A Virtual Environment for the Modelling, Simulation and Manufacturing of Orthopaedic Devices

A thesis submitted for the degree of Doctor of Philosophy

By

Khaled Rasheed Alrashdan

School of Engineering and Design

Brunel University

May 2011

ABSTRACT

The objective of this work is to investigate whether the game physics based modelling is accurate enough to be used in modelling the motion of the human body, in particular musculoskeletal motion. Hitherto, the implementation of game physics in the medical field focused only on anatomical representation for education and training purposes. Introducing gaming platforms and physics engines into orthopaedics applications will help to overcome several difficulties encountered in the modelling of articular joints. Implementing a physics engine (PhysX), which is mainly designed for video games, handles intensive computations in optimized ways at an interactive speed. In this study, the capabilities of the physics engine (PhysX) and gaming platform for modelling and simulating articular joints are evaluated. First, a preliminary validation is carried out for mechanical systems with analytical solutions, before constructing the musculoskeletal model to evaluate the consistency of gaming platforms. The developed musculoskeletal model deals with the human joint as an unconstrained system with 6 DOF which is not available with other joint modeller. The model articulation is driven by contact surfaces and the stiffness of surrounding tissues. A number of contributions, such as contact modelling and muscle wrapping, have been made in this research to overcome some existing challenges in joint modelling. Using muscle segmentation, the proposed technique effectively handles the problem of muscle wrapping, a major concern for many; thus the shortest path and line of action are no longer problematic. Collision behaviour has also shown a stable response for colliding as well as resting objects, provided that it is based on the principles of surface properties and the conservation of linear and angular momentums. The precision of collision detection and response are within an acceptable tolerance controllable by varying the mesh density. An image based analysis system is developed in this thesis, mainly in order to validate the proposed physics based modelling solution. This minimally invasive method is based on the analysis of marker positions located at bony positions with minimal skin movement. The image based system overcomes several challenges associated with the currently existing methods, such as inaccuracy, complication, impracticability and cost. The analysis part of this research has considered the elbow joint as a case study to investigate and validate the proposed physics based model. Beside the interactive 3D simulation, the obtained results are validated by comparing them with the image based system developed within the current research to investigate joint kinematics and laxity and also with published material, MJM and results from experiments performed at the Brunel Orthopaedic Research and Learning Centre. The proposed modelling shows the advantageous speed, reliability and flexibility of the proposed model. It is shown that the gaming platform and physics engine provide a viable solution to human musculoskeletal modelling. Finally, this thesis considers an extended implementation of the proposed platform for testing and assessing the design of custom-made implants, to enhance joint performance. The developed simulation software is expected to give indicative results as well as testing different types of prosthetic implant. Design parameterization and sensitivity analysis for geometrical features are discussed. Thus, an integrated environment is proposed to link the real-time simulation software with a manufacturing environment so as to assist the production of patient specific implants by rapid manufacturing.

ACKNOWLEDGEMENT

First, I would like to praise to Allah, lord of all worlds, for giving me the health and strength to do this work. Then, prayers and peace upon our Prophet Mohammad (peace be upon him).

It is a pleasure to express my hearty thanks to my supervisor, Professor Ibrahim Esat, for his guidance, help and valuable advice all over the years of research.

Special thanks to:

My parents, Rasheed Alrashdan and Ruqaya Faheem, for their unlimited support, encouragement and prayers.

My wife, Reem, for her encouragement, patience and care.

The happiness of my life, my children, Waleed, Deema, Ruqaya and Alya, who colour my world with their own hands.

Not to forget my colleagues, Mohammad Alrashidi, Nawaf Alhaifi and Neriman Ozada, for their continuous support during my research.

And every one who has made this work possible.

Thank you all,

Khaled Alrashdan

LIST OF ORIGINAL PUBLICATIONS

Alrashdan, K., Alrashidi, M., Esat, I. & Alhaifi, N. Year. Elbow Joint Laxity and Stability Using Image-Based Analysis. *In:* The Atlas T3 Annual Meeting Proceedings, May 23-28 2010a George Town, Texas, USA. 132-136.

Alrashdan, K., Alrashidi, M., Esat, I. & Ozada, N. Year. Modelling and Simulation of Human Articulated Joint Using a Physics Engine. *In:* The Atlas T3 Annual Meeting Proceedings, May 23-28 2010b George Town, Texas, USA. 212-216. *(This study was entitled for a grant from the Kuwait Foundation for the Advancement of Science (KFAS)).*

Alrashdan, K., Alrashidi, M., Esat, I. I. & Alhaifi, N. 2010c. D-1 Elbow Joint Laxity and Stability Analysis Using Image Based Method. *Journal of biomechanics,* 43**,** S68-S68.

Alrashdan, K., Alrashidi, M., Esat, I. I. & Ozada, N. 2010d. B-6 Modelling and Simulation of Human Articulated Joints Using a Physics Engine. *Journal of biomechanics,* 43**,** S25-S26.

Alhaifi, N., Alrashdan, K., Poli, S. & Esat, I. I. 2010. D-17 Human Hip Joint Simulator with Feedback Control System. *Journal of biomechanics,* 43**,** S73-S74.

Alrashidi, M., Yildiz, I., Alrashdan, K. & Esat, I. 2009. Evaluating elbow joint kinematics with the Stewart Platform Mechanism. *Modelling in Medicine and Biology Viii,* 13**,** 181-189

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

$\mathbf{F}$	External force acting on a body
m	Mass
$\mathbf{a}$	Acceleration
x, y, z	Cartesian coordinates
x_c, y_c, z_c	Coordinates of the centre of rotation
r_{cor}	Distance from the centre of rotation to the origin
r_c	Distance from marker position to the centre of rotation
r_o	Distance to the origin of the coordinate system
i	Frame number
V	Velocity
d	Projection distance of OBB along the separating axis
t	Time (s)
h	Time step between two successive frames
$\mathbf{r}$	Offset of the applied force from the centre of mass
$\mathbf{L}$	Angular momentum
L	Distance between geometrical centres
$L_{\perp}$	Elongation of the spring perpendicular to the moment arm
$\mathbf{J}$	Moment of inertia
ε	Coefficient of restitution
$\mathbf{\Omega}$	Orientation matrix
$\boldsymbol{\omega}$	Angular velocity
v	Displacement per time step *h*
θ	Rotational angle per unit time step *h*
$\boldsymbol{\tau}$	Torque
P	Momentum
P_n	Geometrical centre of object *n*
u	Normalized time
S	Separating axis
s	Slope
$\mathbf{n}$	Normal vector
f	Impulse force factor
k	Spring stiffness *(N/m)*
T	Periodic time (s)
C	Damping coefficient *(N.s/m)*

α, β, γ	Angles of triangle
a, b, c	Sides of triangle; normal vector components
A	Flexion angle (degrees)
V	Voltage (v)

CHAPTER 1

INTRODUCTION

1.1 Introduction

Nowadays, orthopaedic device development has received considerable attention from public health organizations, the orthopaedics industry and research institutes. The aim in this is to achieve better solutions for problems associated with human joints and hence better joint mobility for those with such problems. The need for such solutions is notably on a steady increase, due to the trend of an ageing population.

Orthopaedic markets have an extensive range of prosthetic joint replacements for restoring joint performance. However, there are many challenges and concerns about prosthetic joints, such as their material related limitations, design and surgical procedures to install them. What we are mainly aiming at here is to develop tools and means whereby orthopaedic personnel can make use of to maintain the joint kinematics as natural as possible, since most available prosthetic implants may influence the natural behaviour of the joint, which in turn directly affects its performance and the failure of the implant.

The accurate modelling of the human joint has received progressively increasing attention from orthopaedic industries. The search for better solutions in human joint diagnosis is a challenge in problems with human joint articulation. Human joints are much more sophisticated than basic idealized engineering joints with a restricted degree of freedom (DOF). Many recent researchers have modelled joint articulation in terms of standard mechanical joints; however, this is not a realistic approach. More reliable models of human joint articulation consider the surface geometry and contact surfaces while allowing 6 DOF for the articulated joint. Such models make the joint kinematics difficult to analyse. Although the main findings in this research concern the modelling and simulation of human joints, modelling and understanding joint kinematics is the key solution for joint associated problems, such as laxity.

Even the relatively recent literature presents human joint prostheses as idealized engineering joints. The articular joint naturally does more complex motion than basic rotation, for instance, the movement of the forearm around a fixed axis. Modelling a joint in terms of meshed surfaces rolling/sliding over each other poses many problems, not always but often relating to the complexity of the surfaces. Joint articulation driven by the contact surfaces makes the joint kinematics challenging to investigate. For example, since the observed elbow joint kinematics is naturally combined with that of the shoulder joint, it is difficult to investigate its kinematics in vivo conditions. With the intention to design artificial joint replacement which can restore the natural mobility of the joint, the kinematics measurements needs to be precisely investigated by allowing 6 DOF.

In reality, although human joints are kinematically unconstrained to allow 6 DOF between articulating bones, many available commercial human joint modellers treat the joints as standard mechanical joint, such as hinge or spherical joints. Joint articulation is effected by bone surfaces, tendons, ligaments and muscles which surrounding the joint tightly. Concentrating on the main elements of the joint complex in the presented model may help to answer many questions regarding the biomechanics of joints and their abnormal functionality. An eminent problem with the human joint is joint laxity, which is, in general, due to looseness or slackness in the ligaments, tendons and soft tissue surrounding the joint. Laxity may also be due to a change in the surface geometry as a result of injury or prosthesis implantation. To diagnose laxity diagnosis in in vivo conditions, several tools and techniques are available in the market; however, their main challenges are accuracy, practicability and cost. Most of the challenges can be met by a proposed simpler way of analysing the elbow joint, based on digital image analysis. The developed image based system is a minimally invasive method which can be used to investigate joint kinematics and laxity. Simplicity, practicability and accuracy are the main advantages of the proposed system. This system uses markers positions to analyse the joint kinematics and thus estimate the extent of abnormal laxity.

1.2 Background of Physics Based Modelling

The substantial increase in the computational power of computers has pushed computer modelling and simulation to higher levels of speed, stability and realism. This proportional growth in digital processing tools has provided limitless solutions for modelling virtual reality environments. Most virtual reality applications have been industrialized for video games and entertainment purposes. Lately, video games have become a major attraction for many developers, given the expanding international markets. Their multi-billion industry has brought the virtual environment of video games and physics engines to superior levels of sophistication and verisimilitude. Physics based modelling and simulation has mainly been developed to allow game engines and animation applications to achieve realistic simulation in the virtual environment. As end users have demanded better accuracy and realism in simulations, physics based simulation has found its way into game engines. The introduction of game physics into game engines has helped the game developers to achieve better simulation with easier modelling. The introduction of the laws of physics into computer simulation has been the key to obtain realistic motion behaviour, while the computer processor has taken care of the required computations. The journey of physical modelling and simulation started with rigid bodies, then articulated bodies, deformable objects and some other types in turn. The three dimensional physics-based modelling starts by defining the geometry of the objects to be involved in the simulation, the objects' physical properties (i.e. mass, density, stiffness, etc.), constraints (i.e. hinge joint, fixed points, etc.) and the environmentally related characteristics (i.e. gravity, other forces, etc.). Simulating such systems of interacting bodies is not easy. The balance between accuracy, computational processing time and realism has to be carefully handled. Simplifying the problem is a choice which will enhance the computational time but should be made with caution, for the choice represents a compromise between accuracy and effectiveness.

The investigation carried out in this project involves the study and analysis of human joints using a physics engine and game authoring platform and validating this analysis using an image based system developed for the investigation of joint kinematics. In addition, the simulation results are validated by comparative studies which involve published materials, Musculoskeletal Joint Modeller software (MJM) and experiments performed at the Brunel Orthopaedic Research and Learning Centre. A perfect match for the simulation results with other methods and literature materials is not possible. However, comparable joint kinematics and joint behaviour will be sufficient to verify the suitability of the physics engine and gaming platform in orthopaedic applications.

1.3 Research Objectives

The objectives of the current research are addressed here:

- To evaluate the capabilities of a physics engine such as PhysX and gaming platforms for modelling diarthrodial joints.
- To create a model of the articular human joint which is capable of dealing with the natural 6 degrees of freedom, where the joint articulation is driven only by contact geometry and the stiffness of the surrounding tissues.
- To establish an integrated environment for manufacturing custom-made implants for patients based on a simulation analysis of the articular joint.
- To develop a minimally invasive system to examine joint disorders experimentally with image based analysis.

1.4 Research Significance

The contributions introduced in this research will have a significant impact on the modelling and simulation of articular joints. The novel implementation of game physics in orthopaedic application will overcome a number of existing challenges faced by other joint modelling software packages, challenges such as contact modelling, muscle wrapping, stability of the simulation environment and cost effectiveness. This research has focused on the local mobility of the musculoskeletal joint with its natural 6 degrees of freedom (DOF). The simulation is performed at an interactive speed, which is not available with other modellers. The evaluation of game physics and its appropriateness for implementation in orthopaedic application shows it to be a promising system with an extended range of applications. In addition, the physics based virtual system will be used to analyse in the proposed integrated manufacturing environment for producing patient-specific implants by rapid prototyping. This integrated environment is expected to be a natural extension of the proposed physics based modelling approach.

1.5 Thesis Structure

The work carried out in this research is presented in nine chapters, followed by references and appendices. An overview of each chapter is presented here.

Chapter 1: Introduction

In this chapter a general background to the research problem, the need, the research objectives, significance and thesis structure are presented

Chapter 2: Literature Review

The literature chapter serves mainly to present the related work of other researchers. This chapter is divided into sections and subsections to categorize the main issues related to this project.

Chapter 3: Physics Based Modelling

This chapter illustrates the method used in developing the physics based model and the development of the physics based software by implementing gaming platforms and physics engine. This chapter also includes a preliminary validation of game and physics engines by virtual experiments.

Chapter 4: The Development of Musculoskeletal Model

In this chapter, the framework for modelling and simulating the musculoskeletal structure on the gaming platform (DX Studio) is examined in detail.

Chapter 5: Simulation Results

This chapter presents the results obtained from a physics based simulation of the elbow joint as a case study.

Chapter 6: The Development of a Novel Image Based Elbow Laxity Measurement System

This chapter presents the image based system developed mainly for the purpose of validating the physics based modelling and simulation. The software and the hardware development of the image based system are illustrated in detail, as is the system calibration.

Chapter 7: Comparative Evaluation of the Simulation Results

This chapter discusses and validates the results obtained from the physics based modelling and simulation software. The validation process was carried out with another modeller, the Musculoskeletal Joint Modeller, together with experimental results obtained from the image based system and Stewart parallel platform at the Brunel Orthopaedic Research and Learning Centre. In addition the obtained results were compared with other published materials.

Chapter 8: Manufacturing of Custom-Made implant Based Physical Simulation

A framework is proposed here for an integrated virtual environment for modelling the simulation, testing, design assessment and manufacturing of custom made implants.

Chapter 9: Discussion, Conclusion and the Future work

This chapter discusses further the proposed modelling environment, the physics based modelling, the associated problems, the limitations and reliability of the image based system, going on to discuss its accuracy and sources of possible error. Finally, concluding remarks about the major contribution of this research are presented.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Many researchers have considered human joint modelling in their pioneering studies. Although these studies aim at dynamic joint modelling, each has looked at the problem from a different standpoint. For some, the skeleton was the priority while for others it was muscle modelling. The area of the application can vary. Some models were created for entertainment games and animations (Chadwick et al., 1989), for a virtual reality environment (Roehl, 1998), for human motion animation in athletic games (Hodgins et al., 1995) or for many other purposes. In addition, graphics-based kinematics and dynamics musculoskeletal modelling techniques have been applied through commercial software such as LifeMOD (LifeMOD™, 2008), Anybody (Damsgaard et al., 2006), VIMS (Chao et al., 2007) and SIMM by Delp et al. (1990). Famous commercial software programs such as ADAMS (MSCSoftware) has been developed in the early 1970s. LifeMOD began as a plug-in to ADAMS for the advanced modelling of musculoskeletal motion. The VIMS software developed by Chao et al. (2007) was developed mainly to understand complex musculoskeletal problems. It is widely used in the planning stage of total joint replacement and joint simulation for rehabilitation. A graphics based software known as Musculoskeletal Modelling in Simulink (MMS) has been developed by Davoodi and Loeb (2002). MMS provides a rewarding framework for prosthesis virtual prototyping, beside its capacity for modelling and controlling musculoskeletal motion. Esat and Ozada

(2010) have developed Musculoskeletal Joint Modeller software (MJM), which is the most recent development capable of modelling human articular joint with 6 DOF.

2.1.1 Multi body Modelling Approaches

a) Actual Physical model

Physical modelling may be as simple as constructing an actual musculoskeletal structure which contains the skeletal tissues (bones, ligaments, muscle-tendons and cartilage). The physical product is usually manufactured from materials such as wood, wire or epoxy-plastic resin. The actual medical dimensions of the produced model are acquired from the original skeletal structure. As an example, a physical model of the human wrist in vitro was built by Jacob et al (1992). Ligaments were modelled as stout strings mounted on their particular locations which were determined as the joint was segmented. Silicon-latex moulds were chosen to create epoxy-resin casts by which the original form of the cartilage surfaces was preserved. Ignoring the material properties is, however, very common with this modelling technique. Physical modelling is generally used for anatomical purposes and basic diarthrodial joint touchable representation. Recent studies have focused on Rapid Prototyping (Abdel-Malek et al., 2007) implementing CAD and CAM (Andrea et al., 2004) systems in order to produce more precise models within an integrated environment (S. Singare, 2010, McGurk et al., 1997).

b) Analytical model

Analytical methods in joint modelling have wide capability. The analytical modelling can be used in computer simulation, joint motion evaluation and graphical rendering. However, its accuracy and realistic behaviour depend upon the assumptions and user input parameters. In analytical modelling, the objects are usually considered to be within a standard defined geometry. For example, regarding the knee and the elbow joints, a standard hinge joint with restricted motion for only 1 DOF is used. For the hip and shoulder, a ball and socket with a maximum 3 DOF is commonly used. Although this restriction in DOF makes an easier approach for modelling and simulation, it has very limited interest for realistic computer simulation and orthopaedic applications.

c) Rigid body model

As the joint consists mainly of bones, the interactions of the involved bones can be used to model the joint without any attention to any possible deformation. Soft tissues such as tendons, cartilage and ligaments may be modelled as springs or rigid shells (Delp and Loan, 2000, Sirkett et al., 2004). The deformation and wrapping of soft tissue are usually neglected, since in rigid body modelling interpenetrations are not allowed. The model characteristics here are based on the data acquired about the bone surfaces and anatomical background. Blankevoort et al. (1991) used medical images and data of the joint to construct a rigid body model which can predict motion. Although this approach seems very approximate, it does have the capacity to provide the full 6 DOF for articulated joints, with real time simulation.

d) Deformable models

More advanced modelling approaches accommodate deformable bodies as well as rigid bodies for articular joints. For example, bones are considered rigid objects, while soft tissues such as ligaments, tendons, muscles and cartilages are modelled as deformable. Modelling techniques based on deformable and rigid bodies are more accurate and realistic; however, they are computationally intensive.

The multi-body modelling of diarthrodial joints can be expressed in several ways. If treated the human joint as idealized joint, it can be defined as a linkage mechanism. The type of connecting joint in a 3D environment can be revolute, spherical, universal, translational or planar. It is essential to be accurate about the type of joint because it constrains motion by means of its degrees of freedom. Understanding joint kinematics is an important aspect of multi body joint modelling for kinematic and dynamic analyses and control.

2.2 Kinematics of Articular Joints

Kinematics by definition is concerned only with body motion and has no interest in what causes this motion. In articulated joints, the kinematics is mainly used to understand and study motion. The kinematics of the joint can represent the number of DOF. For an articular joint, the DOF is what indicates the possibility and range of the movement available to the joint. DOF is a term used in mechanics to describe the number of independent displacements and rotations which identify the placement and orientation of a specific body. In three-dimensional space an unconstrained the body provided with three rotational angles and three translational axes, making a total of 6 DOF. In general, most articular joint modellers treat the skeletal joint as a standard mechanical joint, as in Figure 2.1. In such modelling, the DOF is restricted by the joint kinematics. For example, a hinge joint can revolute only around a fixed axis of rotation and thus it has only 1 DOF, while a spherical joint has 3 DOF and planar joints have 2 DOF (Figure 2.1). So far, recent research has shown that almost all the kinematic mobility representations associated with skeletal joints are actually taken from standard joints as outlined above. Should the skeletal joint not be treated as a standard joint type, it will not be considered as a bilaterally constrained joint; however, the mobility of this joint is also called a unilaterally contact based constraint. In modelling approaches based on joint kinematics, articulation is primarily based on the geometry, without the need to involve related forces. Moreover, when describing the particular kinematic movement for a system, the initial decision should always be whether or not the system is made up of standard joints.

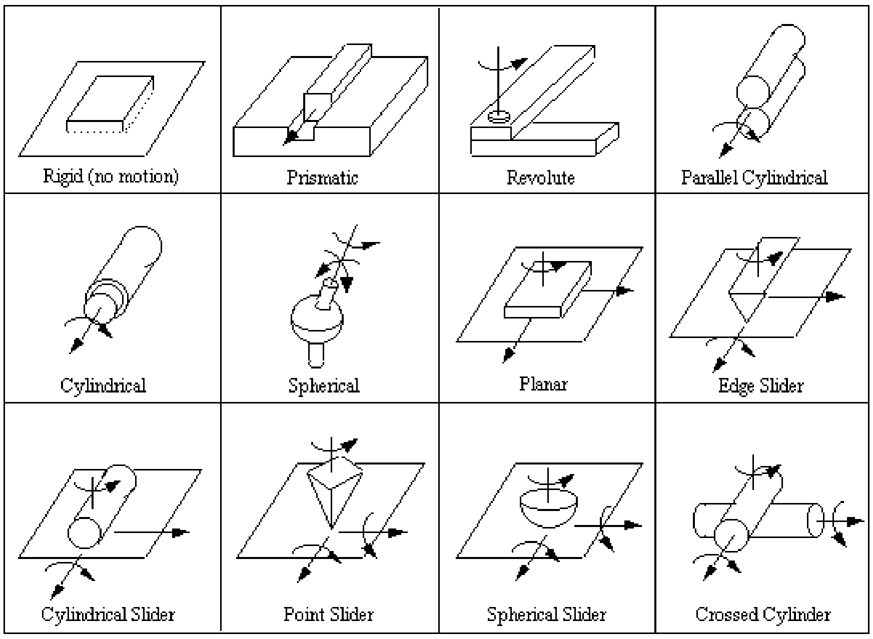


Figure 2.1: Standard mechanical joints

In skeletal human joints, the mobility of the articular joints may be classified under three headings to show the different articulation kinematics (Saladin, 2010).

- Synarthroses joint: No considerable movement, as in the skull.

- Amphiarthroses joint: Slight movement is allowed due to the covering fibrocartilage on the bone surfaces and the ligaments surrounding the joint, an example being the vertebral column.
- Diarthrodial joint: Also known as the synovial joint, this joint is movable with higher mobility range. Typical diarthrodial joints the contact surfaces of each bone are covered with an articular hyaline cartilage and an outer fibrous capsule. The diarthrodial joint is the most frequently used type of articular joint modelling.

With regard to diarthrodial joint modelling, the following types of joint can also model it:

- Planar: a gliding movement between concave and convex surfaces of bones
- Uniaxial: Only a rotation motion about a single axis, as with a hinge joint
- Biaxial: one rotational and one translational movement
- Multi-Axial: multiple rotation and multiple translational movements.

System coordination is another important issue to attend in modelling articular bodies. For musculoskeletal structures, several coordinate systems are used to signify orientation, location and translation (Zatsiorsky, 1998). For example, a fixed reference frame is used to describe joint motion. An intersection between the vertical, frontal and transverse planes is also used to evaluate the relative positions of the joint. Another popular coordinate system based on anatomical landmarks, such as principal axes of local inertia and centre of mass, is often employed.

Forward and inverse kinematic analyses are two approaches used for describing body motion. In forward kinematics, the main concern is to find the location and orientation of the articulated bodies, while the inverse kinematics it is the reverse, since the final position is already known. A very well-known method developed by Denavit and Hartenberg (1955) is commonly used in forward kinematic analysis and still referenced in the studies of articulated bodies. In the work of Denavit and Hartenburg, the translation and rotation of articular joint components is described by setting up a transformation matrix. Transformation matrices are based on a coordinate system for each element involved in the articular joint. Conversely, where the end position and orientation of a given body are well known, as in inverse kinematics, numerous methods are used to obtain a tuned and optimized movement and avoid undesired joint moves or link positions, such as decoupling, iterative numerical methods and inverse transformation (Jazar, 2010).

It is challenging to investigate how the diarthrodial joint might be modelled. A better understanding of joint kinematics is essential for accurate modelling and several approaches to it, with varying levels of complexity, may be taken. A human joint such as the elbow joint used to be modelled as a simple hinge joint with a single DOF, or a ball-and-socket with 3 DOF for hip or shoulder. Among the very first 3D kinematics ways of modelling human joints, Engin and Tumer (1989) presented a kinematics model of the shoulder complex using standard joints with 3 DOF. The main objective was to construct a mechanism of the joint with a sinus rotational range of motion. An accurate modelling approach may require more realistic description of the articular joint, including bone surfaces, ligaments, muscle tendons and cartilage. In 1998, a shoulder complex was introduced with specific medical data for bones and soft tissues (Maurel, 1998). In this model, the joints were handled as idealized joints with an inverse kinematics driving algorithm. The deformation of soft tissues is not permitted during joint motion. Maurel and Thalmann (2000) also discussed the ROM for the shoulder joint. They ignored the translation motion, given the relatively small translation motion compared with the range of motion. Many other comprehensive kinematics based models of a standard joint with a fixed centre of rotation have been developed in this area, such as SANTOS, which uses Denavit-Hartenberg method (Abdel-Malek et al., 2007). An upper limb model was developed by Rab et al., (2002) which employed a 3D video captured with skin reflective markers.

Although there are slight local translations of the centre of rotation (COR) during joint motion, this translation is often not considered and is replaced by a fixed centre of rotation, yielding a simplified solution. Eliminating the local translation of COR may be valid only when the main concern is with gross body motion for the purpose of medical diagnosis, for example, a diagnosis of general functionality, estimating the range of motion and rehabilitation. Many researchers consider daily activities in their studies of joint disorders (Cooper et al., 1993, Magermans et al., 2005, Petuskey et al., 2007). Consequently, many approaches have been developed to investigate experimentally the kinematics of the upper limb. The investigation of the relative motion between two body segments in an experimental procedure based on connecting spatial linkage mechanism has also been developed (Kinzel et al., 1972). More advanced levels of joint modelling involve contact surface geometry to denote more accurate and natural joint motion.

2.2.1 Centre of Rotation (COR)

A body which entirely undergoes absolute translation, the instant or instantaneous centre of rotation is positioned at infinity. This instant centre of rotation follows the same path, since the body is moving. Whenever such linear translation does not exist and the rigid body simply rotates around a stationary point, this point is considered as COR for particular instant. In biomechanical studies of articular joints in the lower and upper extremities, the centre of rotation has received considerable attention in the investigation of joint functionality; for instance, in analysing the knee, elbow, ankle and shoulder joints. Such understanding helps in developing artificial joint implants and prostheses for articular joints such as knee, elbow and shoulder.

Earlier studies have modelled human joints as if they had a fixed COR. Such an assumption makes the joint easier to handle for simulation purposes. However, estimating the COR for the diarthrodial joint is a challenge, because the COR is moving as the joint articulates. Ehrig et al. (2006) presented different methods for estimating the COR of ball joints. Sphere fit methods and transformation techniques were also studied and the results compared. For all approaches, it was noted that the Root Mean Square error (RMS) increases exponentially when the range of motion decreases (Katsuaki et al., 1997). This study also presents a technique called the “Symmetrical Centre of Rotation Estimation” or SCoRE by Katsuaki et al., (1997). This method considers a moving COR and can find even the smallest errors. Gamage and Lasenby (2002) present new least square solution for the estimation of the COR and the AOR. The method proposed in this study does not assume any strict rigidity of the moving body. This paper estimates the stationary COR of a moving body, hence obviating the translation of the COR. To estimate the average COR, Chang and Pollard (2007) propose a constrained least squares optimization technique. The results are improved by using a normalization scheme. These writers concentrate on the relative motion of adjacent body segments and achieve better and less time-consuming results than other COR techniques.

2.2.2 Kinematics Measurements and Applications

Numerous techniques and systems are currently employed in the assessment of the kinematics of articular joints. Nowadays, most researchers prefer to employ an image based motion evaluation analysis of kinematics and a kinetic investigation of complex articulation. Generally in this approach reflective markers or identifiable geometry on the articulated bodies are automatically recognized through a sequential image recording system. Throughout the joint motion the coordinates of the artificial markers in the 2D or 3D scene are used to establish the motion parameters and then to analyse the actual motion as an inverse kinematic problem. This concept was initiated in the 1960s; it led eventually to the development and commercialization of several tracking systems. The first automated tracking system was developed in the 1970s by the Vicon Company. Later on, substantial research produced more advanced tracking systems, in particular once the rapid development of digital imaging began. As human joint motion analysis has become a useful tool in biomechanical studies, researchers from the American Sports Medicine Institute (ASMI, 2007) have used a digital motion analysis system developed by the Motion Analysis Corporation to investigate joint kinematics such as angles and velocities, as well as joint dynamics such as forces and torques, but this system is mainly oriented towards sports activities and motion simulation. Most of the human motion tracking studies are intended mainly to simulate virtual reality and gross body motion (Dariush, 2003, Tao et al., 2007). A kinematic analysis technique using an electro-magnetic motion monitoring system has also been used to assess the kinematics of the elbow joint in passive motion (Bottlang et al., 2000). Another method of analysing 3D motion uses ProReflex motion capture camera system (Qualisys, 2010). This system consists of a sophisticated optoelectronic camera system which generates clean and precise three-dimensional data files. The analysis of the generated data is performed with a software program developed from MATLAB (Mathworks).

A common method for measuring the range of motion (ROM) of human limbs in various static positions uses an instrument called a Goniometer. This device is primarily based on protractor and gravity concepts (Figure 2.2).

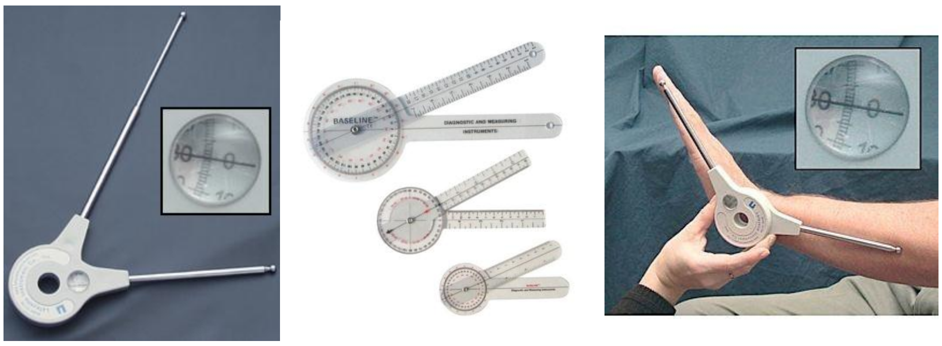


Figure 2.2: Varieties of types and sizes of the Goniometer

An Electro-goniometer is a yet another instrument meant for angle measurement; it is made up of a potentiometer positioned at the middle of the joint along with a pair of extensions fastened to the body elements which form the joint. The range of movements throughout the inspected joint when attached to the electro-goniometer is usually read via an oscilloscope, recording instrument or a pc. The advantage of working with electro-goniometry is that it keeps track of the motion at the joint which is not noticeable to the examiner. Moreover, it can easily record instant angular displacement in regard to time (Adrian and Cooper, 1995).

Roentgen Stereo-photogrammetric analysis is another remarkably precise method for evaluating the 3D movement of prostheses. The method is used to assess the movement regarding the ulna and humerus during valgus stress and within the radius and humerus throughout the maximum pronation of the forearm. This investigation can describe a joint's motion after a picky transection of the MCL (medial collateral ligament) complex (Eygendaal et al., 2002). It has proved to be an accurate way of determining the micro-motion of prosthetic implant with respect to the adjacent bone. This testing system is employed to evaluate the 3D movements associated with bone components. Eygendaal et al. (2002) in their research made use of three small Plexiglas disks (Medis) and used a threaded metal shaft made up of eight tantalum beads as markers for specimens of bony components. The three Plexiglas disks were fixed to the proximal-radial, the coronoid-process and the medial-epicondyle of the ulnar bone. The test setup was made up of a pair of Roentgen tubes positioned over the Plexiglas calibration stand along with x-ray cassette inserted underneath it (Figure 2.3).

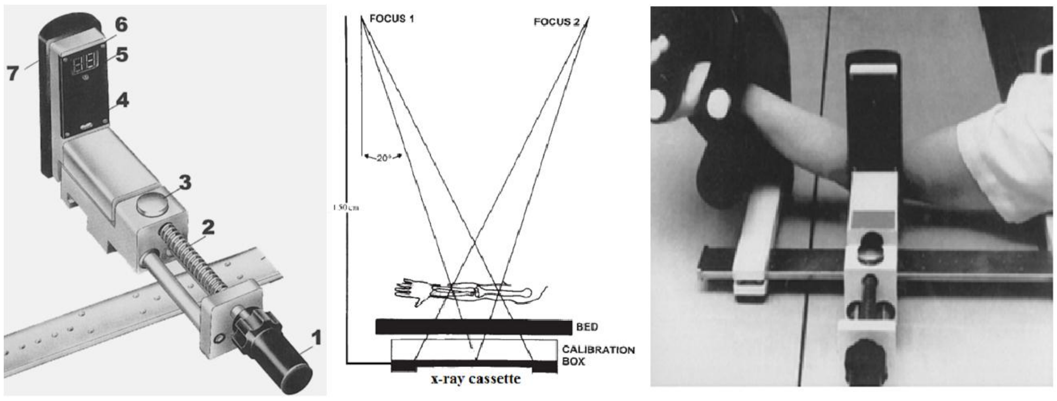


Figure 2.3: Experimental setup (Eygendaal et al., 2002)

The detailed translations were computed by using the Roentgen stereo photogrammetric analysis software program provided by Medis (Medis) which systematically recognizes markers and provides precise measurements for digitized radiographs.

Furthermore, electromagnetic motion tracking systems appear to be useful in the kinematics analysis of the human elbow joint (An et al., 1988, King et al., 1993, Pomianowski et al., 2001, Tanaka et al., 1998). Many of these medical studies sought to compare particular kinematic factors before and directly after complete joint arthroplasties. The first three-dimensional description of elbow joint kinematics based on electromagnetic motion tracking data were carried out by Tanaka et al. (1998). The Eulerian angles based analysis presents a comprehensive explanation of the actual joint angles. Simultaneously, the joint naturally translates along with its rotation but the translation motions were not available with an Eulerian angle based analysis.

Another motion analysis technique is based on "Inertial and Magnetic Measurement Systems" or IMMS, which are available commercially in the market (MicroStrain, Xsens, InterSense Inc). IMMS is made up of a number of sensing units, usually by means of light, portable boxes. Every sensing unit combines an inertial measuring device, composed of an accelerometer and a gyroscope, together with a magnetometer. The accelerometer provides the data, while the gyroscope and magnetometer are joined together by means of sensor-fusion algorithms to measure the 3D positioning sensing unit. Provided with this kind of three dimensional orientation, such solutions offer an approach to estimating joint kinematics (Cutti et al., 2008).

2.2.3 Laxity and Laxity Measuring Devices

A number of systems and devices are available to estimate joint laxity in the elbow or glenohumeral joints. Many of these devices were primarily intended to evaluate joint laxity in the knee. Joint laxity is, in general, caused due to looseness or slackness in the muscles, ligaments and soft tissue associated with the joint. This undermines the stiffness and stability of the elbow joint. Stavlas et al. (2007) describe testing the cadaveric upper limbs with their experimental setup; it was found that the measured deformation was below 0.47º. They said that the range for noticeable laxity is not specifically identified by clinicians. Most studies are mainly interested in the laxity of the varus, because it is the most common type of laxity, but valgus laxity is also studied. The transecting of the lateral collateral ligament or LCL, has been performed by Olsen et al. (1996). LCL is the main ligament involved in varus deformation. They found that the role of LCL is very important for joint laxity, which can reach as much as 24.5º with forced varus.

A further investigation related to LCL is introduced by Jensen et al. (2005) considers the varus deformation following excision of the LCL at different flexion angles for the elbow joint. With respect to valgus laxity of the elbow joint, Floris et al. (1998) mentioned in their paper that, while in unusual, “medial elbow instability” often has a traumatic source and is often a consequence of a partial or even total damage of the anterior set of the medial collateral (MCL). They made use of an illustrated experimental setup to conduct their assessments (see Figure 2.4.)

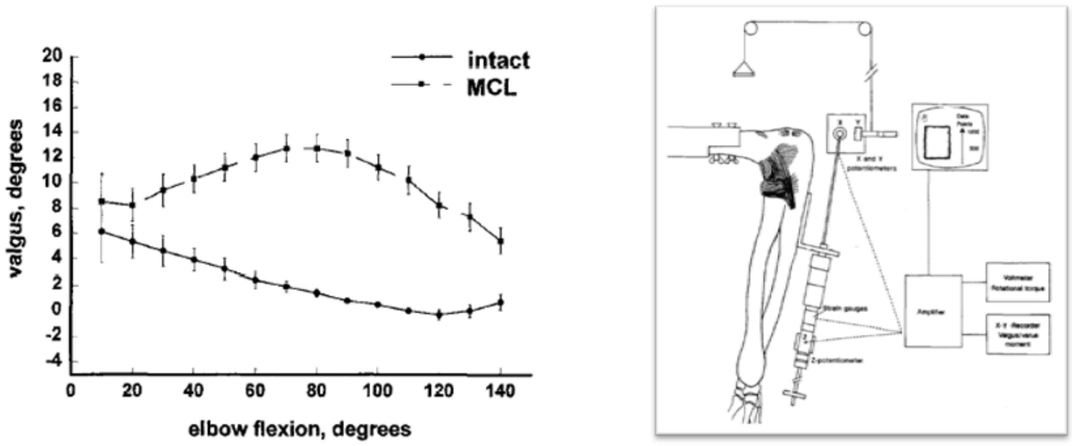


Figure 2.4: Experimental frame and measured valgus laxity (Floris et al., 1998)

The chart in Figure 2.4, produced by Floris et al. (1998), indicates the range of valgus deformation in healthy elbow joints and in severely injured elbow joints. The laxity tests were performed at different flexion angles. This particular type of joint laxity is the one in which we are mainly interested. Sauers et al. (2001) explain that the glenohumeral joint laxity is mostly evaluated via actual physical checking; nevertheless, imprecise measurements of physical check-up may possibly be attributed to different causes, for example, the practical skill of the physician, amount of load or patient positioning. Furthermore, muscular stress and anxiety near the joint may possibly considerably alter the detected magnitude.

Another team of researchers (Hatzel et al., 2006) came up with well-conceived apparatus which measures glenohumeral joint laxity, making use of a KT-2000 knee arthrometer. As already pointed out in almost all articles on this subject, the glenohumeral joint is one which is likely to be injured, in particular in sports activities where moves at great speed or even high momentums are usually needed, along with instantaneous stresses of the joints. Hatzel et al. (2006) thought that although glenohumeral joint permits a large range of motion in different directions, in the course of movement only a very small translation takes place within the humeral head. Moreover, the malfunctioning of the human joint depends on whether increased or decreased translation has affected the humeral head. This is precisely why it is necessary to assess the degree associated with the translation of the humeral head. Their contribution was to modify the existing knee arthrometer (KT-2000) and make use of it for measuring shoulder joint laxity. This arthrometer, to put it simply, includes a pair of sensors which fixed to the main frame. Before each test, these two sensors should be examined to make sure that they are parallel.

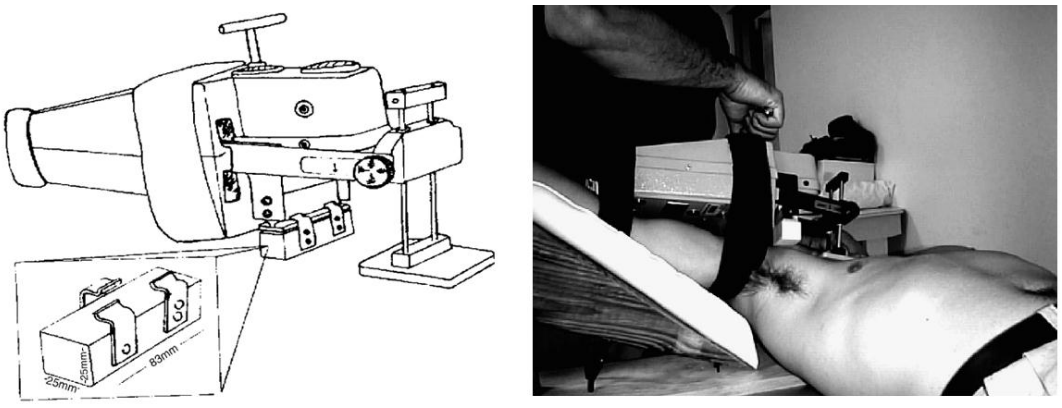


Figure 2.5: KT-1000 and KT-2000 instrumented arthrometer (Hatzel et al., 2006)

2.3 Dynamic modelling of articular joint

As mentioned in the previous section, kinematics is concerned only with motion, without any regard to the causes of this motion. However, to establish a physics based simulation for a multi body system, a kinematics model must be combined if required with some kinetics related aspect such as force, mass, the centre of mass and the centre of inertia. Thus, in dynamic simulation, the study of the motion and the causes of this motion are considered. This coupling is essential in establishing the equations of motion for the system. Considering the rigid bodies only, mass and inertia tensors are required to construct the equation of motion. Mirtich (1996) has presented an algorithm for handling such dynamic equation of motion. The Mirtich algorithm has been mixed with the Gauss Divergence theorem and modified for polyhedral meshes.

Many theories and algorithms exist for solving such equations of motion. For example, Lagrange, the Featherstone algorithm, Newton-Euler and many others are widely recognized methods and employed to handle such dynamic equations of motion. As mentioned by Flores (2008), the Newton-Euler method is the easiest method to implement for the analysis of multi-body dynamic systems.

