
          :move? t :align? nil :orient? nil)
        )
      )
    (truncated-cone1 :class truncated-cone-object
      display? nil
```

```
      start-diameter (* 0.6 ^^size)
      end-diameter (* 0.8 ^^size)
      height (* 0.4 ^^size)
      orientation (list
        (translate
          (list 0
            (* 0.5 ^^pelvis-height)
            (* -0.25 ^^pelvis-depth)
            )
          )
        )
      )
      (ellipsoid-object3 :class ellipsoid-object
        display? nil
        x-diameter (* ^^pelvis-width 0.105)
        y-diameter (* ^^pelvis-height 0.33)
        z-diameter (* ^^pelvis-depth 0.24)
        orientation (list
          (rotate -18 :x-axis)
          (rotate 25 :z-axis)
          (translate
            (list
              (* -0.365 ^^pelvis-width)
              (* 0.05 ^^pelvis-height)
              (* 0.35 ^^pelvis-depth))))
        )
      (ellipsoid-object4 :class ellipsoid-object
        display? nil
        x-diameter (* ^^pelvis-width 0.105)
        y-diameter (* ^^pelvis-height 0.33)
        z-diameter (* ^^pelvis-depth 0.24)
        orientation (list
          (rotate -18 :x-axis)
          (rotate -25 :z-axis)
          (translate
            (list
              (* 0.365 ^^pelvis-width)
              (* 0.05 ^^pelvis-height)
              (* 0.35 ^^pelvis-depth))))
        )
      )
)
```

Listing B.6: sacrum.aml

```
(define-class sacrum-object
  :inherit-from (cone-object bone-object)
  :properties (
    part-topology-list nil
    orientation (list (align
      (center-of-face (the superior) 1)
      (normal-to-face (the superior) 1)
      nil
      (^point1)
      (subtract-vectors ^point2 ^point1)
      nil
      :move? t :align? t :orient? t
      )
    )
    color 'khaki4
    reference-coordinate-system
    (the superior superior self)
    point1 (nth
      (nth 0 ^part-topology-list) ^points-list)
    point2 (nth
      (nth 1 ^part-topology-list) ^points-list)
    point3 (nth
      (nth 2 ^part-topology-list) ^points-list)
    point4 (nth
      (nth 3 ^part-topology-list) ^points-list)
    height (vector-length
      (subtract-vectors ^point2 ^point1))
    height-t5 (vector-length
      (subtract-vectors ^point4 ^point3))
    diameter (* 1.2 ^height-t5)
    )
  )
```

Listing B.7: tibia.aml

```
(define-class tibia-object
  :inherit-from (bone-object)
  :properties(
    property-objects-list
    (list
      "Tibia Properties"
      (the superior tibia-body-radius self)
      (the superior femur-condyle-length self)
      (the superior femur-condyle-height self)
      (the superior femur-condyle-width self)
      (the superior tibia-bottom-radius self)
      )
    (tibia-bottom-radius :class
      'editable-data-property-class
      label "Tibia Bottom Radius"
      formula (default 4)
    )
    (tibia-body-radius :class
      'editable-data-property-class
      label "Tibia Body Radius"
      formula (default 1.5)
    )
    (femur-condyle-length :class
      'editable-data-property-class
      label "Femur Condyle Length"
      formula (default 6)
    )
    (femur-condyle-height :class
      'editable-data-property-class
      label "Femur Condyle Height"
      formula (default 5)
    )
    (femur-condyle-width :class
      'editable-data-property-class
      label "Femur Condyle Width"
      formula (default 6)
    )
    tibia-bottom-position (nth
      (nth 0 ^part-topology-list) ^points-list)
    tibia-condyle-position (nth
      (nth 1 ^part-topology-list) ^points-list)
    )
   :subobjects(
    (tibia-body :class 'cylinder-object
      height (vector-length
        (subtract-vectors
          ^^tibia-condyle-position
          ^^tibia-bottom-position))
      diameter (* 2 ^^tibia-body-radius)
      orientation (list (align
        (center-of-face (the superior) 0)
        (normal-to-face (the superior) 0)
        nil
        (^^tibia-bottom-position)
        (subtract-vectors
          ^^tibia-condyle-position
          ^^tibia-bottom-position)
        nil
        :move? t :align? t :orient? t
        )
      )
      reference-coordinate-system
      (the superior superior self)
      )
    (condyle :class 'difference-object
      object-list (list ^con ^cutcon))
    (con :class 'sphere-object
      display? nil
      diameter (* 1.1 ^^femur-condyle-width)
      orientation (list
        (translate ^^tibia-condyle-position))
      )
    (cutcon :class 'ellipsoid-object
      display? nil
      x-diameter ^^femur-condyle-width
      y-diameter ^^femur-condyle-height
      z-diameter ^^femur-condyle-length
      orientation (list (translate
        ^^tibia-condyle-position)
        (translate
          (list 0
           (/ (the diameter
             (:from (the con))) 1.5) 0)))
     )
    (bottom :class 'sphere-object
      diameter ^^tibia-bottom-radius
      orientation (list
        (translate ^^tibia-bottom-position))
      )
    )
  )
```

Listing B.8: vertebrae.aml

```
;;;---------------------------------------------------------
;;; System: Musculoskeletal-system
;;; Purpose: Class-definitions of vertebrae
;;;
;;; Author : Carl Otto Gjelsvik
;;;---------------------------------------------------------

(define-class vertebrae-cylinder
  :inherit-from (bone-object cylinder-object)
  :properties (
    display? nil
    )
  )
; The following two classes defines the
; geometry for each spinular bone.
(define-class spinal-cord-section
  :inherit-from (difference-object)
  :properties (
    inner-diameter 1
    outer-diameter 2.5
    height 1.0
    display? nil
    object-list (list (the bone) (the hole))
    )
  :subobjects (
    (bone :class 'vertebrae-cylinder
      height ^^height
      diameter ^^outer-diameter
      solid? t
      display? nil
      orientation (list (rotate 0.0 :x-axis))
      )
    (hole :class 'vertebrae-cylinder
      height ^^height
      diameter ^^inner-diameter
      solid? t
      display? nil
      orientation (list (rotate 0.0 :x-axis))
      )
    )
  )

(define-class vertebrae-object
  :inherit-from (union-object)
  :properties(
    point1 (nth (nth 0 ^part-topology-list) ^points-list)
    point2 (nth (nth 1 ^part-topology-list) ^points-list)
    orientation (list (align
      (center-of-face (the superior) 0)
      (normal-to-face (the superior) 1)
      nil
      (^point1)
      (subtract-vectors ^point2 ^point1)
      nil
      :move? t :align? t :orient? t
      )
    )
    reference-coordinate-system (the superior superior self)
    height (vector-length (subtract-vectors ^point2 ^point1))
    diameter (* 1.2 ^height)
    spinal-cord-section-height (/ (^height) 2)
    spinal-cord-section-outer-diameter (* (/ 5 6) (^height))
    spinal-cord-section-inner-diameter
    (/ (^spinal-cord-section-outer-diameter) 2)
    spinal-trans-x 0.0
    spinal-trans-y (* 0.25 ^height)
    spinal-trans-z (* -0.80 ^height)
    articular-process-height (* 0.7 ^height)
    articular-process-diameter (* 0.25 ^height)
    articular-trans-x (* 0.65 ^height)
    articular-trans-y (* 0.25 ^height)
    articular-trans-z (* -1.0 ^height)
    spinious-process-height (* 0.7 ^height)
    spinious-process-diameter (* 0.2 ^height)
    spinious-trans-x (* 0.0 ^height)
    spinious-trans-y (* 0.0 ^height)
    spinious-trans-z (* -1.4 ^height)
    display? t
    color 'khaki4
    render 'shaded
    object-list (list (the front-section) (the cord-section)
      (the left-articular-process)
      (the right-articular-process) (the spinious-process)
      )
    )
:subobjects (
  (front-section :class 'vertebrae-cylinder
    height ^^height
    diameter ^^diameter
    orientation (list (rotate 90.0 :x-axis))
    display? nil
    )
  (cord-section :class 'spinal-cord-section
    inner-diameter ^^spinal-cord-section-inner-diameter
    outer-diameter ^^spinal-cord-section-outer-diameter
    height ^^spinal-cord-section-height
    orientation (list
      (rotate 90.0 :x-axis)
      (translate (list
        (^^spinal-trans-x)
        (^^spinal-trans-y)
        (^^spinal-trans-z))
      )
      )
    display? nil
    )
  (left-articular-process :class cylinder-object
    height ^^articular-process-height
    diameter ^^articular-process-diameter
    orientation (list
     (rotate -5 :x-axis)
     (rotate 60 :y-axis)
     (translate (list (* -1
      (^^articular-trans-x))
      (^^articular-trans-y)
      (^^articular-trans-z)))
     )
    display? nil
    )
  (right-articular-process :class cylinder-object
    height ^^articular-process-height
    diameter ^^articular-process-diameter
    orientation (list
     (rotate -5 :x-axis)
     (rotate -60 :y-axis)
     (translate (list
      (^^articular-trans-x)
      (^^articular-trans-y)
      (^^articular-trans-z)))
     )
    display? nil
    )
  (spinious-process :class cylinder-object
    height ^^spinious-process-height
    diameter ^^spinious-process-diameter
    orientation (list
     (rotate -40 :x-axis)
     (translate (list
      (^^spinious-trans-x)
      (^^spinious-trans-y)
      (^^spinious-trans-z)))
     )
    display? nil
    )
  )
)
```

Listing B.9: intervertebral-disc.aml