Beside the substantial increase of computational power over the last few decades, many software packages have already been developed to handle multi-body dynamic simulation. A review of the available software packages is outlined here as it highly relevant to the main work of this thesis. Implementing the computational modelling in dealing with physical problems is important in many applications, such as biomechanical analysis, digital analysis, virtual simulation and many others. For the case of biomechanical modelling and simulation, Finite Element Analysis (FEA) and multibody dynamics are the most primary employed methods. For the case of multi body dynamics, the simulation of rigid as well as deformable objects is usually carried out by means of numerical algorithms and dynamic concepts. One of the pioneering and widely used software applications for multi-body dynamic simulation is the "Automatic Dynamic Analysis of Mechanical Systems" or ADAMS, an earlier version of it is known as DAMN or "Dynamic Analysis of Mechanical Networks"; which was developed at the University of Michigan in 1970 by Chace (1970) and Chace and Korybalski (1970). A further release, called "Dynamic Response of Articulated Machinery" or DRAM was developed in 1971 and improved in 1977 (Chace and Angell, 1977). The initial software forming the MSC.ADAMS was presented by Orlandea (1976). This introduced a three dimensional workspace rather than the two dimensional one of previous versions. The code in this software program is primarily based on a sort of Lagrange equation to solve the constrained multi-body problem. Besides the rigid body problems, this software package also provides solutions with regard to human body modelling, vehicle dynamics and the dynamics of flexible bodies using finite element dynamics and many other add-ons.

In addition, a very well-known commercial plugin module for ADAMS called LifeMOD™ for the dynamic simulation and analysis of the musculoskeletal system was developed (LifeModeler, 2010). This modeller is a very popular software program used in many areas, such as sports, ergonomics and orthopaedic applications. LifeMOD offers a virtual simulation environment, simple user interface, anthropomorphic data source, inverse and forward dynamics, life-like human movement, muscle modelling, automatic joint generation and effective post processing; however, its main downside is that it models the joint as a standard engineering joint. This type of model reduces the DOF and introduces a fixed centre of rotation. However, unilateral contact constraints are also employed for specific geometries and contact conditions for a contact driven articulation; consequently, the full coordinate system is reduced because of the kinematic restrictions when unilateral contact is enabled. DADS or “Dynamic Analysis and Design Software” (Haug, 1992), is another commercial software program for general purpose multi-body problems which is comparable with ADAMS. This software is built in accordance with the Newton-Euler method with explicit constraints (Haug, 1992). A more specific software package for musculoskeletal system modelling, such as SIMM “Software for the Interactive Musculoskeletal System” was introduced in the early 1990s (Delp et al., 1990, Delp and Loan, 1995, Delp and Loan, 2000). SIMM has a graphic interactive environment and it can be used to analyse lower and upper extremities joint mechanics, it requires a specific joint kinematics or employs a standard type of joint such as ball and socket for the shoulder or the hip. An additional “Musculoskeletal Modelling Software” or MMS for creating and simulating models for articular joints was introduced by Davoodi and Loeb (2002). MMS software is able to predict a joint’s kinematics under different control conditions. It can also be used to examine and analyse the prosthesis in a virtual environment before implantation. More recent software for virtual human modelling and simulation known as the Virtual Interactive Musculoskeletal System (VIMS) was introduced by Chao (2003). This software is mainly designed as a biomechanical analysis tool with graphic modelling, which analyzes the musculoskeletal structure under static, kinematic and kinetic conditions. It also incorporates a model library and a database of medical data for customized analysis and building more specific models which can be used in joint reconstruction, injury management, re-habilitation and orthopaedic device development (Chao et al., 2007). Damsgaard et al. (2006, 2001) have developed what they call the AnyBody modelling software (AnyBody, 2010). The analysis of the musculoskeletal system in this software is carried out as a rigid body system. AnyBody features inverse dynamics, customized loading conditions and motion description, consequently controlling the dynamic simulation over the assigned activity. Although there are several software packages available for the dynamic modelling of human joints, none of them can provide a platform for analyzing the local mobility of an articular joint with 6 DOF in an interactive simulation environment.

Finite Element Analysis FEA is another approach used for modelling musculoskeletal systems, primarily based on continuum mechanics. A variety of commercial and open source software packages for general engineering applications is also available, such as ABAQUS, ANSYS, ADINA, Nastran, ParaFEM and many other software packages. Although most of these software packages are general in their application, some software programs, such as MADYMO and RECURDYN implement multi-body dynamics in their structure. More specialized software for biomechanical analysis has been named FEBio (FEBio). This software is an open source application which facilitates sliding contact, nonlinear static and dynamic analysis and large three dimensional deformations (Bonet and Wood, 2008). Compared with the wide range of FEA applications for musculoskeletal system modelling in many engineering fields, FEA seems to have had limited implementation, in particular in terms of muscle and soft tissue interaction modelling, which generates very complicated equations of motion which are extremely hard, if not impossible, to solve. These make the FEA not a favorite method with regard to musculoskeletal system modelling, as a result of the complexity of dealing with continuum mechanics in articulated bodies.

2.4 Physically Based Modelling

Computers and data processors are now considered essential tools in modelling and simulation. As computational capability improves, users and applications look for higher degrees of realism in these areas. This trend is obvious with computer graphics, in which more complex geometric shapes of physical objects are nowadays modelled in the context of complicated physical environments. Modelling and simulating these, based on physics, have demonstrated the possibilities of generating automated action and motion which are exact replicas of their genuine equivalents (Baraff, 1993). Since all physical objects in this world follow the laws of physics, it is logical to bring in a physical consideration of action. Research into modelling based on physics derived from studying the movements of rigid bodies. Eventually, it expanded sufficiently to contain the motions of articulated objects and deformable bodies, along with other computational models (Lee, 2001). In a physics-based framework, the user may set the physical properties of interacting objects, constrain the range of motion of each object and specify shapes inside a three dimensional environment. This simulation environment requires a sophisticated virtual system if the accuracy of the output is important and not to be reduced for the sake of increased computational speed. One way of simplifying the problem is preferably to use hard physical objects which do not deform on collision. Regarding these, a number of studies discussing physics-based simulation have been, some of which are outlined below.

Starting with Newton's law of motion, the simulation of a simple moving object X can easily be described by Newton's law of motion $f = ma$, where f is the external force acting on a body of mass m and causing an acceleration of a. The same equation can be considered as the second derivate of location x, a=$x" = \frac{f}{m}$ Defining the initial conditions, the location and velocity of the moving object X can be determined at any given time. The simulation is more difficult when moving objects interact with each other. Solving such interaction problems needs further investigations regarding the place and time of the collision. In general, resolving the collision problem is possible only if the exact time and place of the collision are known; otherwise, objects can penetrate one another without a proper collision response being expected. Object surface contact is also important to study as collision and contact are related.

The collision of rigid bodies assumes that the bodies are not deformed during the collision. In this situation, the collision response is obtained by applying corresponding impulses to the colliding objects. These impulse forces change the respective velocities of the objects as soon as collision takes place. To achieve the realistic behaviour in a collision of non-deformable bodies, one must consider the bouncing coefficient or what is called the coefficient of restitution. This denotes how much energy is lost during the collision. The restitution value is between 1 for perfectly elastic materials with no loss of energy and 0 for the complete resting of materials after collision.

In virtual reality simulation systems, the modelling of the friction force is very important for capturing realistic motion behaviour. Modelling static friction is in some ways tricky to handle mathematically, because it introduces discontinuity. With static friction the object remains static only until the applied force overcomes the static friction force and then the friction force is reduced to dynamic friction, which is much easier to model. This is why most physics based simulators do not model static friction (Baraff, 1991).

Modelling the geometrical shapes is important in collision detection. There are several modelling methods which can do this, classified as polygonal or non-polygonal. The polygonal model is a method of representing a shape by patches. The higher the number of patches the higher the resolution and therefore the longer the processing time (Armstrong and Green, 1985). Non-polygonal is used for parametric surfaces; it can present a very smooth surface and is generally easier to handle than the polygonal, but can at times be difficult and even very complicated.

The simulation results are affected by the precision of the model, for not everything can be modelled and calculation errors arise from the fact that the virtual environment operates discretely while reality itself is continuous (Baraff, 1996).

The physical simulation of elasticity is of interest to Terzopoulos et al (1987). They suggest that the physical simulation of elasticity is very important for creating an interesting animation. In this case, models must comply with the law of elasticity, or we can say that the simulation must be performed according to Newtonian laws. In addition, a physics-based modelling approach which assists model creation and can combine complex geometries and realistic motion was later introduced by Terzopoulos and Witkin (1988). They developed a hybrid formulation to combine rigid and non-rigid dynamics leading to model free form elastic deformation by exploiting a relatively simple linear theory. The main objective in introducing a physics based model was to replace the key frame animation with automatic model simulation.

Redon and et al. (2002) state that most approaches for rigid bodies simulation are formulated in the contact space. With the Gauss' principles of least constraints, the frictionless dynamics can be formulated in the motion-space, which uses less memory and requires fewer computations. Contact space and motion space formulations are mathematically equivalent but computationally different.

2.5 Contact Modelling

2.5.1 Collision detection

Collision detection simply deals with the four following questions. Will the object collide into something? If yes, then what are the involved bodies? When? And where?

Collision detection is one of the main components in physical based simulation. Realism, efficiency and accuracy are most important aspects of collision detection. A virtual environment is a computer-generated environment. Interactive objects in a virtual environment are expected to give the user the feeling of presence, for example, objects are expected not to pass through other objects freely and to move as expected when pulled, pushed or colliding. Achieving such a virtual environment requires the simulation system to calculate collision detection accurately, but accuracy is not the only issue to consider in developing a collision algorithm. Objects in a virtual environment can be hundreds or thousands in number; such environments demand a fast and interactive collision algorithm to ensure the right computation of the collision detection. Developing such algorithms requires some assumptions and simplification of the objects in the scene. Video games are not the only applications for collision detection. Collision detection can also be found in training and education systems, virtual surgery and robot path control. Simulators based on collision detection are inexpensive to operate because they use the virtual environment for testing and verification.

Before dealing with collision problems, a definition of this may be given as follows;: in three dimensional environmental space, S, and a moving object, O, a small data structure is pre-processed to check if there any intersection between O and S. The answer is provided with each time frame. Then pre-processing the data continues as the objects move. The collision detection problem becomes more complicated when dealing with higher number of objects in the scene. Clearly, this is an undesirable event in all collision detection algorithms. The collision detection problem is easier to handle for approximated bounding volume than for actual surface geometry. If the bounding volume cannot intersect, then the object will not collide. The approximation method is used for complex shapes and is based on the BVH.

Collision detection methods can be categorized in several ways, depending on what is of interest. Some methods are suitable for rigid bodies, while others are more sophisticated and can handle deformable bodies as well. Moreover, some collision detection methods are more suitable than others for different techniques of object modelling. Collision prevention is sometimes required by certain collision detection methods. However, most of the existing methods of collision detection are developed mainly from an optimization problem or from one basic method or more. In this literature, several collision detection algorithms should be mentioned. They are classified according to the principles they are based on.

2.5.2 Native collision detection algorithms

Collision detection is simply described by checking the targeted object geometry for possible interpenetration. A static interference test is used to check such interpenetration. Basic algorithms are divided into two types: distance calculation algorithm and intersection testing.

2.5.2.1 Separating distance calculation

In this algorithm the distance between the specified features (edges, faces, vertices, etc.) of each object is calculated to check the nearest feature for each object. The distance between the objects' features is tracked until a negative distance is achieved, which means that active penetration has taken place and a collision has occurred. Several algorithms which are based on distance calculation are presented by various writers (Dobkin and Kirkpatrick, 1990, Mirtich, 1998, Lin, 1991, Joukhadar et al., 1996).

2.5.2.2 Intersection tests:

Intersection testing can simply indicate the existence of a collision whenever two primitives are overlapped or intersected. With this method, a collision is shown to have occurred whenever two primitives are in intersection. This test can be briefly expressed for intersecting circles named A and B, by the following statement:

$$L_{A\text{-}B} < r_A + r_B$$

When the above condition exists this means the intersection exists.

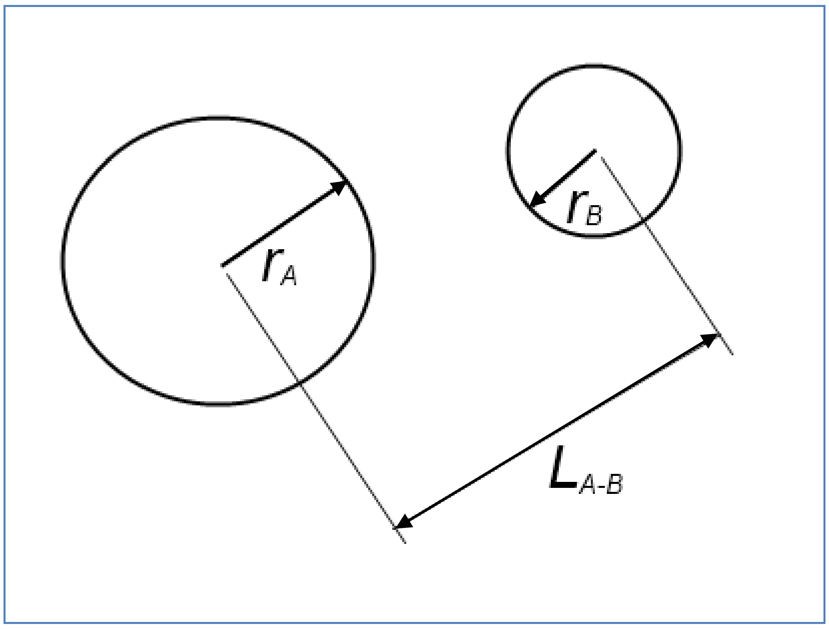


Figure 2.6: Illustration of simple intersection illustration between two circles

The separating axis test proposed by Gottschalk (1996) can be used to test bounding volumes. Since most three dimensional polygonal models are composed of triangle meshes, an effective intersection test for triangles is presented by Muller (1997).

An effective and practical collision detection algorithm may be required to handle such intersection detection for possibly large numbers of objects with fewest computations. In dynamic simulation, a huge number of static interference tests have to be checked for the intersection of primitives. Several approaches to boost collision detection by simplifying the problem are presented here.

2.5.2.3 Bounding Volume Based Algorithms

The collision detection problem is easier to handle for approximated bounding volume than actual surface geometry, if interference calculations must be speeded up. If the bounding volume cannot intersect, then, of course the objects inside these volumes will not collide. Bounding volumes are basically intended to simplify the shapes of the object so as to avoid complex computations. The approximation method is used for abridging complex shapes into more basic shapes and is based on the bounding volume hierarchies called BVH. BVH is basically a tree structure of bounding volumes. Sorting bounding volumes in a tree type structure improves the performance of the collision detection by eliminating unnecessary tests. The structure of the bounding volume hierarchies BVH is built by organizing all the bounding volumes of the geometric objects to form the nodes of the tree. These nodes are then grouped into larger bounding volumes and so on until a single bounding volume for the entire model is attained. When a collision test is performed for one and is negative, then its children are missed too and there is no need to examine them. The construction of bounding volumes hierarchies can be performed in several ways: top-down, which is used by most algorithms; or bottom-up used to produce better trees and insertion methods for a better updates at runtime (Goldsmith, 1987). After the construction of the BVH, the nodes on the tree are checked for any possible overlap. To make sure that there is no overlap, the children of this node are examined too.

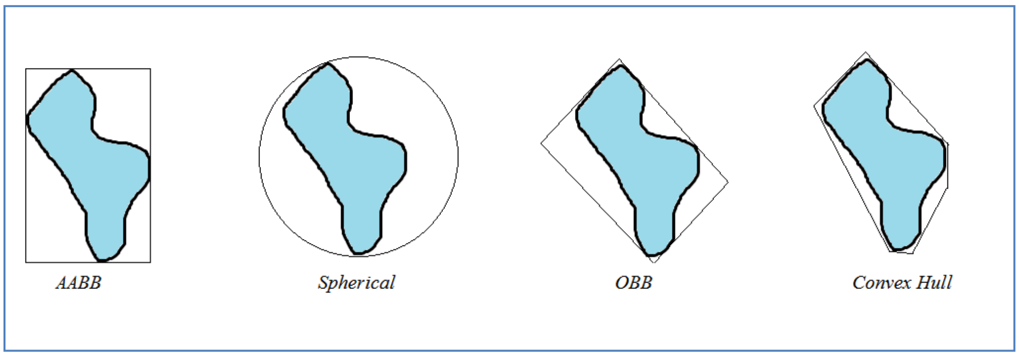


Figure 2.7: The most common types of Bounding Volumes, AABB, Spherical, OBB and Convex Hull

Bounding volumes hierarchies can be applied on deformable bodies as well as rigid bodies. With rigid bodies there is only a possible translation or rotation and bounding volumes are only refitted into the object. However, as the geometry changes with deformable objects, the situation here is more complex than with rigid bodies. An updating process for the BVH tree is required with every new simulation frame. Complete reconstruction of the BVH tree at every frame is avoided by optimization methods, (Mezger et al., 2003). Optimization is also used in the case of fixing the existing BVH tree (Larsson and Möller, 2001). For deformable objects, the update process is the most challenging and many research studies are concerned with it.

A variety of bounding volumes are used by a number of hierarchies which consequently use different tree constructions and update strategies. Therefore, a different simulation environment may prefer different types of BVH from others.

Common examples of BV features Axis-Aligned Bounding Boxes (AABB) and spheres which are primarily picked for their robust overlap check.

Spheres are rotationally invariant and, as regards bounding volumes, they are very quick to update and simple to calculate the distance between them for any chance of overlap. The drawback of this bounding volume is that some objects cannot be closely approximated efficiently (Palmer and Grimsdale, 1995).

An AABB algorithm is considered the simplest algorithm that can be used in the broad stage of the collision detection. Its main advantages are robustness and very fast collision detection. The reason behind this is because it is a rotationally invariant. The disadvantage of the AABB is the wide range of approximation used in representing the object, which is seen as a tight box containing the object. Approximation accuracy depends on the actual shape of the object and its orientation. This is the weak point in simulating complex shapes. Therefore, using AABB for complex shapes may result in a poor simulation.

From its definition, the bounding volume should cover all the geometry at all times. Although AABB is aligned along the global axis system of the virtual scene, the actual object is allowed to rotate due to the environmental interactions. Bounding volumes in AABB can be expressed in two ways. The first is to establish a fixed sized volume which covers the entire object at all times regardless of the orientation changes of the object itself. The second is to make this bounding volume dynamically change its dimensions with every time step to maintain the tightest possible bounding volume. Checking for collisions with AABB is simple because it is axis aligned. The AABBs values are sorted in the x, y and z directions independently. Collisions can only occur if the min/max values of the AABBs overlap over the x, y and z axes. If two or more objects satisfy this condition, it is added to the active contact list (ACL).

Samet and Webber (1988) presented Octree's algorithm. In this algorithm, the object is contained by a volume and is sub-divided into eight octants. Only the sub-divided parts of the octants containing parts of the object are considered as nodes. The benefit of such data structure is its straightforwardness and it can be implemented automatically. Its problem is that the levels of the hierarchy do not fit the underlying object precisely.

The OBB-Tree is another hierarchical method based on arbitrary oriented bounding boxes (Eberly, 2002). It functions better than common AABB, since the orientation of the bounding box may help to gain a tighter fit for the object and so improve the approximation. This type of bounding box is more efficient when the aspect ratios of the shape are higher. According to Gottschalk et al. (1996), the overlap testing for OBB is carried out on fifteen axis projection test. Compared with the sphere test, the OBB-tree is slower to perform and update since it is orientation sensitive.

Another method which uses hierarchies of the convex discrete orientation polytopes bounding volumes is presented by Klosowski et al. (1998). These bounding volumes are also called k-DOPs. The bounding volumes in this method overcome the loose fit of AABB and the slow overlap checking with OBB. Since this method is a generalization of AABB which can be considered as a 6-DOP, it also has a problems with dynamic updates for the nodes.

2.5.2.4 Spatial partitioning representations

Spatial partitioning representations are also considered as a hierarchical bounding volume. In this approach the three dimensional space is subdivided into small volumes or subspaces. Objects occupying the same subspace have possible contact. Octree’s is the most common method used in spatial partitioning (Bandi and Thalmann, 1995, Hamada, 1996). However, several partitioning strategies can subdivide the 3D space into subspaces in a hierarchical way to boost the detection of collision (Klosowski et al., 1998, Garcia-Alonso et al., 1994, Fabio Ganovelli 2000) .

2.5.3 GPU and Image based and methods

In the image-based method, the projection of the object is processed for collision detection. This may enhance the processing time and speed up the computations. A pioneer example for “image based collision detection” is presented by Shinya and Forgue (1991). In their approach, the object projection is rendered wholly in pixel information. Then the z values are read in a comparative manner to the z-buffer to check for any possible overlap. The drawback of this method is not supporting self-collision or non-convex objects and also that only rigid bodies are involved. Other researchers introduced image based algorithms for non-convex objects (Heidelberger et al., 2003) as well as self-collision detection (Heidelberger et al., 2004). Lately, Image based collision detection algorithms have received considerable attention in graphics applications such as CULLIDE which employs a graphics card and calculates a possibly colliding set using visibility issues (Govindaraju et al., 2003). The primary concern associated with image based collision detection is the reliance on image-space resolution. Consequently, missing relatively small polygons may result in a simulation with an inappropriate collision response.

Due to the recent and impressive development in graphics hardware, Graphics Processing Unit (GPU) is capable of supporting additional tasks with its substantial capability, exactly like collision detection. Recently, GPU has had considerable attention in many collision detection algorithms. Such algorithms, which are based on GPU, capture the advantages of the parallel processors of the GPU to boost the computations significantly by moving them from the CPU to the GPU. Collision detection algorithms based on GPU can be divided into two groups: algorithms which are based on the depth of buffer information to compute interference (Govindaraju et al., 2007, Govindaraju et al., 2006, Govindaraju et al., 2005, Govindaraju et al., 2003, Baciu et al., 1999, Shinya and Forgue, 1991); and algorithms which use the fast calculations of distance fields for proximity queries (Kenneth E. Hoff et al., 1999, Kenneth E. Hoff et al., 2001, Heidelberger et al., 2003, Heidelberger et al., 2004, Govindaraju et al., 2005, Govindaraju et al., 2003).

2.5.4 Public Domain Systems

Several public software packages are available for collision detection problems, for example, RAPID (Gottschalk et al., 1996), V-COLLIDE (Hudson et al., 1997), I-COLLIDE (Cohen et al., 1995), SOLID (Bergen), and V-Clip (Mirtich, 1998). Most of these packages are available for polygonal models as well as large simulation environments with quantities of moving bodies. A fair comparison between these available domains is difficult to achieve, since each one performs differently under different simulation attributes. However, a brief description of the well-known RAPID is supplied here.

RAPID “robust and accurate polygon interference detection” is a library that works only with intersecting triangles and is suitable for polygon soups (Gottschalk et al., 1996). RAPID is based on Oriented Bounding Boxes hierarchies (Bobbert and Schenau, 1990) and uses top-down strategy to construct them. Then OBBs undergo overlap tests to detect any possible collision. If a higher levels of OBBs overlap, the lower levels of the overlap pair are tested for further verification. In cases where no overlap exists, the pair has not collided and the algorithm stops. Subsequently, a list of triangle pairs is generated, representing the contact pair for each collision. The main drawback of RAPID is that self-detection is inapplicable.

2.5.5 Multi-Phase Collision Detection

A collision detection system may also be combined with more than one algorithm for optimized and efficient computations. It can be carried out on two phases, and then is called hybrid. Hybrid collision detection is proposed to deal with complex situations. The hybrid approach refers to any collision detection method which is based on two phases, one broad and one narrow. In the broad phase, a rough estimation of the collision is performed. Then more accurate calculations for the collision are made to identify the parts of the object which will be involved in the collision. Thus, the collision detection in the hybrid approach is handled as a multi-phase process. The initial phase, also called the broad, is called mostly for rough estimation and to reduce the pairs of object which cannot possibly collide. Different methods can be used in this phase to achieve this rough collision detection, such as Sweep and Prune, overlapping tables or global-bounding tables. In the refined (narrow) phase of the hybrid collision detection method, more accurate calculation is required to narrow down the intersected regions of objects. The narrow phase of the hybrid method uses several algorithms such as I-Collide, V-Clip or enhanced GJK. Dividing collision detection into two phases is meant to avoid unnecessary calculations and therefore reduce the computational coast.

2.5.6 Collision Response

Collision response is the determination of the resulting behaviour of the colliding objects. Several approaches have been developed to handle the collision response: constraint based (Baraff, 1994), impulse based (Mirtich and Canny, 1995) and penalty methods are the most common methods used in physics based simulations. In contact problems, modelling friction is another issue that directly affects the collision response. Several works discuss the friction modelling and the difficulties associated with it (Baraff, 1991, Baraff, 1994).

2.5.7 Penalty Method

In the penalty method, temporary springs are inserted amongst the colliding sets at contact points. This method is quite simple to implement and to understand. The standard sort of penalty method uses Hook's law $F = -kx,$ where F is the force applied on the colliding objects in the opposite direction of collision. Even though this method presents a solid foundation, in practice it experiences some obstacles with approximations and massive computations, in particular when *k* values (of spring stiffness) are higher and smaller time steps are required to solve the problem with acceptable accuracy. The reason behind this is in accordance with Baraff (1993), in penalty method; the infinite quantities are modelled with finite quantities. The main challenge in implementing this method is to accurately find the proper stiffness constants. Otherwise, the penalty method presents a simple and powerful collision response method which is robust and can be used with a variety of surface conditions.

2.5.8 Constraint based method

In this method, interactions of objects during physical contact are described by constraints. Collision and contact are distinct. The normal relative speed of the colliding set of points is calculated and checked for further actions. Should the normal relative speed be negative, this would mean that the objects are colliding or positive, for separating or zero for resting. Additional correcting force constraints are found to eliminate any external acceleration, these may cause interpenetration. As a result, this method completely restricts interpenetration. However, discontinuity is introduced here because of the simplification provided by rigid bodies only. A number of non-linear equations are to be solved in such situations (Baraff, 1989).

2.5.9 Impulse based method

An approach to dynamic simulation termed ‘impulse based simulation’ is proposed by Mirtich and Canny (1995). The recognized attribute with this technique is the implementation of almost every sort of contact (sliding, rolling, resting and colliding) within a single platform. The approach is simpler and more robust than previously mentioned constraint based methods. In addition to the robustness and simplicity of the proposed method, the simulation accuracy is highly acceptable compared with the experimental results. Impulse based simulation can be briefly described as an effective collision detection scheduling scheme and a complete general process of frictional collision. Mirtich and Canny (1995) concentrate on two issues: accuracy of the physical simulation and the efficiency of the computations. They also compare impulse based method with the constraint based method but in the end they do not suggest replacing one with another, but rather combine the strengths of each. For instance, constraint based method could be used for stress and strain studies during collision, but would be difficult to use for real-time simulation; however, impulse based method can be effective is such situations. With impulse based method, the collision between bodies is analysed only with respect to impulses and not material deformation. The computation of the impulses of the colliding bodies is carried out with some assumptions.

An extended formulation of Baraff (1994) is presented by Katsuaki et al. (1997). This formulation is proposed to simulate the impulsive friction force acting on the colliding objects. The results have also been compared with the Impulse-based method by Mirtich and Canny (1995). A simulation system employing the extended formulation of Baraff (1994) has been developed for simulating the interaction of rigid bodies with impulses, contact force and friction force. A comparison of the three methods is shown in Table 2.1.

Table 2.1: A comparison study between penalty, constraint based and impulse based collision response methods.

	Penalty	Constraint Based	Impulse based
Body types	Rigid, deformable	Rigid, deformable,	Rigid
Principle and integration	Simple	Complex	Medium
Computations	High	Low	Medium
Time steps required	High	Low	Medium
Problems entailed	Stiffness of contact	Changing contact type	Resting contact
Parallel processors	Possible	Difficult	Potential
Accuracy	Depends on time steps	Accurate	Accurate
Verifications	Difficult	Easy	Easy

2.6 Deformation modelling

Deformable bodies have, in general, the possibility of relative internal changes, which are certainly not permitted with rigid bodies. The modelling of deformable bodies could be treated in a number of ways. These methods are usually categorized as physical and non-physical based approaches. Non-physical based modelling approaches are based on pure geometrical deformation of the body; accuracy and realism are not usually a priority since these approaches are mainly employed for computer animation and entertainment application. The other group of methods that is based on physical deformation considers the continuum mechanics which take into consideration external forces, material properties, internal deformation and constraint conditions (Zatsiorsky, 1998). Introducing physically based deformation in graphic applications demands huge computational power, which became possible only a few years ago.

In the current research, more attention should be given to physically based deformation methods because the current case under scrutiny deals with the musculoskeletal system, which demands the highest possible accuracy and realism.

2.6.1 Methods based on physical deformation

In the non-physical modelling (where the physical properties of the objects are not defined), the system has no knowledge of the behaviour of the deformable bodies in the simulation, since everything depends on the user specifications for the expected output. However, in the physically based methods, motion and deformation are controlled by a partial differential equation which must be solved and this creates an extremely complex situation. Moreover substantial computation power is needed to solve such partial differential equations. These obstacles can be overcome by finding an optimized, simplifying model for each specific problem, such as using an effective numerical method for such complex differential equations (Terzopoulos et al., 1987, Saladin, 2010). Commonly used techniques in computer graphics for physically based deformation are found to belong to the finite element method: mass spring systems and the particle system.

2.6.2 Mass-Spring System

Mass spring systems are very widespread in computer graphics. Although this method is a physically based method, it is very efficient at modelling deformable bodies. In the mass spring system, the deformable body is modelled as a collection of discrete masses connected by springs in an organized structure. If the connected springs are excited from their rest position, the structure will exert force at its terminal on the attached masses. Deriving the equation of motion, for each mass we can write the following equation:

$$F = kx + C\dot{x} + m\ddot{x}$$

Applying this equation for the complete set of mass-spring elements in the system will lead to a system of differential equations which can be solved by several algorithms. Damping is also introduced to maintain the stability of the system and ensure realistic behaviour.

2.7 Physics Engine and Software Implementation

Recently, medical applications have received substantial attention in computer graphics modelling and virtual reality simulation software. A number of commercial and non-commercial simulation software programs have already been designed to assist physicians as well as pupils to understand human anatomy in addition to performing operative surgical preparations, training and testing within a digital platform. Such physics-based simulators may also be used for joint disorder diagnosis, kinematic and dynamic analysis, rehabilitation and prosthetic assessment at lowest risk and overall cost with the help of virtual environment simulation.

A physics engine is essentially a software code which generates a simulation of Newtonian physical models within the simulation environment, for instance, rigid and deformable body dynamics with collision detection and response. Game industries and film productions are probably the most active areas for implementing physics engines. However, the concept of the physics engine may be used in general to illustrate any kind of software program used for physical simulation. Working with parameters, such as mass or density, velocity, acceleration, friction and stiffness, they can imitate and predict outcomes according to various circumstances which probably mimic the activities occurring in real life. Physics engines are mainly developed to be implemented in gaming platforms (game engines) for game development, film and animation production and others to enhance the simulation as it imitates a physical scenario and to deliver remarkably convincing games and animated graphics. Physics engines with high accuracy may demand considerably more computing capability in order to calculate highly accurate responses. Hence, they are typically used for scientific research and high standard movie productions. Other physics engines commonly used for interactive computer applications for prompt real-time simulation take advantage of optimized algorithms and simplified calculations to carry out computations just in time to react in an interactive manner. One of the earliest physics engines were developed on an ENIAC computer in 1946 by the Unites States army to calculate approximately the range of various shells at different angles (Goldstine, 2001, Martin H. Weik, 1961). Since then, physics engines have found their way to various applications. For example, they have been used for modelling fluid dynamics on supercomputers in the 1980s, where force vectors are assigned for each fluid particle to show flowing streams. Tyre designers have begun to use physics simulations to verify the way in which different tyre tread styles may function in different road conditions, with different materials and loadings. To date, however, very little research has considered implementing physics engines in the biomechanical analysis of articular joints. As with object rendering, contact, collision, muscle wrapping and deformation are other important and computation consuming components in the simulation of articular joints. Getting the most out of GPU power for physics based simulation was a point of interest for many researchers in medical applications (Georgii and Westermann, 2005, Mosegaard and Sorensen, 2005, Taylor et al., 2008, Pang et al., 2010). However, some limitations and issues have emerged. For instance, geometrical models should be carefully handled, intensive CPU computations are still required and a special algorithm is needed for managing parallel processing.

Most recently, a new specific computer hardware accelerator for physics engines referred to as a PPU or Physical Processing Unit has been launched mainly for handling the computations involved in physical simulation, typically in video games (Nealen et al., 2006). Putting PPU into action in surgical training has been carried out by Pang et al. (2007), but in their pioneer work in this area, they have suffered from hardware limitations.

2.7.1 Middleware available physics engine

The thought of implementing physics engine such as PhysX to improve articulation modelling and simulation has been encouraged by the versions of virtual reality games which have recently become available. An approach has been devised which channels video game technological innovations to medical applications, where they can deliver cost-effective systems. Since the physical principles in biomechanical applications are the same as the principles applied in serious gaming platforms, the prospect of using game physics may introduce significant improvements in medical applications.

PhysX (AGEIA) is a widely used and well known middleware SDK physics engine provided originally by Ageia. It was later acquired by Nvidia in 2008. It was initially created to deliver processing ability for dynamic interactions in video gaming. A PhysX engine is incorporated into the recent versions of GPUs to permit hardware acceleration, while PhysX has the capacity to use hardware acceleration. It is referred to as a Physics Processing Unit or PPU (Davis et al., 2005). As PhysX was developed mainly for games, using PPU may enhance the processing time to boost system performance. Parallel processing acceleration enhancement has been introduced by Nvidia represented as CUDA (Nvidia) cores. The earliest Physics Processing Unit (PPU) was launched by Ageia in 2006 (Derek, 2006); PPU operates in a comparable way with the Graphic Processing Unit (GPU). Incorporating PhysX with parallel processing would relieve the Central Processing Unit (CPU) of most of the heavy physical computations and take them to CUDA parallel processors.

CHAPTER 3

PHYSICS BASED MODELLING

3.1 Introduction

The continuous revolution in the digital computations and graphics hardware raised virtual modelling and simulation to elevated levels of speed, stability and realism. The recent developments in digital entertainment and gaming have influenced the need for efficient and effective processors to carry on the heavy computations at an interactive speed. Video gaming is a multi-billion industry which attracts massive business and research resources around the world. Modern video games nowadays are more convincing than ever; they gain sophistication by implementing the principles of mechanics to achieve virtual simulation through natural-seeming interactions of physical objects. The core component of physically based modelling and simulation on a game authoring software or game engine is the physics engine. The natural behaviour of interacting objects in the simulation scene results from the implementation of the laws of physics in the game engine. Physics engines such as PhysX support a number of physical objects with rigid, deformable or soft bodies, as well as fluid dynamics. In addition, different types of constraints are available (i.e. hinge joint, point, etc.) for use in the constructing the physical system. Physics engines are also responsible for simulating physical interactions, such as collisions and responses. Physical objects need to be defined by certain attributes, such as density or mass, surface geometry, stiffness and other things, which influence the physical behaviour of the object. The principles and theories of physics behind physics engines and gaming platforms are investigated in this chapter. For a preliminary validation of game physics modelling, a number of virtual experiments of mechanical systems with analytical solution have been performed. The preliminary validation is useful to confirm this platform for further investigation with the musculoskeletal joint model.

3.2 Development Architecture

A preferred pattern when building a physically based model using a game engine is usually to divide the application development into three main components, as demonstrated in Figure 3.1.

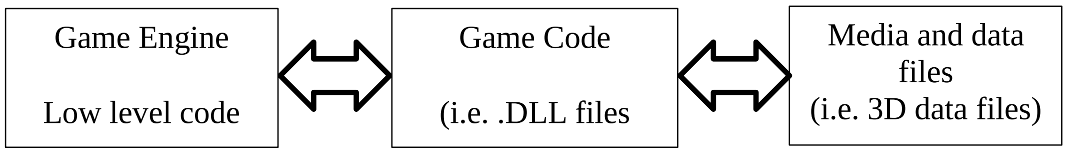


Figure 3.1: Main components of a game engine

Part one (the .exe) includes the low-level code, which is responsible for a number of tasks including object rendering, managing the memory and carrying out physics calculations. It manages all the operations of working with computer hardware, for instance, I/O hardware. Moreover, this code hyperlinks almost all APIs in the interests of game development. Part two, which is the game code, is a higher-level code than the engine code and is abstracted from the engine specified code. The major functions of this component are usually to deal with the action logic. The game code describes how the interactions are practised. Quite often, gaming platforms are integrated with a scripting language, such as JAVA. The last part of the game engine consists of the media and data files used in the game. It is likely to include nearly anything from images or 3D data to sounds. The advantage of employing this kind of structure is its overall flexibility. It makes it possible for the creator to open-source the code, allowing users to make changes for anything up to an entire application, with no need to gain access to the engine code.

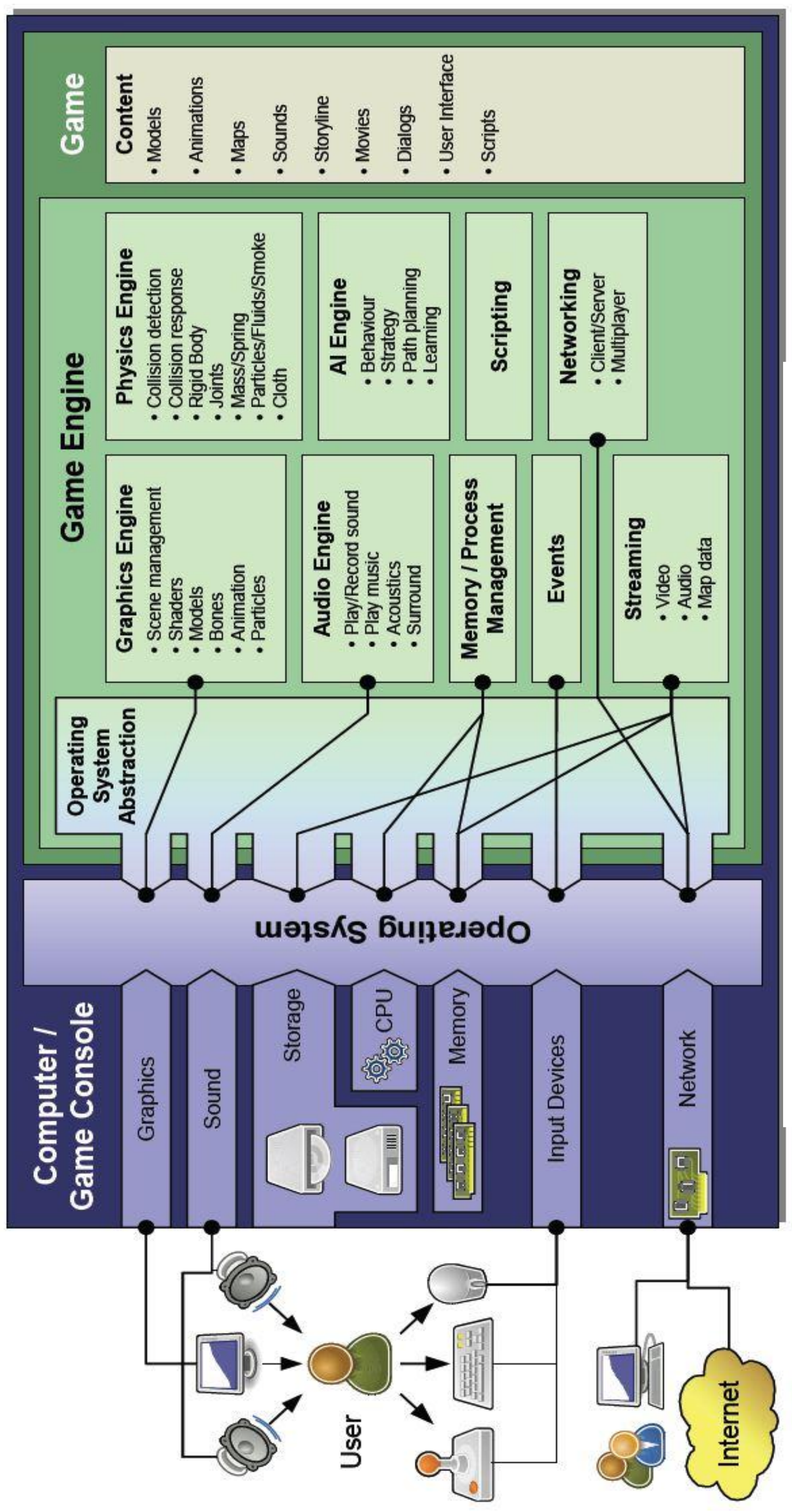


Figure 3.2: Game engine block diagram (Marks et al., 2007)

The assortment of best available game authoring platforms began together with an assessment of the available game engines according to an online data base (DevMaster, 2010). At the beginning of this evaluation, more than 300 game engines were available to consider. However, to reduce this number, some engines which were only in an early stage of development and others which were out of date were ignored. The selected game engines were critically reviewed so as to choose a reliable physics engine to support collision detection, collision response, rigid bodies, soft bodies and mass spring systems. Engines which lacked essential editing, import and export components were also eliminated. Out of those which remained, DX Studio was chosen for in depth review, as in our opinion it was one of the best available game engines and might suitable for our application.

In the virtual environment, physical objects are required to exhibit natural motion as they interact with one another. Solving such motion and interaction in virtual simulation requires the laws of physics to be implemented in the simulation environment to capture realistic behaviour. This kind of virtual simulation can be carried out with the aid of a physics engine, which can be simply illustrated as a software library which delivers an approximate simulation of a physical system based on the laws of physics. Most physics engines were mainly developed as middleware for video games to assist real-time interaction with the virtual environment. Recently, physics engine have become more highly developed and sophisticated than ever, due to the huge volume of investment these days in game industries.

The core components of the physics engine are collision detection, collision response, soft and rigid body dynamics and fluid dynamics. Although most physics engines are based on similar principles, they do vary in their components and algorithms. Some may offer more options such as soft body and fluid dynamics; some others are more optimized and support multiple processors. More advanced physics engines, such as PhysX are employed to simulate real problems for research purposes, such as surgical planning and simulation (Pang et al., 2010). In addition to the above core components of physics engines, PhysX from Nvidia comes with further features. PhysX is a leading sophisticated physics engines; it has an extended range of options and capacities, such as supporting a Physics Processing Unit (PPU), volumetric fluid simulation, soft bodies, deformable bodies, cloth, springs, dynamic triangular mesh geometry, static and dynamic friction, as well as continuous collision detection for objects with high speed and many other things. In addition, a wide variety of options and system customization is available to establish a tailored simulation environment. Since PhysX is a commercial physics library, its internal details are unavailable. However, game physics are usually derived from Newtonian physics.