```
;;;---------------------------------------------------------
;;; System: Musculoskeletal-system
;;; Purpose: Class-definitions of intervertebral discs
;;;
;;; Author : Carl Otto Gjelsvik
;;;---------------------------------------------------------

; Lumbar and cervical section
(define-class intervertebral-disc-object
  :inherit-from (cartilage-object elbow-object)
  :properties(
    part-topology-list nil
    point1 (nth (nth 0 ^part-topology-list) ^points-list)
      point2 (nth (nth 1 ^part-topology-list) ^points-list)
      point3 (nth (nth 2 ^part-topology-list) ^points-list)
      point4 (nth (nth 3 ^part-topology-list) ^points-list)
      vector1 (subtract-vectors ^point2 ^point1)
      vector2 (subtract-vectors ^point4 ^point3)
      katet (vector-length(subtract-vectors ^point3 ^point2))
      angle (angle-between-2-vectors ^vector1 ^vector2)
      halfangle (/ ^angle 2)
      elbow-radius (/ (/ ^katet 2) (sind ^halfangle))
      boneheight (vector-length
        (subtract-vectors ^point4 ^point3))
    diameter (* 1.2 ^boneheight)
    orientation (list (align
      (center-of-face (the superior) 1)
      (normal-to-face (the superior) 1)
      nil
      (^point2)
      (subtract-vectors ^point2 ^point1)
      nil
      :move? t :align? t :orient? t
      )
    )
    reference-coordinate-system (the superior superior self)
    )
  )

; Thoracic section
(define-class thoracic-cartilage-object
  :inherit-from (cartilage-object elbow-object)
  :properties(
    part-topology-list nil
    point1 (nth (nth 0 ^part-topology-list) ^points-list)
      point2 (nth (nth 1 ^part-topology-list) ^points-list)
      point3 (nth (nth 2 ^part-topology-list) ^points-list)
      point4 (nth (nth 3 ^part-topology-list) ^points-list)
      vector1 (subtract-vectors ^point2 ^point1)
      vector2 (subtract-vectors ^point4 ^point3)
      katet (vector-length(subtract-vectors ^point3 ^point2))
      angle (angle-between-2-vectors ^vector1 ^vector2)
      halfangle (/ ^angle 2)
      elbow-radius (/ (/ ^katet 2) (sind ^halfangle))
    boneheight (vector-length
      (subtract-vectors ^point4 ^point3))
    diameter (* 1.2 ^boneheight)
    orientation (list
      (rotate 180 :y-axis)
      (align
      (center-of-face (the superior) 1)
      (normal-to-face (the superior) 1)
      nil
      (^point2)
      (subtract-vectors ^point2 ^point1)
      nil
      :move? t :align? t :orient? t
      )
    )
    reference-coordinate-system
    (the superior superior self)
    )
  )
```

Listing B.10: geometry.aml

```
;;;-------------------------------------------------------
;;; System: Musculoskeletal-system
;;; Purpose: Class definitions that generates instances
;;;     of geometric objects
;;; Derived from Mechanism Program by Ole Ivar Sivertsen
;;; Author : Carl Otto Gjelsvik
;;;-------------------------------------------------------

(define-class points-class
  :inherit-from (coordinate-system-class)
  :properties(
    length 10
    colors '(red green blue)
    transformation-matrix
    (default '(
      (1.0 0.0 0.0 0.0)
      (0.0 1.0 0.0 0.0)
      (0.0 0.0 1.0 0.0)
      (0.0 0.0 0.0 1.0)
      )
    )
    )
  )

(define-class link-coordinate-systems
  :inherit-from (series-object coordinate-system-class)
  :properties(
    label nil
    draw-box? nil
    draw-label? t
    draw-axes? t
    Quantity (length ^^bone-data-list)
    series-prefix 'coordinate-system
    class-expression 'coordinate-system-class
    init-form '(
    origin
    (nth
      (nth 0
        (nth 2
          (nth !index ^bone-data-list)
          )
        )
      ^points-list
      )
    vector-i
    (subtract-vectors
      (nth
        (nth 1
          (nth 2
            (nth !index ^bone-data-list)
            )
          )
        ^points-list
        )
      (nth
        (nth 0
          (nth 2
            (nth !index ^bone-data-list )
            )
          )
        ^points-list
        )
      )
    vector-j (cross-product
      (nth 3
        (nth !index ^bone-data-list )
        )
      (subtract-vectors
        (nth
          (nth 1
            (nth 2
              (nth !index ^bone-data-list)
              )
            )
          ^points-list
          )
        (nth
          (nth 0
            (nth 2
              (nth !index ^bone-data-list)
              )
            )
          ^points-list
          )
        )
      )
    label (nth 0
      (nth !index ^bone-data-list)
      )
    length 0.2
    )
  reference-coordinate-system
  (the superior superior self)
  )
)
(define-class cartilage-coordinate-systems
  :inherit-from (series-object coordinate-system-class)
  :properties(
  label nil
  draw-box? nil
  draw-label? t
  draw-axes? t
  Quantity (length ^^cartilage-data-list)
  series-prefix 'coordinate-system
  class-expression 'coordinate-system-class
  init-form '(
    origin (nth
      (nth 0
        (nth 2
          (nth !index ^cartilage-data-list)
          )
        )
      ^points-list
      )
    vector-i (subtract-vectors
      (nth
        (nth 1
          (nth 2
            (nth !index ^cartilage-data-list)
            )
          )
        ^points-list
        )
      (nth
        (nth 0
          (nth 2
            (nth !index ^cartilage-data-list)
            )
          )
        ^points-list
        )
      )
    vector-j (cross-product
      (nth 3
        (nth !index ^cartilage-data-list)
        )
      (subtract-vectors (nth
        (nth 1
          (nth 2
            (nth !index ^cartilage-data-list)
            )
          )
        ^points-list
        )
      (nth
        (nth 0
          (nth 2
            (nth !index ^cartilage-data-list)
            )
          )
        ^points-list
        )
```

```
        )
        )
      label (nth 0 (nth !index ^cartilage-data-list))
      length 0.2
      )
    reference-coordinate-system
    (the superior superior self)
    )
)
(define-class bone-part
  :inherit-from (series-object coordinate-system-class)
  :properties(
    property-objects-list
    (list
      "Femur Properties"
      (the superior outer-bone-density self)
      (the superior inner-bone-density self)
      (the superior average-stiffness self)
      )
    (outer-bone-density :class 'editable-data-property-class
      label "Outer Bone Density"
      formula (default 1)
    )
    (inner-bone-density :class 'editable-data-property-class
      label "Inner Bone Density"
      formula (default 0.5)
    )
    (average-stiffness :class 'editable-data-property-class
      label "Average Stiffness"
      formula (default 1.5)
    )
    part-topology-list nil
    the-ref-property nil
    label nil
    Quantity 1
    series-prefix 'bonepart
    bone-type nil
    class-expression '(case ^bone-type
      ("vertebrae" 'vertebrae-object)
      ("sacrum" 'sacrum-object)
      ("pelvis" 'pelvis-object)
      ("femur" 'femur-object)
      ("tibia" 'tibia-object)
      (nil   'default-bone-object)
      )
    init-form'(
      part-topology-list ^^part-topology-list
      outer-bone-density ^^outer-bone-density
      inner-bone-density ^^inner-bone-density
      average-stiffness ^^average-stiffness
      )
    )
  )

(define-class cartilage-generator
  :inherit-from (series-object coordinate-system-class)
  :properties(
    part-topology-list nil
    the-ref-property nil
    label nil
    Quantity 1
    series-prefix 'cartilagepart
    cartilage-type nil
    class-expression '(case ^cartilage-type
      ("intervertebral-disc"
      'intervertebral-disc-object)
      ("thoracic-cartilage"
      'thoracic-cartilage-object)
      (t    'default-cartilage-object)
      )
    init-form'(
      part-topology-list ^^part-topology-list
      )
    )
  )
(define-class bone-collection
  :inherit-from (
    series-object
    coordinate-system-class
    cylinder-object
    )
  :properties(
    draw-box? nil
    draw-label? nil
    the-ref-property nil
    Quantity (length ^^bone-data-list)
    series-prefix 'bone
    bone-type nil
    class-expression 'bone-part
    init-form '(
      part-topology-list (nth 2
        (nth !index ^bone-data-list )
        )
      bone-type (nth 4
        (nth !index ^bone-data-list )
        )
      the-ref-property (list (the superior))
      label (nth 0 (nth !index ^bone-data-list))
      )
    )
  )
(define-class cartilage-collection
  :inherit-from (series-object coordinate-system-class)
  :properties(
    draw-box? nil
    draw-label? nil
    the-ref-property nil
    Quantity (length ^^cartilage-data-list)
    series-prefix 'cartilage
    class-expression 'cartilage-generator
    init-form '(
      part-topology-list (nth 2
        (nth !index ^cartilage-data-list )
        )
      cartilage-type (nth 4
        (nth !index ^cartilage-data-list )
        )
      the-ref-property (list (the superior))
      label (nth 0
        (nth !index ^cartilage-data-list )
        )
      transformation-matrix-list
      (loop for ix in
        (remove-duplicates
          (nth 2
            (nth !index ^cartilage-data-list)
            )
          )
      do
      collect (list
        '(1.0 0.0 0.0 0.0)
        '(0.0 1.0 0.0 0.0)
        '(0.0 0.0 1.0 0.0)
        (append
          (nth ix ^points-list)(list 1.0))
        )
      )
      reference-coordinate-system
      (the superior superior self)
      )
    )
  )

(define-class mechanism-triads
  :inherit-from (series-object coordinate-system-class)
  :properties(
    draw-box? nil
    draw-label? nil
    length 0.2
    Quantity (length ^^points-list)
    series-prefix 'triad
    class-expression 'points-class
    init-form  '(
      transformation-matrix (list
        '(1.0 0.0 0.0 0.0)
        '(0.0 1.0 0.0 0.0)
```

```
       '(0.0 0.0 1.0 0.0)
       (append (nth !index ^points-list)(list 1.0))
       )
     orientation (list
       (apply-matrix ^transformation-matrix)
       )
     reference-coordinate-system
     (the superior superior self)
     )
   )
  )
(define-class mechanism-model-class
  :inherit-from (series-object coordinate-system-class)
  :properties (
    )
  :subobjects(
    (bones :class 'bone-collection
     )
    (cartilages :class 'cartilage-collection
     )
    (coordinate-systems :class 'link-coordinate-systems
     )
    (triads :class 'mechanism-triads
     )
    )
  )
```

Listing B.11: draw.aml

```
;;;----------------------------------------------------------
;;; System: Musculoskeletal-system
;;; Purpose: Class definitions that draws/undraws geometry
;;;         and labels
;;; Derived from Mechanism Program by Ole Ivar Sivertsen
;;; Author : Carl Otto Gjelsvik
;;;----------------------------------------------------------

(define-class edit-coordinates-class
  :inherit-from (object)
  :properties (
    sub-folder-list nil
    sub-folder-hash-table (let* (
      (ht (make-hash-table))
      )
    (loop for folder-info in ^sub-folder-list
      for folder-id = (nth 1 folder-info)
      do
      (setf (gethash folder-id ht) folder-info)
      )
    ht
    )
    (pop-up-selection :class 'option-property-class
      labels-list (loop for pair in ^^sub-folder-list
        collect (nth 0 pair)
        )
      options-list (loop for pair in ^^sub-folder-list
        collect (nth 1 pair)
        )
      mode 'menu
      formula (nth 0 !options-list)
      )
    mechanism-info (gethash ^pop-up-selection
      ^sub-folder-hash-table)
    mechanism-class (nth 2 ^mechanism-info)
    mechanism-label (nth 0 ^mechanism-info)
                nrows nil
                )
  )
(define-method property-classification-list
  edit-coordinates-class ()
  (let* ((link-objects (the mechanism bones))
    (cartilage-objects (the mechanism cartilages))
    (node-label-objects (the superior node-label))
    (bone-label-objects (the superior bone-label))
    (cartilage-label-objects (the superior cartilage-label))
    )
  (list
    (list "Main Properties"
      (remove nil
        (list
          "User Input"
          '(pop-up-selection (automatic-apply? t))
          (the superior points-list self)
          :blank
          "Draw/Undraw"
          (list 'nrows
            (list 'label "Draw Bones"
              'button1-action `(
                draw-bones ,link-objects :undraw? nil)
              'button3-action `(
                draw-bones ,link-objects :undraw? t)
              )
            'ui-access-button-class
            )
          (list 'nrows
            (list 'label "Draw Cartilages"
              'button1-action `(
                draw-cartilages ,
                cartilage-objects :undraw? nil)
              'button3-action `(
                draw-cartilages ,
                cartilage-objects :undraw? t)
              )
            'ui-access-button-class
            )
          (list 'nrows
            (list 'label "Draw Bone Names"
              'button1-action `(
                draw-bone-labels ,
                bone-label-objects :undraw? nil)
              'button3-action `(
                draw-bone-labels ,
                bone-label-objects :undraw? t)
              )
            'ui-access-button-class
            )
          (list 'nrows
            (list 'label "Draw Cartilage Names"
              'button1-action `'(
                draw-cartilage-labels ,
                cartilage-label-objects :undraw? nil)
              'button3-action `'(
                draw-cartilage-labels ,
                cartilage-label-objects :undraw? t)
              )
            'ui-access-button-class
            )
          (list 'nrows
            (list 'label "Draw Point Numbers"
              'button1-action `'(
                draw-node-labels ,node-label-objects :undraw? nil)
              'button3-action `'(
                draw-node-labels ,node-label-objects :undraw? t)
              )
            'ui-access-button-class
            )
          )
        )
      )
    )
   )
  )
  (define-method draw-bones bone-collection (&key (undraw? nil))
    (if undraw?
      (undraw !bones)
      (draw   !bones)
      )
    )
  (define-method draw-cartilages cartilage-collection
    (&key (undraw? nil))
    (if undraw?
      (undraw !cartilages)
      (draw   !cartilages)
      )
    )
  (define-class mechanism-text-object-class
    :inherit-from (series-object coordinate-system-class)
  :properties(
    draw-label? nil
    draw-box? nil
    quantity (length ^^points-list)
    series-prefix 'node
    class-expression 'text-object
    init-form '(
      coordinates (list (nth 0 (nth !index ^^points-list))
        (+ (nth 1 (nth !index ^^points-list)) 0.06)
        (nth 2 (nth !index ^^points-list))
        )
      height 1.0
      text-string (format nil "~a" !index)
      reference-coordinate-system (the superior superior self)
      )
    )
  )
(define-class part-text-object
  :inherit-from (text-object)
  :properties (
    part-topology-list nil
    lastpoint nil
    color nil
    view-normal? t
    )
  )