3.3 Rigid Body Dynamics for Physics Engines

Assuming that the deformation of the object is negligible compared to the size of motion, an object will be recognized in the virtual environment as a rigid body. Rigid bodies will be able only to change position and orientation. Driving the motion in physics engine usually starts with Newton’s law of motion. The simulation of a simple moving object X can be easily described by Newton’s second law of motion

$$\mathbf{F} = m\mathbf{a} \tag{3.1}$$

where **F,** is the external force acting on a body of mass m and causing an acceleration of **a**. The acceleration, **a,** may also be expressed as:

$$\mathbf{a} = \frac{d\mathrm{v}}{dt} = \frac{d^2\mathrm{x}}{dt^2} \tag{3.2}$$

By integrating over time, the displacement and velocity at any given time are written as:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i t + \frac{1}{2}\mathbf{a}t^2 \tag{3.3}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a}t \tag{3.4}$$

The force, **F**, is also the derivative of the momentum, P,

$$\mathbf{F} = \frac{d\mathrm{P}}{dt} = m\frac{d\mathrm{v}}{dt} = m\frac{d^2\mathrm{x}}{dt^2} \tag{3.5}$$

By defining the initial conditions and external forces, the location and velocity of the moving object **X** can be determined at any given time. For a particular body, the equation of motion is obtained by Newton's law. Then the ordinary differential equations representing the equations of motion of the rigid bodies in the simulation environment can be solved by numerical integration methods such as Euler's method and the Runge-Kutta method, which integrates the equations for every time step, *h*. The force, **F**, is a function which is dependent on position, **x**(*t*) and velocity **v**(*t*) are written as:

$$\boldsymbol{F}(\mathbf{x}(t), \mathbf{v}(t)) = m.\acute{\mathbf{v}}(t) \tag{3.6}$$

Assuming, the time step, h is very small, the definition of the derivative can be simplified and rearranged as follows:

$$\acute{f}(t) = \lim_{h \to 0} \left(\frac{f(t+h) - f(t)}{h} \right) \tag{3.7}$$

$$\acute{f}(t) \approx \left(\frac{f(t+h) - f(t)}{h} \right) \tag{3.8}$$

Or in a different arrangement;

$$f(t+h) \approx f(t) + h\acute{f}(t) \tag{3.9}$$

In this case, the value of the function for the next step is determined by substituting the current variable in the function and its derivative.

By using Euler's method, the position and velocity may be obtained at every time step, *h*, as in Equations (3.10) and (3.11).

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i \tag{3.10}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + h\mathbf{a}_{\boldsymbol{i}} \tag{3.11}$$

where the net force applied on the centre of mass, F, and the acceleration, a, is obtained as follows:

$$\mathbf{F} = \sum_k \mathbf{F}_k \tag{3.12}$$

$$\mathbf{a}_i = \frac{\mathbf{F}}{m} \tag{3.13}$$

Then, the formula mentioned above provides a complete modelling for the translational motion of a rigid body; however, it is not satisfactory for many cases of virtual simulation since it is unable to take into account the rotational motion of the body.

The rotational motion occurs due to the offset of the applied force from the centre of mass. This offset causes torque, $\boldsymbol{\tau}$, which will drive the rotation.

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F} \tag{3.14}$$

The angular speed, $\boldsymbol{\omega}$, can be calculated by knowing the angular momentum, **L,** and the moment of inertia, **J**.

The angular momentum, **L**, is given by

$$\mathbf{L} = \mathbf{J}\boldsymbol{\omega} \tag{3.15}$$

So the angular velocity, $\boldsymbol{\omega}$, will be

$$\boldsymbol{\omega} = \mathbf{J}^{-1}\mathbf{L} \tag{3.16}$$

Since the moments of inertia, $\mathbf{J}$, is a 3×3 matrix which expresses how an object rotates about different axes according to its geometrical parameters and density, it will be computationally expensive to calculate it at every time step; thus simplification is needed. One common assumption is that the density is uniformly distributed along the objects and the calculation is based on the geometry.
Therefore Equation (3.16) may be rewritten as:

$$\mathbf{J}_{\mathbf{i}}^{-\mathbf{1}}\mathbf{L}_{\mathbf{i}} = \Omega_{\mathrm{i}}\mathbf{J}_{\mathbf{0}}^{-\mathbf{1}}\Omega_{\mathrm{i}}^{\mathrm{T}}\mathbf{L}_{\mathbf{i}} \tag{3.17}$$

Then the new rotation matrix is found to be

$$\Omega_{\mathrm{i+1}} = \Omega_{\mathrm{i}} + h\widetilde{\omega}_{i}\Omega_{\mathrm{i}} \tag{3.18}$$

where

$$\tilde{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{3.19}$$

The angular equations for the new orientations may be expressed as:

$$\boldsymbol{\tau} = \sum_k r_k \times \mathbf{F}_{\boldsymbol{k}} \tag{3.20}$$

And by integrating the torque, $\boldsymbol{\tau}$, the new angular momentum will be

$$\mathbf{L}_{\mathrm{i+1}} = \mathbf{L}_{\mathrm{i}} + h\tau \tag{3.21}$$

And $\mathbf{J}_{\mathrm{i+1}}^{-1}$ can be found to be

$$\mathbf{J}_{\mathrm{i+1}}^{-1} = \Omega_{\mathrm{i+1}} \mathbf{J}_0^{-1} \Omega_{\mathrm{i}}^{\mathrm{T}} \tag{3.22}$$

The angular velocity is thus:

$$\boldsymbol{\omega}_{\mathrm{i+1}} = \mathbf{J}_{\mathrm{i+1}}^{-1} \mathbf{L}_{\mathrm{i+1}} \tag{3.23}$$

It can be seen that that by knowing torque $\boldsymbol{\tau}$ from (3.20), the change in the angular momentum can be obtained from (3.21) and as $\mathbf{J}_{\mathrm{i+1}}^{-1}$ can be calculated from (3.22) the new angular velocity $\boldsymbol{\omega}_{\mathrm{i+1}}$ can be obtained by (3.23).

3.4 Contact Modelling in Physics Engines

3.4.1 Collision detection

Before managing the collision issue, the concept of such a situation, is described as follows. In virtual 3D environmental space, S, including a moving object, O, a small data structure, is pre-processed to check if there any intersection between O and S.

The answer is provided with each time frame and pre-processing the data continues as the objects move. The collision detection problem becomes more complicated when the number of objects in the scene rises. Obviously this would be undesirable in all collision detection algorithms. Collision detection problem are easier to handle for approximated bounding volume than the geometry described by the mesh coordinates. If the bounding volume cannot intersect, then the object will not collide. The approximation method is used for complex shapes and is based on the bounding volume hierarchies.

Most physical objects in the simulation environment are represented by two different geometries. The rendered geometry on the screen illustrates the real shape of the object, which is normally complex. The other shape, referred to as the bounding volume, is a simplified geometry which is generated by the physics engine to optimize the computations when it comes to collision detection, in particular when the number of physical objects in the scenes rises. PhysX, as well as most physics engines, takes advantage of bounding volumes in speeding up the simulation for physical real-time interaction. In PhysX, as well as other physics engines, a collision primitive may be expressed as a bounding box, convex hull, sphere, capsule or plane. These bounding volumes are usually found in the broad phase of the collision detection to greatly reduce the unnecessary computations before the narrow phase begins, where the mesh surfaces of the objects are checked for any collision. As mentioned earlier, the physics engine and gaming platforms offer a variety of bounding volumes which can be used to simplify the collision geometries. Several tree constructions and update strategies are employed. Accordingly, some simulation environments may be more likely to have different types of BVH than others. Very frequent types of bounding volumes which are implemented in the physics engine and the gaming platform are discussed here. The development platform features Axis-aligned bounding boxes (AABB); Object oriented bounding boxes (OBB) and spheres are primarily preferred for their robust overlap check. An AABB algorithm is recognized as being the simplest algorithm that may be used during the broad phase of processing. The drawback of the AABB is its imprecise approximation in representing the object, which makes it unsuitable for some geometries. Objects represented by AABB will be considered as tight boxes containing the objects. Approximation accuracy depends on the actual shape of the object and its orientation. This is the weak point of simulating complex shapes. Therefore, using AABB for complex shapes may result in a poor simulation. From the definition, the bounding volume should cover all the geometry at all times. Although AABB is aligned with the axis system of the virtual scene, the actual object is allowed to rotate due to the environmental interactions. Bounding volumes in AABB can be expressed in two ways. The first is to establish a fixed sized volume which covers the entire object at all times, regardless of the orientation changes of the actual object. The second way is to make this bounding volume dynamically change its dimensions with every time step, to maintain the tightest possible bounding volume. Checking for collisions with AABB is simple, because its axis is aligned. The AABBs values are sorted in x, y and z directions independently.

Collision can occur only if the min/max values of the AABBs overlap over the x, y and z axes. If two or more objects satisfy this condition, it will be added to the active contact list. Figure 3.3 shows an illustration of the AABB detection check. In this AABB collision detection check, the sides of the bounding box are aligned with the x, y and z axes of the world coordinate of the scene. Then checking for any overlap is carried out as follows. Assuming two moving objects termed A and B are contained in two separate AABBs

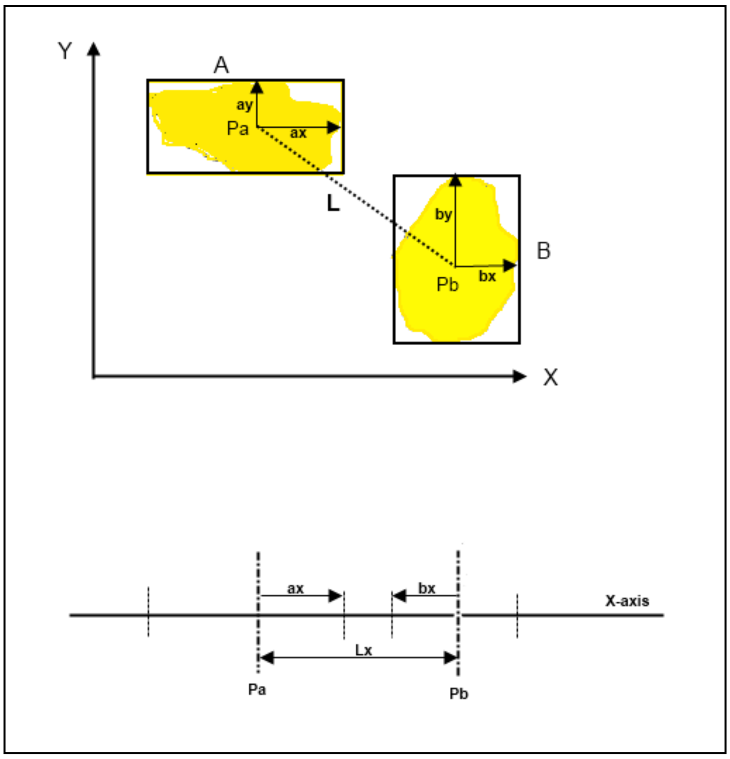


Figure 3.3: AABBs for A and B in x-y plane and the separating axis for the X-axis

L represents the distance between the geometrical centres, P_a and $P_{b,}$ and is given by $L = |Pb - Pa|$. A separating axis test is carried out along each axis to see if there is any separating axis. Along each axis *i*, if $L_i > |a_i + b_i|$, there is no overlapping, for, if there were, the axis, *i*, would not be a separating axis. When there is no penetration between A and B, one separating axis must be present. While A and B are continuously moving, sweeping AABBs is also possible, to search out the initial overlap as shown in Figure 3.4. The displacements, noted as v_a and v_p for *A* and *B* respectively, are calculated at every time step.

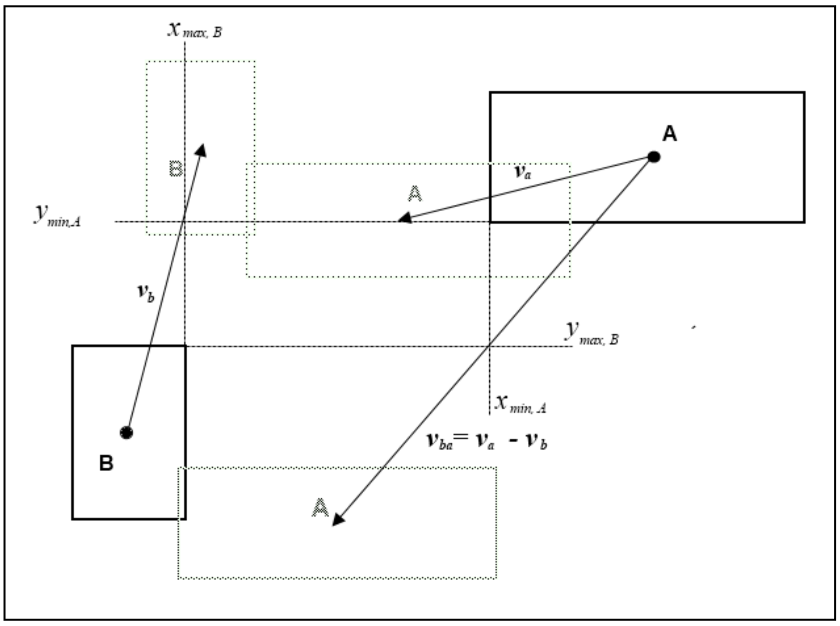


Figure 3.4: Illustration of AABB collision test in x, y plane

The minimum and maximum axis values for every side of the bounding box $(For\ this\ example, x_{min,A}, x_{max,B}, y_{min,A}, y_{man,A})$ are acquired to obtain the normalized time, as follows:

$$u_x = \frac{x_{max,B} - x_{min,A}}{v_x} \tag{3.24}$$

$$u_y = \frac{y_{max,B} - y_{min,A}}{v_y} \tag{3.25}$$

where, $u_x, and\ u_y$ represent the normalized times at which the x and y axes extension are required to overlap. The overlapping of the AABBs exists only if the extensions of all axes overlapping in a shared time interval and once any axis extensions cease to overlap, the boxes are separated. The time when overlapping begins for each axis extension may be expressed as $u_{x1}, u_{y1}, and\ u_{z1}$ for x, y and z directions respectively. Similarly, $u_{x2}, u_{y2}, and\ u_{z2}$ are the times when the overlapping ends for each axis. Thus, the beginning of a possible penetration of AABBs is termed by:

$$u_1 = \max(u_{x1}, u_{y1}, u_{z1}) \tag{3.26}$$

and the instant when the AABBs are separated may be expressed as:

$$u_2 = \max(u_{x2}, u_{y2}, u_{z2}) \tag{3.27}$$

Therefore, if the bounding volumes are to penetrate, the following condition must be satisfied:

$$u_1 \leq u_2$$

Then the positions where the penetration begins and ends may undergo linear interpolation with us.

The OBB-Tree is another hierarchical method based on arbitrarily oriented bounding boxes. It functions better than typical AABB, since the orientation of the bounding box may help in a tighter fit for the object and thus manage an improved approximation. This type of bounding box is most efficient when the aspect ratios of the shape are higher. The overlap testing for OBB is carried out on fifteen axis projection tests. Compared with AABB, the OBB-tree is slower to perform and update, since it is orientation sensitive. In OBB, the bounding boxes are oriented according to the local axes, as in Figure 3.5.

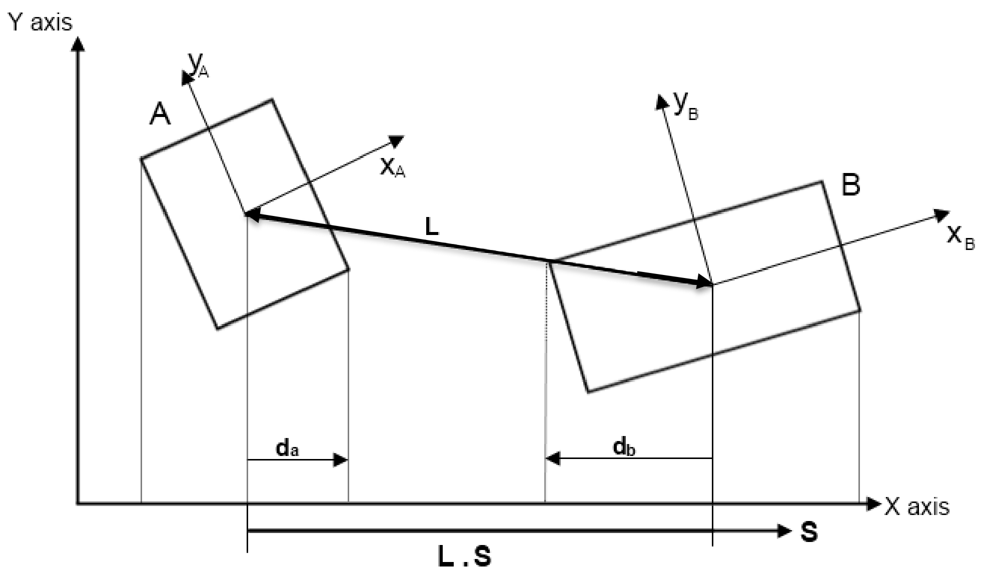


Figure 3.5: An illustration of the OBB in 2D

At every time step, the projections of the bounding boxes on the vector which forms the separating axis, S, may be used to carry out the separating axis test for any possible overlapping. The distance between the two geometrical centres of A and B along the separating axis forms an interval to be checked against the projection distance for both boxes, as illustrated in Figure 3.5.

The distance of projection for each OBB along the separating axis X is determined as follows.

For OBB, A:

$$d_a = a_1|X_A . S| + a_2|Y_A . S| + a_3|Z_A . S| \qquad (3.28)$$

And so for B:

$$d_b = b_1|X_B\,.S| + b_2|Y_B\,.S| + b_3|Z_B\,.S| \tag{3.29}$$

Thus, satisfying the following condition is mandatory in order to say that S is a separating axis

$$|L.S| > \; d_a + d_b$$

The separating axes between A and B are checked along the six principal axes and the consequent nine cross products. If any of the 15 axes shows a valid separating axis, the OBBs are separated, otherwise penetration exists. A simplified version of this test may be performed if the local coordinate system is transformed into the other box.

The bounding volumes discussed earlier in this section are only approximations of the actual geometry, which is more complex than a standard rectangular shape or a sphere. The main advantage of the bounding volumes for the complex geometries is to recognise objects which may possibly collide in the broad phase of the collision detection. If the bounding volumes do not intersect, then the mesh surface of the object will not, of course, intersect and no further investigation for collision is required. For cases when the bounding volumes collide, a more detailed interference check is required, which is also termed the narrow phase of the collision detection. The actual interference test should be then carried out using the original geometry of the model as described by the triangular mesh. The triangle, the unit component of the surface geometry, is represented as a three point convex combination.

There are several methods to check for such a triangle intersection in the physics engine. One algorithmic check implemented in the gaming platform is illustrated as follows:

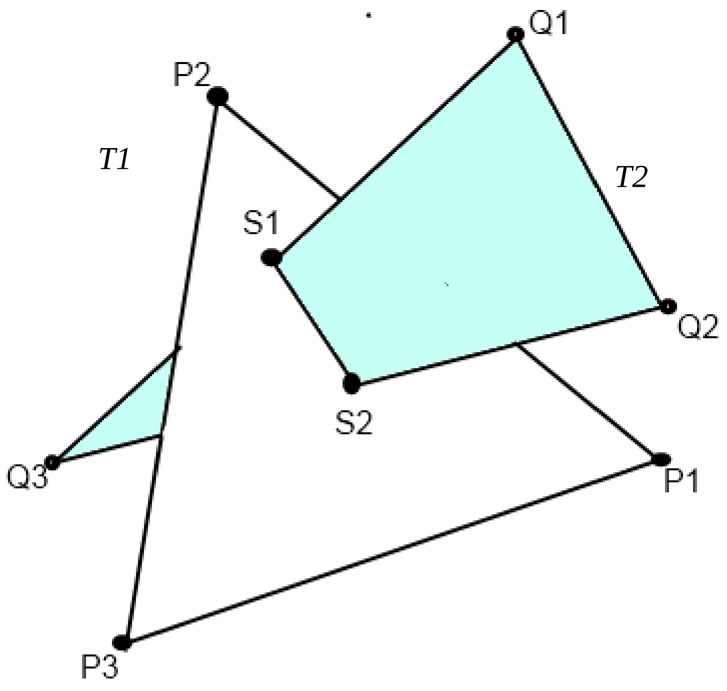


Figure 3.6: Two intersecting triangles

First, for the first triangle, T_1, which consists of points *P1*, *P2* and *P3*, with a normal vector **n** = *(a, b, c)* and point on the plane $P1(x_1, y_1, z_1)$, then the equation for this plane will be:

$$a(x - x_1) + b(y - y_1) + c(z - z_1) = 0 \tag{3.30}$$

Or in a different arrangement,

$$ax + by + cz - d = 0 \tag{3.31}$$

Where d is:

$$d = ax_1 + by_1 + cz_1 \tag{3.32}$$

where $a, b,$ and c are the components of the normal vector which can be calculated by normalizing the cross products of the two vectors on the plane.

$$\mathbf{n} = (P2 - P1) \times (P3 - P1) \tag{3.33}$$

Then points *Q1*, *Q2* and *Q3* of the other triangle T_2 are substituted in the plane equation of the first triangle to check whether or not the three points are located on one side of the first triangle. This can be checked out according to the sign generated from the plane equation. If all points have the same sign, then they are all on one side; otherwise, they are not and triangle-triangle intersection exists. If opposite signs exist, then the *d*s are stored for each point from the plane equation to find which points of triangle *T2* intersect the plane of *T1*. This test is performed for each pair of points on the triangle *T2*. The intersecting triangles in Figure 3.6, show that the pairs *(Q1, Q3)* and *(Q2, Q3)* experience opposite signs, which make it clear that *Q3* is on the opposite side of *Q1* and *Q2*. The intersection edge (S1-S2) between the triangles can be determined by Equation (3.35).

$$S = Q_i + \frac{d_i}{(d_i - d_j)}(Q_i - Q_j) \tag{3.34}$$

Performing this for every pair of points will produce the intersection edge as illustrated by the line S1-S2 in Figure 3.6.

3.4.2 Collision response

Collision response is the determination of the resulting behaviour of the colliding objects. The two main situations of collision response occur when objects collide and bounce back or rest after the first collision, as they do when rolling. Several approaches are available to handle collision response. Constraint based (Baraff, 1994), impulse based (Mirtich and Canny, 1995) and penalty methods are the most common collision response methods used in physically based simulations. In contact problems, modelling friction is another issue that directly affects the collision response. A number researchers have discussed friction modelling and the difficulties associated with it (Baraff, 1991, Baraff, 1994). The most popular solution for collision response when used for virtual simulation is the impulse based method. An illustration of the linear collision response of two colliding spheres named A and B, with velocities V_A and V_B is discussed below.

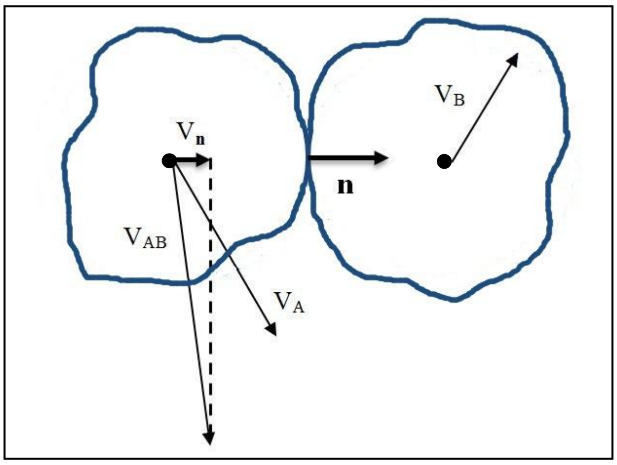


Figure 3.7: Linear collision of two spheres

The impulse force, *f*, is generated along the collision normal and placed at the collision point.

The relative approaching velocity between the two spheres is calculated by:

$$V_{AB} = V_A - V_B \tag{3.35}$$

and the normal component of V_{AB} is:

$$V_{\mathbf{n}} = V_{AB}.\mathbf{n} \tag{3.36}$$

The coefficient of restitution, ε, is required to calculate an appropriate impulse force. This coefficient reflects the physical nature of the colliding objects and the energy absorbed during the collision. Its value is limited to between 0 and 1. The value 0 is for sticky contact and is 1 when the collision is fully elastic and no energy is absorbed during it. It is represented as the ratio between the velocities along the normal just before and after collision.

$$\acute{V}_{\mathbf{n}} = -\varepsilon\, V_{\mathbf{n}} \tag{3.37}$$

Or

$$(\acute{V}_A - \acute{V}_B).\mathbf{n} = -\,\varepsilon\,(V_A - V_B).\mathbf{n} \tag{3.38}$$

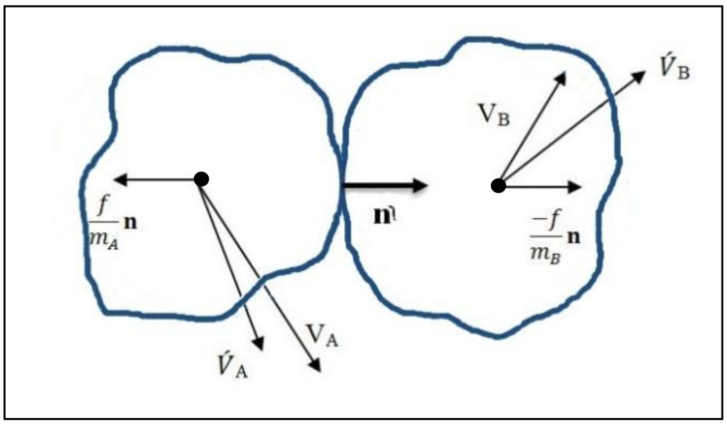


Figure 3.8: Separating velocities for a linear collision response

Just before and after collision, the total momentum of the system is conserved, so

$$m_A\, V_A + f\mathbf{n} = m_A\, \acute{V}_A \tag{3.39}$$

and

$$m_B\, V_B - f\mathbf{n} = m_B\, \acute{V}_B \tag{3.40}$$

Then, solving for the impulse factor, *f*, substituting the Equations (3.40) and (3.41) into (3.39) will give:

$$f = \frac{-(1+\varepsilon)(V_A - V_B).\mathbf{n}}{\mathbf{n}.\mathbf{n}\left(\frac{1}{m_A} + \frac{1}{m_B}\right)} \tag{3.41}$$

Therefore, the velocities of the objects just after the impact will be:

$$\acute{V}_A = V_A + \frac{f}{m_A}\mathbf{n} \tag{3.42}$$

and

$$\acute{V}_B = V_B - \frac{f}{m_B}\mathbf{n} \tag{3.43}$$

No rotational motion is considered so far; however, introducing the angular motion is similar to the procedure carried out above, with the addition of the angular velocities.

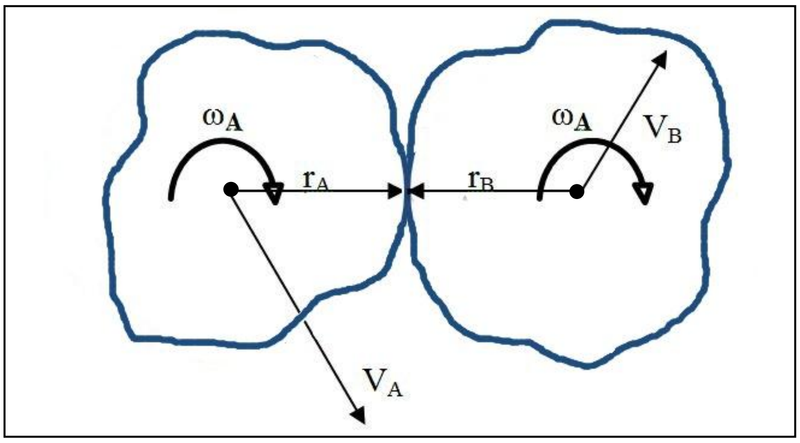


Figure 3.9: Collision with angular velocities

The velocities at the contact point are:

$$\bar{V}_A = V_A + \omega_A \times r_A \tag{3.44}$$

and

$$\bar{V}_B = V_B + \omega_B \times r_B \tag{3.45}$$

So the relative velocity here will be

$$V_{AB} = \bar{V}_A - \bar{V}_B \tag{3.46}$$

Similarly, with Equation (3.39)

$$(\acute{\bar{V}}_A - \acute{\bar{V}}_B) = -\,\varepsilon\,(\bar{V}_A - \bar{V}_B) \tag{3.47}$$

In addition to the linear momentum, the conservation of angular momentum is also considered here.

$$\mathbf{J}_A\,\omega_A + r_A \times f\mathbf{n} = \mathbf{J}_A\,\acute{\omega}_A \tag{3.48}$$

So,

$$\acute{\omega}_A = \omega_A + \mathbf{J}_A^{-1}(r_A \times f\mathbf{n}) \tag{3.49}$$

Similarly,

$$\acute{\omega}_B = \omega_B + \mathbf{J}_B^{-1}(r_B \times f\mathbf{n}) \tag{3.50}$$

Then solving for the impulse, *f*, substituting Equations (3.49) and (3.50) into (3.47) will result in the impulse equation being:

$$f = \frac{-(1+\varepsilon)(V_A - V_B).\mathbf{n}}{\mathbf{n}.\mathbf{n}\left(\frac{1}{m_A}+\frac{1}{m_B}\right) + \left[\left(\mathbf{J}_A^{-1}(r_A \times \mathbf{n})\right) \times r_A + \left(\mathbf{J}_B^{-1}(r_B \times \mathbf{n})\right) \times r_B\right].\mathbf{n}} \tag{3.51}$$

The penalty method is a different method employed in physics engines to handle collision response. In the penalty method, temporary springs are placed at the contact points of the surfaces with possible collision. The inserted springs at the contact points will generate an opposite force on the colliding objects to push them apart. The force is generated according to Hook's law:

$$F = -kx \tag{3.52}$$

where *F*, is the force exerted by the spring at the contact points, *k* is the spring stiffness and the *x* is the spring compression length or penetration depth.

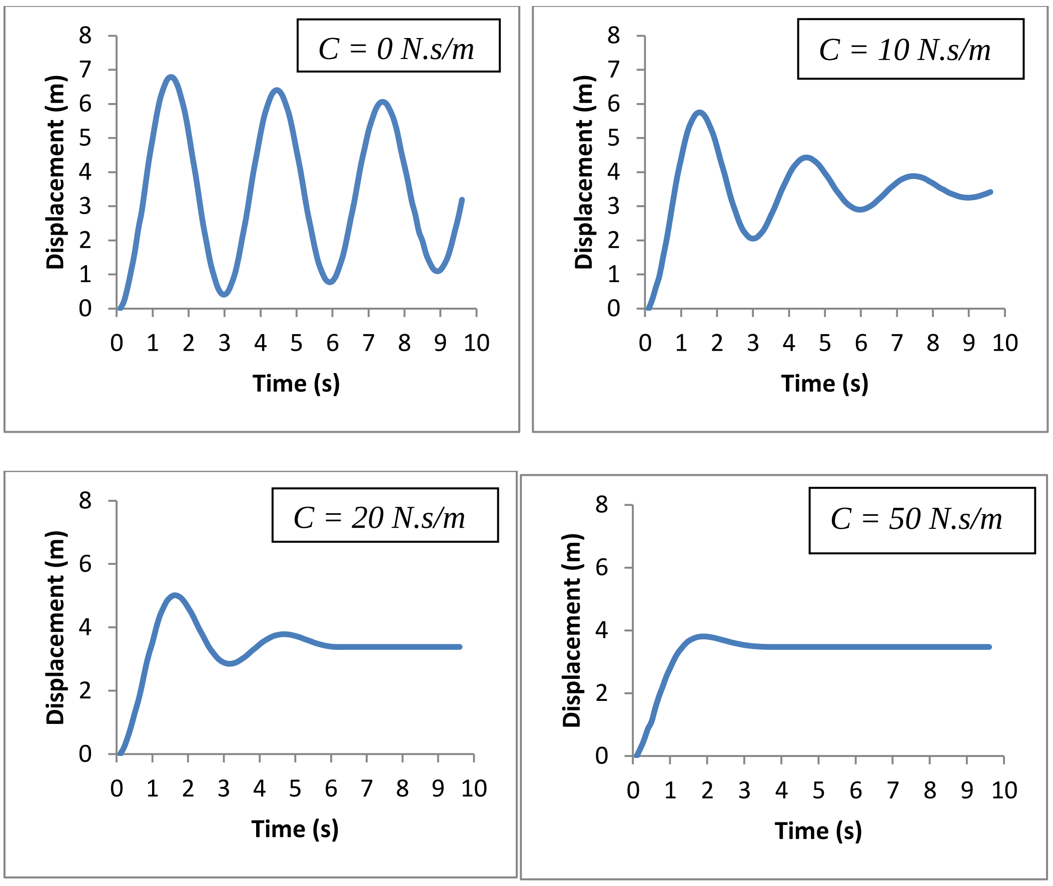
Although this method looks simple to use and implement, it requires very small time increments, which are computationally expensive (as will be discussed in section 3.5.2). Otherwise, it becomes unstable, in particular with higher *k* values. This is because the infinite quantities are modelled with finite quantities. The essential concern when using the penalty method is to find the appropriate spring stiffness, *k*. However, this method delivers an effective collision response which can be used efficiently with complex geometries.

3.5 Preliminary Validation of the Physics Engine, PhysX

In order to assist the physical accuracy of the game physics to validate such a platform for medical applications, this section has been introduced. Some basic mechanical systems with analytical solutions have been tested in the virtual environment powered by the physics engine, PhysX, to verify its feasibility for solving authentic problems.

3.5.1 Spring-mass system

For a mass spring system, with mass, m = 25 kg, spring stiffness, k = 100 N/m and damping coefficients, C = 0, 10, 20, 50 and 100 $N.s/m$, the following system responses obtained from the physics based simulation are plotted in Figure 3.10.



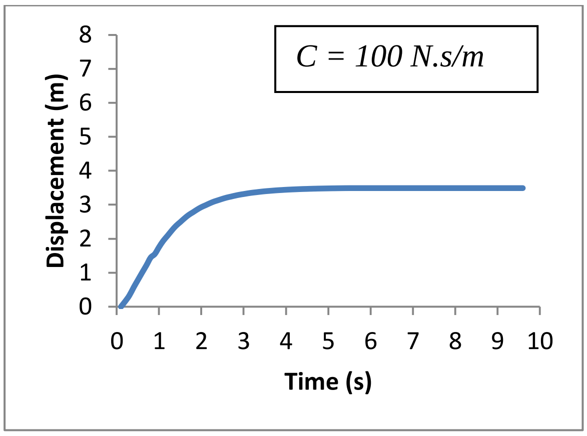


Figure 3.10: System responses for different damping ratios

Figure 3.10 shows how the damping changes the response of the mass spring system under different damping ratios.

The calculation of the periodic time, *T*, is calculated as:

$$T = \frac{2\pi}{\sqrt{\frac{k}{m}}} = \frac{2\pi}{\sqrt{\frac{100}{25}}} = 3.142\ s$$

The critical damping coefficient can be calculated for this system as follows:

$$C = \sqrt{4km} = \sqrt{4(100)(25)} = 100\ N.s/m$$

From the physics based simulation, the system response for the critical damping coefficient is plotted in Figure 3.10 *(C = 100 N.s/m).*

Although the undamped system experiences some damping, it is caused by the fact that physics engines are mainly developed for gaming platforms and this demands a stable simulation environment. Thus, a completely undamped system may result in an unstable system response, in particular when physical interactions take place with multiple objects in real time simulations. For this reason, internal damping in a physics engine exists, to provide some stability for the simulation environment. This is because the implementation of penalty function is algorithmically difficult and may cause a build-up of energy in the system, as discussed in the following section.

The mass spring system illustrated above is validated to an acceptable level. The physics engine, PhysX, handles the linear vibration problem in a realistic manner.

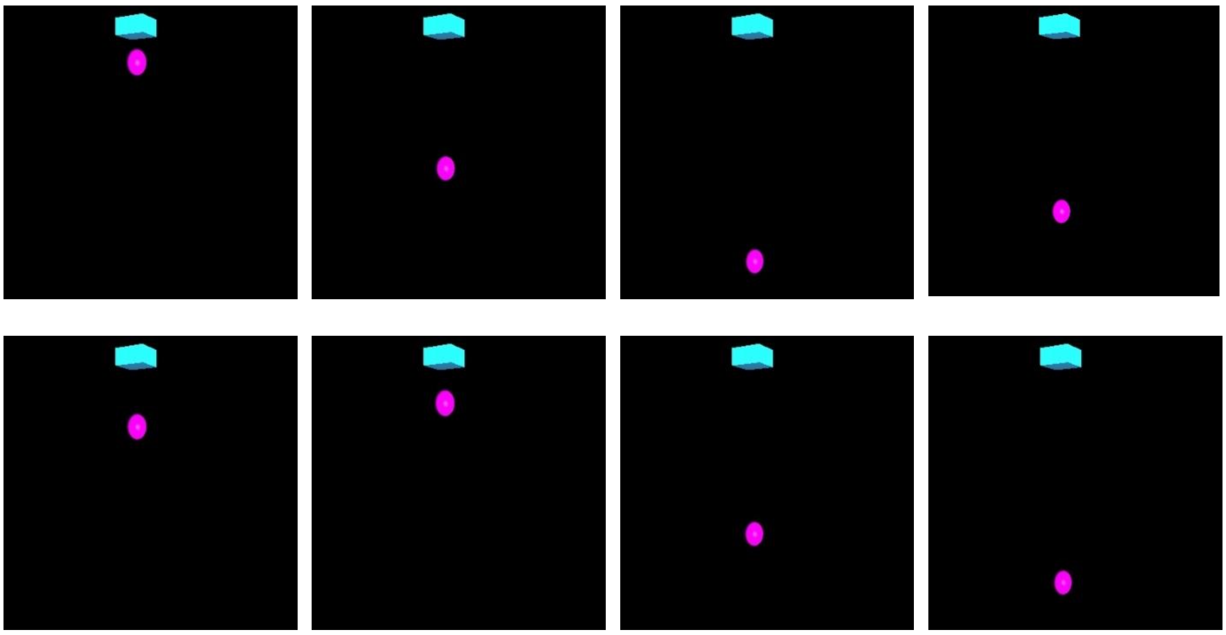
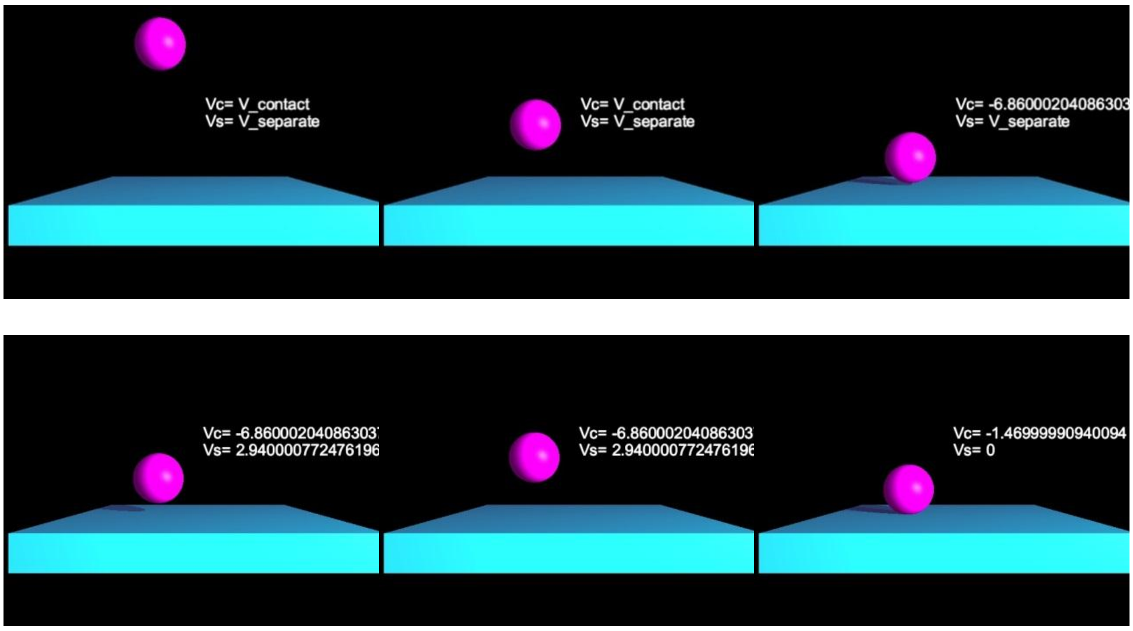


Figure 3.11: Snapshots of the mass spring system simulation

3.5.2 Contact modelling

The second virtual experiment was carried out to examine the viability of the system in dealing with contact problems. The approaching and separating velocities were recorded just before and after physical contact occurred. The coefficient of restitution was set to zero, where completely elastic collision takes place, and 0.5, which allows some energy dissipated during collision to achieve a condition of rest after a few collisions. Gravity was the driving force over the moving sphere, as shown in Figure 3.12. It has been found that when collision is completely elastic, there is some build-up of energy in the system. The simulation results show that the separating velocity is slightly higher than the colliding velocity. As the collision test is carried out at timed increments, there is a chance that the collision is detected just after a certain penetration. In physical terms, this means that objects are penetrated without losing kinetic energy and this causes an increase of energy in the system; it can be observed that the separation velocity increases. Higher accuracy can be achieved by increasing the frames per second FPS, so the collision can be detected more precisely. Although this accuracy can be increased, the penetration between the system time steps cannot be avoided, due the discrete environment of the virtual simulation, where all calculations are carried out on discrete time steps. Reducing these time steps would enhance the overall accuracy of the simulation; but more computations are introduced.



```
Vc= V_contact
Vs= V_separate

Vc= V_contact
Vs= V_separate

Vc= -6.86000204086303
Vs= V_separate

Vc= -6.86000204086303
Vs= 2.940000772476196

Vc= -6.86000204086303
Vs= 2.940000772476196

Vc= -1.46999990940094
Vs= 0
```

Figure 3.12: Screen shots of the collision simulation

For the other situation when the collision is not fully elastic, ε = 0.5, some energy loss is expected. It can be seen that the approaching speed is almost twice the separating speed for the first collision. Thus, the coefficient of restitution value explains this change of linear momentum.

Another example of physical objects sliding along each other on a frictionless surface is illustrated in Figure 3.13.