(define-class bone-text-object-class
  :inherit-from (series-object coordinate-system-class)
  :properties(
    draw-label? nil
    draw-box? nil
    part-topology-list nil
    quantity (length ^^bone-data-list)
    series-prefix 'bone-label
    class-expression 'part-text-object
    init-form '(
      part-topology-list (nth 2
        (nth !index ^bone-data-list ))
      lastpoint (- (length ^part-topology-list) 1)
      coordinates (list  (progn (+ 5 (nth 0
        (nth (nth 0 ^part-topology-list) ^^points-list))))
        (progn (/ (+ ((nth 1 (nth (nth 0 ^part-topology-list)
          ^^points-list)))((nth 1 (nth
          (nth ^lastpoint ^part-topology-list)
          ^^points-list)))) 2))
        (progn (+ 5 (/ (+ ((nth 2 (nth
          (nth 0 ^part-topology-list) ^^points-list)))
        ((nth 2 (nth (nth ^lastpoint ^part-topology-list)
          ^^points-list)))) 2)))
        )
      height 1.0
      text-string (nth 0 (nth !index ^bone-data-list ))
      color 'red
      reference-coordinate-system (the superior superior self)
      )
    )
  )
(define-class cartilage-text-object-class
  :inherit-from (series-object coordinate-system-class)
  :properties(
    draw-label? nil
    draw-box? nil
    part-topology-list nil
    quantity (length ^^cartilage-data-list)
    series-prefix 'cartilage-label
    class-expression 'part-text-object
    init-form '(
      part-topology-list (nth 2
        (nth !index ^cartilage-data-list ))
      lastpoint (- (length ^part-topology-list) 1)
      coordinates (list  (progn (+ 5 (nth 0 (nth
        (nth 0 ^part-topology-list) ^^points-list))))
        (progn (/ (+ ((nth 1 (nth (nth 0 ^part-topology-list)
          ^^points-list)))((nth 1 (nth
          (nth ^lastpoint ^part-topology-list)
          ^^points-list)))) 2))
        (progn (+ 5 (/ (+ ((nth 2 (nth
          (nth 0 ^part-topology-list)
          ^^points-list)))((nth 2
          (nth (nth ^lastpoint ^part-topology-list)
            ^^points-list)))) 2)))
        )
      height 1.0

      text-string (nth 0 (nth !index ^cartilage-data-list ))
      color 'white
      reference-coordinate-system (the superior superior self)
      )
    )
  )
(define-method draw-node-labels
  mechanism-text-object-class (&key (undraw? nil))
  (if undraw?
    (undraw !node-label)
    (draw   !node-label)
    )
  )
(define-method draw-bone-labels
  bone-text-object-class (&key (undraw? nil))
  (if undraw?
    (undraw !bone-label)
    (draw   !bone-label)
    )
  )
(define-method draw-cartilage-labels
  cartilage-text-object-class (&key (undraw? nil))
  (if undraw?
    (undraw !cartilage-label)
    (draw   !cartilage-label)
    )
  )
```

Listing B.12: io.aml

```
;;;-------------------------------------------------------
;;; System: Musculoskeletal-system
;;; Purpose: User input and GUI
```

```
;;;
;;; Derived from Mechanism Program by Ole Ivar Sivertsen
;;; Author : Carl Otto Gjelsvik
;;;---------------------------------------------------------


(define-class musculoskeletal-program
  :inherit-from (object)
  :properties(
    library-root-path
    "D:\\Technosoft\\AML\\AML5.85_x64\\workspace\\
    musculoskeletal-program\\models"
    valid-root-path (directory? ^library-root-path)
    sub-folder-list (if ^valid-root-path
      (file-popup-list (the superior)) nil)
    pop-up-selection
    (the pop-up-selection (:from ^start-ui))
    mechanism-class
    (the mechanism-class (:from ^start-ui))
    mechanism-label
    (the mechanism-label (:from ^start-ui))
    selected-mechanism
    (nth (- ^pop-up-selection 1) ^sub-folder-list)
    directory-path (nth ^pop-up-selection
      (rest(directory
        "D:\\Technosoft\\AML\\AML5.85_x64\\workspace\\
        musculoskeletal-program\\models"
        )))
    loaded-points-list
    (if
      (or
        (equal ^valid-root-path nil)
        (equal ^sub-folder-list nil)
        )
      ^default-points-list
      (read-selected-points-list (the superior))
      )
    points-quantity (length ^loaded-points-list)
    (points-list :class 'data-matrix-property-class
      mode 'pop-up
      default-data-type 'not-a-string
      label "Points List"
      columns-labels-list (list
       "X-coord"
       "Y-coord"
       "Z-coord"
       )
      rows-labels-list
      (loop for row from 1 to ^^points-quantity
       collect (format nil "Node ~a"
         (- row 1)
         )
       )
      formula ^loaded-points-list
      )
    bone-data-list
    (if
      (or
        (equal ^valid-root-path nil)
        (equal ^sub-folder-list nil)
        ) ^default-bone-data-list
      (read-selected-bone-data-list (the superior)
        )
      )
    cartilage-data-list
    (if
      (or
        (equal ^valid-root-path nil)
        (equal ^sub-folder-list nil)
        )
      ^default-cartilage-data-list
      (read-selected-cartilage-data-list (the superior)
        )
      )
    save-sub-directory-name "test-model"
    save-directory-path (create-directory
      (format nil "~a~a~a"
        (append !library-root-path) '"\\"
        (append !save-sub-directory-name))
      )
    display-tree (with-open-file
      (file (logical-path !directory-path "tree.txt")
       :direction :output
       :if-exists :overwrite
       )
      (print-tree
        (the superior)
        :show-class? t :expand? nil :stream file)
      )
    default-points-list '(
      (0.0 0.0 0.0)
      (0.0 0.15 0.0)
      (0.3 0.0 0.0)
      (0.3 0.375 0.0)
      (0.6 0.6 0.0)
      )

    default-bone-data-list '(
     ("input-link" "FE-input-link" (0 1) (0.0 0.0 1.0))
     ("coupler-link" "FE-coupler-link" (1 3 4) (0.0 0.0 1.0))
     ("output-link" "FE-output-link" (2 3) (0.0 0.0 1.0))
     ("Ground" nil (0 2) nil)
     )
    default-cartilage-data-list '(
     ("input-link" "FE-input-link" (0 1) (0.0 0.0 1.0))
     ("coupler-link" "FE-coupler-link" (1 3 4) (0.0 0.0 1.0))
     ("output-link" "FE-output-link" (2 3) (0.0 0.0 1.0))
     ("Ground" nil (0 2) nil)
     )
    )
:subobjects(
  (start-ui :class 'edit-coordinates-class
                      sub-folder-list ^^sub-folder-list
                      )
  (mechanism :class 'Mechanism-model-class
    )
  (mechanism-label :class 'text-object
    height 5
    coordinates '(0 -0.06 0)
    text-string (format nil "~a" (nth 0 ^^selected-mechanism))
    )
  (node-label :class 'mechanism-text-object-class
    )
  (bone-label :class 'bone-text-object-class)
  (cartilage-label :class 'cartilage-text-object-class)
  )
)


(define-method file-popup-list musculoskeletal-program ()

  (loop for line in (rest
   (rest
    (directory
      "D:\\Technosoft\\AML\\AML5.85_x64\\workspace\\
      musculoskeletal-program\\models"
      )
    )
   )
  do
  for x = (string-to-delimited-token-list line
    :delimiter #\\
    :string-token? t
    :blank-token? nil
    )
  for ix from 1 to ( - (length
    (directory
      "D:\\Technosoft\\AML\\AML5.85_x64\\workspace\\
      musculoskeletal-program\\models"
      )) 2)
  for name = (first (last x))
  for class-name = (read-from-string
    (concatenate name "-class"))
  for class-exists? = (find-class class-name)
  collect (list name ix (when class-exists?
   class-name)))
  )

(define-method read-selected-points-list
  musculoskeletal-program ()
  (with-open-file (file (format nil "~a~a"
    (append !directory-path) '"\\points.txt")
    :direction :input
    )
  (loop for line = (read-line file nil nil)
    while line
    for points = (read-from-string (format nil "(~a)" line))
    collect points
    )
  )
  )
(define-method read-selected-bone-data-list
  musculoskeletal-program ()
  (with-open-file (file (format nil "~a~a"
    (append !directory-path) '"\\bones.txt")
    :direction :input
    )
  (loop for line = (read-line file nil nil)
    while line
    for x = (string-to-delimited-token-list line
      :delimiter #\tab
      :string-token? nil
      :blank-token? nil
      )
    for bone = (list (nth 0 x)
     (nth 1 x)
     (read-from-string (nth 2 x))
     (read-from-string (nth 3 x))
     (nth 4 x)
     )
    collect bone
    )
  )
  )
(define-method read-selected-cartilage-data-list
  musculoskeletal-program ()
  (with-open-file (file (format nil "~a~a"
    (append !directory-path) '"\\cartilage.txt")
    :direction :input
    )
  (loop for line = (read-line file nil nil)
    while line
    for x = (string-to-delimited-token-list line
      :delimiter #\tab
      :string-token? nil
      :blank-token? nil
      )
    for cartilage = (list (nth 0 x)
     (nth 1 x)
     (read-from-string (nth 2 x))
     (read-from-string (nth 3 x))
     (nth 4 x)
     )
    collect cartilage
    )
  )
  )
```

## B.2 Example Input Files

The following text files is input for the model which can be seen in Figure 5.18.

Listing B.13: points.txt

```
-11.4 0.0 0.0
-9.4 0.0 0.0
-10.4 0.5 1.0
-15.0 45.0 -1.0
-11.0 46.5 -1.0
-10.0 -40.0 0.0
-10.0 -3.0 0.0
9.4 0.0 0.0
11.4 0.0 0.0
```

```
10.4 0.5 1.0
15.0 45.0 -1.0
11.0 46.5 -1.0
10.0 -40.0 0.0
10.0 -3.0 0.0
0.000000 46.000000 -10.000000
0.000000 55.350000 -6.260000
0.000000 56.078651 -5.863603
0.000000 57.320678 -5.289171
0.000000 58.605108 -4.817110
0.000000 59.397994 -4.573399
0.000000 60.668350 -4.269238
0.000000 61.958403 -4.064102
0.000000 62.783904 -3.982797
0.000000 64.027148 -3.937006
0.000000 65.270392 -3.982797
0.000000 66.095589 -4.067141
0.000000 67.262141 -4.257044
0.000000 68.412568 -4.528017
0.000000 69.204551 -4.774645
0.000000 70.252671 -5.168640
0.000000 71.272400 -5.631176
0.000000 71.804301 -5.847060
0.000000 72.770093 -6.214281
0.000000 73.745978 -6.553772
0.000000 74.290657 -6.735015
0.000000 75.247982 -7.030569
0.000000 76.213125 -7.299502
0.000000 76.768328 -7.445353
0.000000 77.712098 -7.671859
0.000000 78.661604 -7.872968
0.000000 79.225036 -7.982824
0.000000 80.150464 -8.143250
0.000000 81.079745 -8.279599
0.000000 81.649073 -8.353005
0.000000 82.551698 -8.450638
0.000000 83.456488 -8.525592
0.000000 84.029359 -8.562246
0.000000 84.905065 -8.600668
```

```
0.000000 85.781446 -8.617864
0.000000 86.355487 -8.617612
0.000000 87.200525 -8.600665
0.000000 88.044937 -8.563983
0.000000 88.617775 -8.526828
0.000000 89.428776 -8.458582
0.000000 90.238037 -8.372111
0.000000 90.807301 -8.298206
0.000000 91.581290 -8.182930
0.000000 92.352604 -8.050940
0.000000 92.915940 -7.940591
0.000000 93.650347 -7.782713
0.000000 94.381317 -7.609621
0.000000 94.936393 -7.463284
0.000000 95.629061 -7.267361
0.000000 96.317695 -7.057697
0.000000 96.862214 -6.875978
0.000000 97.511406 -6.646663
0.000000 98.156117 -6.405036
0.000000 98.595847 -6.266389
0.000000 99.260853 -6.076487
0.000000 99.931715 -5.908444
0.000000 100.381856 -5.808651
0.000000 101.060882 -5.677430
0.000000 101.743838 -5.568496
0.000000 102.200964 -5.508315
0.000000 102.888842 -5.436774
0.000000 103.578693 -5.387779
0.000000 104.039325 -5.367667
0.000000 104.730821 -5.356352
0.000000 105.422317 -5.367667
0.000000 105.882949 -5.387779
0.000000 106.572800 -5.436774
0.000000 107.260678 -5.508315
0.000000 107.717804 -5.568496
0.000000 108.400760 -5.677430
0.000000 109.079786 -5.808651
0.000000 109.529927 -5.908444
0.000000 110.200789 -6.076487
```

```
0.000000 110.865795 -6.266389
```

Listing B.14: bones.txt

```
"Femur-Left"  "FE-Femur"      "(0 1 2 3 4)" "femur"
"Tibia-Left"  "FE-Tibia"      "(5 6)"      "tibia"
"Femur-Right" "FE-Femur"      "(7 8 9 10 11)" "femur"
"Tibia-Right" "FE-Tibia"      "(12 13)"    "tibia"
"Pelvis"   "FE-Pelvis"     "(4 11)"   "pelvis"
"S1"       "FE-Sacrum"     "(14 15 16 18)" "sacrum"
"L5"       "FE-Lumbar-bone" "(16 18)"    "vertebrae"
"L4"       "FE-Lumbar-bone" "(19 21)"    "vertebrae"
"L3"       "FE-Lumbar-bone" "(22 24)"    "vertebrae"
"L2"       "FE-Lumbar-bone" "(25 27)"    "vertebrae"
"L1"       "FE-Lumbar-bone" "(28 30)"    "vertebrae"
"T12"      "FE-Thoracic-bone" "(31 33)"    "vertebrae"
"T11"      "FE-Thoracic-bone" "(34 36)"    "vertebrae"
"T10"      "FE-Thoracic-bone" "(37 39)"    "vertebrae"
"T9"       "FE-Thoracic-bone" "(40 42)"    "vertebrae"
"T8"       "FE-Thoracic-bone" "(43 45)"    "vertebrae"
"T7"       "FE-Thoracic-bone" "(46 48)"    "vertebrae"
"T6"       "FE-Thoracic-bone" "(49 51)"    "vertebrae"
"T5"       "FE-Thoracic-bone" "(52 54)"    "vertebrae"
"T4"       "FE-Thoracic-bone" "(55 57)"    "vertebrae"
"T3"       "FE-Thoracic-bone" "(58 60)"    "vertebrae"
"T2"       "FE-Thoracic-bone" "(61 63)"    "vertebrae"
"T1"       "FE-Thoracic-bone" "(64 66)"    "vertebrae"
"C7"       "FE-Clavi-bone"    "(67 69)"    "vertebrae"
"C6"       "FE-Clavi-bone"    "(70 72)"    "vertebrae"
"C5"       "FE-Clavi-bone"    "(73 75)"    "vertebrae"
"C4"       "FE-Clavi-bone"    "(76 78)"    "vertebrae"
"C3"       "FE-Clavi-bone"    "(79 81)"    "vertebrae"
"C2"       "FE-Clavi-bone"    "(82 84)"    "vertebrae"
"C1"       "FE-Clavi-bone"    "(85 87)"    "vertebrae"
```

Listing B.15: cartilage.txt