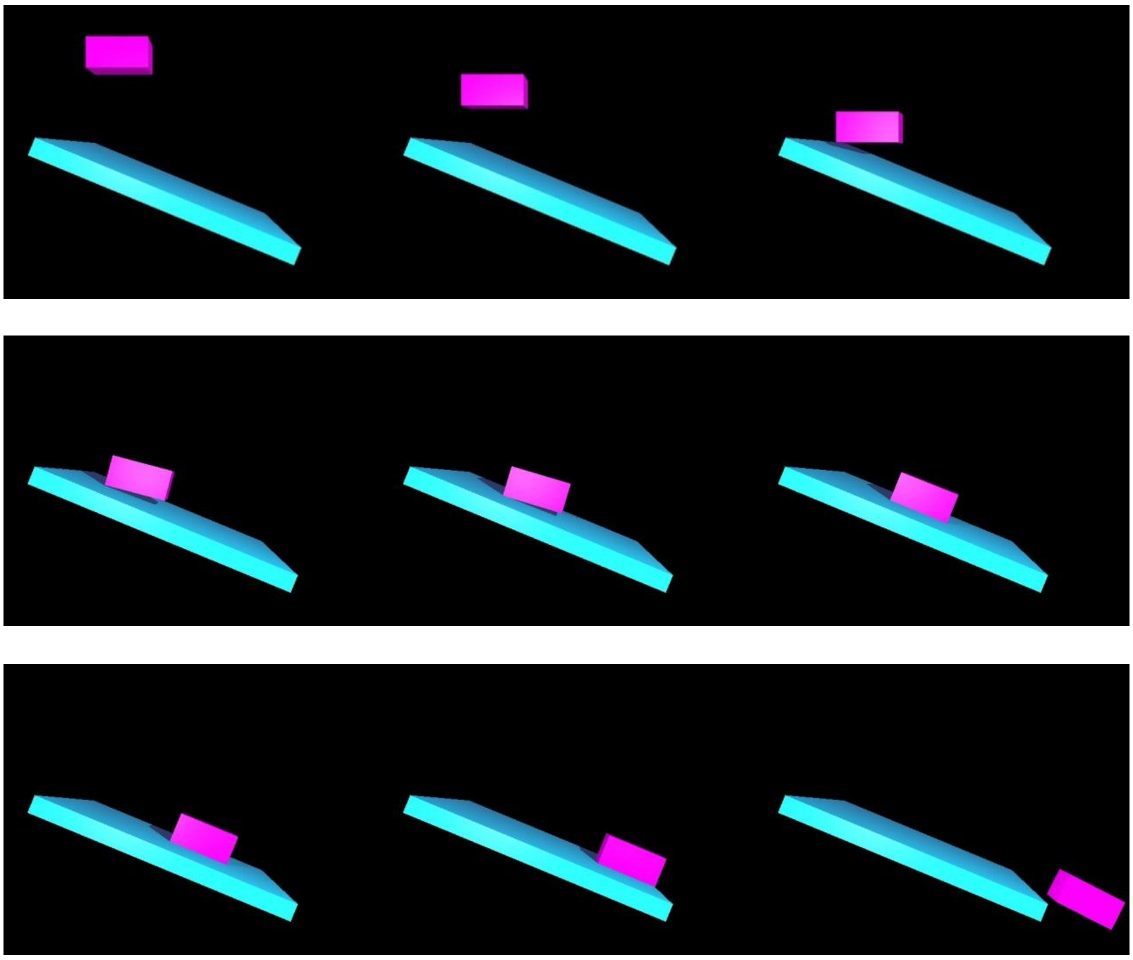


Figure 3.13: Sliding contact between two physical objects

It can be seen that for the above rectangular object in Figure 3.13, the contact was complete over the sliding face of the object. The collision shows several rotational bounces of the object on the rigid surface until it settles. The sliding motion is driven by the force of gravity smoothly over the frictionless surfaces. Increasing the surface friction coefficient would merely slow down the sliding velocity.

3.5.3 Wrapping by segmentation

Another important issue to examine here is the muscle and tissue wrapping by segmentation. A number of theories have been proposed for muscle wrapping; however, the best way of doing this is by finding the minimum potential energy of a spring (tissue) between two points over a surface. The shortest tissue path has the minimum level of potential energy and this is the reason that stretching springs come under consideration. Achieving minimum potential energy is possible by stretching tensioned springs between these two points and allowing them to slide over the surface to settle at the lowest energy level, thus finding the shortest path. The simulation of this technique is carried out by dividing the tissue into segments of mass spring systems, as in Figure 3.14 and 3.15, where the masses are not permitted to penetrate the wrapping surface. This has also been tested for basic geometry to examine the accuracy and feasibility of this technique before it is applied to muscle and tissues in the articular joint model, as shown in Figure 3.14 and 3.15.

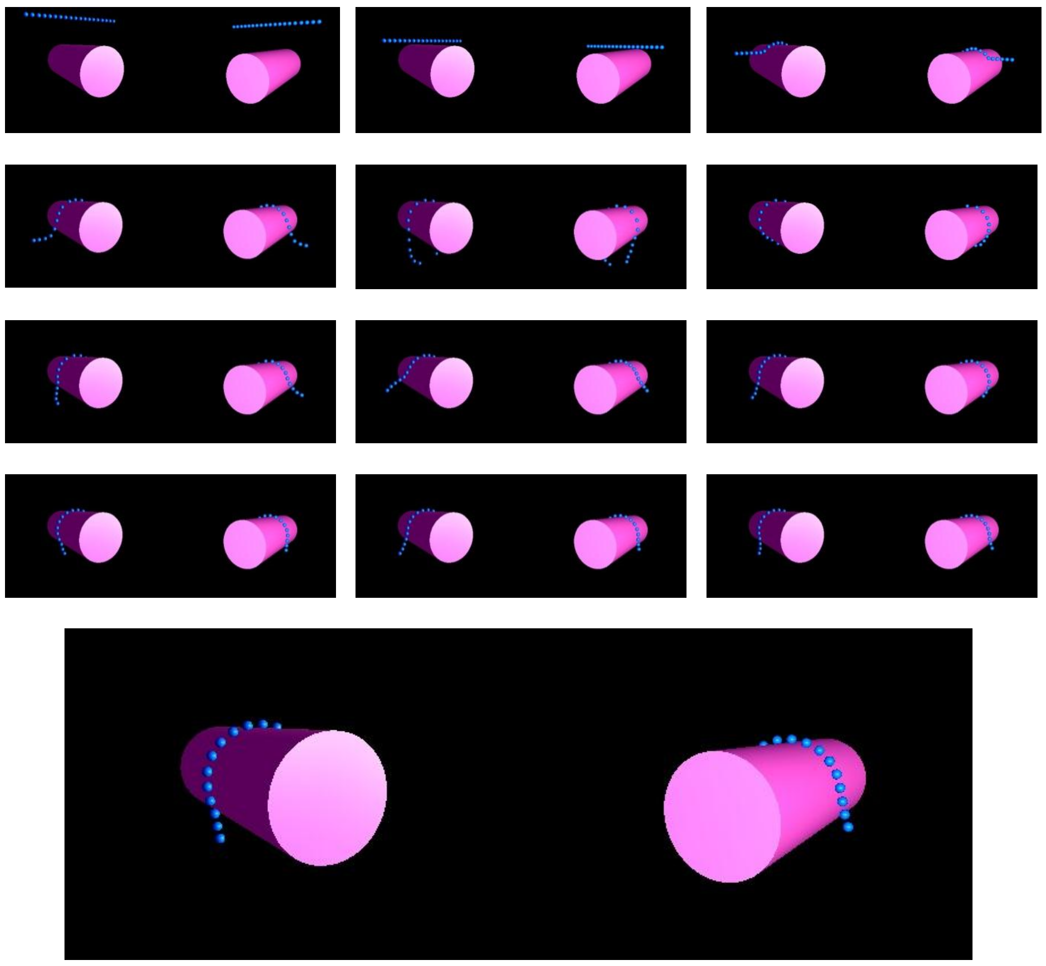


Figure 3.14: Snapshots of muscle and tissue wrapping by segmentation

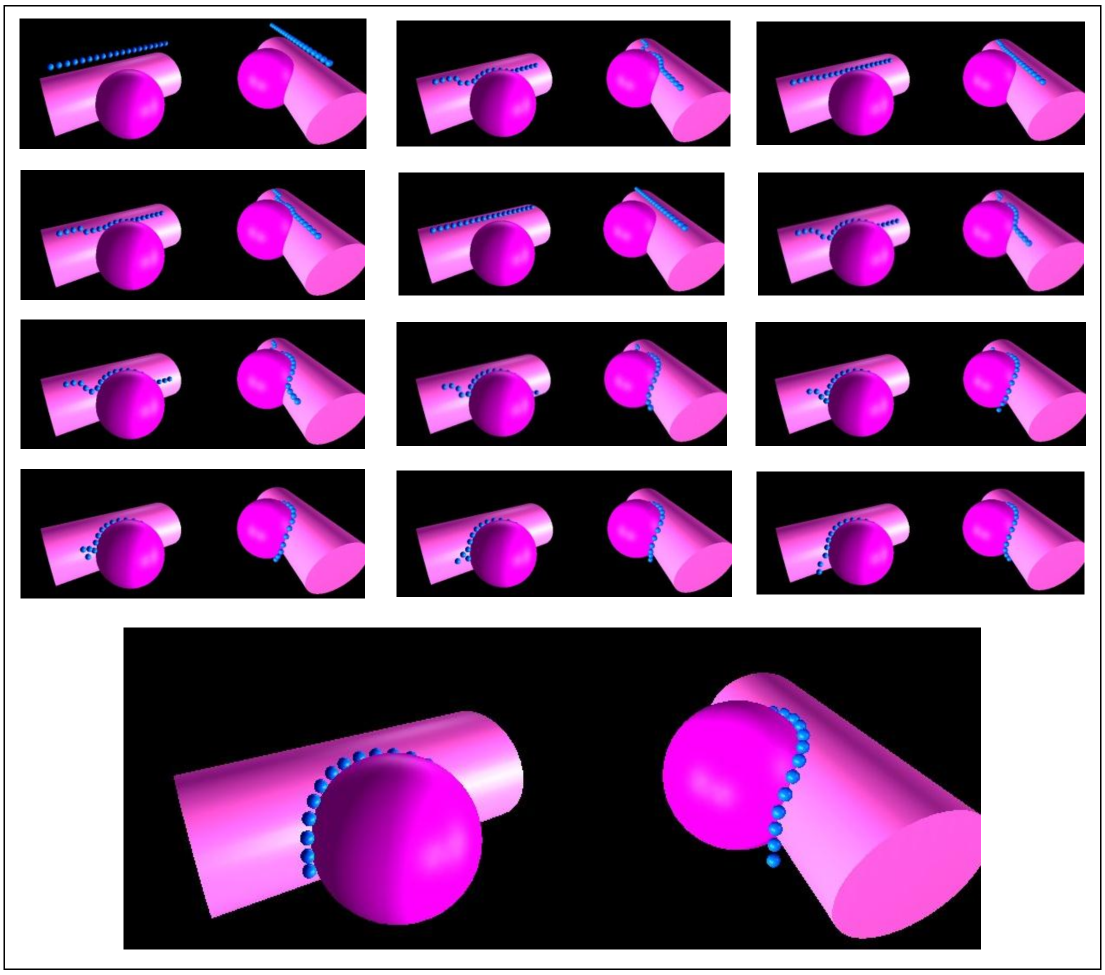


Figure 3.15: Snapshots of muscle and tissue wrapping by segmentation

The wrapping technique works efficiently in real-time simulation. The demonstration carried out here was under the natural force of gravity for a system of masses (blue spheres) connected together by springs with stiffnesses of 0.5 *kN/m* and masses of 100 grams each. Damping was also introduced to exhibit the viscoelastic behaviour of the tissues and provided the system with extra stability. Spherical shapes were selected to be the unit of the masses because the sphere is rotationally invariant and therefore the collision detection and response can be performed faster, taking into account that a single muscle or tissue consists of a number of spheres connected together by springs. A higher number of segments would result in a smoother wrapping; the converse would also be true, but more computations would then be required. In addition, as the masses are connected by springs, it is important for the gap between the masses not to expand too much to avoid the springs becoming overstretched and therefore passing through the meshed surface. This can be dealt with by increasing the number of masses or increasing the stiffness of the springs, or both, to ensure that there is surface contact at all times. It is preferable for surface friction to have the lowest possible value in order to allow the masses to slide freely on the frictionless surface and to maintain an accurate shortest path for the muscle or tissue. The preliminary validation of this technique shows a promising approach for muscle and tissue wrapping, despite being a simple and non-invasive method.

The preliminary assessment of the physics engine and gaming platform has uncovered a convenient way for modelling and simulating a mechanical system by implementing a physics engine and gaming platforms. The virtual experiments carried out in this chapter have demonstrated the reliability of such platforms. However, further evaluation is carried out in the next chapters by developing a musculoskeletal system based on medical data.

CHAPTER 4

THE DEVELOPMENT OF MUSCULOSKELETAL MODEL

4.1 Introduction

The preliminary assessment of the physics engine (PhysX) and gaming platform (DX Studio) carried out in Chapter 3 has shown the appropriateness of these platforms for further investigation in orthopaedic applications. In this chapter, it is intended to construct a musculoskeletal model of an articular joint on a gaming platform, which will be subjected to further evaluation. The human articular joint model will be provided with the natural 6 DOF, which is not an option for other joint modellers because of the model complexity and the associated challenges when dealing with 6 DOF. The articulation is driven by contact geometry and surrounding tissue stiffnesses to obtain joint motion which is as natural as possible. The developed musculoskeletal structure in the simulation environment has to be provided with its surface geometry and some physical attributes, such as mass and the coefficient of restitution for the contact surfaces. The dynamic behaviour of the musculoskeletal model is achieved by defining each component of the joint as a physical object. The centre of mass, moment of inertia and the principal axes are computed automatically and therefore these will determine the velocity, orientation and positioning of each physically defined object. The proposed framework of the model development is illustrated in Figure 4.1. It shows the steps required to create and manipulate the musculoskeletal model on a gaming platform. Muscle and tissue wrapping are constructed by segmentations, because this technique has shown a promising approach in the preliminary evaluation in section 3.5.3. Kinematics analysis, including the instantaneous centre of rotation and valgus-varus motion, as well as moment arm determination of the elbow joint, are also presented in this chapter.

4.2 Framework

The musculoskeletal joint model is developed as illustrated in Figure 4.1. This chart shows the method used for the musculoskeletal model development and simulation of articular joint based on anatomical surface information.

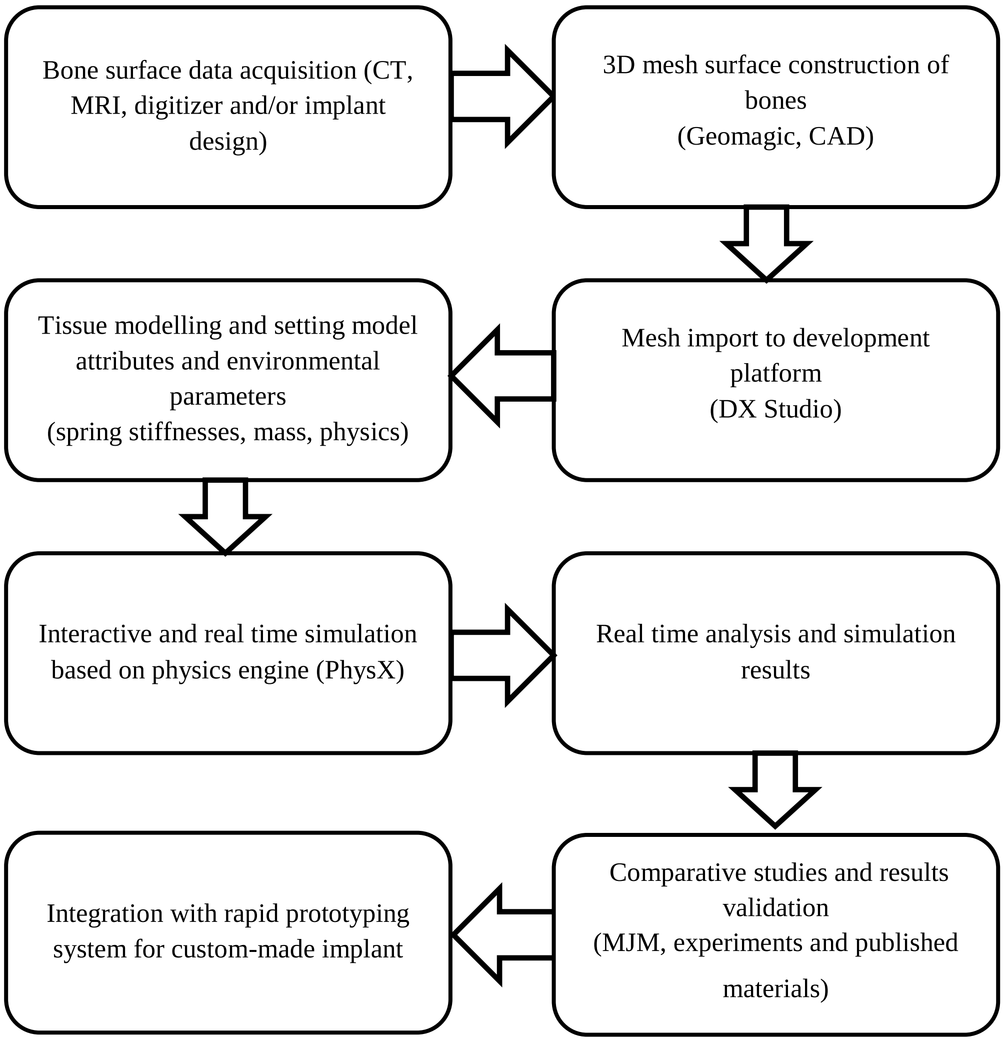

Bone surface data acquisition (CT, MRI, digitizer and/or implant design)
3D mesh surface construction of bones (Geomagic, CAD)
Mesh import to development platform (DX Studio)
Tissue modelling and setting model attributes and environmental parameters (spring stiffnesses, mass, physics)
Interactive and real time simulation based on physics engine (PhysX)
Real time analysis and simulation results
Comparative studies and results validation (MJM, experiments and published materials)
Integration with rapid prototyping system for custom-made implant

Figure 4.1: Framework for the development of the physically based model

Subsequent sections in this chapter will describe in detail the steps and issues involved in creating the musculoskeletal joint model.

4.2.1 Human musculoskeletal system

The proposed physically musculoskeletal joint is modelled as an unconstrained articular joint provided with 6 DOF mobility. As the primary objective here would be the analysis of the joint articulations, driven by contact surfaces, it is crucial to have rich geometrical details for the contact surfaces. As noted above, the musculoskeletal system is made up of several types of joint, such as synarthroses (immovable), amphiarthroses (slightly movable) and diarthrodial joints (highly movable). With this classification, every type of joint can be categorized in accordance with common standard kinematic mobility. As an example, slightly movable joints are supposed to have less movability than diarthrodial joints. In spite of this, in the this thesis, the motivation is to focus on giving every joint, including those with limited mobility, the full 6 DOF. The level of joint mobility is influenced through the ligaments as well as the muscle tendons surrounding the joint. Nevertheless, the appropriate modelling of the joint contact geometries and setting up of adjacent joint tissues is without doubt a time intensive procedure; hence the performed investigation is limited to one type of highly movable joint (diarthrodial), namely, the elbow joint. With regard to the elbow joint, the movements tend to be tagged by flexion extension, varus valgus and supination pronation motions. Because of the characteristics of joint modelling presented in the present research, mobility is identified with three dimensional translations together with three dimensional rotations, hence 6 DOF.

4.2.2 Bone surface construction

The surface geometries of the joint bones are required to construct the articular joint model. Former research workers at the Brunel Orthopaedic Research and Learning Centre have studied the cadaveric limp and digitized the bones by using the Faro Platinum Arm. The digitizer output is an ".iges" file which contains a cloud of points distributed to compose the particular form of the scanned bone. The ".iges" cannot be directly imported into the modelling platform since the extension ".iges" is not readable. Consequently, an ".iges" data file was opened in a Geomagic Studio 9 (Geomagic, Inc., North Carolina, USA). A Geomagic Studio enables the user to digitally rebuild an unlimited range of real-world shapes from scanned 3D data and produce a precise model which represents either the design objective or the as-built component. Geomagic Studio can accelerate the design process by using re-engineering, artefact design, engineering analysis, fast prototyping, mass customization and digital archiving. In Geomagic, the humerus was something like what is presented in Figure 4.2. At this stage, the humerus showed a great deal of noise and scatter dots which had to be cleaned off and removed before all the points could be connected together and the actual surface generated by wrapping the dots together. The first thing to do was to clean off and remove these unwanted points from the current geometry. The process commences by importing the particular raw data file to the Geomagic Studio 9. Usually the raw data sets are acquired in various conditions and furthermore, every bone may have in excess of thirty thousand points upon its surface area.

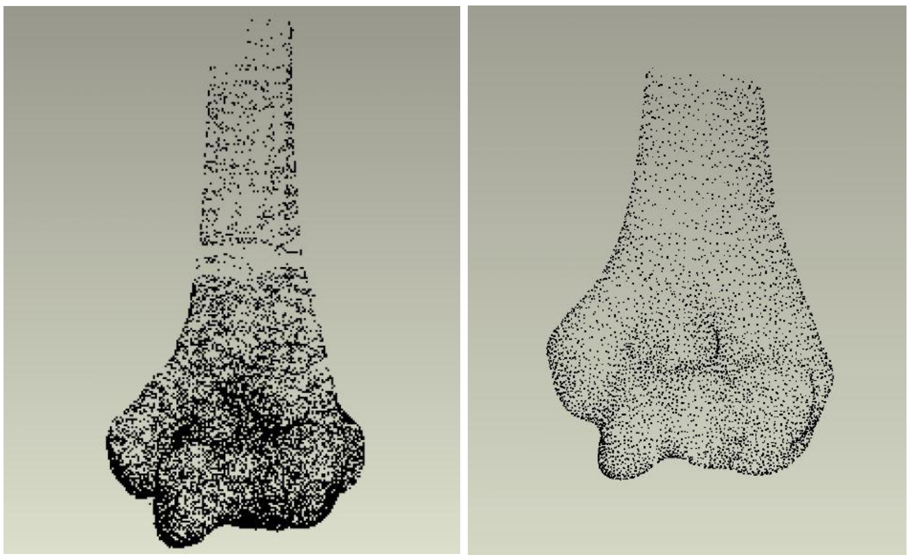


Figure 4.2: Digitized humerus in Geomagic Studio 9 before (left) and after (right) noise reduction

Throughout the cloud are many unwanted points identified as noise. Most of the disconnected points are removed by simply using reduce noise functions, which in turn help to eliminate undesirable points outside the bone surface.

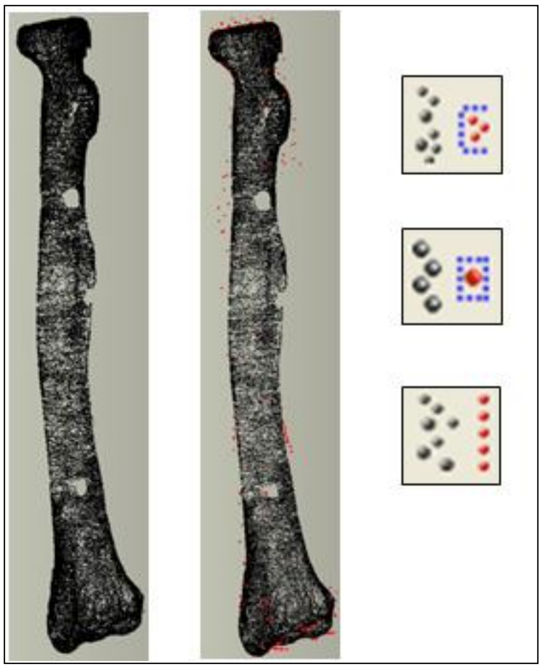


Figure 4.3: Removing disconnected points with reduce noise functions

Afterwards the wrapping feature may be used so as to generate the surface triangles with recommended point spacing. The automated triangle generation also relies on the point cloud at which smoother surface areas tend to be attained through denser point clouds. Following the generation of the surface triangles, quite possibly gaps in the surfaces will tend to result, which may be the outcomes of bone surface areas which have not been successfully scanned. The function called Fill-Hole is needed here to occupy present openings over the surface area of the bone fragments. If the cases of holes are simple in shape, the usual Fill-Hole function is acceptable; however, when the surface condition is extreme or where the basic Fill-Hole function is simply not adequate, different solutions can be combined with Fill-Partial, Create-Bridge and the Clean-up and Move options. Furthermore, unsuitable surface parts or improper wrapping are often reviewed by manually deleting unwanted surface parts and filling the missing holes with the enhanced fill functions. After applying the advanced Fill-Hole options such as the Create-Bridge option, the created surface area may possibly have straight line connections all-round the outer surface. Such surface problems may easily be smoothed by means of Sandpaper options. This operation makes it possible to thoroughly clean the parts of the outer surface which require it, by fine-tuning the level of smoothness. Once the desired surface area is attained without any problems, the created mesh data can be transformed in a number of data files (e.g. .stl, .wrl, .obj, .x etc…) in order to be used in many designs, modelling, game development platforms and various other applications. These kinds of data file are recommended when the triangulated surfaces are sufficiently refined to represent the object surface. In the present research, the platform used to create the musculoskeletal model is capable of supporting several file formats; in this case, an “.x” and “.dae” file extensions with a triangulated surface mesh were found to be suitable. In order to enhance the processing time and optimize the heavy computations during runtime, the number of triangles had to be carefully considered for optimal performance. Less mesh density (fewer triangles) means less surface quality but improved processing time; however, the accuracy and reliability of the analysis may be directly influenced in cases where the analysis relies upon the smoothness of the surface level. The balance between these variables should be carefully considered. In order to deal with this issue, the number of triangles in the bone surface was reduced using the Decimate Polygons feature. As a result, the number of triangles of the bone surface was optimized by providing contact surfaces with a higher mesh density than the bone surfaces far away from the joint, where surface smoothness is not an issue. Once performing the specified functions for each segment, the initial 75000 triangles of radial bone surface were reduced to 7000 triangles with no considerable loss of smoothness. Nevertheless, with regard to many software systems, mesh data need additional post-processing for creating patches and splines to create some control point throughout the mesh surface. The Nonuniform Rational Basis Spline often called the NURBS is widely used in computer graphics for representing curves and surfaces with excellent flexibility and also accurate controlling for the simulation of a geometrically based deformation. These treatments and others were performed for the ulna, humerus, radius, scapula and clavicle bones.

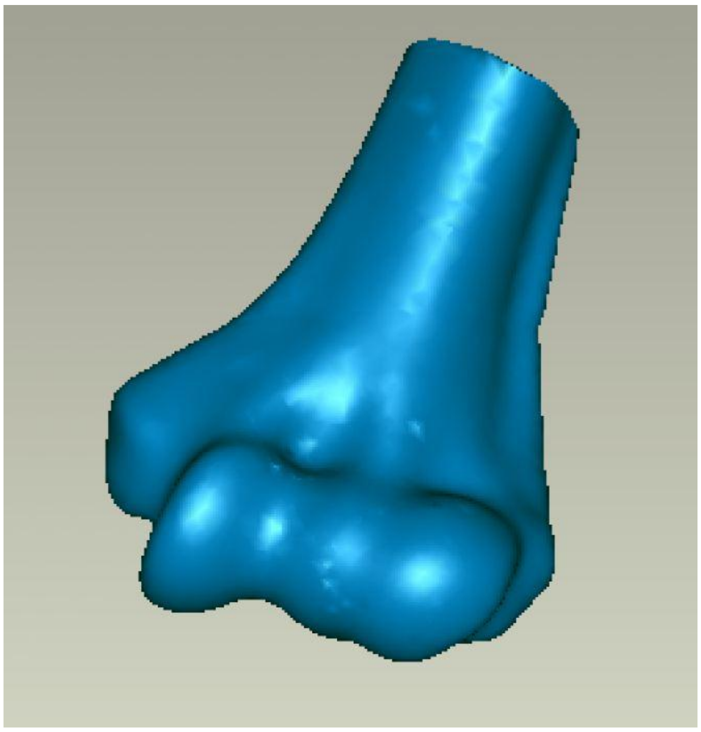


Figure 4.4: Final shape of the humerus

When the bones were finalized in Geomagic, the image was saved as a ".dae" file and another file for ".x", in formats which are compatible with DX Studio. (DX Studio can read other file formats as well). Figure 4.4 shows the humerus in its final form, when it was ready to be transferred to DX Studio.

In this research the analyses mainly focus on the elbow joint. With the particular proposed 6 DOF and surface geometry based joint analysis for the articular joint, bone segments of the elbow joint complex were developed simply as unconstrained objects.

4.2.3 Model development in DX Studio gaming platform

4.2.3.1 An overview about the DX Studio

In DX Studio the different scenes are organized and managed by the DX Studio document, which is the top level and consists of one or several 2D and 3D scenes. Each scene has its own variables, attributes, graphics, objects and scripts. The scenes within the document can be displayed as layers overlapping one another or in sequential layout. Every scene within the document possesses its own DX Studio editor.

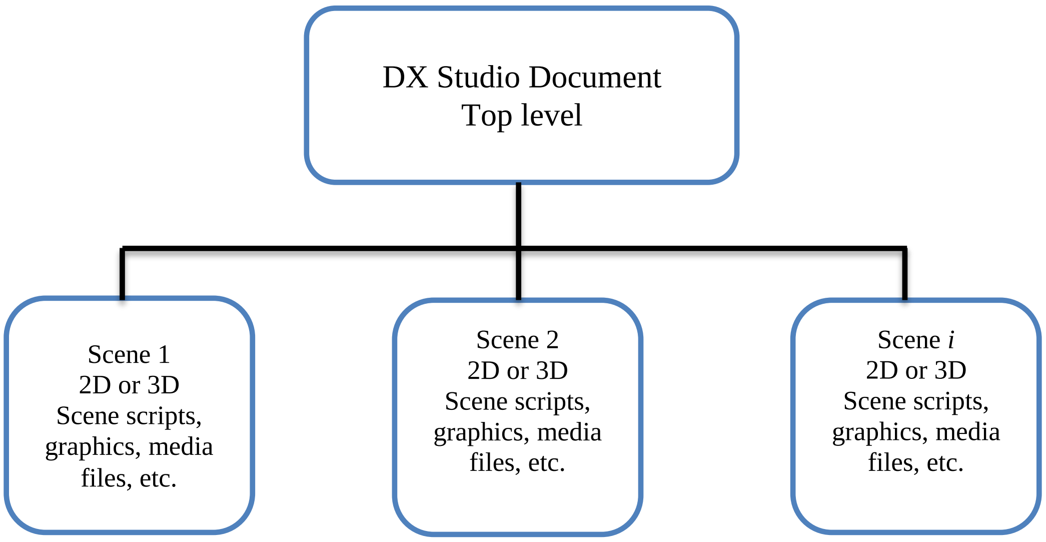


Figure 4.5: DX Studio organization layout

The DX Studio Editor

The DX Studio Editor stands out as the principal built-in development platform for constructing the document. It enables the user to import, export, build and identify the way that the user is interacting. The script in the DX Studio editor is based on

ECMAScript, which is also known as JavaScript. Scripting with the editor has a tolerating capacity and extended customization in developing the simulation environment.

DX Studio Player

The purpose of the DX Studio player is mainly viewing the document. Its low lever C++ code allows it to run in most environments without the need for .Net Framework. It accepts ".dxstudio" file extensions for playing.

Redistributing with DX Studio

Execution files generated from a DX Studio document can easily be distributed for non-commercial use; however, other distributions may require registered a DX Studio version.

Importing and building the model in DX Studio

Several types of file format are supported for mesh import (.dae, .x, .fbx). As regards the elbow, the 3D mesh file for the humerus, radius and ulna were imported from ".x" files each with 1000 faces and 3000 vertices (Figure 4.6).

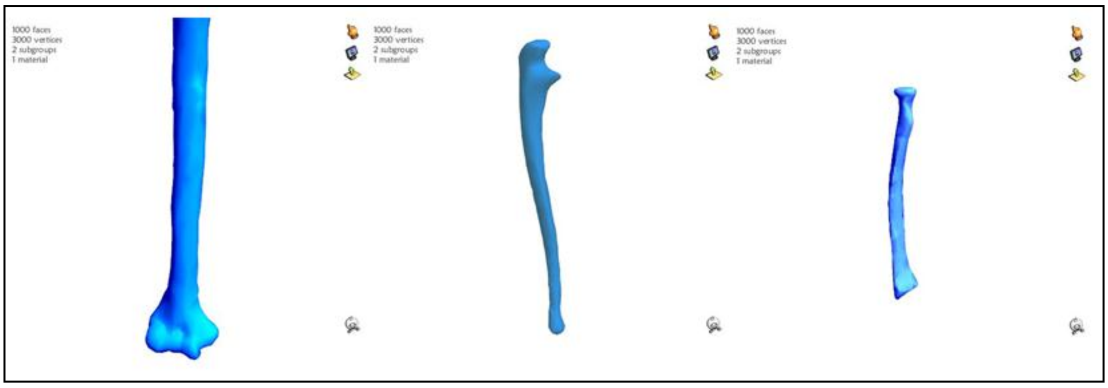


Figure 4.6: Imported mesh files (humerus, ulna and radius)

The meshes then have to be assembled in anatomical order, as in Figure 4.7.

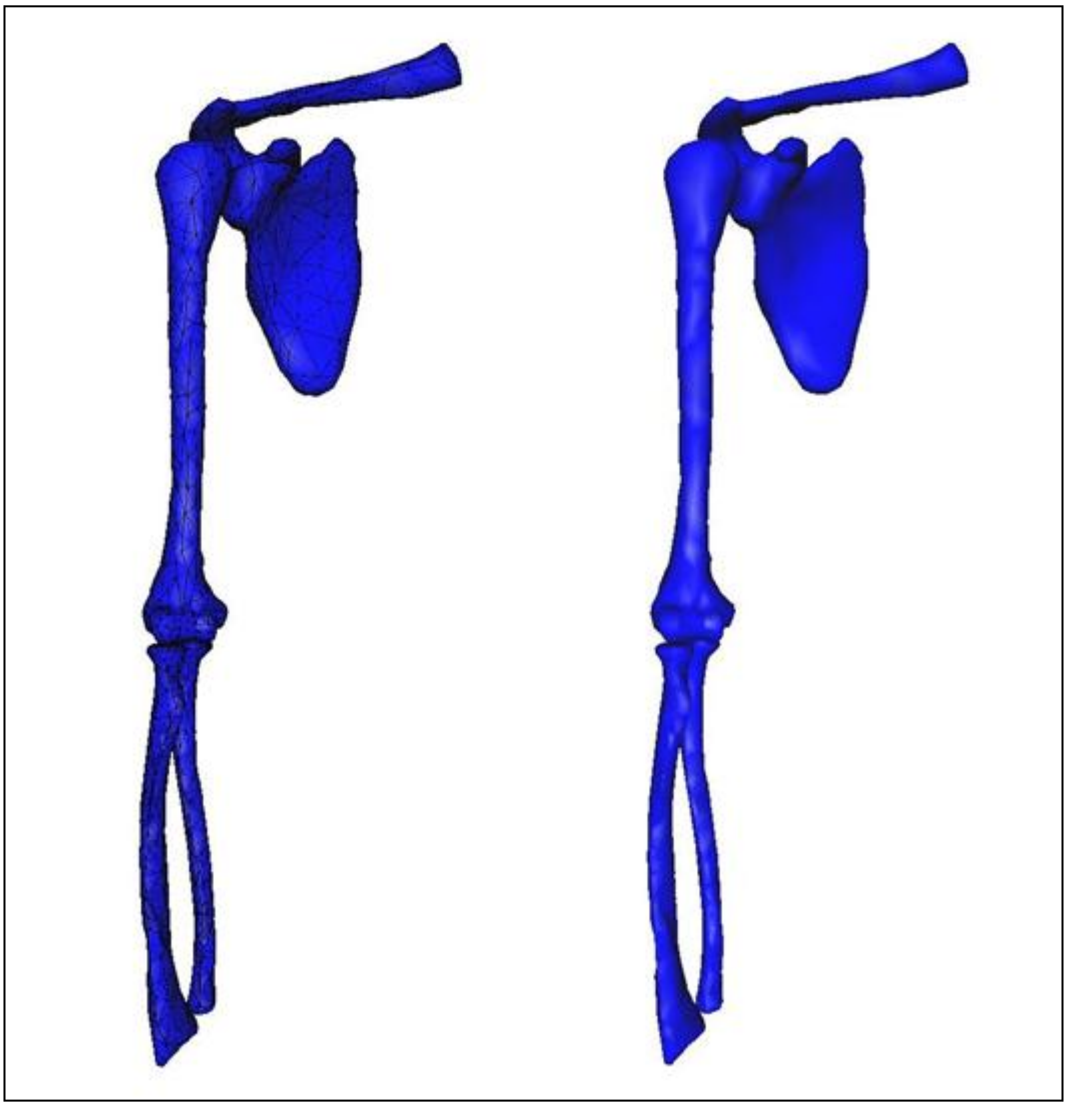


Figure 4.7: Assembled complete right arm mesh.

Then the objects are identified as physical objects to enable the physical properties to be added and recognized by the physics engine. Once these are in place, the mesh of the object is passed through to the physics engine, PhysX. For a particular object, this can be performed with the following script:

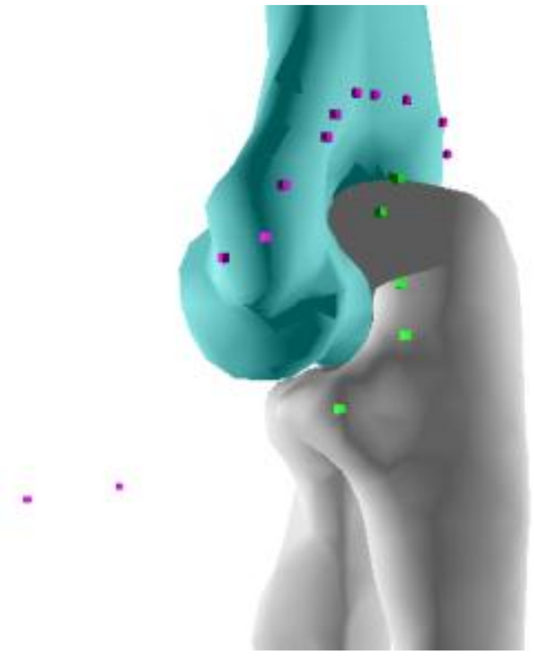


Figure 4.8: Assembled right arm with the insertion points

```
function onInit()
{
    object.physics.enable=true;//switch on physics
    object.physics.friction=0.1;//friction coefficient
    object.physics.setDamping(0.5,0.5);//The amount of linear and angular damping to apply.
    object.physics.useRotation=true;//Allow rotation we well as translation
}
```

Physical parameters may be added for the objects, such as mass or density, the coefficient of restitution, constraints and others, as shown in the scripts in the Appendix A1.

4.3 Stiffness Configurations with Mass-Spring Model

A crucial challenge when modelling the musculoskeletal structure with mass-spring systems is usually the selection of appropriate spring stiffness. A proper stiffness value may enhance the reality of the physics based simulation and allow the model to exhibit the behaviour of the involved material more accurately. A mass-spring model mimics a continuous real shape with a finite pair of masses and the springs connecting them. The displacement of the masses in the mass-spring system illustrates the deformation of the body which is caused by the applied forces. But because of the discrete digital computations of the mass spring systems, configuring such parameters for the physically based modelling of the musculoskeletal system is not apparent as a procedure. Fine-tuning the stiffness values of the mass-spring systems may require several trials before a realistic simulation response is achieved. However, this approach is time consuming. Other heuristic methods have already been proposed with a view to figuring out the stiffness parameters through the use of a precise simulated deformation. Optimization approaches could be employed to find the required stiffness parameters, for example, genetic algorithms (Louchet et al., 1995), simulated annealing (Deussen et al., 1995) and evolutionary algorithms (Bianchi et al., 2003, Bianchi et al., 2004). Nevertheless, establishing optimization types of procedure demands special expertise as well as very careful design to carry out the parameter configuration settings.

As discussed above, ligaments were modelled as segmented spring mass systems. Although the masses are set to small values, the presence of the masses along the ligaments is introduced here to maintain a good wrapping around the joint capsule.

At an earlier stage in the creation of this model, the articular joint simulation experienced very unstable conditions, due to many factors such as the interpenetration of objects, surface stiffness and undamped spring excitation. Figure 4.9 (b) below shows a snapshot from the model development at an early stage. Cylindrical shapes were used to model the mass segments which were replaced later with spherical shapes; see Figure 4.9 (a). Spheres are rotationally invariant and the wrapping was more effective because the orientation of the mass segment was no longer an issue.

Muscles were modelled as mass-spring systems just similar to the ligaments; however, springs in the muscle are active only when acquired and inactive when released. Damping is introduced to muscles to provide stability and smooth motion under variable muscle loading and unloading. Joint motion is caused by muscle activation, which can be done by activating the springs. Those forming the muscle should be active at certain timed or whenever the user clicked on ‘activate muscle’. Deactivation is similarly performed. Each muscle can be activated and deactivated independently.

As explained above in section 3.5.3, muscle wrapping is carried out by dividing the whole length of the muscle into mass spring segments. Each segment consists of mass-spring elements. The reason behind this is to avoid any penetration that might occur between the muscle and the bone surfaces and good wrapping is performed to prevent this.

The stiffnesses of the springs and the damping values used to connect the segments together are provided by the following script for each muscle and ligament segment:

```
var Km;

var Dm;

function onInit()
{
Km  = 50;

Dm = 25

scenes.scene_1.objects.seg_12.physics.constraintType = point

scenes.scene_1.objects.seg_12.physics.constraintPosSpringStrength = Km;

scenes.scene_1.objects.seg_12.physics.constraintPosSpringDamping = Dm;

}
```

Complete scripts and more functions and events for setting the segments together are included in Appendix A1 of this thesis.

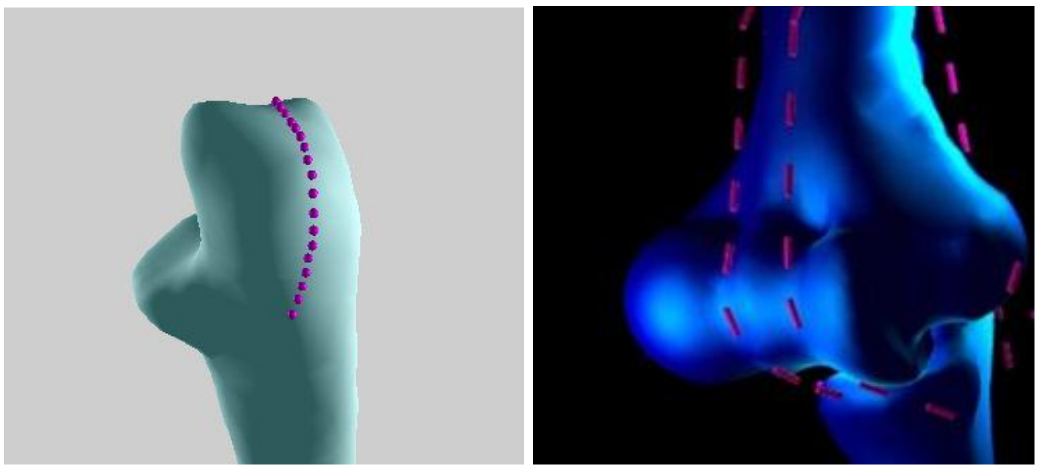

(a) (b)

Figure 4.9: Muscle and tissue wrapping

In the physics engine used here, penetration between physical objects is not allowed and therefore, masses along the line of action will prevent penetration and keep the muscle wrapped around any physical object. More mass-spring segments mean better wrapping. However, more physical objects in the scene require more processing time and heavier computation. Muscle forces are calculated on the basis of spring elongations. As friction is considered to be set to zero, it is assumed that the muscle under tension maintains the same force on all segments. Therefore, calculating the spring elongation between any two successive masses beside the known spring stiffness should provide the amount of force in the muscle.

4.4 Collision Detection and Collision Response

Since the articulation of the proposed multi-body model of the joint is based on the contact geometry of the bones, contact and collision are considered here in order to obtain an accurate simulation response. In the current study, the collision handling is implemented with three methods; these can work simultaneously.

The first method (*onPhysicsCollision* and *physics.useForCollisions*), the physics engine, PhysX from Nvidia, provides the physical behaviour of the simulation environment automatically and internally. As well as collision detection, PhysX supports full collision response. However, the user does not have much control over the simulation environment and what may be happening next.

The other method (*Events.onCollision (ObjectID)*) is only used for detecting collisions based on the collision detection system which is independent of the physics engine and based on the AABB method combined with the partitioned space tree. The built in system in the framework can generates system events if any collision is detected with any particular object. In this method, the event can be acquired by script to indicate collision. Although the collision detection here is very efficient, the system does nothing to separate the colliding objects. The collision response is not automatic and the user decides what to do when two meshes intersect. Thus, a collision response algorithm is required.

The Raycasting system is another method implemented in the framework to determine objects which might possibly make contact. In this technique, a contact point is determined on the basis of an arbitrary line intersection with any object in its path. The collision can be checked between any meshes in the scenes with *Object.notifyCollisionCheckFaces* and *onNotifyCollision* event. With these events set to true, a face to face check test is performed and any possible intersection reported. Raycasting offers the maximum control over the simulation environment; however, this is not the concern of the present research. These three methods work together and in this way improve the simulation with optimized computations.

4.5 Joint Kinematics

4.5.1 Instantaneous Centre of Rotation, COR

This section illustrates the method implemented for finding out the instant centre of rotation of the articular joint within the 3D space. Finding the centre of rotation has to be formulated since DX Studio does not give it. When tracing a certain point on the moving bone segment of the joint in the 3D space, provided that the other bone segments are stationary, 4 points are needed to carry on the calculation for the sphere equation for a specific time step.

Four coordinates of the traced point recorded during joint motion are as follows:

Point Position 1: (x_1, y_1, z_1)

Point Position 2: (x_2, y_2, z_3)

Point Position 3: (x_3, y_3, z_3)

Point Position 4: (x_4, y_4, z_4)

From these four position coordinates, a temporary sphere may be formed and its centre is the instant centre of rotation of the articular joint. The equation of the sphere can be found by solving the next determinant.

$$\begin{vmatrix} x^2+y^2+z^2 & x & y & z & 1 \\ {x_1}^2+{y_1}^2+{z_1}^2 & x_1 & y_1 & z_1 & 1 \\ {x_2}^2+{y_2}^2+{z_2}^2 & x_2 & y_2 & z_2 & 1 \\ {x_3}^2+{y_3}^2+{z_3}^2 & x_3 & y_3 & z_3 & 1 \\ {x_4}^2+{y_4}^2+{z_4}^2 & x_4 & y_4 & z_4 & 1 \end{vmatrix} = 0 \qquad (4.1)$$

Related to the above determinant, the 4 points to be used in formulating the sphere have to satisfy some conditions (outlined below) in order to be valid for generating the sphere equation, otherwise, the sphere equation may not be undefined with none or several solutions, or an infinite number of solutions.

- No combination of more than 2 points may be collinear (passing the same line)
- The four points cannot be coplanar (on a same passing plane)

Next, the formula may be noted merely as:

$$(x^2+y^2+z^2)\begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} - x\begin{vmatrix} {x_1}^2+{y_1}^2+{z_1}^2 & y_1 & z_1 & 1 \\ {x_2}^2+{y_2}^2+{z_2}^2 & y_2 & z_2 & 1 \\ {x_3}^2+{y_3}^2+{z_3}^2 & y_3 & z_3 & 1 \\ {x_4}^2+{y_4}^2+{z_4}^2 & y_4 & z_4 & 1 \end{vmatrix} +$$

$$y\begin{vmatrix} {x_1}^2+{y_1}^2+{z_1}^2 & x_1 & z_1 & 1 \\ {x_2}^2+{y_2}^2+{z_2}^2 & x_2 & z_2 & 1 \\ {x_3}^2+{y_3}^2+{z_3}^2 & x_3 & z_3 & 1 \\ {x_4}^2+{y_4}^2+{z_4}^2 & x_4 & z_4 & 1 \end{vmatrix} - z\begin{vmatrix} {x_1}^2+{y_1}^2+{z_1}^2 & x_1 & y_1 & 1 \\ {x_2}^2+{y_2}^2+{z_2}^2 & x_2 & y_2 & 1 \\ {x_3}^2+{y_3}^2+{z_3}^2 & x_3 & y_3 & 1 \\ {x_4}^2+{y_4}^2+{z_4}^2 & x_4 & y_4 & 1 \end{vmatrix} + \qquad (4.2)$$

$$\begin{vmatrix} {x_1}^2+{y_1}^2+{z_1}^2 & x_1 & y_1 & z_1 \\ {x_2}^2+{y_2}^2+{z_2}^2 & x_2 & y_2 & z_2 \\ {x_3}^2+{y_3}^2+{z_3}^2 & x_3 & y_3 & z_3 \\ {x_4}^2+{y_4}^2+{z_4}^2 & x_4 & y_4 & z_4 \end{vmatrix} = 0$$

The above matrices may be also expressed as:

$$(x^2+y^2+z^2)\,M_{11} - x\,M_{12} + y\,M_{13} - z\,M_{14} + M_{15} = 0 \qquad (4.3)$$

The centre of rotation COR (x_c, y_c, z_c) and the radius from this centre to the centre of a body r_c moving in the 3D space may be found from the equation of the sphere shown below:

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = {r_c}^2 \tag{4.4}$$

Putting together these two equations leads to finding out radius r_c and the centre of rotation coordinate (x_c, y_c, z_c) may be written as:

$$x_c = \frac{M_{12}}{2\, M_{11}} \tag{4.5}$$

$$y_c = -\frac{M_{13}}{2\, M_{11}} \tag{4.6}$$

$$z_c = \frac{M_{14}}{2\, M_{11}} \tag{4.7}$$

However, radius, r_c, may be expressed as:

$${r_c}^2 = {x_c}^2 + {y_c}^2 + {z_c}^2 - \frac{M_{15}}{M_{11}} \tag{4.8}$$

Keep in mind that these formulations cannot be resolved whenever the matrix term M_{11} equals zero, since this refers to the non-quadratic terms whereby the four point positions are not fulfilling the earlier noted condition and so they are possibly collinear or coplanar.

Another method which can be used in determining the COR is from the angular and translational motions of the body. During the articulation of the joint, the relative motion between the bones is considered to calculate the instantaneous centre of rotation. If the moving rigid body combines angular velocity w with its translational velocity v, then an instantaneous centre of rotation may exist and can be calculated. From standard kinematic analysis, in order to find the centre of rotation from the angular and translational velocities, ω and V respectively, only the perpendicular translational velocity vector component, $V_{\perp}$, that is, to the angular velocity ω need be involved.

The distance to the centre of rotation, r_c, is given by:

$$r_c = \frac{|V_{\perp}|}{|\omega|} \tag{4.9}$$

where $V_{\perp}$ is the perpendicular component to ω , of the translation velocity vector.

$$r_{COR} = r_o + r_c \tag{4.10}$$

where r_o is the distance to the origin of the coordinate system.

4.5.2 Calculation of moment arm

The moment arm can be described by the perpendicular distance between the line of the acting force and the line passing through the centre of rotation and parallel to the line of action. The moment arm can be expressed by the relationship between the tissue or spring perpendicular elongation, $L_{\perp}$ and the related angular motion θ. Thus it can be expressed by:

$$MomenArm = \frac{L_{\perp}}{\theta} \tag{4.11}$$

Note that $L_{\perp}$and θ are perpendicular to each other.

CHAPTER 5

SIMULATION RESULTS

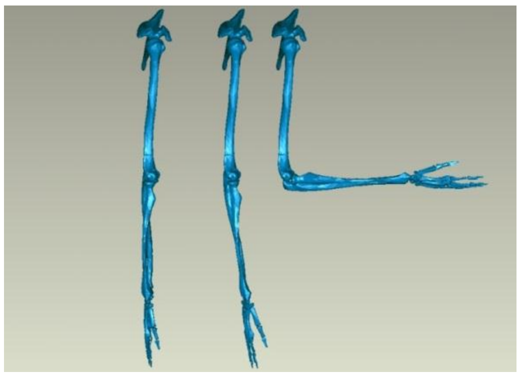
5.1 Introduction

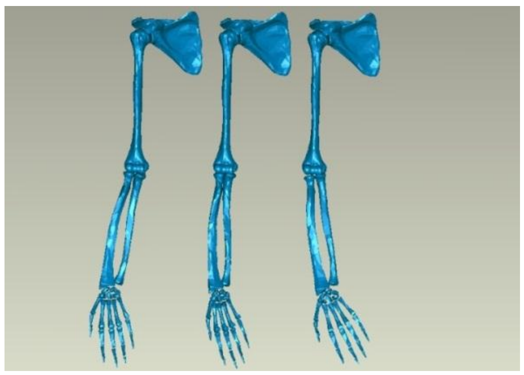
Paying particular attention to the modelling and simulation based on the physics engine, “PhysX”, the diarthrodial joints were modelled as multi-body systems consisting of the biological bones, tendons and ligaments, so as to form the musculoskeletal structure of the articular joint with no reduction in the DOF of the joint. The main innovation in such a modelling approach relies on implementing the latest developments in gaming industry to overcome many of the existing challenges involved in the modelling of articular joints.

Throughout the literature review, no equivalent technique was found for handling such a biomechanical problem, not even in general engineering applications. Moreover, the muscle wrapping technique by segmentation is another novel method for biological tissue, which also maintains a sensible line of action during joint articulation. Dealing as well as possible with some challenging problems, such as collision detection and collision response was also proposed, by implementing some of the state of the art algorithms in the current model, making use of the hardware acceleration for further acceleration. The virtual interactive simulation of the musculoskeletal model was performed on a Windows PC with an Nvidia graphics card GT240. This graphics card and many others from Nvidia support PhysX for parallel processing to lower the processing load from the CPU.

5.2 Case Study of the Elbow Joint

Although the proposed approach can be used with any joint in the musculoskeletal system, the elbow joint has been considered in this study, because this joint is interesting to investigate in many respects. Moreover, the availability of the medical data for the elbow joint made it possible to model with the current platform. First of all, the typical motion actions associated with the elbow joint are generally outlined as valgus-varus, flexion-extension and pronation-supination movements, as in Figure 5.1. From the literature, such elbow activities and range of motion of the elbow joint are illustrated in Appendix A3. Because some data obtained from the published material consider the joint to have a single DOF, it might not be valid to compare it with the results obtained in our model, which deals with the joint as a 6 DOF.


a) Flexion-Extension


b) Valgus-varus

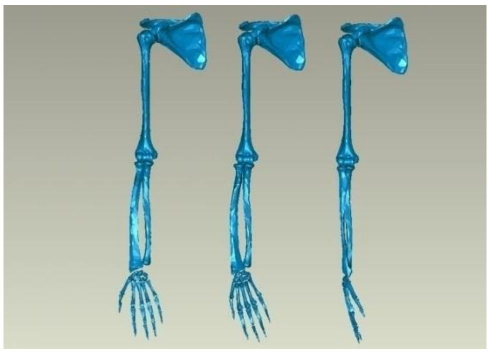

c) Pronation-supination

Figure 5.1: Human elbow joint movements

As mentioned above in Chapter 3, the muscles and ligaments were modelled as linear damped mass-spring segments. The muscles and ligaments consist of a finite number of masses connected by springs to assist wrapping and maintain accurately the shortest path. Although muscles and ligaments are composed of similar structures, ligament forces (passive tissue) were tending towards resting positions while muscles are only active when excited. The generated force due to the extension of the ligaments or the excitation of the muscle passes through all the masses, those forming the shortest path leading to the attachment point. However, the surfaces are set to be frictionless to maintain proper tissue wrapping.

The surface geometry of the bones is acquired by employing a mechanical digitizer and is followed by post geometric processing software (Geomagic Studio 9) to assist in creating readable 3D mesh data which have been imported to our modelling platform, detailed description can be found in section 4.2.2. Then the muscles and ligaments are constructed by the segmentation technique proposed and validated as in sections 3.5.3 and 4.3. The insertion points and properties are described below:

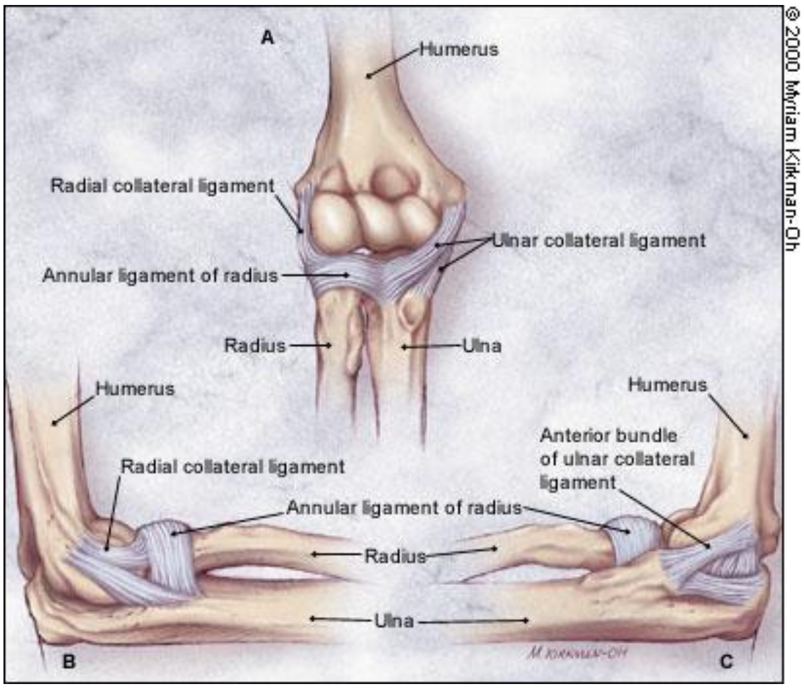

A
Humerus
Radial collateral ligament
Ulnar collateral ligament
Annular ligament of radius
Radius
Ulna
Humerus
Humerus
Anterior bundle of ulnar collateral ligament
Radial collateral ligament
Annular ligament of radius
Radius
Ulna
B
C
© 2000 Myriam Kirkman-Oh
M. KIRKMAN-OH

Figure 5.2: Ligaments of the elbow joint (AAFP, 2000)

(A) Anterior view (B) Lateral view (C) Medial view

The ligaments of the elbow joint are illustrated in Figure 5.2, the main ligaments being the:

- Annular Ligament (AL)
- Lateral Ulnar Collateral Ligament (UCL)
- Radial Collateral Ligament (RCL)
- Medial Collateral Ligament (MCL)

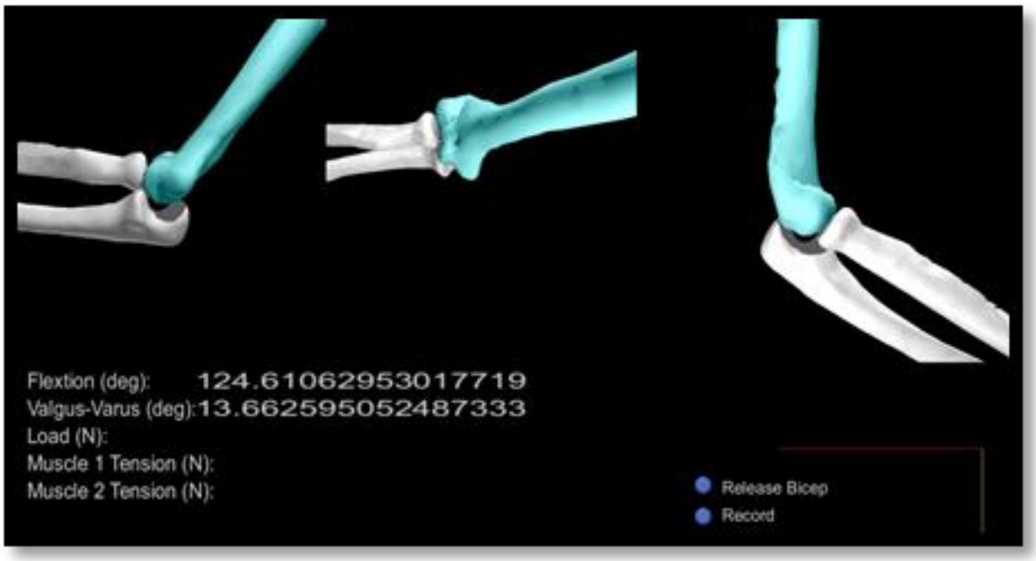
Similarly, the muscles acting through the elbow joint are:

- Biceps
- Brachialis
- Brachioradialis
- Triceps
- Pronator teres
- Extensor carpi radialis longus

The insertion points for these ligaments and muscle tendons are adapted from Tortota and Grabowski (2007), as in Appendix A3 and are attached interactively on the screen. Those ligaments and muscles were modelled as linear damped mass spring systems. In addition, during the flexion-extension movements, the humerus was fixed to the shoulder, allowing only the radial and ulna to articulate as well as enabling the primary muscles alone to move, adding to the effectiveness of this movement. The stiffness properties of ligaments are obtained from the literature and are listed in Appendix A3.

5.2.1 Flexion-extension and valgus-varus movements

The musculoskeletal joint model developed in Chapter 4 is placed for kinematic analysis. At first, the simulation based on the physics engine (PhysX) for the presented joint model is carried out for the elbow joint, mainly to study the flexion-extension movements of the intact elbow joint as in Figure 5.3 and Figure 5.4. The coordinates of some markers on the moving bones are stored in the software and immediately analysed to obtain the flexion angle and valgus-varus motion.



```
Flextion (deg):        124.61062953017719
Valgus-Varus (deg):13.662595052487333
Load (N):
Muscle 1 Tension (N):
Muscle 2 Tension (N):
```

Figure 5.3: Flexion extension movements of the elbow joint

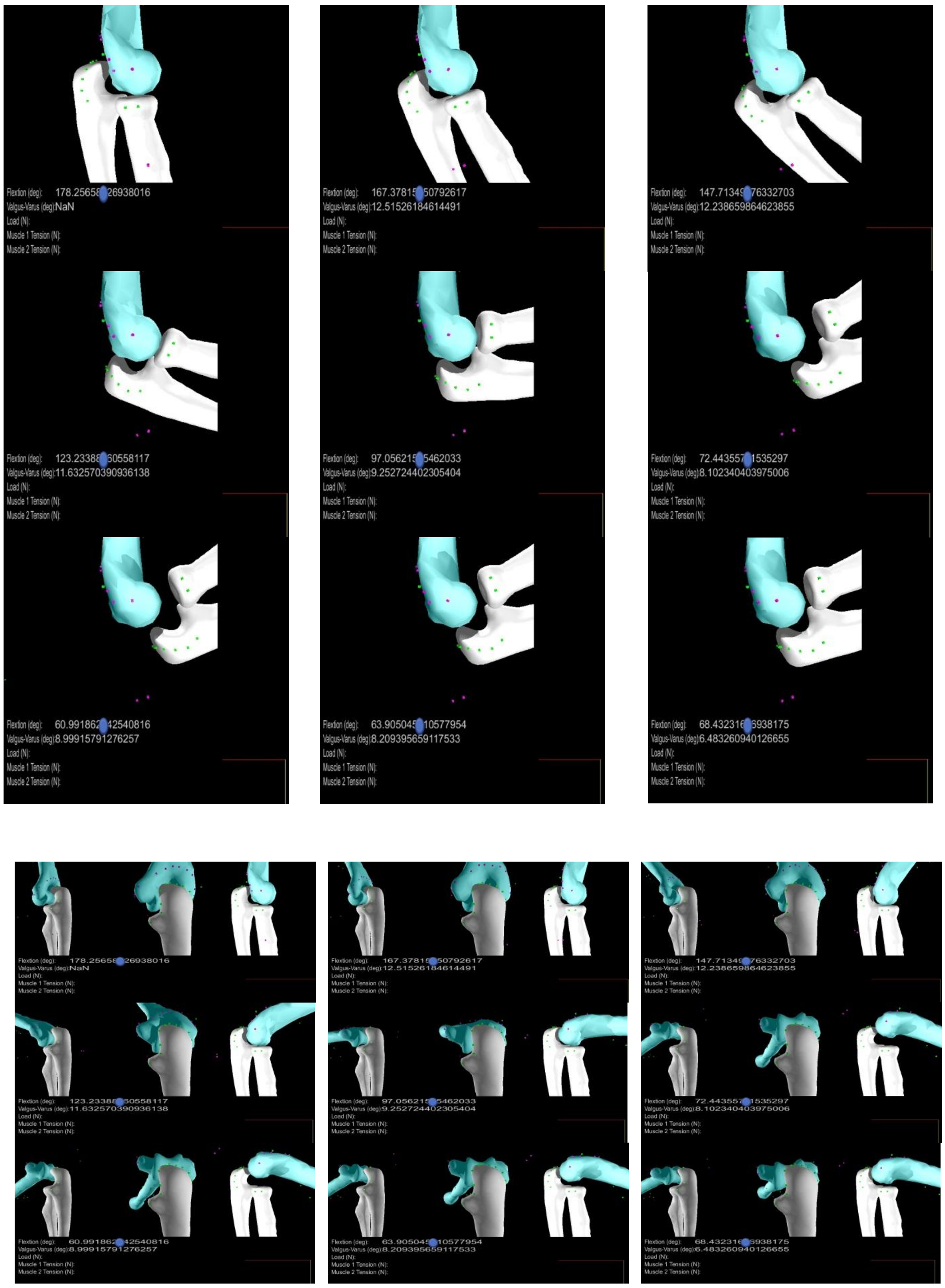

Flexion (deg): 178.25658[illegible]26938016
Valgus-Varus (deg):NaN
Load (N):
Muscle 1 Tension (N):
Muscle 2 Tension (N):
Flexion (deg): 167.37815[illegible]50792617
Valgus-Varus (deg):12.51526184614491
Flexion (deg): 147.71349[illegible]76332703
Valgus-Varus (deg):12.238659864623855
Flexion (deg): 123.23386[illegible]60558117
Valgus-Varus (deg):11.632570390936138
Flexion (deg): 97.056215[illegible]5462033
Valgus-Varus (deg):9.252724402305404
Flexion (deg): 72.443557[illegible]1535297
Valgus-Varus (deg):8.102340403975006
Flexion (deg): 60.991862[illegible]42540816
Valgus-Varus (deg):8.99915791276257
Flexion (deg): 63.905045[illegible]10577954
Valgus-Varus (deg):8.209395659117533
Flexion (deg): 68.432316[illegible]6938175
Valgus-Varus (deg):6.483260940126655

Figure 5.4: Snapshots of simulation of the elbow joint

The movement analysis starts by recording the flexion angle and the valgus-varus angle during joint flexion and extension without any varus or valgus load. Such motion may give an indication of the varus-valgus movement during this unloaded joint motion for an intact joint. The data extracted from the simulation are plotted in the following figure, showing that the valgus-varus range is within 9.5^o. The starting position was set to be at the full extension of the arm or flexion, 180^o.

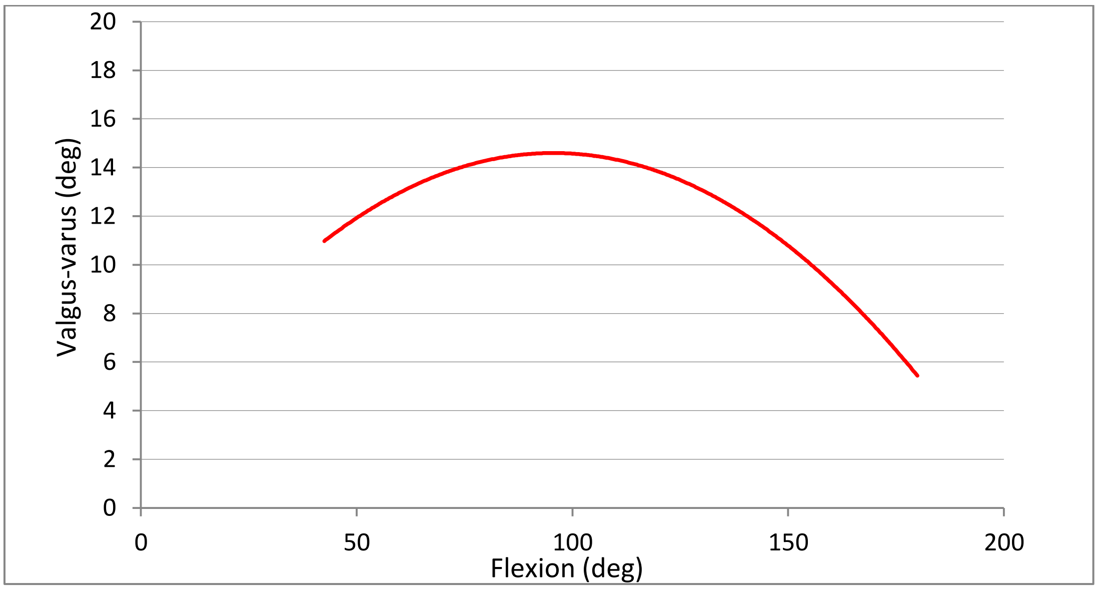
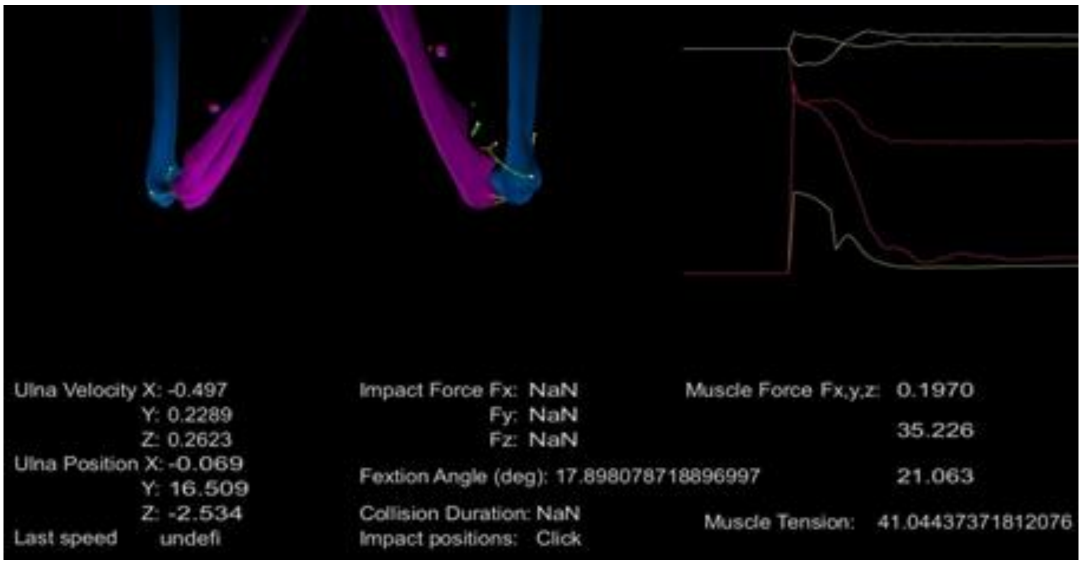


Figure 5.5: Valgus-varus motion during extension-flexion using the physics based method

From the plot in Figure 5.5 it can be seen that the valgus-varus deformation has a maximum value at flexion angle, around 90^o. Such a result indicates the natural behaviour of the elbow joint in the performed simulation. Real time analysis was carried out as shown in Figure 5.6. Interactive simulation with useful information on run time provides the user with a wide range of results with an interactive interface.

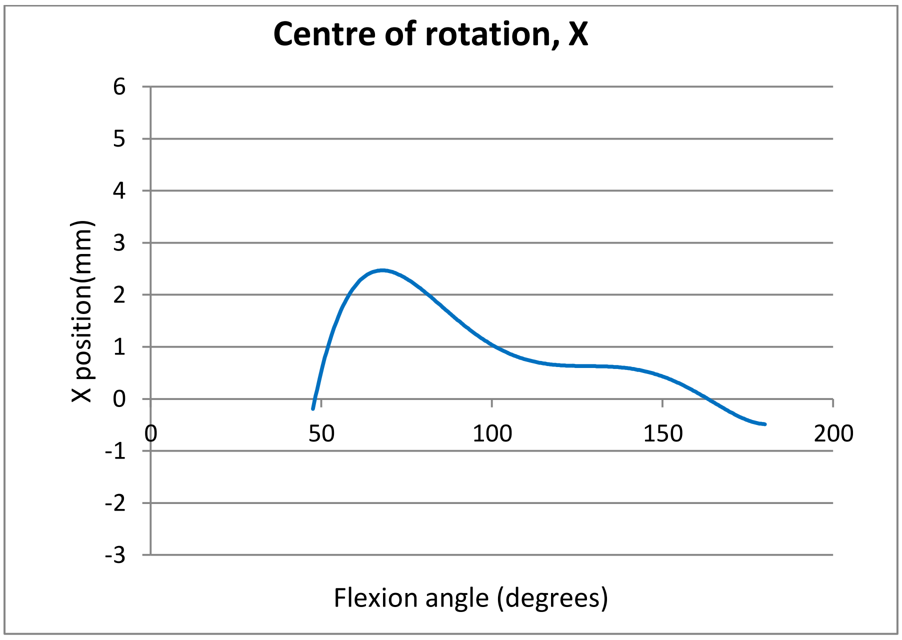


```
Ulna Velocity X: -0.497            Impact Force Fx:  NaN           Muscle Force Fx,y,z:  0.1970
              Y: 0.2289                         Fy:  NaN
              Z: 0.2623                         Fz:  NaN                                35.226
Ulna Position X: -0.069
              Y: 16.509           Fextion Angle (deg): 17.898078718896997                 21.063
              Z: -2.534
Last speed      undefi            Collision Duration: NaN          Muscle Tension:   41.04437371812076
                                  Impact positions:   Click
```

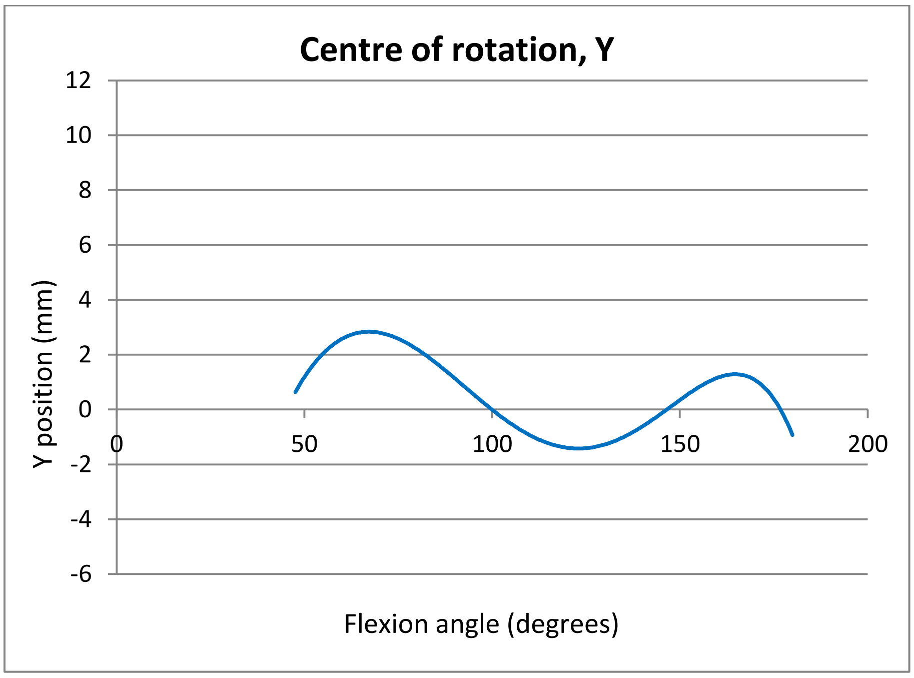
Figure 5.6: Real time analysis during joint simulation

5.2.2 Calculations of the instant centre of rotation

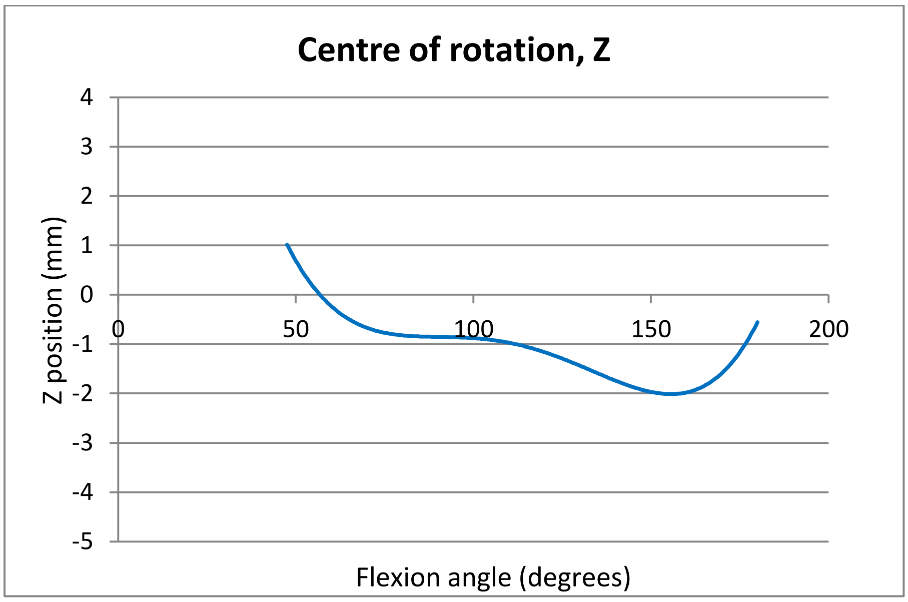
With the current 6 DOF model of the elbow joint, it is expected that the COR would be changed instantaneously for the contact driven articulation. The calculations of the instantaneous centre of rotation COR, as described in section 4.5.1, were carried out for an intact elbow joint during flexion movements and are shown in Figure 5.7.



(a)



(b)



(c)

Figure 5.7: Centre of rotation during flexion-extension of the elbow joint in x (a), y (b) and z (c)

It can be noted that the instant centre of rotation is moving in the 3D space during joint motion. This is interesting since the COR is naturally not fixed. The range of variations of COR is noted to be within 3.0 mm.

5.2.3 Joint laxity investigation

Performing elbow laxity investigation in the physically based model may serve as a tool for more advanced diagnosis in medical applications. As mentioned above, the musculoskeletal system has been built using anatomical data; this may indicate the possibility of further extension in this direction for joint laxity diagnosis. The joint laxity for musculoskeletal joint model has been investigated using the physics engine and gaming platform. The simulation has been performed with all the ligaments surrounding the joints attached to their insertion points. At first no force was applied except the muscle activation forces to drive joint movement. Then valgus-varus forces with values of 7 and 15 N were applied all the way during flexion movement, as shown in Figure 5.8.

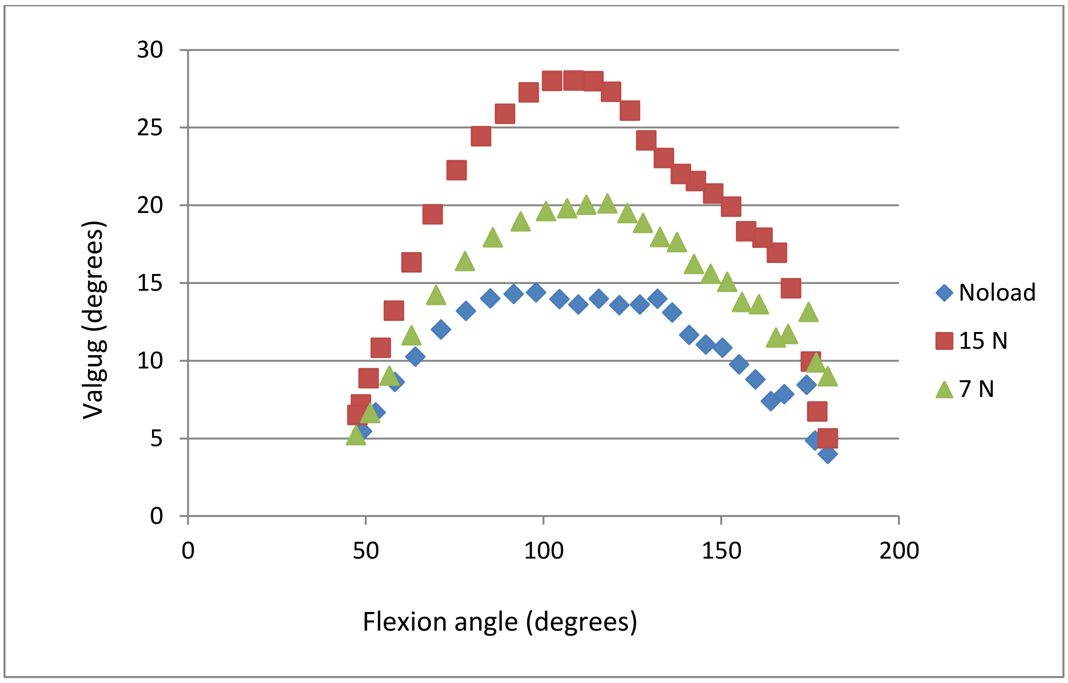


Figure 5.8: Elbow joint laxity for different varus forces

The curves in Figure 5.8, illustrates the valgus-varus angle for flexion angle range from 47^o to 180^o under different loading conditions. These data have been obtained by adding the laxity generated due to tissue looseness to the normal varus-valgus deformation during elbow flexion, without any load. Thus, the laxity may be indicated by the difference between the no load curve and the other loaded curves.

5.2.4 The moment arm

The moment arm calculation, as described in section 4.5.2, was performed during elbow flexion. The calculation is continuously updated with each time step to involve recent muscle force vectors and COR for moment arm calculations. Although the moment arm calculations can be used with any muscle, it has been done only with the biceps; the other moment arms can be calculated with the same procedure. The moment arm calculation file was saved and imported manually into Excel, showing the following graph in Figure 5.9.

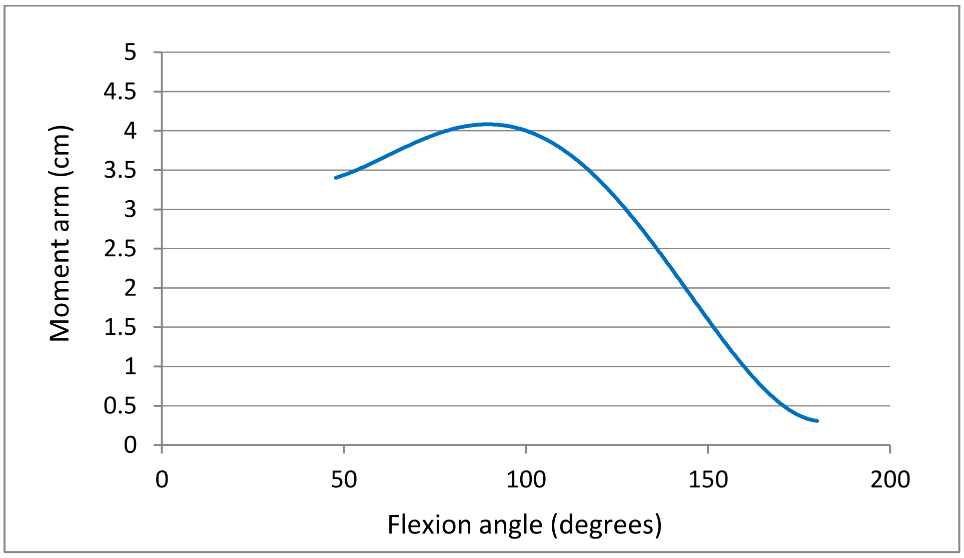


Figure 5.9: Moment arm during flexion for bicep muscle

It can be seen that the maximum moment arm value is found to be at 90^o; it then declines as the joint extends. This explains why the Biceps is most effective at 90^o, where the muscle has its maximum moment arm value.

CHAPTER 6

THE DEVELOPMENT OF A NOVEL IMAGE BASED ELBOW LAXITY MEASUREMENT SYSTEM

6.1 Introduction

The significance in the field of orthopaedic applications of understanding the kinematics of the diarthrodial joint has recently grown. Many researchers are investigating more effective solutions for human joint diagnosis. This is a well-known challenge in the assessment of articular joint kinematics. For example, human elbow joint is far more sophisticated than a mere basic hinge joint rotating around a fixed axis. Although a great many scientific studies have modelled the articulation of the diarthrodial joint as an idealized mechanical joint, it is simply not naturally correct to assume this. More accurate modelling should consider that the articular joint is driven by the contact surfaces. Although the primary findings in this thesis concern the evaluation of a physics engine in modelling and simulating a diarthrodial joint, it is clear that accurate kinematic measurement technique is required for the purpose of experimentally validating the result.

In this chapter, a new technique based on marker position analysis is proposed for the kinematic investigation of the elbow joint. A variety of instruments and techniques is readily available in the market to study joint kinematics, disorder and laxity diagnosis in in vivo conditions; at the same time, accuracy, practicability and cost remain the principal issues. Most of these issues may be dealt with but the developed image based system, requiring analysis of the kinematics of the elbow joint based on digital image analysis. The main contribution of the proposed diagnosis system is its non-invasive setup and process. In this chapter, an image-based analysis system is presented for the experimental study of joint kinematics and laxity. The main advantages of the proposed image based analysis system are its simplicity, practicability and accuracy.

6.2 Method

In this method three skin markers are used (see Figure 6.1): two markers are placed on epicondyles at the distal edge of the humerus close to the elbow and the third is positioned at the apex of the ulnar process. The positions of these markers entail the least relative motion between the skin and the bones. These marker positions were inspected for possible skin movement by placing a reference marker on the ulna and between the two markers on the epicondyles, as illustrated in Figure 6.1 (close to the centre of the elbow joint). Having these three markers (with the reference marker in the middle) on the same line during the application of lateral force can point to any considerable skin movement. At a flexion angle of 90°, a lateral force was applied to investigate the relative motion between the two markers on the epicondyles and the reference marker. The three markers appeared to be on the same line; this suggests that there was no significant skin movement at these marker positions.