```
"S1-L5"    "FE-cartilage" "(14 15 16 18)" "intervertebral-disc"
"L5-L4"    "FE-cartilage" "(16 18 19 21)" "intervertebral-disc"
"L4-L3"    "FE-cartilage" "(19 21 22 24)" "intervertebral-disc"
"L3-L2"    "FE-cartilage" "(22 24 25 27)" "intervertebral-disc"
```

```
"L2-L1"    "FE-cartilage" "(25 27 28 30)" "intervertebral-disc"
"L1-T12" "FE-cartilage" "(28 30 31 33)" "thoracic-cartilage"
"T12-T11"  "FE-cartilage" "(31 33 34 36)" "thoracic-cartilage"
"T11-T10"  "FE-cartilage" "(34 36 37 39)" "thoracic-cartilage"
"T10-T9" "FE-cartilage" "(37 39 40 42)" "thoracic-cartilage"
"T9-T8"    "FE-cartilage" "(40 42 43 45)" "thoracic-cartilage"
"T8-T7"    "FE-cartilage" "(43 45 46 48)" "thoracic-cartilage"
"T7-T6"    "FE-cartilage" "(46 48 49 51)" "thoracic-cartilage"
"T6-T5"    "FE-cartilage" "(49 51 52 54)" "thoracic-cartilage"
"T5-T4"    "FE-cartilage" "(52 54 55 57)" "thoracic-cartilage"
"T4-T3"    "FE-cartilage" "(55 57 58 60)" "thoracic-cartilage"
"T3-T2"    "FE-cartilage" "(58 60 61 63)" "thoracic-cartilage"
"T2-T1"    "FE-cartilage" "(61 63 64 66)" "thoracic-cartilage"
"T1-C7"    "FE-cartilage" "(64 66 67 69)" "intervertebral-disc"
"C7-C6"    "FE-cartilage" "(67 69 70 72)" "intervertebral-disc"
"C6-C5"    "FE-cartilage" "(70 72 73 75)" "intervertebral-disc"
"C5-C4"    "FE-cartilage" "(73 75 76 78)" "intervertebral-disc"
"C4-C3"    "FE-cartilage" "(76 78 79 81)" "intervertebral-disc"
"C3-C2"    "FE-cartilage" "(79 81 82 84)" "intervertebral-disc"
"C2-C1"    "FE-cartilage" "(82 84 85 87)" "intervertebral-disc"
```

The following text file is input for the model which can be seen in Figure 5.17.

Listing B.16: points.txt-file for a spine

```
0.0000 0.0000 -0.0000
0.0000 9.3500 3.7400
0.0000 10.4150 4.3193
0.0000 12.2302 5.1589
0.0000 14.1075 5.8488
0.0000 15.2663 6.2050
0.0000 17.1230 6.6496
0.0000 19.0084 6.9494
0.0000 20.2149 7.0682
0.0000 22.0320 7.1351
0.0000 23.8490 7.0682
0.0000 25.0551 6.9449
0.0000 26.7601 6.6674
0.0000 28.4414 6.2714
0.0000 29.5990 5.9109
```

```
0.0000 31.1308 5.3351
0.0000 32.6212 4.6591
0.0000 33.3986 4.3435
0.0000 34.8101 3.8068
0.0000 36.2364 3.3106
0.0000 37.0325 3.0457
0.0000 38.4317 2.6138
0.0000 39.8423 2.2207
0.0000 40.6537 2.0076
0.0000 42.0331 1.6765
0.0000 43.4208 1.3826
0.0000 44.2443 1.2220
0.0000 45.5968 0.9876
0.0000 46.9550 0.7883
0.0000 47.7871 0.6810
0.0000 49.1063 0.5383
0.0000 50.4287 0.4287
0.0000 51.2660 0.3752
0.0000 52.5459 0.3190
0.0000 53.8267 0.2939
0.0000 54.6657 0.2943
0.0000 55.9008 0.3190
0.0000 57.1349 0.3726
0.0000 57.9721 0.4269
0.0000 59.1574 0.5267
0.0000 60.3402 0.6531
0.0000 61.1722 0.7611
0.0000 62.3034 0.9296
0.0000 63.4307 1.1225
0.0000 64.2541 1.2838
0.0000 65.3274 1.5145
0.0000 66.3958 1.7675
0.0000 67.2070 1.9814
0.0000 68.2194 2.2677
0.0000 69.2259 2.5741
0.0000 70.0217 2.8397
0.0000 70.9705 3.1749
0.0000 71.9128 3.5280
0.0000 72.5555 3.7307
```

```
0.0000 73.5274 4.0082
0.0000 74.5079 4.2538
0.0000 75.1658 4.3997
0.0000 76.1582 4.5914
0.0000 77.1564 4.7507
0.0000 77.8245 4.8386
0.0000 78.8298 4.9432
0.0000 79.8381 5.0148
0.0000 80.5113 5.0442
0.0000 81.5220 5.0607
0.0000 82.5326 5.0442
0.0000 83.2058 5.0148
0.0000 84.2141 4.9432
0.0000 85.2195 4.8386
0.0000 85.8876 4.7507
0.0000 86.8857 4.5914
0.0000 87.8781 4.3997
0.0000 88.5360 4.2538
0.0000 89.5165 4.0082
0.0000 90.4885 3.7307
```

## B.3 Spine Input Generator - Source Code

To generate points for spine models, the following program was developed in matlab.

Listing B.17: spine_points_generator.m

```
%------------------------------------------------------------
% System: spine_point_generator
% Purpose: Shorten time when making model for
%        Musculoskeletal Program.
%
% Author: Carl Otto Gjelsvik
%------------------------------------------------------------
%Set height of the person
height       = 187;
%Edit spineHeight to overwrite the spine height.
spineHeight  = 0.38*height;
```

```
cArcLength     = 0.2276*spineHeight;
tArcLength     = 0.5195*spineHeight;
lArcLength     = 0.273*spineHeight;
cSectionAngle   = 30;
tSectionAngle  = 40;
lSectionAngle  = 45;
numberOfCSections = 7;
numberOftTSections = 12;
numberOfLSections = 5;
cRadius = cArcLength*180/(pi*cSectionAngle);
tRadius = tArcLength*180/(pi*tSectionAngle);
lRadius = lArcLength*180/(pi*lSectionAngle);
%enter a startpositon to connect with other models.
startposition = [0.0, 0.0*height, -0.0*height];
A = zeros(60,3);
A(1,:,:) = [startposition(1),
        startposition(2),
        startposition(3)];
A(2,:,:) = [A(1,1)+0,
        (A(1,2)+(height*0.05)),
        (A(1,3)+(0.02*height))];
lengthOfCartilage = (lArcLength/(numberOfLSections-1))*0.25;
lengthOfBone = (lArcLength/(numberOfLSections-1))*0.75;
k = 1;
l = 1.1;
for i=1:3:15
   length = l*lengthOfBone;
   A(2+i,1:3) = nextPoint(lSectionAngle,
           lRadius,
           numberOfLSections,
           -((length/2)+lengthOfCartilage),
           -length/2,
           A(i+1,:,:),
           k);
   A(3+i,1:3) = nextPoint(lSectionAngle,
           lRadius,
           numberOfLSections,
           -(length/2),
           0,
            A(i+2,:,:),
            k);
    A(4+i,1:3) = nextPoint(lSectionAngle,
            lRadius,
            numberOfLSections,
            0,
            length/2,
            A(i+3,:,:),
            k);
    k = k+2;
    l = l - 0.05;
end
lengthOfCartilage = (tArcLength/(numberOftTSections-1))*0.25;
lengthOfBone = (tArcLength/(numberOftTSections-1))*0.75
k = 1;
l = 1.2;
for i=1:3:36
    length = l*lengthOfBone;
    A(17+i,1:3) = nextThoracicPoint(tSectionAngle,
       tRadius,
    numberOftTSections,
     -((length/2)+lengthOfCartilage),
     -length/2,
    A(i+16,:,:),
    k);
    A(18+i,1:3) = nextThoracicPoint(tSectionAngle,
       tRadius,
    numberOftTSections,
    -(length/2),
    0,
    A(i+17,:,:),
    k);
    A(19+i,1:3) = nextThoracicPoint(tSectionAngle,
       tRadius,
    numberOftTSections,
    0,
    length/2,
    A(i+18,:,:),
    k);
```