A gradually increasing static force in the valgus-varus direction is applied to inspect the stiffness of the joint. For the analysis, two images are required at each flexion angle; the first image is to be captured before lateral force is applied and the second one with the application of this force. Then the positions of the markers are analysed by a camera-integrated VB.Net program. The results are obtained immediately after images are captured. The VB.Net program has been developed to work on a Windows PC and on Windows Mobile devices.

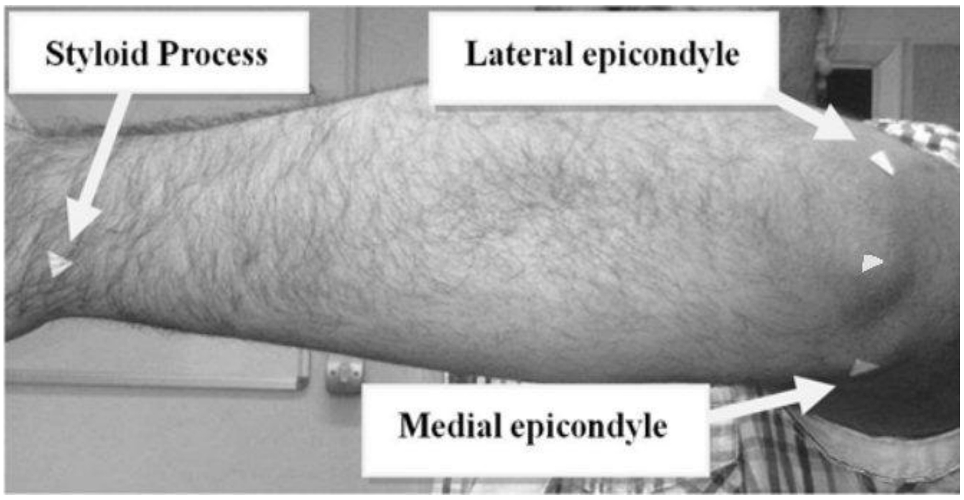


Figure 6.1: Positions of the markers

6.3 Software Development

A Visual Basic program has been developed to capture, manipulate, process and analyse the digital images. The layout of the developed Windows PC program is shown in Figure 6.2. Another version for Windows Mobile has also been developed. The developed software starts with images being uploaded to the picture box window directly from the integrated camera or from saved images. Each image is captured in the required frame. As the image is loaded, the marker position on the image may be clicked and then the coordinate of the points will appear automatically in the corresponding coordinate's boxes. As the marker positions shown in Figure 6.1 are clicked, the program detects the pixel coordinate in the *x* and *y* axes of the captured frame. For each picture, a triangle is drawn showing the selected marker positions, as in Figure 6.4.

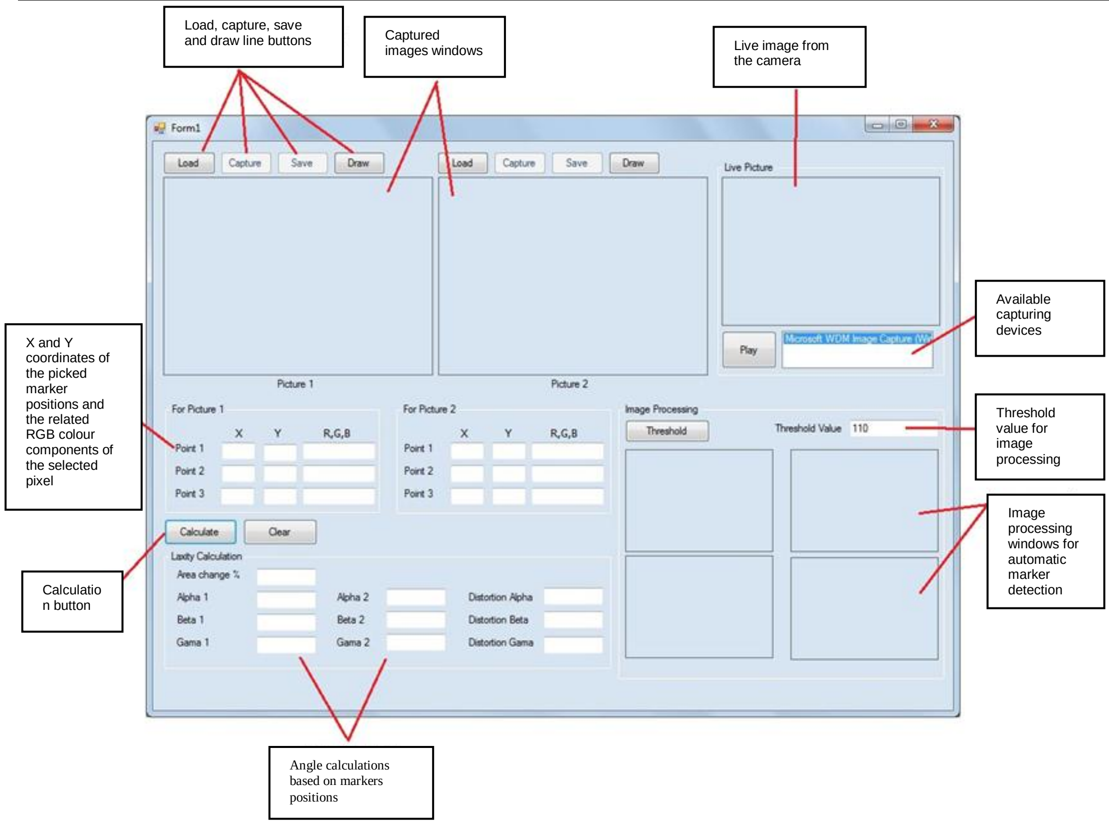

Load, capture, save and draw line buttons
Captured images windows
Live image from the camera
Form1
Load
Capture
Save
Draw
Live Picture
Picture 1
Picture 2
Play
Microsoft WDM Image Capture (Wi
Available capturing devices
X and Y coordinates of the picked marker positions and the related RGB colour components of the selected pixel
For Picture 1
For Picture 2
X
Y
R,G,B
Point 1
Point 2
Point 3
Image Processing
Threshold
Threshold Value
110
Threshold value for image processing
Image processing windows for automatic marker detection
Calculate
Clear
Calculation button
Laxity Calculation
Area change %
Alpha 1
Beta 1
Gama 1
Alpha 2
Beta 2
Gama 2
Distortion Alpha
Distortion Beta
Distortion Gama
Angle calculations based on markers positions

Figure 6.2: User interactive interface

The image samples which may be uploaded towards PictureBox 1 and PictureBox 2 are generally in one of several file formats (e.g. .bmp, .png, .jpg, .jpeg and .gif). The image will fit the picture window frame whatever its original size. The images which have been recorded while using the system camera are in ".bmp" file format, but the capture buttons are able to operate only in sequence, which means that the capture button for PictureBox 2 will not function unless the capture button for PictureBox 1 is pressed first. The captured images can be saved using the save buttons in a default folder directory in the 'My Documents/Laxity' folder.

As the images are loaded, marker points within the image can be selected by clicking and then the coordinates of the selected points will show up automatically within the coordinate show column. The clicking feature is based on the 'Mouseclick' in the command term. When the marker positions are clicked, this software will pick up on the coordinate through the ‘e.X.ToString’ and ‘e.Y.ToString’ command for coordinates in the X-axis and Y-axis for each selected point. In this software, the *x* and *y* coordinates in PictureBox 1 of the first selected position is termed $(X11, Y11)$ the second position to be clicked is $(X12, Y12)$ and the last is $(X13, Y13)$ whereas the coordinates in PictureBox 2 for the first selected marker is named $(X21, Y21)$ the second position is $(X22, Y22)$ and last position is $(X23, Y23)$. The VB code illustrated below shows the way in which the coordinates of the pixel were used to represent the marker positions of the first picture.

```
Private Sub picturebox1_Mouseclick(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseClick
Dim bmp As New Bitmap(PictureBox1.Image)

      If txtX11.Text = Nothing Then
            txtX11.Text = e.X.ToString
            txtY11.Text = e.Y.ToString
            txtRGB11.Text = bmp.GetPixel(e.X, e.Y).R.ToString & "," &
bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString

        ElseIf txtX12.Text = Nothing Then
            txtX12.Text = e.X.ToString
            txtY12.Text = e.Y.ToString
            txtRGB12.Text = bmp.GetPixel(e.X, e.Y).R.ToString & "," &
bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString

        Else
            txtX13.Text = e.X.ToString
            txtY13.Text = e.Y.ToString
            txtRGB13.Text = bmp.GetPixel(e.X, e.Y).R.ToString & "," &
bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString
       End If
       End Sub
```

After this, three lines are drawn to connect the selected marker positions so as to clarify the triangle which will later be subjected to further analysis. This is done using the ‘DrawLine’ command, accompanied by the label of the appropriate coordinate. For example, to connect the points in Picture 1 the ‘Drawline’ command is used, naming it PictureBox1.CreateGraphics.DrawLine $(f, X11, Y11, X12, Y12)$’

which allows a line to be drawn from the first coordinate point to the second coordinate point. Soon a red line appears because of the command for ‘f’ as ‘New Pen (Color.Red, 2)’. The VB code of the line to be drawn in PictureBox 1 is as follows:

```
Private Sub Draw1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Draw1.Click

        Dim f As Pen = New Pen(Color.Red, 2)

        Dim X11, X12, X13 As Integer
        Dim Y11, Y12, Y13 As Integer
        X11 = txtX11.Text
        Y11 = txtY11.Text
        X12 = txtX12.Text
        Y12 = txtY12.Text
        X13 = txtX13.Text
        Y13 = txtY13.Text
        PictureBox1.CreateGraphics.DrawLine(f, X11, Y11, X12, Y12)
        PictureBox1.CreateGraphics.DrawLine(f, X12, Y12, X13, Y13)
        PictureBox1.CreateGraphics.DrawLine(f, X13, Y13, X11, Y11)
    End Sub
```

When the calculation button is pressed, the software performs the angle calculations of the triangles in both pictures. This button functions according to the mathematical

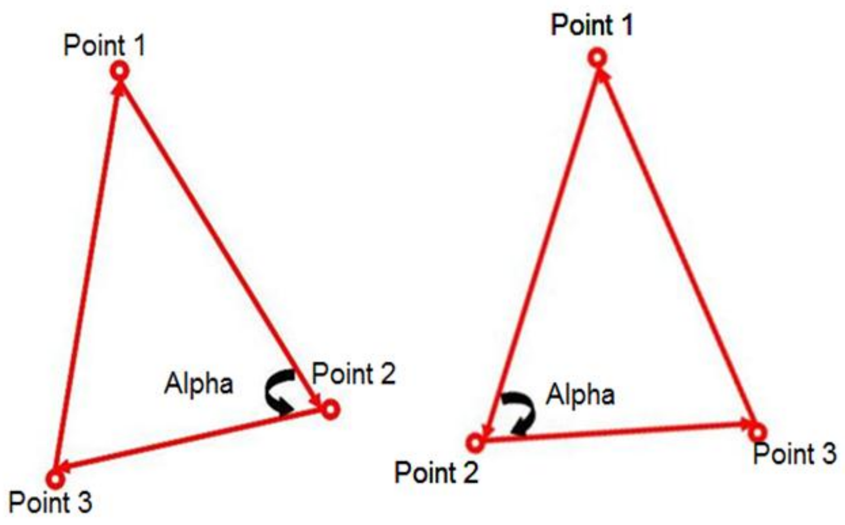


Figure 6.3: Location of Alpha in triangles

calculations expressed in this system code. The ‘Alpha 1’ and ‘Alpha 2’ values are the angle magnitude in degrees of alpha for the triangles in Pictures 1 and 2. Alpha is the angle at the second point clicked on, that is, between line 1 (from point 1 to point 2) and line 2 (from point 2 to point 3), as shown in Figure 6.3. The calculation is then performed for the angles of the triangles in both pictures and the corresponding distortion for each angle is displayed. In order to correctly compare the angles in the two pictures, each angle should be compared to its corresponding angle in the other picture. This is done by labelling the selected points according to the sequence in which they were picked. As a result, the selection procedure of the point should be the same in both images; if it is not, the obtained data is misleading. For example, Figure 6.3 shows the alpha angle as a corresponding angle to the second picked point, regardless of its position. The angles are calculated on the basis of the simple cosine law because only the coordinates of the points on the triangle are known.

$$a^2 = b^2 + c^2 - 2bc \cos\alpha \quad (6.1)$$

$$b^2 = a^2 + c^2 - 2bc \cos\beta \quad (6.2)$$

$$c^2 = a^2 + b^2 - 2bc \cos\gamma \quad (6.3)$$

Also,

$$\alpha = cos^{-1}\left(\frac{-a^2+b^2+c^2}{2bc}\right) \quad (6.4)$$

$$\beta = cos^{-1}\left(\frac{a^2-b^2+c^2}{2ac}\right) \quad (6.5)$$

$$\gamma = cos^{-1}\left(\frac{a^2+b^2-c^2}{2ab}\right) \quad (6.6)$$

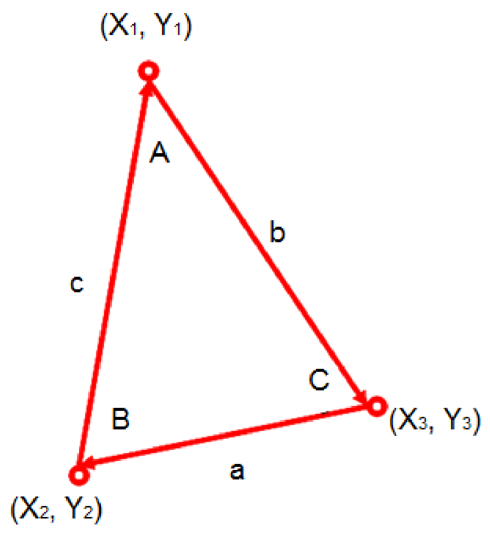


Figure 6.4: Drawing a triangle based on the marker positions

where a, b and c represent the triangle sides which can be directly calculated because their coordinates are known, using the Cartesian equations for coordinates:

$$a = \sqrt{(y_1 - y_3)^2 + (x_1 - x_3)^2} \tag{6.7}$$

$$b = \sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2} \tag{6.8}$$

$$c = \sqrt{(y_2 - y_3)^2 + (x_2 - x_3)^2} \tag{6.9}$$

The area for the triangle above can be found using the formula below:

$$Area = \frac{bc}{2}\sin\alpha \quad = \frac{ac}{2} sin\,\beta \quad = \frac{ab}{2}\sin\gamma \tag{6.10}$$

In the calculation parts, all the formulas above are used in both angle calculations for Pictures 1 and 2.

For the area change and distortion of alpha, the calculations are as follows:

$$Area\ Change\ \% = \frac{Area\ of\ triangle\ 1}{Area\ of\ triangle\ 2} \times 100 \qquad (6.11)$$

$$Angle\ Distortion\ = \alpha\ (in\ picture\ 1) - \alpha\ (in\ picture\ 2) \qquad (6.12)$$

The VB code to carry out these calculations is listed in Appendix A2 of this thesis; an example of this code is given below.

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        ' for the first picture

        Dim X11, X12, X13 As Integer
        Dim Y11, Y12, Y13 As Integer

        ' for the second picture
        Dim X21, X22, X23 As Integer
        Dim Y21, Y22, Y23 As Integer
        ' Sides and angles of triangle 1 and triangle 2
        Dim a1, b1, c1 As Double
        Dim a2, c2, b2 As Double
        Dim area1, area2 As Double
        Dim alpha1, beta1, gama1 As Double
        Dim alpha2, beta2, gama2 As Double
        X11 = txtX11.Text
        X12 = txtX12.Text
        X13 = txtX13.Text
        Y11 = txtY11.Text
        Y12 = txtY12.Text
        Y13 = txtY13.Text
        X21 = txtX21.Text
        X22 = txtX22.Text
        X23 = txtX23.Text
      Y21 = txtY21.Text
      Y22 = txtY22.Text
      Y23 = txtY23.Text

        ' for picture 1
        a1 = ((Y11 - Y13) ^ 2 + (X11 - X13) ^ 2) ^ 0.5
        b1 = ((Y11 - Y12) ^ 2 + (X11 - X12) ^ 2) ^ 0.5
        c1 = ((Y12 - Y13) ^ 2 + (X12 - X13) ^ 2) ^ 0.5
 alpha1 = Math.Acos((-a1 ^ 2 + b1 ^ 2 + c1 ^ 2)/(2 * b1 * c1))
 beta1 = Math.Acos((a1 ^ 2 - b1 ^ 2 + c1 ^ 2)/(2 * a1 * c1))
 gama1 = Math.Acos((a1 ^ 2 + b1 ^ 2 - c1 ^ 2)/(2 * a1 * b1))

        ' for picture 2
        a2 = ((Y21 - Y23) ^ 2 + (X21 - X23) ^ 2) ^ 0.5
        b2 = ((Y21 - Y22) ^ 2 + (X21 - X22) ^ 2) ^ 0.5
        c2 = ((Y22 - Y23) ^ 2 + (X22 - X23) ^ 2) ^ 0.5
```

```
alpha2 = Math.Acos((-a2 ^ 2 + b2 ^ 2 + c2 ^ 2)/(2 * b2 * c2))
beta2 = Math.Acos((a2 ^ 2 - b2 ^ 2 + c2 ^ 2)/(2 * a2 * c2))
gama2 = Math.Acos((a2 ^ 2 + b2 ^ 2 - c2 ^ 2)/(2 * a2 * b2))

        area1 = a1 * b1 * Math.Sin(gama1)/2
        area2 = a2 * b2 * Math.Sin(gama2)/2
        AreaChange.Text = (area1/area2) * 100 - 100

        txtalpha1.Text = alpha1 * 180/3.1416
        txtalpha2.Text = alpha2 * 180/3.1416
distorsionAlpha.Text = alpha2 * 180/3.1416 - alpha1 *180/3.1416

        txtBeta1.Text = beta1 * 180/3.1416
        txtBeta2.Text = beta2 * 180/3.1416
DistorsionBeta.Text = beta2 * 180/3.1416 - beta1 * 180/3.1416

        txtGama1.Text = gama1 * 180/3.1416
        txtGama2.Text = gama2 * 180/3.1416
DistorsionGama.Text = gama2 * 180/3.1416 - gama1 * 180/3.1416

    End Sub
```

6.4 Hardware Development and the Design of an Arm and Forearm Device (Braces)

The image-based system proposed here consists of a basic Windows compatible webcam; Windows operated PC, camera holding braces and the developed Visual Basic software. The design of the braces uses the idea of an electro goniometer with a potentiometer in place at the connector hinge. However, the whole experiment uses the concept of a motion camera analysis, using the device with the camera imaging software which has been developed. This newly designed system features a mini camera attached, playing the main role in this system.

6.4.1 Mechanical Part

The design of the device is presented in Figure 6.5, below. It comprises of three connected parts, namely, the arm brace, forearm brace and the camera stand. The arm and forearm brace are worn on the arm and forearm of a patient. The camera stand is where the camera should function once the camera imaging software has been connected. The material used to manufacture and fabricate this device is aluminium.

The inner parts of the braces are made of plastic used as a grip on the arm and forearm surfaces.

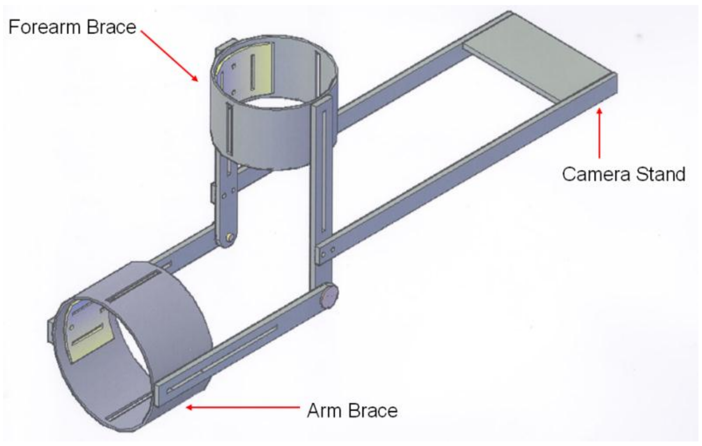


Figure 6.5: 3D image of the arm and forearm device design

As shown in Figure 6.6, the camera stand and the forearm brace should be perpendicular to each other. This is because the camera should be placed parallel to the forearm to view the points on the patient’s forearm.

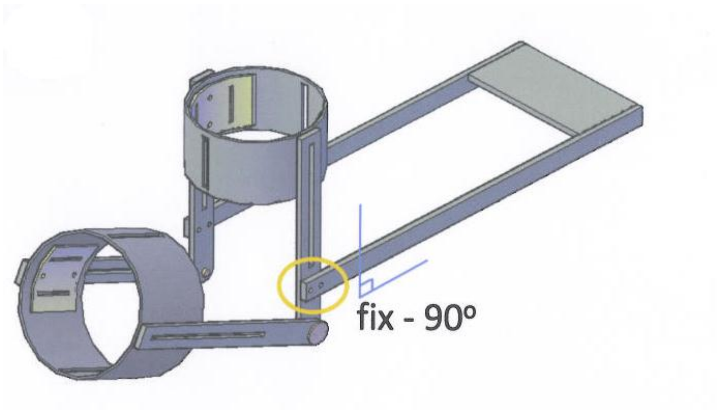


Figure 6.6: Forearm brace and perpendicular camera stand

Only the parts between the arm brace and the forearm brace will move, as shown in Figure 6.8. The motion is rotational, to represent the movement of the elbow in flexion and extension motion. Figure 6.8 describes the motion between these two braces.

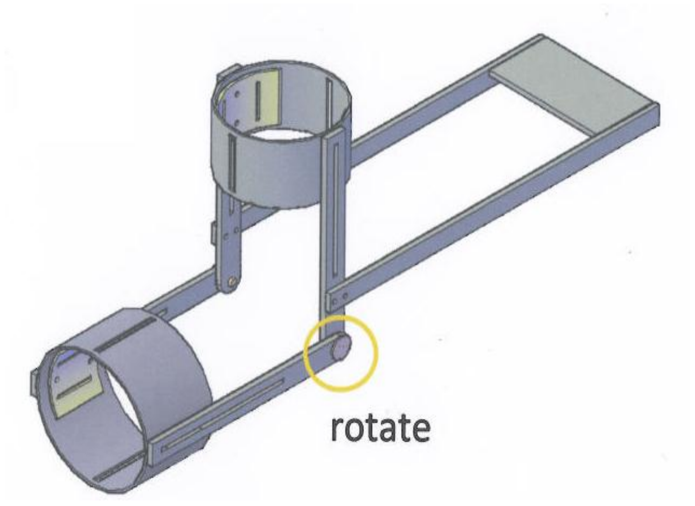


Figure 6.7: The connection between the arm brace and forearm brace – the only moving part

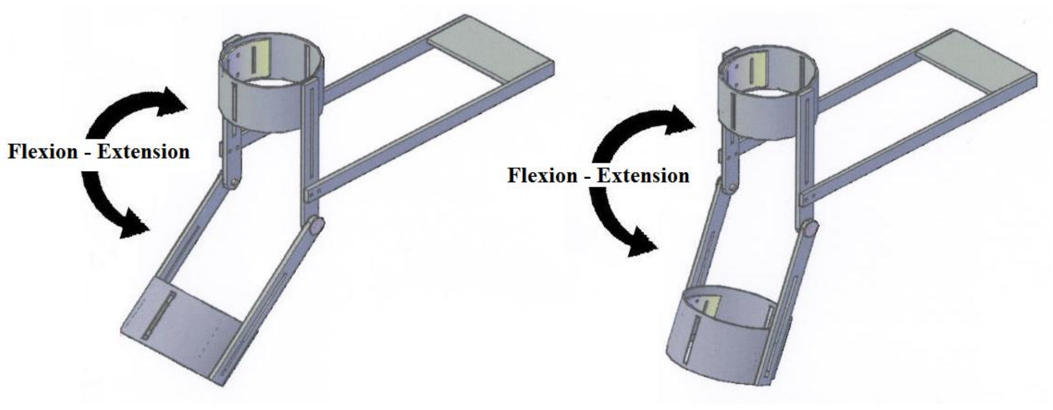


Figure 6.8: Representing the flexion-extension motion of the elbow joint

The arm and forearm go into the metal cylinders. The axis of rotation of the elbow joint has to be close to the axis of the rotation of the braces. The braces therefore are attached to the arm and forearm using straps. The flexion angle is measured using a potentiometer which is attached to a point between the arm brace plate and the forearm brace plate.

6.4.2 Electronic Part

The electronic part of this device is the potentiometer connection. The connection consists of a potentiometer and a terminal board.

6.4.2.1 Potentiometer

The potentiometer is a device which changes the resistance to a flow of electric current in a circuit. In the project, the potentiometer is used to measure the angle between the arm brace and the forearm brace. It is connected to a terminal board so that the output produced can be read digitally, using Pico Technology software. However, the output signal from the potentiometer measures the voltage. Hence, to get the angles when the two braces are moving, the voltage outputs need to be inserted in a conversion equation. This equation can be written by calibrating the device with the potentiometer. Figure 6.9 shows how the potentiometer is connected to the device and the terminal board.

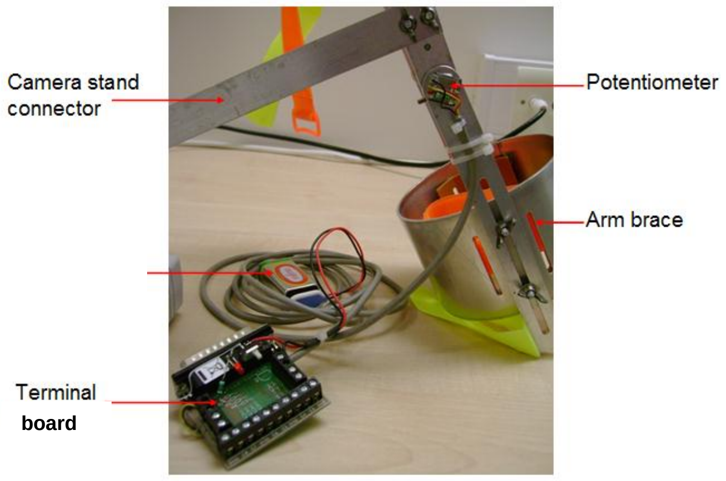


Figure 6.9: Potentiometer installation

6.4.2.2 Terminal Board

The terminal board used is from Pico Technology ADC-11, part of the ADC-11 Data Logger. It enables sensor circuits to be built which take measurements for the data logger to process. The screw terminal allows sensor wires to be attached directly to the board without the need for soldering.

Table 6.1 shows the purpose of each of the terminals and empty component sites relative to ADC-11 terminal board.

Table 6.1: Terminal board connections

Terminal or site	Description
C1 to C11	Connections to ADC channels 1 to 11
D0	Digital output. It can be used as a low-current supply to power sensors
GND	Connection to ground.
Q1	Site for LM35 temperature sensor. Will not be used in this project.
R1 to R3	Sites for resistors between D0 and Channels 1 to 3.
R4 to R7	Sites for series resistors in inputs to channels 5 to 8.
R8 to R11	Sites for stunt resistors between channels 5 to 8 and GND.

The terminal board is connected directly to the analogue connector on the ADC-11 Data Logger.

6.4.2.3 PicoData Logger

The terminal board used to connect the potentiometer has to be read using the PicoData Logger. The ADC-11 PicoData Logger is a medium speed ADC which comes in many versions. For this project, the version used is as follows:

Product:ADC-11/10 USB

Resolution: 10 bits

Channel: 11

In connection: D25

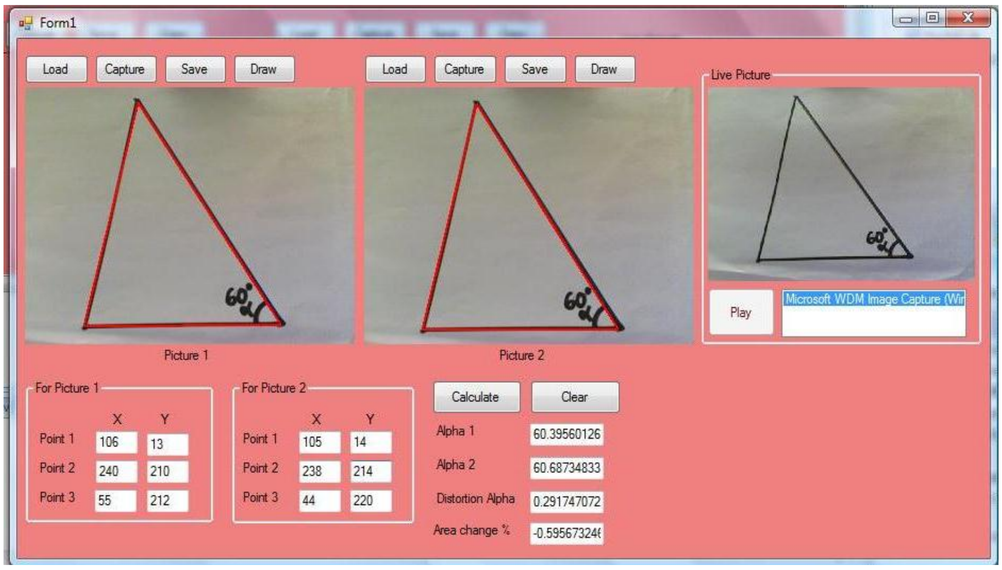
Out connection: USB

The terminal board is connected to the 25-way D-type connections. This is mainly to offer port input connectors for the 11-input channels. This ADC can be function as an oscilloscope with the PicoScope software as well as PicoLog. The developed program can be used to acquire and process the ADC data. The drivers for the logger are installed during the installation of PicoLog software from the CD driver provided.

This ADC-11 is built for a low voltage range (0-2.5 volts). For the case when the voltage arises out of the specified range for any reason, an overvoltage warning message is displayed on the PicoLog active window. Any voltage changes greater than $\underline{+}$30V may cause permanent damage to the unit.

6.4.3 System Calibration

The developed image based program is calibrated to verify the accuracy of the measuring technique of angle calculations. The system calibration is made by drawing a triangle with known angles values on a piece of paper. Then the paper is placed parallel to the camera so that the image can be taken using the software produced. The angles values on the picture are calculated by the system and then compared with the actual measured angles. Figure 6.10 shows that the measurements of the angles corresponds with the actual angle drawn manually, which is 60º.



For Picture 1

	X	Y
Point 1	106	13
Point 2	240	210
Point 3	55	212

For Picture 2

	X	Y
Point 1	105	14
Point 2	238	214
Point 3	44	220

Alpha 1	60.39560126
Alpha 2	60.68734833
Distortion Alpha	0.291747072
Area change %	-0.59567324

Figure 6.10: Calibration for the developed imaging software

6.4.3.1 Potentiometer Calibration

The potentiometer should be calibrated to find the relation than converts the voltage produced to the flexion angle. To do this, the device is placed in a known angular position, as shown in Figure 6.11. The voltage output for this specific position is noted.

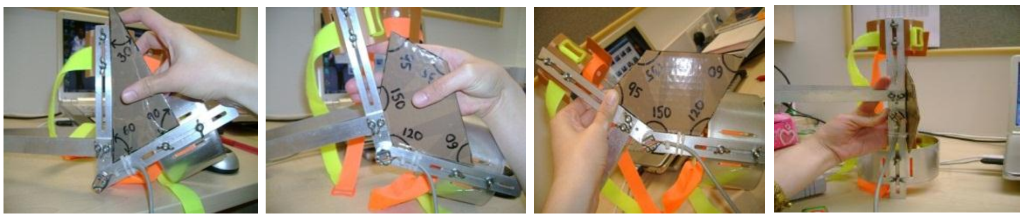


Figure 6.11: Different angle positions for calibrating the potentiometer

Table 6.2: Calibration results

Position	Position 1	Position 2	Position 3	Position 4	Position 5
Angle (degrees)	60	90	120	150	180
Voltage (V)	0.226	0.466	0.776	1.043	1.324

The calibration results from Table 6.2 are plotted in Excel. The linear graph is shown in Table 6.2. A linear equation may be used to find this relation that links the voltage and the angle.

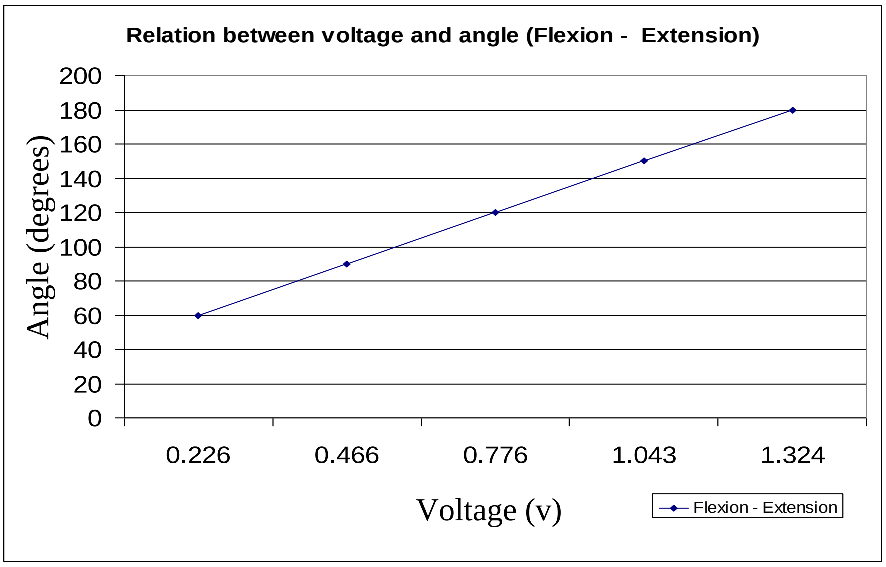


Figure 6.12: Graph of the relationship between the voltage and the angle

The linear equation is:

$$A = sV + c \tag{6.13}$$

Where *A* is the flexion angle, *V* is the voltage, *s* is the slope and *c* is a constant.

The slope, *s*, is found a follows:

$$s = \frac{A_2 - A_1}{V_2 - V_1} \tag{6.14}$$

$$s = \frac{150-90}{1.043-0.466} = 103.99 \qquad \left(\frac{degrees}{V}\right)$$

and the constant *c* can be calculated as follows:

$$c = A - sV \tag{6.15}$$

$$c = 120 - 103.99 * 0.776 = 39.31 \quad \textit{(degrees)}$$

Therefore, Equation (6.13) can be rewritten as:

$$A = (103.99)V + 39.31 \quad \textit{(degrees)} \tag{6.16}$$

This equation (6.16) which relates the flexion angle and the voltage is used to record the flexion angle, *A* during the experiment.

6.5 Experimental Procedure

The test procedure starts by attaching three skin markers: two are stuck on the lateral and medial epicondyles at the distal side of the humerus near the elbow and the third is placed at the styloid process, which is a bony trajectory at the end of the ulna near the wrist, as shown in Figure 6.13. These markers, as noted above, are positioned where there is least relative motion between the skin and the bones. For the analysis, two frames are required; the first frame being recorded before any force is applied on the forearm and the second frame after a static force is applied on the forearm. The applied force has a magnitude of 7N in the valgus-varus direction.

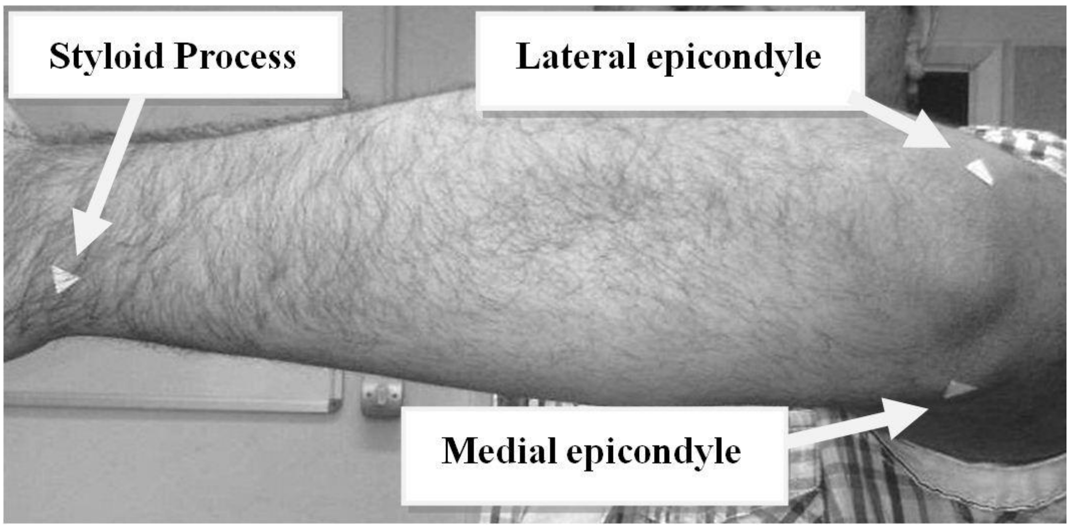


Figure 6.13: Positions of the skin markers

As the humerus is not fixed, the biggest part of the elbow joint movement in varus-valgus plane originates from the glenohumeral joint. However, this should not affect the analysis of the actual motion of the elbow valgus-varus, because the analysis is performed based on the relative positions of the skin markers. For each frame, the three skin markers form a triangle and this enables the angles of the two triangles to be compared to one another to check for distortion. The changes in the angles should indicate the valgus-varus deformation of the joint under scrutiny.

CHAPTER 7

COMPARATIVE EVALUATION OF THE SIMULATION RESULTS

7.1 Introduction

In Chapter 4 and Chapter 6 of the thesis, which use modelling based on game physics and marker position analysis respectively, novel methods have been presented for investigating articular joints. The current chapter studies the reliability, practicability, performance and consistency of the proposed modelling technique. The validation of such simulation is not a simple task, because no comparable joint treatment is available, in a situation where the modelling approach is based on contact driven articulation and the joint flexibility is determined by the surrounding tissues. Thus, the validation has to evaluate the proposed method by another existing method which has exactly the same joint geometry and parameters, not an easy condition to fulfil. However, the simulation result of joint modelling based on game physics and the image based analysis were compared with one another, in addition to some of the related studies and experiments at the Brunel Orthopaedic Research and Learning Centre. Moreover, it was found possible to examine the consistency of the developed system to at least an acceptable level when the results were compared using Musculoskeletal Joint Modeller software (MJM), which provides contact driven articulation with 6 DOF.

From the point of view of software development, while the simulation of the current software remains within the estimated range, it may be capable in the future of further improvement and updating, since the implementing of such technology in biomechanical investigations is still at an early stage. Consequently the framework for evaluative analyses of the current modelling approach may consider the expected range as a base line. Thus, some specific inquiries should be looked at carefully for further analysis, such as:

- The stability and reliability of the simulation under different conditions and parameters
- The realistic behaviour of the simulation so that it stays within acceptable limits for its use in medical applications
- The possibility of assessing the obtained results after comparison with those of the image based system, MJM, published materials and the Orthopaedic Research and Learning Centre at Brunel University.

7.2 Comparative Analysis of the Simulation Results

Due to the unique nature of the joint kinematics to be analysed and the results obtained in Chapter 5, the process of validating the results requires different validation platforms. In this section, the modelling software program Musculoskeletal Joint Modeller (MJM) will be used, since it the only modeller with 6 DOF. Another comparison study is performed to compare the obtained valgus-varus movements and laxity of the elbow joint with the experimental results. The simulation results is compared with experimental results obtained from the minimally invasive method developed in this research and presented in Chapter 6, along with the Stewart platform, which was developed by our research group in the Brunel Orthopaedic Research and Learning Centre. Published materials have also been considered in this comparative study.

7.2.1 Image based analysis system

The experimental setup and procedure for the image based system is explained in detail in Chapter 6. However, this system was developed mainly to examine the valgus-varus deformation of the elbow joint during flexion extension movement in addition to joint laxity. Before the system was used in the experiment, it had been calibrated as illustrated in section 6.4.3. Its consistency is due to the simplicity of the design and the minimum noise measurement, based on marker coordinates.

7.2.1.1 Valgus-varus deformation

Flexion extension joint movement was carried out experimentally to provide some grounds for confidence for the physics based approach, and the image based system. As mentioned above, in validating the simulation results, perfect match was not expected. However, validating a similar pattern of joint behaviour may be sufficient to prove the reliability of the physics based model used in the joint modelling. In analysing joint behaviour, it may not be sufficiently accurate to consider only one individual elbow joint in vivo, since experiments offer multiple sources of error, such as the accurate location of the marker centre and the possible relative marker-skin movements. In addition, everyone has unique joint behaviour. Hence, the experiment was performed on 7 healthy volunteers, none of them suffering from any elbow joint problems. Figure 7.1 shows the results obtained experimentally for all 7 volunteers.

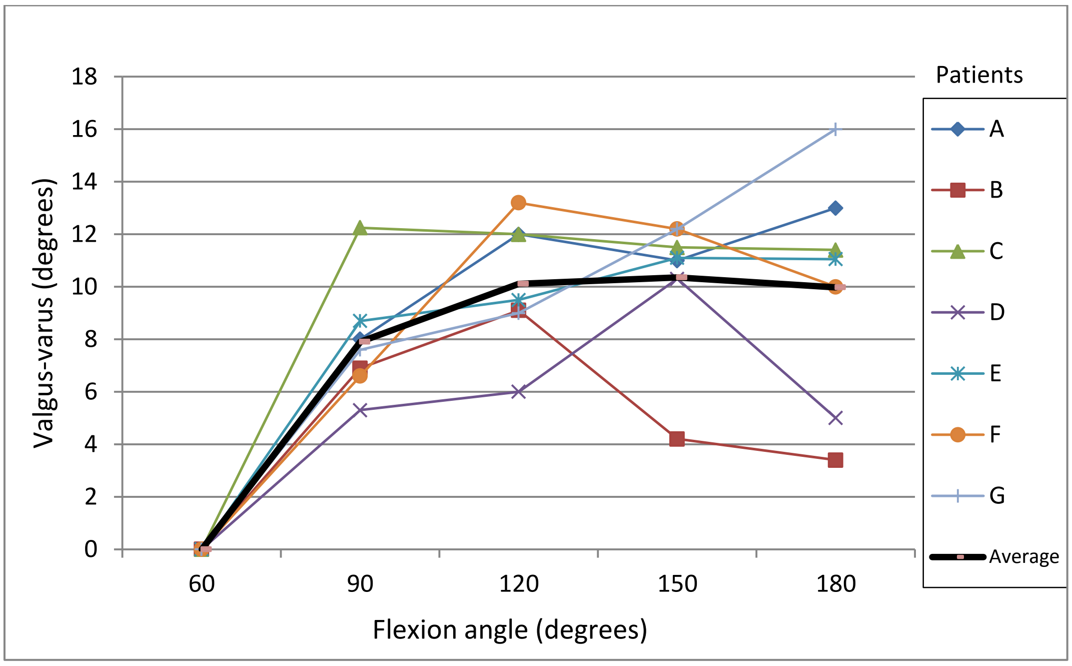


Figure 7.1: Varus-valgus deformation angle during flexion movement (image based analysis)

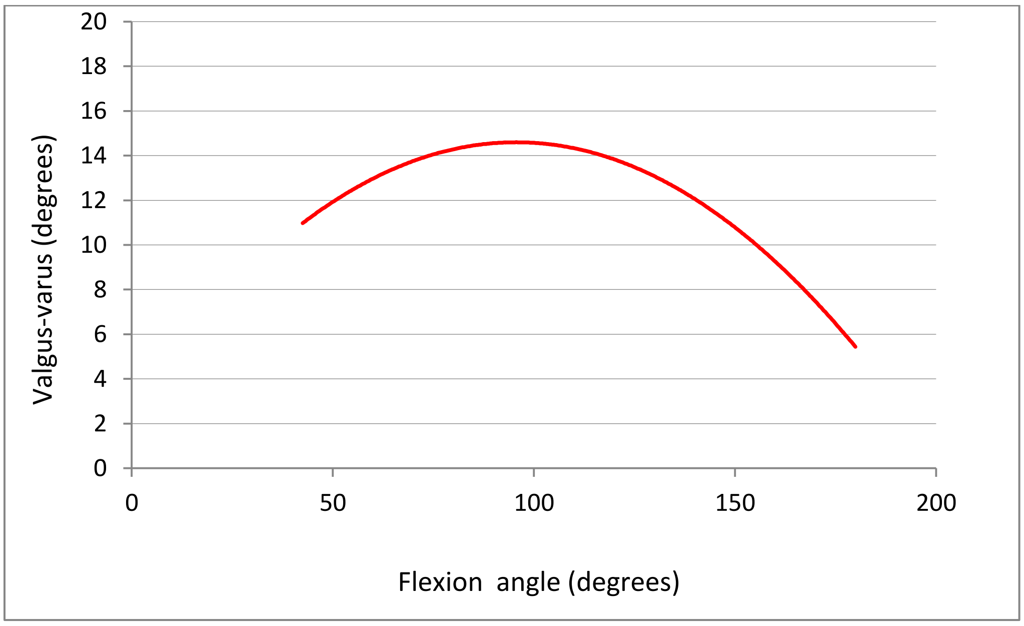
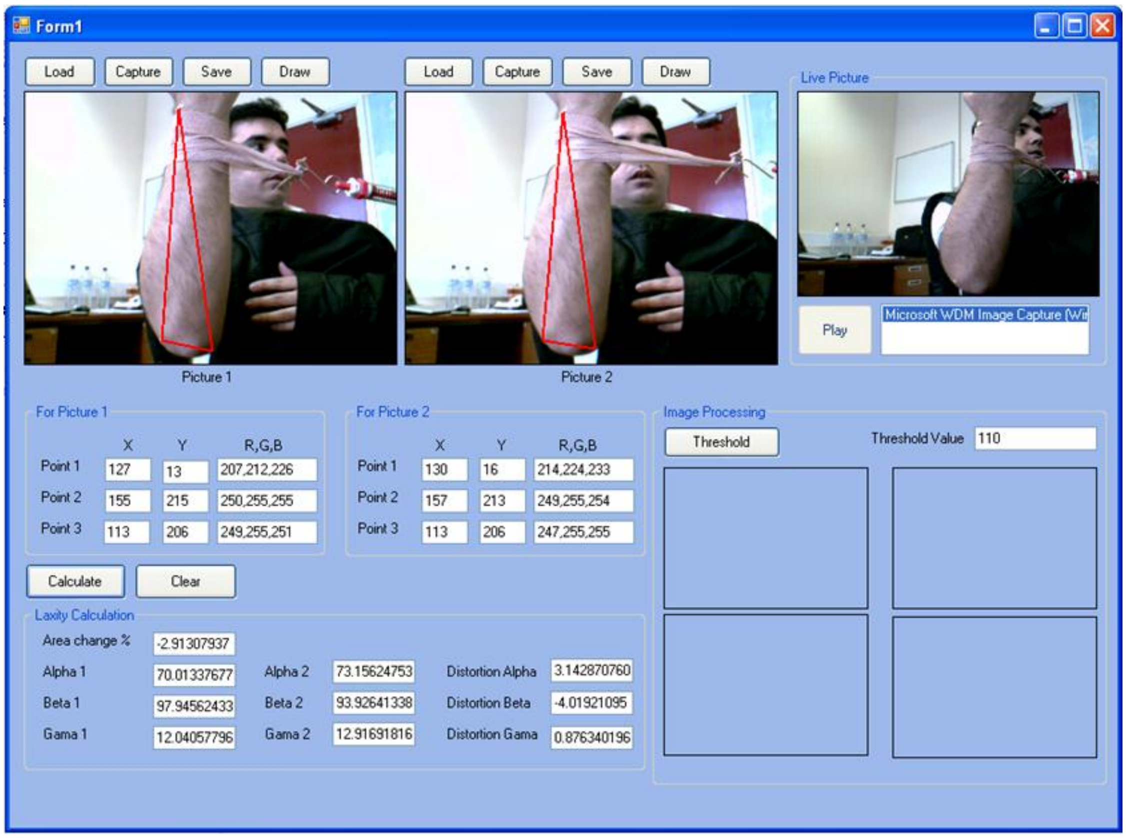


Figure 7.2: Varus-valgus angle during flexion-extension from a physics based simulation

In Figure 7.1 we noted that the maximum valgus-varus angle varies from one person to another. For most volunteers, a value of almost 10^o is reached between the minimum and maximum valgus-varus angles. Note that the range of motion for the experimental investigation was from 60^o to 180^o (120^o), while in the modelling software the range was from 42^o to 180^o (138^o) as shown in Figure 7.2; such variation may also introduce some difference in the expected results, where a larger range of flexion-extension motion may produce valgus-varus angles higher in value. However, the results obtained, as assessed between the values from the physics based simulation and those from the image based systems, are comparable, as seems appropriate.

7.2.1.2 Joint laxity

Soon after the preliminary comparison between the proposed physics based model and the image based systems for varus-valgus deformation during joint flexion, it's been decided to extend the research to investigating joint laxity, because this joint disorder is very difficult to examine in in vivo conditions. The image based system proposed in this thesis was also used experimentally to investigate the laxity of the elbow joint. To begin with, a normal range of elbow joint laxity is required to establish a basis for analysis. The data regarding this are obtained from published materials and therefore they were used for comparing the experimental results using the image analysis to the simulation results with a view to verification. The experiment was performed in two stages; in the first, the range of valgus-varus motion over elbow flexion and extension is established, as shown in Figure 7.1. In the second, the normal range of laxity was studied for the participating volunteers by applying 7N static force in the valgus-varus direction. As all volunteers had a healthy elbow joint, no abnormal laxity was expected. The experimental test was performed on 7 volunteers in the first stage and 9 in the second. A snapshot of the PC Windows version of the software is shown in Figure 7.3.



Threshold Value 110

For Picture 1

	X	Y	R,G,B
Point 1	127	13	207,212,226
Point 2	155	215	250,255,255
Point 3	113	206	249,255,251

For Picture 2

	X	Y	R,G,B
Point 1	130	16	214,224,233
Point 2	157	213	249,255,254
Point 3	113	206	247,255,255

Laxity Calculation

Area change %	-2.91307937				
Alpha 1	70.01337677	Alpha 2	73.15624753	Distortion Alpha	3.142870760
Beta 1	97.94562433	Beta 2	93.92641338	Distortion Beta	-4.01921095
Gama 1	12.04057796	Gama 2	12.91691816	Distortion Gama	0.876340196

Figure 7.3: User graphical interface

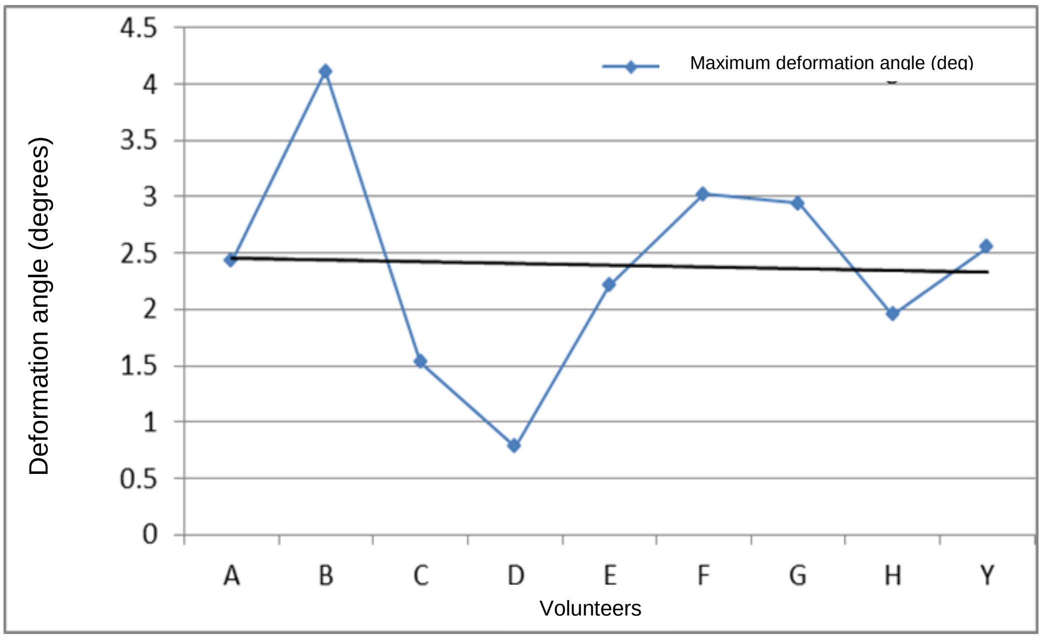


Figure 7.4: Experimental result for deformation angle using image based system

The above results in Figure 7.4 show the experimental valgus-varus deformations for the 9 volunteers with their elbow joints loaded by 7N valgus force at a flexion angle of 90^o. The degree of permitted looseness of the elbow joint may vary from one person to another, since it is associated with many other factors, such as age, gender, daily life activities and many others. Not all volunteers are expected to have similar joint stiffness, apart from other factors. Nevertheless, to compare the experimental result, the effects of the other factors can be reduced by averaging all the experimental results of joint looseness in the volunteers and then comparing the average with that in the published materials, for purposes of verification. As shown in Figure 7.3, this average is found to be slightly below 2.5^o. As none on the participants suffers from any elbow problems, the above chart shows no abnormal laxity.

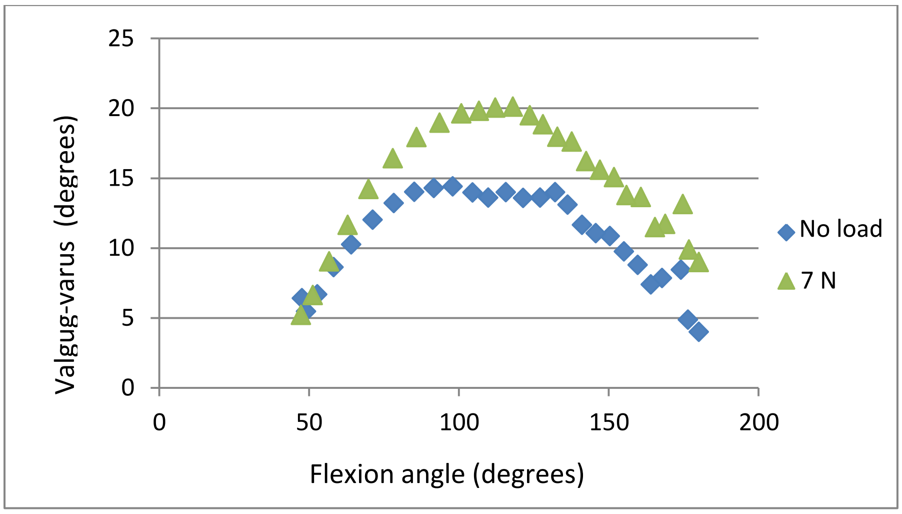


Figure 7.5: Physics based valgus-varus deformation for no load and a 7N loaded joint

The physics based simulation result for valgus-varus deformation is shown in Figure 7.5. It can be seen that, at a flexion angle of 90^o, the difference between the unloaded and 7 N loaded deformation curves is about 4.5^o. Such a difference may relate to different assumptions (such as insertion points and the stiffness of the tissues) in the physics based simulation, or to other sources of error in the experimental results. The uncertainty of the current results obtained experimentally may be due to the manual locating of the marker position and/or to skin movement. The automatic locating of picking of marker positions by pattern recognition technique is recommended and is being currently developed to eliminate human error. Skin movement, in contrast, is difficult to handle and this experiment is more suitable for patients with little body fat than for others, because subcutaneous fat increases skin movement. However, an extended discussion of the uncertainty in image based analysis is provided in Chapter 9.

7.2.2 Musculoskeletal Joint Modeller Software (MJM)

7.2.2.1 Centre of rotation

The simulation results show interesting behaviour for the centre of rotation COR, where a continuously moving centre of rotation in 3D space appears. This behaviour was brought into the comparison by a specialised software package known as Musculoskeletal Joint Modeller or MJM (Esat and Ozada, 2010). MJM is almost the only joint modeller that permits 6 DOF for the articular joint model. The 3D translation of COR obtained from MJM is shown in Figure 7.6. However, the differences between our results and the MJM may be due mainly to tissue arrangements and stiffness, as well as surface mesh density, for higher mesh density gives more accurate simulation results. Moreover, different coordinate systems and also system calibration, ligaments and tendon insertion points may all influence the centre of rotation. Overall, the range of variations in the values of COR is acceptable.

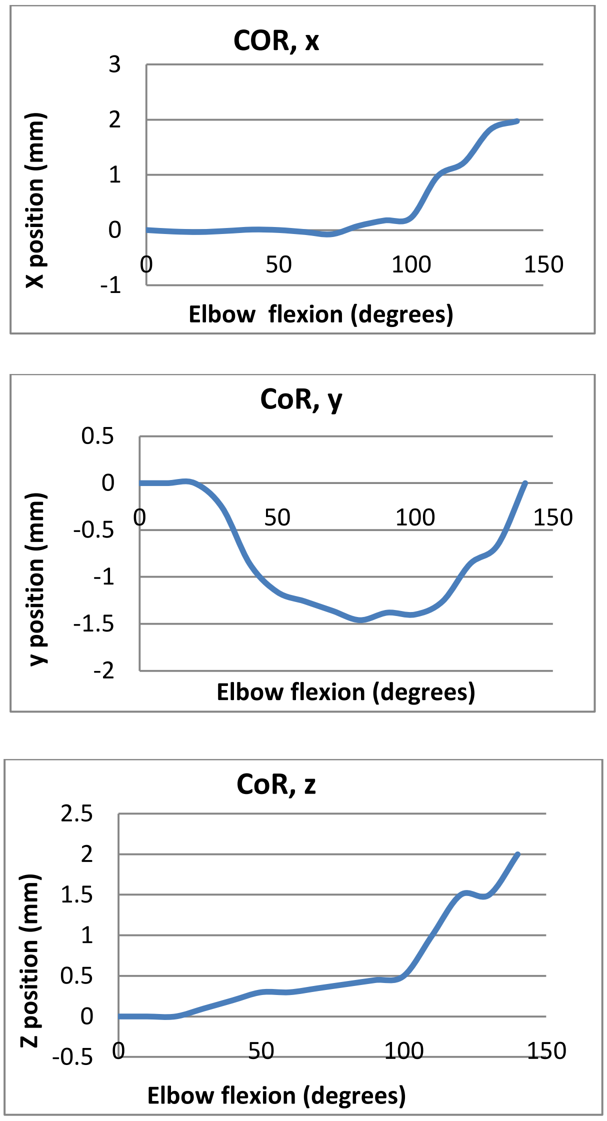


Figure 7.6: 3D change in the centre of rotation (COR) during elbow joint extension, obtained from MJM software

7.2.2.2 Moment arm

Furthermore, the moment arm was compared with the results obtained from the Musculoskeletal Joint Modeller (MJM); it was compared with MJM because MJM allows 6 DOF for the articular joint, as the current study also does. The results from MJM of the moment arm for the biceps are shown in Figure 7.7.

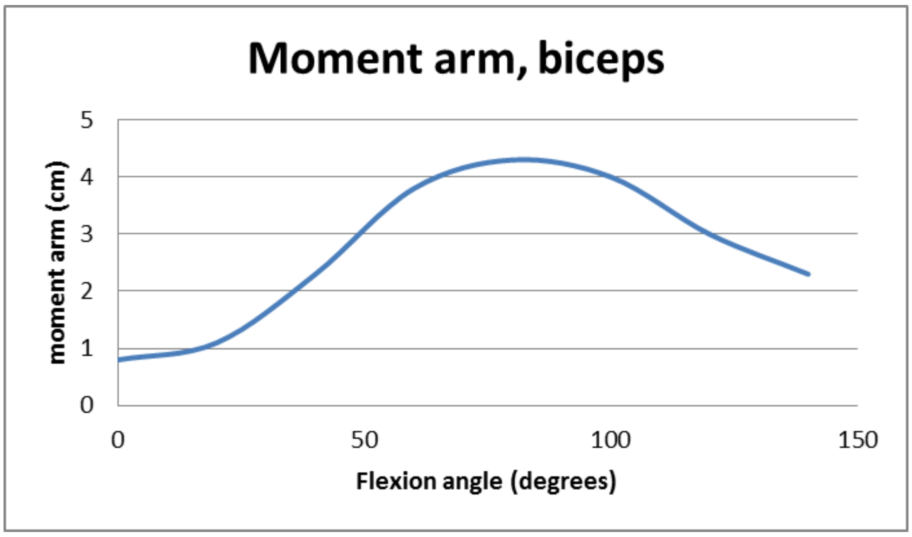


Figure 7.7: Moment of arm for biceps from MJM (Esat and Ozada, 2010)

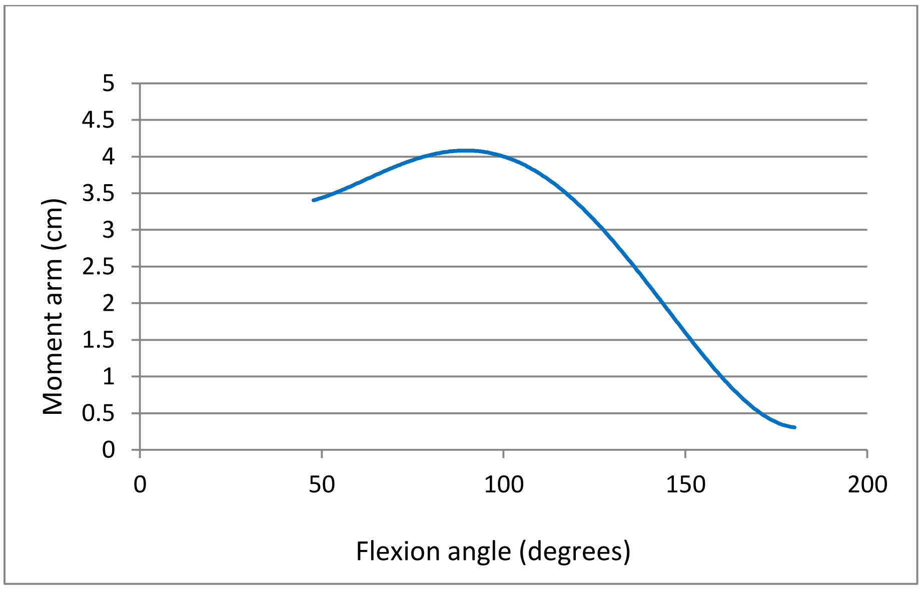


Figure 7.8: Moment of arm for biceps from the physics based simulation

Note that the range of the flexion angle in the developed model is from 50^o to $180^{o,}$ as in Figure 7.8, while with MJM it ranges from 0^o to about $140^{o,}$ as in Figure 7.7 ; this may explain the variation. Thus, the comparison is valid from 50^o to 140^o, which obviously shows similar behaviour of the moment arm between the results of MJM and those of the proposed physics based simulation. As no exact match is expected, the behaviour and range of results obtained from the physics based simulation is comparable with the MJM results.

7.2.3 Experimental results performed at Brunel Orthopaedic Research and Learning Centre (Stewart Platform)

A variety of devices and systems has been developed in the Brunel Orthopaedic Research and Learning Centre. However, a modified Stewart platform designed mainly for investigating elbow joint kinematics is illustrated in Figure 7.9. This device originated as a universal instrument which could evaluate the 6 DOF of joint movements based on a parallel Stewart platform. The Stewart parallel mechanism was implemented by our research group at Brunel Orthopaedic Research and Learning Centre as a kinematic based device for measuring elbow joint kinematics (Figure 7.9). This device comprises 6 linear potentiometers in which the alternation in displacement for each potentiometer is acquired and used for further calculations, by means of the inverse kinematics analysis accessible in MATLAB.

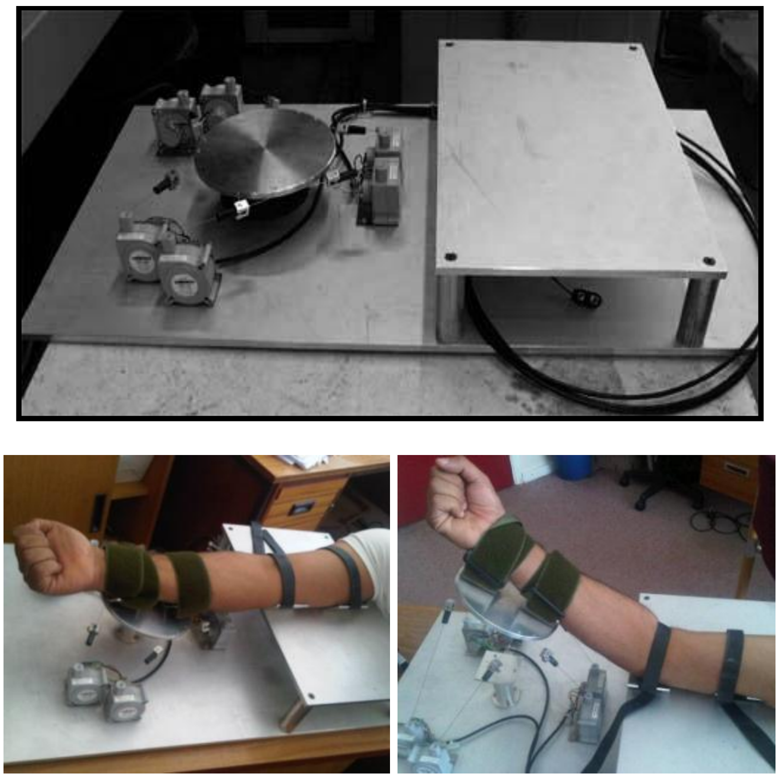


Figure 7.9: Stewart platform developed for measuring joint kinematics (Alrashidi et al., 2009)

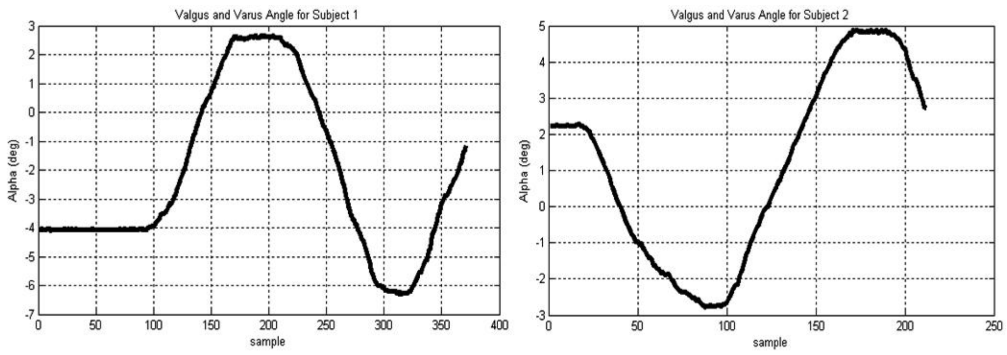


Figure 7.10: Valgus and varus deformation of elbow joint during flexion

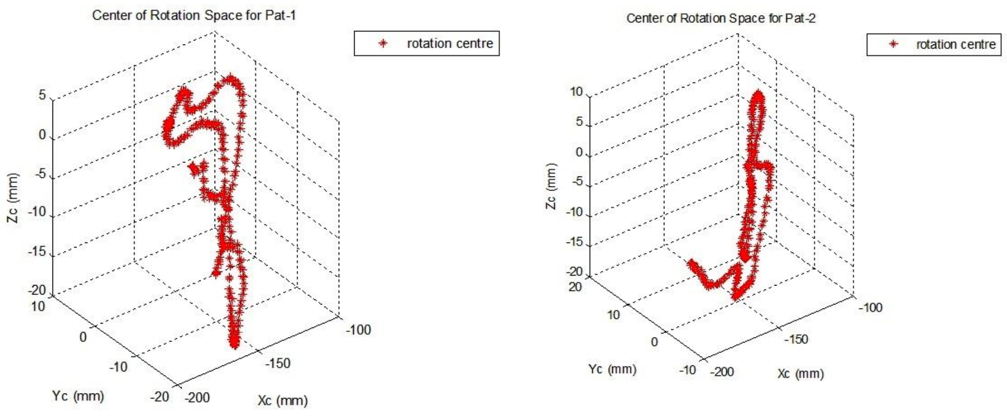


Figure 7.11: Centre of rotation of elbow joint during arm flexion

Figure 7.9 shows the Stewart platform device used for the valgus-valgus deformation analysis of the elbow joint during flexion extension, as shown in Figure 7.10. It can also be used for calculating the centre of rotation by inverse kinematics, as in Figure 7.11. In this experiment, the bones have to be safely and securely fastened in resting positions in order to get accurate results, otherwise, skin movement and looseness may introduce some undesirable uncertainty into the output. The outcomes of the Stewart platform device show clearly that the centre of rotation is moving in the 3D space during joint articulation. In cases of valgus-varus deformation, it shows some considerable variation of the valgus-varus angle (y-axis) during joint flexion as signal samples (x-axis). Similarly, the centre of rotation was moving within a bigger range than expected for a healthy joint. More precise results could perhaps have been reached through the use of a cadaveric specimen, in which skin movement can be avoided by firmly fixing the bones into the platform, and thus allowing the actual laxity of the elbow joint to be evaluated. Indeed, it is worth considering a better fixation device than straps to minimize skin movements to an acceptable limit. However, these results still indicate that the valgus-varus deformation and COR values are just within the range of the data provided.

7.2.4 Published materials

7.2.4.1 Varus-valgus and joint laxity

Comparing the laxity curves obtained from the simulation results with the published results (Floris et al., 1998, Jensen et al., 2005, Stavlas et al., 2007) for validation purposes shows similar elbow joint laxity under loaded conditions. Although this does not exactly match the published material in Figure 7.12, it does indicate to an acceptable level the laxity behaviour of an intact joint.

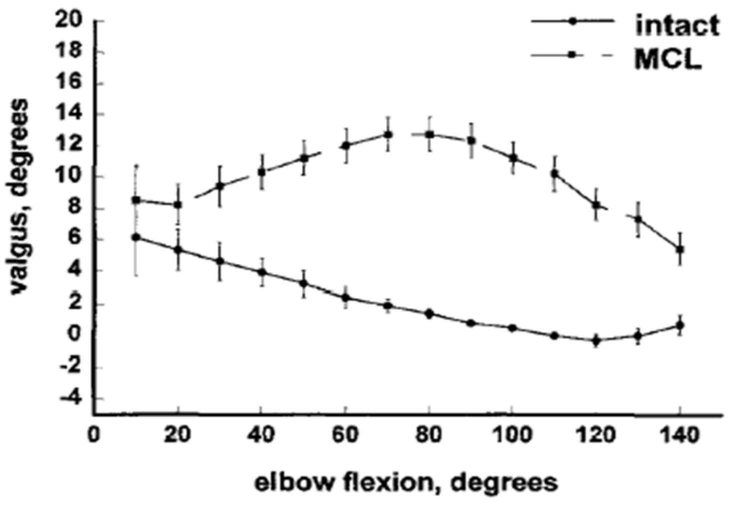


Figure 7.12: Valgus angle for different flexion angles (Floris et al., 1998)

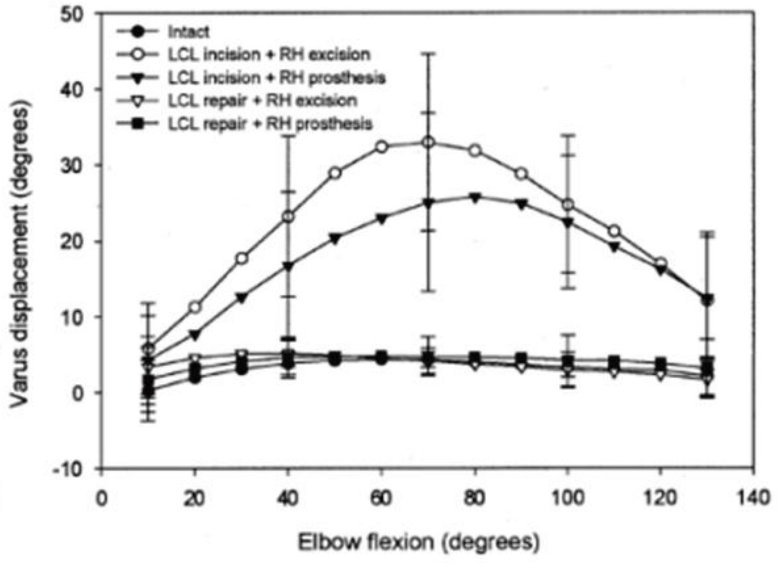


Figure 7.13: Valgus angle for different flexion angles (Jensen et al., 2005)

7.2.4.2 Moment Arm

Additionally, Figure 7.14 illustrates the moment arm of biceps (Murray et al., 2002), it exhibits a very similar trend of moment arm in Figure 7.14, and therefore, these results are in agreement.

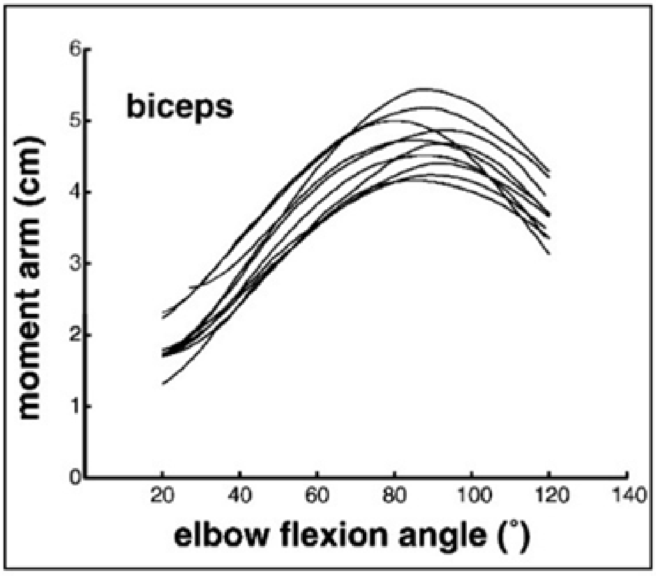


Figure 7.14: Moment arm of biceps for 10 different specimens (Murray et al., 2002)

Table 7.1: Experimental moment arm results for elbow joint muscles (Murray et al., 2002)

Muscle	Mean (cm)	Range of Peaks (cm)	Angle of Peak (degrees)	Range of Angles (degrees)
Brachioradialis	7.7	7.0 - 9.0	108	100-118
Biceps	4.7	4.2 - 5.4	88	80-93
ECRL	3.2	2.6 - 4.5	106	99-115
Brachialis	2.6	2.1 – 3.0	88	76-102
Pronator Teres	1.7	1.3 – 2.0	100	94-113
Triceps	-2.3	-1.8 to -2.8	44	1-62

The comparative studies carried out for the physics based simulation results show that these results are comparable with other results obtained from MJM, both experiments and published materials. It is not possible to have a perfect match, but the overall behaviour of joint kinematics is sufficient to claim the effectiveness of game physics based modelling and simulation. These comparative studies reveal the capabilities of the physics engine and gaming platforms to be used to model articular joints. Moreover, since this is the first investigation of the physics engine (PhysX) and gaming platform (DX Studio) with a musculoskeletal model, it can be said that implementing gaming platform in orthopaedic applications is expected to drive joint modelling to superior levels.

CHAPTER 8

MANUFACTURING OF CUSTOM MADE IMPLANTS BASED ON PHYSICAL SIMULATION

8.1 Introduction

In this chapter, the framework of an integrated environment for manufacturing personalised implants is proposed. The work carried out in previous chapters forms the foundation of the proposed virtual rapid prototyping environment. The integrated environment comprises physics based modelling and simulation as the core components of analysis. Such analysis is responsible for testing and assessing implant design at an early stage of production to discover any abnormal motion or loading. Due to the digital nature of the virtual modelling and simulation based on the physics engine and gaming platforms, integration with virtual manufacturing systems would be an effective route to the manufacture of personalised implants.

The use of an artificial implant is considered to be the most efficient way for patients to recover the regular performance of a diarthrodial joint affected by significant bone problems related to bone tumour resection. In such cases, the manufactured implant is required to satisfy a number of conditions before it can perform as a healthy joint would. Such attributes involve the mechanical requirement that the prosthetic implant will maintain sufficient strength and stiffness to support the structural system under various loads. Another important issue here is that of e geometrical attributes, in particular with replacements customized to match patient-specific requirements

(Hutmacher, 2000, Sun et al., 2004). The implant material also has certain requirements regarding biological factors, which should be considered carefully, for example, biocompatibility and the motivation for bone growth. Considering that each patient has a unique musculoskeletal system, rapid manufacturing is used in the manufacture of customized prosthetic implants for joint reconstruction and repair**.** The very first purpose intended for rapid prototyping seemed to be simply to validate a particular product design and the appearance of concept products. Subsequently it was given a role in the design process. For instance, in medical applications, the rapid prototyping (RP) of bone and soft tissues is now used by surgeons to devise more innovative strategies in surgical treatments and procedures.

Formerly, artificial joint replacements and implants surgeries normally used standard-sized replacement parts picked out from a variety supplied by manufacturers on the basis of anthropomorphic statistics. This has been adequately successful for certain types of treatment, but not many. Clearly, there are always patients beyond the standard range of sizes, as well as distinctive needs and specifications which may be pertain to health issues or even genetics.

8.2 Functional Rapid Prototyping in Medical Applications

Functional Prototype is an effective try in order to mimic the final shape, features, appearance, and materials for the required product. Rapid prototyping was essentially instituted to enhance and accelerate the development of fresh products. Since it has been widely acknowledged, the concept of a rapid prototyping may be simply described as a way of combining several different yet relevant techniques, which were mainly developed for making very complex physical designs, and producing physical models instantly from three dimensional virtual designing or CAD packages.

A number of technologies are available for rapid prototyping, such as Laminated Manufacturing (LM), Stereolithography (SLA), Selective Laser Sintering, Fused Deposition modelling and others. In addition to the several available techniques for rapid prototyping, the modern development of RP systems has introduced a variety of materials to choose from. The flexibility in the material range may enhance the capacity of the RP to be implemented in numerous fields, such as medical applications and the manufacturing of orthopaedic devices. Moreover, initial research clearly shows many opportunities for the modern advance of RP in various industries, including those with medical applications. Since every single patient is one of a kind, RP systems are employed in manufacturing customized orthopaedic implants for reconstructive surgery. Making use of RP in the production of custom-made implants result from the efficiency and consistency of the LM in producing complex and sophisticated shapes. The physical model thus produced may itself be used as a functional prototype.

This chapter illustrates a proposed strategy for manufacturing custom made implants by RP. The proposed method is mainly concerned with the final geometrical design of an artificial orthopaedic device which leads to the best performance in joint articulation. Because the final aim of this study is to establish an integrated manufacturing environment for producing patient specific implants, the modelling and simulation platform developed in this research was used as an analysis platform, which was expected to play an indicative part in the manufacturing process. This chapter addresses the innovative opportunities of integrating RP technology with the modelling and simulation approach developed in this thesis to achieve an integrated manufacturing environment which would contribute to the production of custom made orthopaedic devices.

8.3 Integrated Environment Framework

The layout of the proposed manufacturing integration is illustrated in Figure 8.1; the physics based simulation solution was the core analysis tool in the integrated environment. This analysis was carried out in real time based on physical interaction in the virtual environment, which employs a physics engine. Although medical expertise has an important function in assessing the implant design, the proposed virtual environment was expected to provide a useful solution for assisting medical decisions. The interactive nature of the virtual environment provides endless possibilities of joint configurations and also implant designs and alignments of implant installation. The implementation of this tool in an integrated manufacturing environment was expected to effectively enhance the production of personalised implants.

The flow chart in Figure 8.1 demonstrates the proposed integration between the modelling and simulation solution developed earlier in this thesis and the rapid prototyping systems, in order to establish an integrated virtual environment for manufacturing custom made implants.

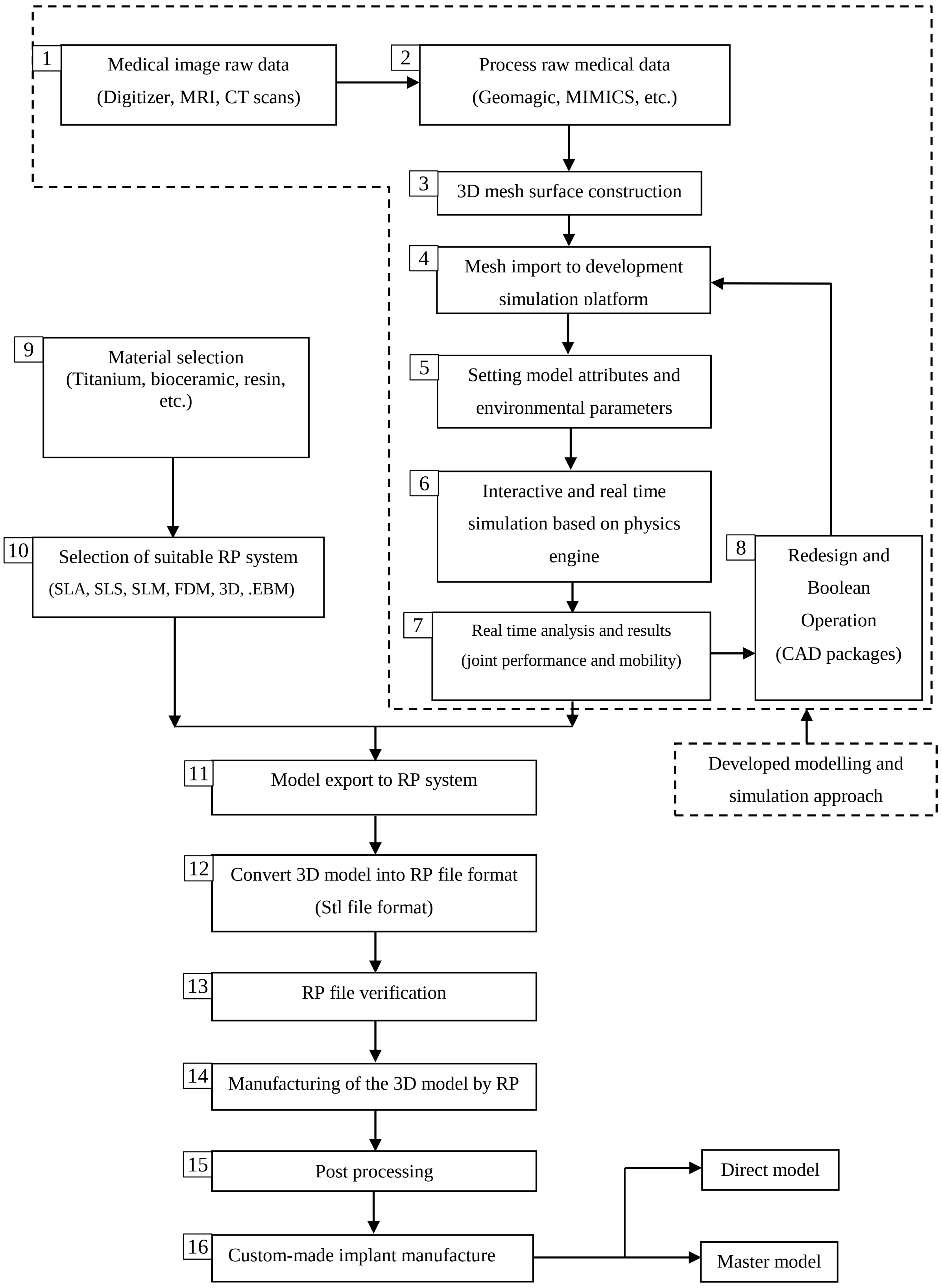

1 Medical image raw data (Digitizer, MRI, CT scans)
2 Process raw medical data (Geomagic, MIMICS, etc.)
3 3D mesh surface construction
4 Mesh import to development simulation platform
5 Setting model attributes and environmental parameters
6 Interactive and real time simulation based on physics engine
7 Real time analysis and results (joint performance and mobility)
8 Redesign and Boolean Operation (CAD packages)
9 Material selection (Titanium, bioceramic, resin, etc.)
10 Selection of suitable RP system (SLA, SLS, SLM, FDM, 3D, .EBM)
11 Model export to RP system
12 Convert 3D model into RP file format (Stl file format)
13 RP file verification
14 Manufacturing of the 3D model by RP
15 Post processing
16 Custom-made implant manufacture
Direct model
Master model
Developed modelling and simulation approach

Figure 8.1: The proposed integration of simulation software with the rapid prototyping for customized implants

1) The raw medical images are acquired as described thoroughly in section 8.3.1. The integrated environment accepts multiple data and images file format; such as CT scans, MRI, iges files, etc.)
2) The raw medical images are imported via software packages such as; MIMICS that can process DICOM image slices, and Geomagic for ".iges" files.
3) The raw data files are then processed in order to build the 3D mesh surface. In this stage, the data file are checked, cleaned and fixed as required for any surface defects or missing data.
4) The completed 3D mesh files are exported to modelling and simulation platform in acceptable file format (.x, .dae, .fbx).
5) The joint model is developed by importing the bone mesh data file. Then the ligaments and muscles are attached to their insertion points. Mass, surface stiffness and tissues stiffnesses values are also assigned to all tissues and then the bones are allowed to rest on each other.
6) The interactive simulation starts to perform the required joint motion.
7) Joint kinematics are investigated and analysed virtually to examine the ROM, COR, laxity, impingement, and any other required investigation.
8) If there was any disorder with the investigated joint, artificial joint replacement is installed and the simulation is performed again. Artificial implant can also be subject to design modification and tested. This can be performed until a satisfaction results are obtained.
9) The material selection depends on several criteria's, such as joint loading, biocompatibility, installation, implants design, etc.

10) The rapid manufacturing system is selected according to the implant material and design. (e.g. for Titanium alloys, solid free form fabrication with electron beam melting is appropriate).
11) After the design is finalized in step 8. The mesh data file is exported from the modelling platform to STL file conversion software like Magics RP.
12) The mesh data file is converted into STL file format, where only the 3D geometrical surfaces are preserved.
13) The STL file is then verified and checked and repaired for any defects, imperfections, flipped triangles, bad edges, holes, gaps, and overlapping surfaces.
14) The STL file is then exported to the rapid manufacturing system or the 3D printing system for production.
15) The manufactured part or prototype may subject for additional post processing to refine surface quality and maintain the tolerance.
16) The product from this virtual environment is used according to the objective it’s been built, whether as a functional prototype, user experience prototype, or a final product that can be used directly.

8.3.1 Medical image acquisition and construction

Medical imaging solutions, including Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) scans are widely employed in medical applications to visualize internal tissues and bones and can also be used to generate 3D images with the aid of image processing packages such as the Mimics program (Materialise, 2010). Most CT scans and MRI imaging systems generate a large number of slices in DICOM file format, which can be used by Mimics software to construct the 3D image of the internal structure (Materialise, 2010). The generated image may then be saved in an appropriate format to import to the developed modelling software, as discussed in Chapter 4.

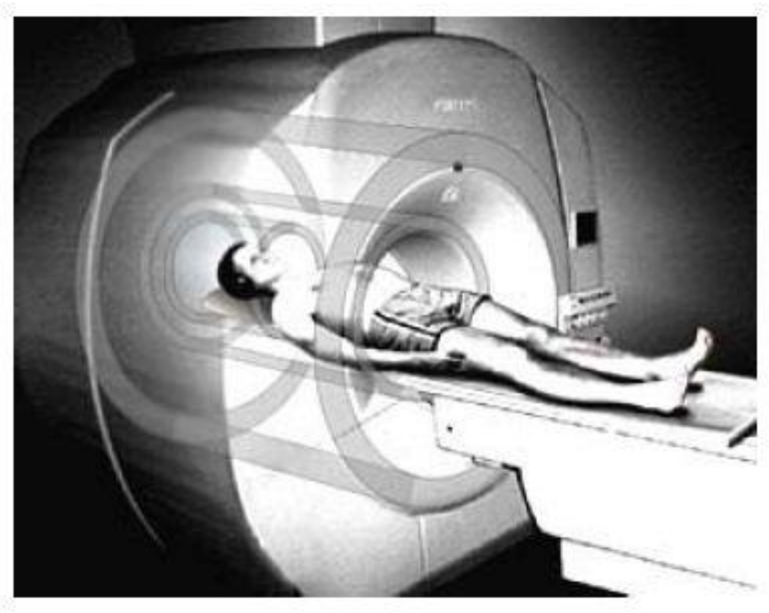


Figure 8.2: Magnetic Resonance Imaging (MRI) unit (Lauterbur and Mansfield, 2003)

The recent technology in medical imaging has contributed to the increase in innovative systems for manufacturing medical products. Internal body structures can be visualized by such medical imaging solutions as MRI and CT scans. The generated medical images may provide practical data files whose function is to acquire the geometries of the internal structures of the human body in a 3D space with the help of commercial software packages such as Mimics. The raw data files are processed and reconstructed in a 3D workspace and then exported, as described above, to build the musculoskeletal model.

In the current project, the raw image data were acquired by the FARO arm digitizer at the Brunel Orthopaedic Research and Learning Centre. Then the cloud of points was processed in a Geomagic software package to build the 3D virtual model of the bone and construct the surfaces. Additional surface enhancements were also used to eliminate any surface roughness and discontinuities. Then it was saved in “.x” file format for exporting to the gaming platform.

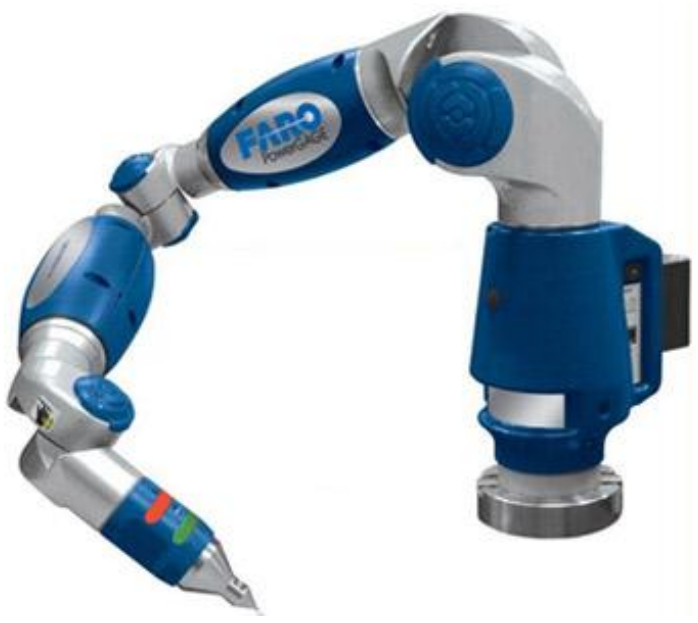
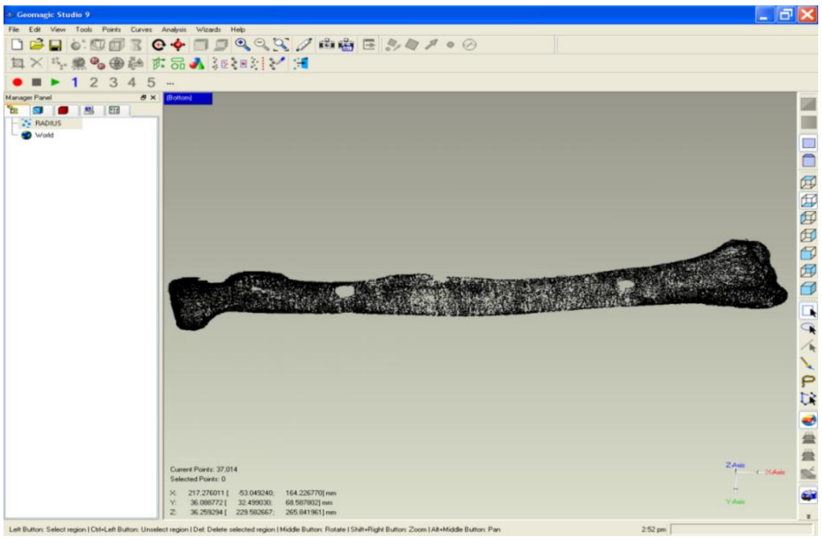


Figure 8.3: Faro Arm digitizer at Brunel Orthopaedic Research and Learning Centre