```
   k = k+2;
   l = l - 0.0364;
end
lengthOfCartilage = (cArcLength/(numberOfCSections-1))*0.25;
lengthOfBone = (cArcLength/(numberOfCSections-1))*0.75;
k = 1;
for i=1:3:21
   A(53+i,1:3) = nextPoint(cSectionAngle,
               cRadius,
               numberOfCSections,
                -((lengthOfBone/2)+lengthOfCartilage),
               -lengthOfBone/2,
               A(i+52,:,:),
               k);
   A(54+i,1:3) = nextPoint(cSectionAngle,
               cRadius,
               numberOfCSections,
               -(lengthOfBone/2),
               0,
               A(i+53,:,:),
               k);
   A(55+i,1:3) = nextPoint(cSectionAngle,
               cRadius,
               numberOfCSections,
               0,
               lengthOfBone/2,
               A(i+54,:,:),
               k);
   k = k+2;
end
write_coords(A);
```

Listing B.18: nextPoint.m

```
%------------------------------------------------------------
% System: spine_point_generator
% Function for finding next point in spine generator
%
% Author: Carl Otto Gjelsvik
%------------------------------------------------------------
```

```
function [ y ] = nextY(
              arcangle,
              radius,
              sectionCount,
              arcLengthToPrevious,
              arcLengthToPoint,
              prev,
              i)

mtemp      = -sectionCount+(i);
mdegrees   = mtemp*(arcangle/((sectionCount-1)*2));
pdegrees   = mdegrees+(180*arcLengthToPrevious)/(radius*pi);
degrees  = mdegrees+(180*arcLengthToPoint)/(radius*pi);
dy         = abs(radius*sind(pdegrees)-radius*sind(degrees));
dz         = radius*cosd(degrees)-radius*cosd(pdegrees);
y = [0.0, prev(1,2)+dy, prev(1,3)+dz];

end
```

Listing B.19: nextThoracicPoint.m

```
%------------------------------------------------------------
% System: spine_point_generator
% Function for finding next point in spine generator
%
% Author: Carl Otto Gjelsvik
%------------------------------------------------------------
function [ y ] = nextY(
              arcangle,
              radius,
              sectionCount,
              arcLengthToPrevious,
              arcLengthToPoint,
              prev,
              i)

mtemp      = -sectionCount+(i);
mdegrees   = 180-(mtemp*(arcangle/((sectionCount-1)*2)));
pdegrees   = mdegrees-((180*arcLengthToPrevious)/(radius*pi));
degrees  = mdegrees-((180*arcLengthToPoint)/(radius*pi));
```

```
dy          = radius*sind(degrees)-radius*sind(pdegrees);
dz          = radius*cosd(degrees)-radius*cosd(pdegrees);
y = [0.0, prev(1,2)+dy, prev(1,3)+dz];

end
```

Listing B.20: write_coords.m

```
%------------------------------------------------------------
% System: spine_point_generator
% Writing the coordinates to a point.txt file in the
% same folder as the script.
%
% Author: Carl Otto Gjelsvik
%------------------------------------------------------------
function [ t ] = write_coords( A )

B = transpose(A);
fileID = fopen('points.txt','w');
fprintf(fileID,'%0.4f %0.4f %0.4f \n',B);
fclose(fileID);
t = 1;

end
```

# Appendix C

# Source Code: Segmentation Tool

## C.1 Source code

Listing C.1: system.m

```
%-----------------------------------------------------------
% System: Segmentation-tool
% Purpose: Segmentation of bone tissue and 3D display of result
%
% Toolboxes: Matlab Image Toolbox
%
% DICOM access reference:
% "http://se.mathworks.com/company/newsletters/
%     articles/accessing-data-in-dicom-files.html"
%
% Author: Carl Otto Gjelsvik
%-----------------------------------------------------------

[FileName,PathName,FilterIndex] = uigetfile('.dcm');
dicomInfo = dicominfo(strcat(PathName, FileName));
nRows = dicomInfo.Rows;
nCols = dicomInfo.Columns;
SliceThickness = dicomInfo.SliceThickness;
nPlanes = dicomInfo.SamplesPerPixel;
nFrames = length(dir(strcat(PathName,'*.dcm')));
format = getSequenceFormat(PathName);
```

```
startFrame = getSequenceFirstFrame(PathName);
lastFrame = getSequenceLastFrame(PathName);
filename = FileName(1:length(FileName)-(4+format));
formatAsString = sprintf('%d', format);
%Allocates matrix X
X = repmat(int16(0), [nRows, nCols, nPlanes, nFrames]);
i = 1;
%Reads frames to matrix X.
for p=startFrame:lastFrame
    fname = strcat(PathName,filename,sprintf(
        ['%0' formatAsString 'd'], p),'.dcm');
    X(:,:,nPlanes,i) = dicomread(fname);
    i=i+1;
end
%High and low values in matrix
minPixels = min(X(:));
maxPixels = max(X(:));
%Linear combination -> increase information to full 16bit
%Easier to see contrast
b = minPixels;
m = 2^16/(maxPixels - b);
Y = imlincomb(double(m), X, double(-(m * b)), 'uint16');
B = squeeze(Y(:,:,1,:));
a = 0;
threshold = 0;
while (a == 0)
    threshold = input('Enter lower threshold');
    C = B;
    C(C<threshold) = 0;
    figure
    imshow(C(:,:,ceil(nFrames/3)));
    a = input('Enter 0 to choose a new
        lower threshold or 1 to continue');
end
B(B<threshold) = 0;
tic;
fprintf('starter iso \n');
figure;
data = smooth3(B);
```

```
patch(isocaps(data,0.2),...
   'FaceColor','interp','EdgeColor','none');
p1 = patch(isosurface(data,0.2),...
   'FaceColor','blue','EdgeColor','none');
isonormals(data,p1)
view(3);
axis vis3d tight
camlight left;
colormap jet
lighting gouraud
fprintf('Time to make isosurface %d \n', toc);
```

Listing C.2: getSequenceFirstFrame.m

```
%-----------------------------------------------------------
% System: Segmentation-tool
% Purpose: Finding first frame in input list
%
% Author: Carl Otto Gjelsvik
%-----------------------------------------------------------

function [ x ] = getSequenceFirstFrame( PathName )

A = dir(strcat(PathName,'*.dcm'));
numbers = {length(A):1};
temp = '';
for i=1:length(A)
    filename = A(i).name(1:length(A(i).name)-4);
    fileNameLength = length(filename);
    while true
        if (isstrprop(filename(fileNameLength), 'digit'))
            temp=strcat(sprintf('%d',str2double(filename(fileNameLength))),temp);
            fileNameLength = fileNameLength - 1;
        else
            break;
        end
    end
    numbers{i} = temp;
    temp = '';
end
```

```
x = min(str2double(numbers));
end
```

Listing C.3: getSequenceLastFrame.m

```
%-----------------------------------------------------------
% System: Segmentation-tool
% Purpose: Finding last frame in input list
%
% Author: Carl Otto Gjelsvik
%-----------------------------------------------------------

function [ x ] = getSequenceLastFrame( PathName )

A = dir(strcat(PathName,'*.dcm'));
numbers = {length(A):1};
temp = '';
for i=1:length(A)
    filename = A(i).name(1:length(A(i).name)-4);
    fileNameLength = length(filename);
    while true
        if (isstrprop(filename(fileNameLength), 'digit'))
            temp=strcat(sprintf('%d',str2double(
                filename(fileNameLength))),temp);
            fileNameLength = fileNameLength - 1;
        else
            break;
        end
    end
    numbers{i} = temp;
    temp = '';
end
x = max(str2double(numbers));
end
```

Listing C.4: getSequenceFormat.m

```
%-----------------------------------------------------------
% System: Segmentation-tool
% Purpose: Finding format of file names
%
```

```
% Author: Carl Otto Gjelsvik
%-----------------------------------------------------------

function [ x ] = getSequenceFormat( PathName )

A = dir(strcat(PathName,'*.dcm'));
format = (length(A):1);
temp = 0;
for i=1:length(A)
    filename = A(i).name(1:length(A(i).name)-4);
    fileNameLength = length(filename);
    while true
        if (isstrprop(filename(fileNameLength), 'digit'))
            temp=temp+1;
            fileNameLength = fileNameLength - 1;
        else
            break;
        end
    end
    format(i) = temp;
    temp = 0;
end
x=min(format);
end
```

| | | | |
|---|---|---|---|
| NTNU HMS | Kartlegging av risikofylt aktivitet | Utarbeidet av | Nummer | Dato |
| | | HMS-avd. | HMSRV2601 | 22.03.2011 |
| | | Godkjent av | | Erstatter |
| | | Rektor | | 01.12.2006 |

**Enhet:** **IPM** **Dato:** 16.09.15

**Linjeleder:** **Torgeir Welo**

**Deltakere ved kartleggingen** (m/ funksjon): **Carl Otto Gjelsvik** (student), **Ole Ivar Sivertsen** (veileder)
*(Ansv. veileder, student, evt. medveiledere, evt. andre m. kompetanse)*

**Kort beskrivelse av hovedaktivitet/hovedprosess: Masteroppgave, Carl Otto Gjelsvik, Modeling and Simulation of the Human Body**

**Er oppgaven rent teoretisk?** (JA/NEI): **JA** *«JA» betyr at veileder innestår for at oppgaven ikke inneholder noen aktiviteter som krever risikovurdering. Dersom «JA»: Beskriv kort aktiviteten i kartleggingskjemaet under. Risikovurdering trenger ikke å fylles ut.*

**Signaturer:** *Ansvarlig veileder:* [illegible] *Student:* Carl Otto Gjelsvik

| ID nr. | Aktivitet/prosess | Ansvarlig | Eksisterende dokumentasjon | Eksisterende sikringstiltak | Lov, forskrift o.l. | Kommentar |
|---|---|---|---|---|---|---|
| 1 | Litteraturstudie og programmering | COG | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| NTNU | Risikovurdering | Utarbeidet av | Nummer | Dato |
|---|---|---|---|---|
| HMS | | HMS-avd. | HMSRV2601 | 22.03.2011 |
| | | Godkjent av | | Erstatter |
| | | Rektor | | 01.12.2006 |

**Enhet: IPM** **Dato:** 16.09.15

**Linjeleder: Torgeir Welo**

**Deltakere ved kartleggingen (m/ funksjon): Carl Otto Gjelsvik** (student), **Ole Ivar Sivertsen** (veileder)
*(Ansv. Veileder, student, evt. medveiledere, evt. andre m. kompetanse)*

**Risikovurderingen gjelder hovedaktivitet: Masteroppgave, Carl Otto Gjelsvik, Modeling and Simulation of the Human Body**

**Signaturer:** *Ansvarlig veileder:* Ole Ivar Sivertsen *Student:* Carl Otto Gjelsvik

| ID nr | Aktivitet fra kartleggings-skjemaet | Mulig uønsket hendelse/ belastning | Vurdering av sannsyn-lighet | Vurdering av konsekvens: | | | | Risiko-Verdi (menn-eske) | Kommentarer/status Forslag til tiltak |
|---|---|---|---|---|---|---|---|---|---|
| | | | (1-5) | Menneske (A-E) | Ytre miljø (A-E) | Øk/ materiell (A-E) | Om-dømme (A-E) | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

## Sannsynlighet vurderes etter følgende kriterier:

| Svært liten<br>1 | Liten<br>2 | Middels<br>3 | Stor<br>4 | Svært stor<br>5 |
|---|---|---|---|---|
| 1 gang pr 50 år eller sjeldnere | 1 gang pr 10 år eller sjeldnere | 1 gang pr år eller sjeldnere | 1 gang pr måned eller sjeldnere | Skjer ukentlig |

## Konsekvens vurderes etter følgende kriterier:

| Gradering | Menneske | Ytre miljø<br>Vann, jord og luft | Øk/materiell | Omdømme |
|---|---|---|---|---|
| E<br>Svært Alvorlig | Død | Svært langvarig og ikke reversibel skade | Drifts- eller aktivitetsstans >1 år. | Troverdighet og respekt betydelig og varig svekket |
| D<br>Alvorlig | Alvorlig personskade.<br>Mulig uførhet. | Langvarig skade. Lang restitusjonstid | Driftsstans > ½ år<br>Aktivitetsstans i opp til 1 år | Troverdighet og respekt betydelig svekket |
| C<br>Moderat | Alvorlig personskade. | Mindre skade og lang restitusjonstid | Drifts- eller aktivitetsstans < 1 mnd | Troverdighet og respekt svekket |
| B<br>Liten | Skade som krever medisinsk behandling | Mindre skade og kort restitusjonstid | Drifts- eller aktivitetsstans < 1uke | Negativ påvirkning på troverdighet og respekt |
| A<br>Svært liten | Skade som krever førstehjelp | Ubetydelig skade og kort restitusjonstid | Drifts- eller aktivitetsstans < 1dag | Liten påvirkning på troverdighet og respekt |

**Risikoverdi = Sannsynlighet x Konsekvens**
Beregn risikoverdi for Menneske. Enheten vurderer selv om de i tillegg vil beregne risikoverdi for Ytre miljø, Økonomi/materiell og Omdømme. I så fall beregnes disse hver for seg.

**Til kolonnen "Kommentarer/status, forslag til forebyggende og korrigerende tiltak":**
Tiltak kan påvirke både sannsynlighet og konsekvens. Prioriter tiltak som kan forhindre at hendelsen inntreffer, dvs. sannsynlighetsreduserende tiltak foran skjerpet beredskap, dvs. konsekvensreduserende tiltak.

| NTNU | Risikomatrise | utarbeidet av | Nummer | Dato |
|---|---|---|---|---|
| HMS/KS | | HMS-avd. | HMSRV2604 | 08.03.2010 |
| | | godkjent av | | Erstatter |
| | | Rektor | | 09.02.2010 |

## MATRISE FOR RISIKOVURDERINGER ved NTNU

| KONSEKVENS | | | | | | |
|---|---|---|---|---|---|---|
| | Svært alvorlig | 🟨 E1 | 🟥 E2 | 🟥 E3 | 🟥 E4 | 🟥 E5 |
| | Alvorlig | 🟨 D1 | 🟨 D2 | 🟥 D3 | 🟥 D4 | 🟥 D5 |
| | Moderat | 🟨 C1 | 🟨 C2 | 🟨 C3 | 🟥 C4 | 🟥 C5 |
| | Liten | 🟩 B1 | 🟩 B2 | 🟨 B3 | 🟨 B4 | 🟥 B5 |
| | Svært liten | 🟩 A1 | 🟩 A2 | 🟩 A3 | 🟨 A4 | 🟨 A5 |
| | | Svært liten | Liten | Middels | Stor | Svært stor |
| | | SANNSYNLIGHET | | | | |

**Prinsipp over akseptkriterium. Forklaring av fargene som er brukt i risikomatrisen.**

| Farge | | Beskrivelse |
|---|---|---|
| Rød | 🟥 | Uakseptabel risiko. Tiltak skal gjennomføres for å redusere risikoen. |
| Gul | 🟨 | Vurderingsområde. Tiltak skal vurderes. |
| Grønn | 🟩 | Akseptabel risiko. Tiltak kan vurderes ut fra andre hensyn. |