```
Current Points: 37,014
Selected Points: 0
X:   217.276011 [   -53.049240;   164.226770] mm
Y:    36.088772 [    32.499030;    68.587802] mm
Z:    36.259294 [   229.582667;   265.841961] mm
Left Button: Select region | Ctrl+Left Button: Unselect region | Del: Delete selected region | Middle Button: Rotate | Shift+Right Button: Zoom | Alt+Middle Button: Pan      2:52 pm
```

Figure 8.4: Radius bone scanned in Geomagic software

Additional details of the model construction can be found in section 4.2.2.

8.3.2 Modelling and simulation based on the physics engine

The modelling approach based on the physics engine and gaming platform which was proposed in Chapters 3 and 4 to model and simulate the human articular joint, was implemented in the integrated environment as the core analysis tool. It was used to test and validate the implant design and installation configuration, for instance in axis alignment. This investigation is based on a patient's anatomical data, acquired by medical imaging as discussed in section 1). This provides a personalised environment which can deal with each patient individually. The interactive simulation can also be used to investigate the joint articulation under different joint loading and movement conditions. The results obtained from the physics based simulation was expected to guide the major design issues, such as maintaining the natural centre of rotation and the range of motion of the diarthrodial joint. As the COR of the human articular joint changes constantly during articulation, any installed implantation should maintain the original COR to avoid the chance of functional disorders of the joint in the future. For instance, a fixed COR during joint motion may cause cartilage wear and hence nonuniform loading and impingement problems. In addition, the natural mobility of the joint will essentially affected by reducing the DOF, which results in undesirably constrained joint articulation.

The geometrical parameters can be dealt with in the virtual environment by avoiding any abnormal joint motion. For example, if the design shows a hypermobility of the joint, the prosthetic implant may be enlarged so as to stabilize the joint. Equally, immobility may be treated by increasing the thickness of the cartilage. The joint can then be tested against the range of motion that it can perform. Such assessments may only be possible in the virtual environment, which thus provides a viable tool for articular joint analysis.

8.3.3 Design assessment

Before the final manufacturing stage, the design of the model should be evaluated before conversion to STL and being sent to the RP system. This evaluation is mainly to allow surgeons to examine the appropriateness of the model from several aspects, such as the fixation device, bone substitution, errors due to medical images, implantation procedure and other factors related to anatomical understanding and surgical procedures. This assessment is performed mainly by surgeons and bioengineers, with the aid of the CAD software packages available. This step is critical for avoiding problems associated with the design and it therefore enhances the quality of the model.

8.3.4 STL file generation and fixation

The STL data format describes only the 3D geometrical surface of the object without considering other attributes, such as colour or texture. After the validation of the design by physically based simulation, it is converted to STL file in order to proceed to the remaining manufacturing stages. The data can be converted from an “.x” file format to STL by several CAD software programs. As shown in Figure 8.5 (a), the STL file may contain some defective triangles, which should be repaired. Their defects may be flipped triangles, bad edges, or holes, gaps and overlapping surfaces. However, Magics RP from Materialise repairs STL files automatically and also manually after they are repaired, the triangulated geometry is ready to be exported to the CAM system. The STL file above has been sent to a 3D printing machine and the manufactured part is produced by rapid prototyping (see Figure 8.5 b).

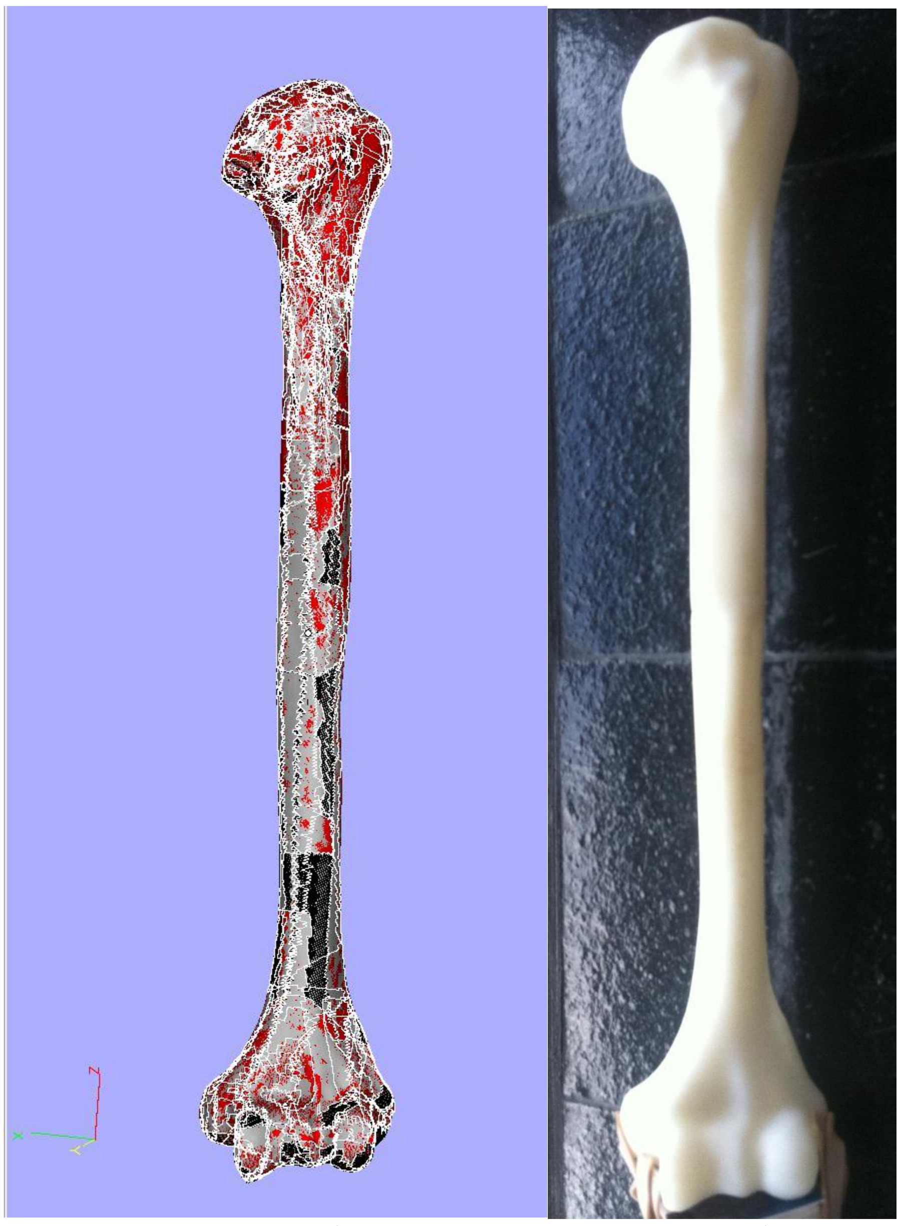


Figure 8.5: a) STL file of the humerus showing the surface defects in white b) Manufactured part of the STL file on a 3D printing machine

8.4 Fabrication of Functional Rapid Prototypes

By means of technologies for rapid prototyping, the physical products are manufactured with the substance being added in successive layers. Usually, all rapid prototyping solutions tend to be established within the same strategy, as described here.

Beginning with a 3D digital design developed or imported into a CAD system, the model must then be imported to the specific rapid prototyping program for the chosen RP system. In this step, the software splits the 3D model into parallel sections and generates the sectional shape of each plane. The RP system successively produces the sectional shape of each plane to build up the complete shape composed of the parallel layers.

8.4.1 Example of rapid manufacturing technology selection

A number of rapid prototyping technologies are available; some of them, however, are considerably better than others for a given situation. For example, material, accuracy, surface finish and mechanical properties limitations are considered when producing functional prototypes. In articular joint modelling and simulation, personalised prosthesis manufacturing for an articular joint seems to have a rich environment for further integration with the modelling and simulation approach.

Once the design is finalized and confirmed, the next step is to select the most appropriate rapid prototyping technology to produce the functional prototype. This selection is influenced by the purpose of the product itself and also its accuracy, material, surface characteristics and mechanical properties, etc..

As the RP technology has determined, the 3D digital file must be imported by the rapid prototyping software in the ".stl" file format. It must not be forgotten that the quality of the final product is greatly affected by any reduction in the 3D image file; thus, any compression of this file may lower the quality of the manufactured product. It should also be pointed out that the alignment, positioning and choosing of the right parameters in the RP system are critical before manufacturing can begin.

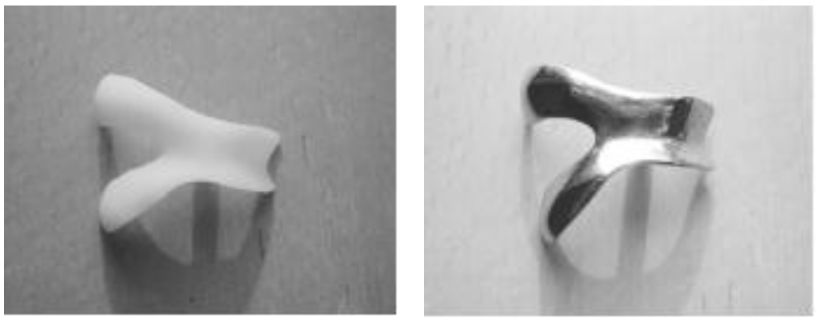


Figure 8.6: SL pattern of a hemi-knee joint (left), the titanium alloy joint (right) (He et al., 2006)

8.4.2 Solid free form fabrication with EBM

After confirming the design for the customized implant, the product should be fabricated from a suitable material. The most usual method for custom implants is CNC machining and, since personalised implants are very complex in their geometrical shape, the 5-axis CNC system is commonly used to manufacture the product. This process of manufacturing the orthopaedic device presents a number of challenges. For example, generating the code which drives the tool path is rather difficult and the installation of parts and setting up of the machinery may demand appreciable skill and time. Furthermore, the cutting process itself requires considerable machining time for removing the volume of the material from the stock piece. The material removed in the form of chips may reach up to 80% of the original stock bar. This may significantly affect the cost of the product since biocompatible materials (e.g. Titanium, Cobalt-chromium) tend to be expensive. Such drawbacks indicate that CNC machining is not the best option for manufacturing customized implants, in particular with regard to fully automated systems and an integrated environment.

Another technique which can be implemented in the manufacturing of custom-made orthopaedic devices is solid freeform fabrication (cladding). Although this method is cost effective and allows complicated customized geometries to be produced, it also entails some limitations when it comes to the biocompatibility of the material to be used. An efficient technique which can be used to produce customized orthopaedic device based on solid free form (SFF) principles rests on electron beam melting. This method is able to produce a complete metal-dense part through multiple layered fusion. The procedure begins simply by spreading a layer of the metal powder (i.e. about 0.12 mm for titanium) on the building chamber base in a high vacuum. Then an electron beam with adjustable power is concentrated on a point in the metal powder with a diameter of about 0.1 mm, causing the metal powder to melt and form a floating melted pool. The electron beam causes this melting pool to spread along the complete cross-section of the layer which it affects until a solid slice is completed. The second round begins by lowering the base while adding a fresh layer of the metal powder and performing the same process for the upper layer. This process is repeated as many times as required until the complete part is created.

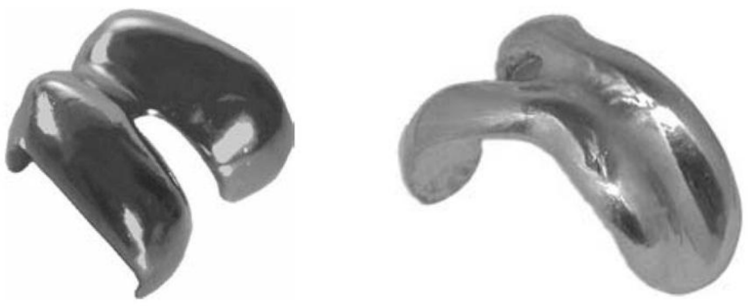


Figure 8.7: Custom- made femoral head by EBM a) Stainless steel b) Titanium alloy

8.5 Integrated Environment for Manufacturing Personalized Orthopaedic Devices

The ideas illustrated in this chapter focus on advanced manufacturing technologies to extend the current principles of direct production into an automated feedback originating from the implant performance in a virtual environment. The proposed framework may significantly enhance the product’s quality, performance and cost-effectiveness, in particular when the rate of production is low and where the design is conveniently customized. It is noted that rapid prototyping technologies are the most convenient systems for integrating with the virtual simulation software because of their capacity for complete automation. For medical aspects, such as the biocompatibility of the implant material, various techniques are available for applying RP to a wide range of materials for use in the manufacture of customised orthopaedic devices.

The proposed framework is expected to provide a potential virtual environment for manufacturing such devices. This integrated environment is thought to be the rational extension of the physics based modelling platform developed in the present research. The proposed virtual environment is an example of the novel implementation of game physics in orthopaedics.

CHAPTER 9

DISCUSSION, CONCLUSION AND THE FUTURE WORK

9.1 Thesis Summary

While the previous chapters have discussed the results obtained and the future application of game physics to orthopaedic applications, this section mainly discusses the overall contribution of the work presented in this thesis and its importance to the field.

The work presented in this thesis was carried out to achieve the research objectives listed in section 1.4 above.

The discussion begins by listing the chapters of the thesis, as follows:

Chapter 1 presented a general introduction, the need and motivating reasons, the research objectives and research significance for such an investigation into orthopaedic applications.

In Chapter 2, the survey of relevant literature work was presented. It focused on multibody modelling in general, as well as the kinematics and dynamics of articular joints. Physically based modelling, contact modelling and physics engines were also reviewed.

In Chapter 3, the principles and theories of game physics were illustrated. In addition, a preliminary validation of the physics engine (PhysX) and the gaming platform (DX Studio) were carried out using virtual experiments.

Chapter 4 showed the framework for developing the musculoskeletal joint model on a gaming platform. The model was constructed using real medical data. Muscle and tissue wrapping by segmentation were also presented. In addition, the calculation of COR and moment arm were shown.

In Chapter 5, the physics based simulation results of elbow joint were presented.

In Chapter 6, a novel image based system for joint kinematics was proposed for experimentally validating the physics based simulation results. This image based system was developed in this work to experimentally analyse the kinematics of the elbow joint.

In Chapter 7, comparative studies were conducted of the physics based simulation results. In this chapter, the comparison was carried out using MJM software and published materials, together with experimental results obtained from the image based analysis and the Brunel Orthopaedic Research and Learning Centre.

In Chapter 8, a new framework was proposed for an integrated manufacturing environment for custom-made orthopaedic devices. The core of this framework was the physics based modelling and simulation presented in this thesis.

9.2 General Discussion

9.2.1 Physics based modelling and simulation

The physics based virtual environment was investigated with some virtual experiments to evaluate the capabilities of gaming platforms to be further investigated with the joint musculoskeletal model. The initial validation was performed with mechanical systems, such as a mass spring system with linear vibration. The system response showed some energy dissipation, even when damping coefficient is set to zero. It was noticed in this experiment that internal damping had been introduced to gaming platforms in order to stabilize the system. In the first virtual experiment, the periodic time from system simulation (*T=3.11 s*) was found to be very close to the analytical solution (*T=3.1416 s*), that is, with less than 0.1% error. Considering that the physics engine and gaming platform have been designed and developed for video games, this percentage of error is within the acceptable tolerance.

After the preliminary assessment of the physics engine and gaming platform carried out in section 3.5, it was decided to go further and investigate the musculoskeletal model on a gaming platform. The musculoskeletal model of the human joint was developed using joint anatomical data, its articulation being based on contact surfaces and the stiffness of the tissues surrounding the joint. The development of the musculoskeletal model was discussed in detail in Chapter 4. Joint model kinematics in game physics simulation were investigated to evaluate the gaming platform for orthopaedic applications. Although any diarthrodial joint can be used in this investigation, the elbow joint was studied because its anatomical data was accessible to our research group. Different movements of the elbow joint were performed to compare it with the results of MJM, published materials and experimental results, as demonstrated in Chapter 7. These comparative studies were performed to see how accurate game physics based modelling was in dealing with articular joints and whether this modelling system could extend to orthopaedic applications.

In the early stage of the current joint model, joint simulation encountered extremely unpredictable behaviour resulting from several factors, such as interpenetration at low FPS, high surface stiffness and undamped spring excitation. However, these issues were handled successfully without affecting the 6 DOF articulations, as is discussed below.

9.2.1.1 Contact modelling

In the developed virtual environment, the collision handling is implemented by means of three methods, empowered to operate together. The first method of collision detection is provided by PhysX from Nvidia, which supports full collision detection and response to physical objects on the scene. The second method is only for detecting collisions between physical and nonphysical objects (objects which take up space but have no inertial properties) on the scene. The third method is the implemented raycasting system which examines the objects with possible contacts.

Contact modelling on gaming platforms can be handled by the game engine, the physics engine, or both simultaneously. However, the physics engine (PhysX) is more accurate than the game engine when it comes to detecting collisions. The reason for its more accurate detection is that it ensures a proper collision response.

Initial investigation about contact problems was reported in section 3.5.2. For colliding objects with complete elastic collision, it was observed that the system after collision gains some energy. This can be explained by the implementation of the penalty method in the discrete environment. To put it another way, the simulation is carried out on a time increment and the collision may occur between two discrete time steps, causing penetration without losing kinetic energy. This in turn pushes the colliding object back, but with higher momentum. The effects of energy build-up in the system (which is associated with the internal implementation of the penalty function) can be moderated by minimizing the time increment (or increasing the frames per second FPS) of the simulation environment, or adjusting the coefficient of restitution of the colliding surfaces to a small value (for example, 0.1) to overcome the energy build-up.

Another issue to discuss here is the friction force for sliding contact. The physics engine (PhysX) supports static as well as dynamic friction force. It was noticed that, even with frictionless surfaces, there is a small friction force which slows down the sliding object. This is intended to maintain a stable simulation in gaming platform, for a stable environment in these platforms is a higher priority than accuracy is.

9.2.1.2 Muscle and tissue wrapping

For many reasons, muscle and tissue wrapping is a critical issue to focus on in joint modelling. Muscle and tissue wrapping is responsible for determining the shortest path between two points on the bone surface which is defined by the mesh data. The segmentation of muscle into multiple masses connected by springs however, reveals an effective technique for muscle and tissue wrapping. The shortest path between two points on the meshed surface should have the minimum energy level. The tensioned springs tend to settle at the lowest potential energy level, which in this situation is the shortest path. In the preliminary validation, a number of masses connected by springs were allowed to wrap around a cylindrical object to imitate the muscle wrapping around the bone. The virtual validation in section 3.5.3 shows a consistent technique. The wrapping was fast and the shortest path was found to be accurate, without any complications. However, the accuracy depends on the number of segments in the muscle model. More segments will lead to a smoother wrapping and more accurate determination of the shortest path. Conversely, a high number of muscle segments will result in more computations and therefore a slower system response. Another issue to discuss here is the gap between the masses which compose the muscle. If the gap between two masses is increased and the spring is over-stretched, the spring may cross the mesh surface and introducing an inaccurate line of action. This can be avoided if the gap size is maintained, increasing the number of mass-spring segments. This use of the ability of the gaming platform to simulate muscle wrapping is one of the most significant contributions of the present research and is not available on industry standard musculoskeletal modelling systems.

9.2.2 Musculoskeletal joint model

Since the developed joint model considers the contact surfaces, surface geometries of the joint bones are required. Geometrical information of bones is obtained by the FaroArm Platinum digitizer, which generates a cloud of points forming the bone surface in “.iges” file format; more details are available in section 4.2.1 and 4.2.2. The accuracy tolerance of the scanned geometry is 0.02 mm for the current working volume, as provided by the FaroArm manufacturer. This accuracy of the data relates only to the scanned raw data of the bone surfaces, but not the processed data files. Processing the ".iges" file in Geomagic, as outlined in Chapter 4, to produce triangular mesh surfaces may affect the accuracy of the generated surfaces. However, a denser cloud of points provides more accurate and smoother surface geometry. Higher mesh density requires more computations and processing at the simulation time. Nevertheless, lower mesh density may directly affect the accuracy of the analysis, since the developed joint model considers contact surfaces for joint articulation. For this reason, it is important to optimize the number of mesh triangles for optimum performance without affecting the accuracy of the analysis. Contact surfaces were provided with the highest available mesh density than other surfaces and these were provided with lower mesh density where the level of surface smoothness was not important. This mesh optimization lowered the number of mesh triangles for the radius bone from 75000 triangles to 7000, without significant change to the smoothness of the contact surfaces.

The selection of tissue stiffness is a critical issue in constructing the joint model. Although the ligament and muscle stiffness were obtained from previous studies, it was used as a starting point. Stiffness tuning was required, to stabilize the joint. Joints with the level of ligament stiffness obtained from the literature exhibited unstable response, for several reasons. For example; insertion points in anatomical tissue are infinite in number over an area which, in the current joint model, is composed on discrete single insertion points. Moreover, the position of the insertion point is selected interactively on the screen, which may not be the natural position.

The muscle and ligament insertion positions raised another issue for discussion. Since muscle and ligament insertion positions are discrete points instead of attachment areas, as with natural joints, the insertion points need not be precisely based on anatomical descriptions. This is because the joint can be stabilized only after a number of trials to examine multiple insertion points which give more stability at dynamic as well as static positions. This adjustment of the insertion position is performed by virtual experiments on the screen to test the positions of the insertion points and examine the stability of the joint with each new configuration of the insertion points. Some tissue arrangements are stable in a static joint position, but in a dynamic joint may require different insertion points because activating the muscles will change the joint loading and therefore its stability.

9.2.3 Image based system

The image based analysis system proposed in this thesis was developed to examine joint kinematics and laxity for the purpose of verification and validation of the results obtained from the physics based simulation. This is a minimally invasive system which can accurately indicate deformation of the valgus-varus during flexion-extension movement of the elbow joint and also joint laxity as discussed in detail in Chapter 6. The marker positions in this technique were chosen according to the minimum skin movement at the bony positions (two markers on epicondyles at the distal side of the humerus and the third marker at the apex of the ulnar process). These marker positions were verified by placing a stationary reference marker (at the centre of the ulna near the elbow joint and between the two markers on the epicondyles) and applying a lateral force at a 90^{o} flexion angle to see how the marker positions are affected. This application shows no significant skin movement at the selected markers positions. However, sources of error for the angle measurements may be due to the following:

1) Errors due to relative skin movement or misplaced marker

This error depends on the patient’s weight and the amount of fat in the arm. The higher the percentage of fat in the body, the greater the skin movement and therefore the higher the error that can be expected. It is sometimes very difficult to determine the right place to locate and stick the markers on the specified bony positions because of the high percentage of fat underneath the skin; in such cases this experiment is hard to perform properly. Thin people are more likely to have minimum skin movement and therefore yield more accurate results. However, if the bony positions are visible and can be easily determined (which is the case with most patients) the skin movement can be neglected.

2) Errors due to location of the marker position

Inaccurately locating the centre of the markers is another source of error in this system. This error cannot be avoided, since the chosen location is manually determined by the user on the screen. This error can be minimized by using smaller markers with reflective colours, which make it easier for the user to identify the marker centre more precisely. In the worst situation, provided that the selected positions are not the centre but on the markers, the error diverges no more than 10% from correct calculated distortion angle. It can be reduced by repeating the process of location the markers several times and averaging the distortion angle, using the same captured images (without retesting the joint). However, the proposed system can be modified to eliminate this drawback by automatically determining marker centre using pattern recognition.

Although the developed system is oriented for the elbow joint, it may be extended to investigate by marker position analysis other human joints based on locating specific points which have minimum skin movement.

9.2.4 Validation of the Results

An evaluation of the presented results is discussed here to validate the physics based modelling approach proposed in this thesis. The results have been compared with other results from different sources. Because of the unique nature of the problem presented in this thesis, it turned out to be challenging to produce matching results and even highly unlikely that they would match due to a number of factors, as discussed in the chapter of comparative studies (Chapter 7). Consequently, it is submitted that the validation process of the results is based on the general behaviour and pattern of the joint kinematics. Comparing the obtained results with the other experimental results obtained at the Brunel Orthopaedic Research and Learning Centre shows comparable results. Similarly, the comparison with the MJM and published material mentioned above confirms the general behaviour and articulation pattern of the human joint.

9.2.4.1 Valgus-varus deformation and joint laxity

Flexion-extension movement for the elbow joint was performed in physics based virtual environment. This movement was also carried out experimentally to examine valgus-varus deformation during flexion-extension without any external lateral force. The physics based simulation showed valgus-varus deformation of 9.5^o during joint movement from 42^o to 180^o flexion angles. The marker based experimental results showed an average valgus varus deformation of 10.2^o during flexion-extension movement from 60^o to 180^o flexion angles. The difference in valgus-varus deformation between the marker based experimental and the physics based simulation results was 7.3%. This percentage is acceptable if it is recalled that in the virtual modelling environment the stiffness and insertion points are fine-tuned to deliver a stable joint. Still, every individual is unique, with his own joint characteristics, and such variation is very common.

For joint laxity, the physics based simulation investigated joint laxity and found that the joint laxity is 4.5^o at a flexion angle of 90^o under a lateral load of 7 N. The joint laxity was also investigated using the developed image based system. The experiment was carried out on 9 volunteers and the obtained results were averaged to eliminate the variations between individuals. With 7 N loading, the average joint laxity was found to be slightly below 2.5^o. Even a difference of 44.4% in the variation between the physics based simulation results and the marker based experimental results can be explained in a similar way by the variation of valgus-varus deformation during joint flexion-extension movement. However, this sudden increase in variation percentages (7.3% for valgus-varus without loaded joint and 44.44% for laxity investigation with lateral load) is caused by the greater sensitivity of joint stability to lateral load, in particular if there was no lateral load when different degrees of stiffness and different insertion point configurations were tested to stabilize the joint.

The obtained simulation and experimental results were compared with published materials (Floris et al., 1998, Jensen et al., 2005). These published results were obtained experimentally for in vito elbow specimens. The difference in lateral applied force value inherently affects joint laxity. Nevertheless, the behaviour of an intact elbow joint is comparable to an acceptable extent with the findings in published materials.

9.2.4.2 Instantaneous centre of rotation, COR

The findings on the instantaneous centre of rotation or COR were compared with the results obtained from MJM and experiments at the Brunel Orthopaedic Research and Learning Centre, using the Stewart platform. Simulation results as well as those of MJM and the Stewart platform show the COR moving in 3D space. Although the behaviour of the COR is not exactly the same for the three methods, this is because each method uses a different axis alignment of the coordinate system. In addition, for contact driven articulation, mesh density can directly affect COR. In such a situation, it is more important to look at the range of COR obtained, which shows a comparable range of data for the three methods. As a first implementation of game physics to obtain the instantaneous centre of rotation in an articular joint and compare it with a specialized software package such as MJM, and with the results of experiments, it can be said that the simulation results obtained are within the appropriate range.

9.2.4.3 Moment arm

The moment arm for bicep is another simulation result which was compared with MJM software and published material. The maximum value of the moment arm from the obtained results was found to be 4.25 cm, with 4.1 cm from the MJM software for the same meshed data files. This difference arises mainly because of the insertion point of the bicep, which is very sensitive when it comes to moment arm calculation. However, a difference of 3.5% in comparing our results with those of a specialized software program can validate the simulation results. Moreover, the published results for calculating moment arm experimentally were also comparable.

9.2.5 Virtual integrated environment for custom-made implants

After evaluating game physics for orthopaedic applications, a potential framework for an integrated manufacturing environment is proposed. Although this was not implemented in the present research, this integration is expected to be a natural extension of it. The modelling and simulation based on game physics will serve as the analysis tool in the integrated environment, which provides virtual manipulation, testing and design assessment in the early stages of implant production. The analysis results will interactively assist the design and testing of custom-made implants. They will also help to investigate how joint mobility can be restored by changing different design parameters of the orthopaedic implant. Another implementation of this environment is the pre-surgery planning which can be used for the rapid prototype joint model based on patients' anatomical joint data. This pre-surgery is preferred, because it avoids unexpected surgical challenges and increases the success rate of surgical operations.

The integration between physics based modelling and simulation, computed aided design and the manufacturing system is highly desirable, but this type of integrated environment does not exist yet. This integrated environment will reduce the cost, achieve better design, improve the performance of artificial joints and speed up the manufacturing of custom made prosthetic implants for individual patients.

9.2.6 Summary

It is highly preferable to model and simulate human articular joints in an interactive environment. It is a considerable challenge to develop interactive software to deliver precise simulation with flexible joint configurations for anatomically based models. Nevertheless, modelling software based on game physics was developed to evaluate the possibility of such a gaming platform for use in modelling articular joints. The modelling software is still in constant development to add features and further integrate it with a rapid prototyping system for producing customized implants based on the local analysis of joints. Although gaming platforms are not designed for joint modelling, by exploring their potentials and their novel implementation in orthopaedic applications is expected to make a significant impact on articular joint modelling and simulation.

9.3 Conclusion

The work presented in this study reveals the novel use of a gaming platform in modelling musculoskeletal structure. Modelling of musculoskeletal structure in their anatomical fidelity has been the ultimate goal of the modelling technology. The difficulty which makes the modelling in this form is almost impossible is due to the fact that the joint is forced contact maintained and bone surfaces are described by high density mesh. Describing the problem in this format poses an array of fundamentally difficult problems, such as contact collision, collision response and time domain integration in a discrete environment, which may result in numerical instability. It is shown in this study that the proposed modelling environment driven by game physics can overcome all the problems presented in this thesis.

This research is summarized by the following concluding remarks:

a) The anatomical bone surfaces of the articular joint are measured by a FaroArm digitizer. The developed joint model is provided with 6 DOF and contact driven articulation at interactive simulation speed, which is not provided by any other joint modeller.

b) The novel implementation of a physics engine (PhysX) and gaming platform in the modelling system helped to tackle a number of difficulties involved in modelling an articular joint such as simulation stability, cost effectiveness and processing time. PhysX employs a physics processing unit (PPU) and this makes it very fast since the computations are processed by the hardware. The performance of the developed solution clearly demonstrates this.

c) The image based system was developed in this study to experimentally investigate joint kinematics and laxity. This minimally invasive method overcame several challenges posed by existing laxity measurement devices, such as inaccuracy, complication, impracticability and cost. This method, which is based on marker position analysis, is a practical and accurate way of investigating joint laxity. As the applied force increases, the change in medial laxity values can literally be seen, which validates the robustness and reliability of the technique despite its simplicity and non-invasiveness. Further statistical study may help to introduce some guidelines for the laxity of the healthy elbow joint.

d) Beside the 3D interactive simulation for elbow joint, valgus-varus deformation, instantaneous centre of rotation, joint laxity and the moment arm of elbow joint were investigated and the findings compared with the experimental results performed by the image based system, Musculoskeletal Joint Modeller software (MJM), experiments at the Brunel Orthopaedic Research and Learning Centre and published materials. The comparative studies of the obtained results demonstrate the fidelity of the proposed physics based modelling technique.

e) The performed simulation analysis shows the advantages of speed, accuracy and flexibility in the proposed model. The simulation results were obtained at interactive speed while it took about 90 minutes on MJM. The obtained results are encouraging and comparable with the experimental results carried out by the research group.

f) As a natural extension of the current research, the physics based modelling and simulation platform is to be integrated with rapid prototyping systems to assist the production of personalized implants. Simulation analysis is expected to produce indicative results to guide the rapid prototyping of custom implants and finally produce manufacturing implants specific to individual patients.

Such integration will speed up the manufacturing of custom-made implants at an optimal cost.

g) This research has a number of contributions to overcome some existing challenges with joint modelling, such as contact modelling, tissue and muscle wrapping and kinematics analysis.

   1) The contact modelling was tested virtually and shown to be stable in a resting position as well as in motion. The use of bounding volumes to avoid unnecessary collision checks optimised the computations required for contact modelling. Objects with possible collision received face to face interference checks automatically to determine the exact point of contact. This multistage contact check provided a stable simulation environment at interactive speed.

   2) Collision response for colliding as well as resting physical objects was carried out using PhysX. The coefficients of restitution and the conservation of linear and angular momentum are the main aspects in deriving the proper collision response. Collision detection and response were found to be within an acceptable tolerance controllable by varying the density of the mesh.

   3) Muscle and tissue wrapping is for many joint modellers a considerable challenge which has so far not been effectively solved. The proposed segmentation method for muscle and tissue wrapping was tested to obtain the shortest path between two points on the bone surface by finding the minimum energy level. The contribution of this new method delivered an effective technique to accurately model muscle or tissue wrapping without intensive computations at run time.

4) The image based joint laxity measuring technique, which was developed on the basis of identifying stationary skin positions around the joint. These positions are the ones which remain visibly stationary when load is applied on the joint. This is a completely novel approach and has never been reported hitherto.

In conclusion, physics based modelling and simulation exploiting gaming platforms and physics engines such as PhysX have shown their suitability for handling unconstrained human articular joints with the natural 6 DOF. This modelling technique provides a promising virtual platform for articular joint diagnosis, laxity investigation, implant testing and design assessment. The contributions presented in this thesis overcome many modelling problems such as; model complexity, intensive computations, collision handling, muscle wrapping, simulation stability and many others. It has been shown that the gaming platform and physics engine provide a viable solution to human musculoskeletal modelling. The novel implementation of game physics in orthopaedic applications is expected to make a significant change in modelling human articular joints.

9.4 The Future Work

The investigation of game physics and whether it can be used for modelling the articular joint in a virtual environment has answered some questions and raised some others. In addition, the proposed virtual integrated manufacturing environment has revealed new vistas for personalised orthopaedics devices requiring further research and investigation.

The following are possible areas where the current research can be extended.

- A specialised system which employs game physics for orthopaedic application will enhance the development of musculoskeletal models. Ligaments and muscle insertion points will be replaced with insertion areas and more control over the cartilage thickness.
- The virtual environment can be used also to virtually evaluate whether the mobility of the artificial joint has been restored.
- The proposed virtual manufacturing environment with rapid prototyping can be useful in pre-surgery planning. Bones and tissue prototypes based on the patient's scanned tissues can be used for offline surgery planning to ensure in advance that the operation goes well.
- Manufactured personalised implants instead of prototypes can be directly implanted in a patient's joint.

REFERENCES

Abdel-Malek, K., Yang, J., Kim, J. H., Marler, T., Beck, S., Swan, C., Frey-Law, L., Mathai, A., Murphy, C., Rahmatallah, S. & Arora, J. 2007. Development of the virtual-human Santos\&\#174. *Proceedings of the 1st international conference on Digital human modeling.* Beijing, China: Springer-Verlag.

Adrian, M. & Cooper, J. M. 1995. *Biomechanics of human movement*, Madison, Wis., Brown & Benchmark.

Ageia, P., Nvidia. *The AGEIA PhysX accelerator* [Online]. Available: http://www.nvidia.com/object/physx_new.html [Accessed 03/11/2010].

Alrashidi, M., Yildiz, I., Alrashdan, K. & Esat, I. 2009. Evaluating elbow joint kinematics with the Stewart Platform Mechanism. *Modelling in Medicine and Biology Viii,* 13**,** 181-189

American Academy of Family Physicians (AAFP). Evaluation of Overuse Elbow Injuries - February 1, 2000 [Online]. Available: http://www.aafp.org [Accessed 08/12/2010].

An, K. N., Jacobsen, M. C., Berglund, L. J. & Chao, E. Y. 1988. Application of a magnetic tracking device to kinesiologic studies. *J Biomech,* 21**,** 613-20.

Andrea, C., Valentina, C. & Aurelio, C. 2004. Estimation of the centre of rotation: a methodological contribution. *Journal of biomechanics,* 37**,** 413-416.

Anybody, T. 2010. *The AnyBody Modeling System™* [Online]. Available: http://www.anybodytech.com [Accessed 05/11/2010].

Armstrong, W. & Green, M. 1985. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer,* 1**,** 231-240.

Asmi. 2007. *American Sport Medicine Institute* [Online]. Available: www.asmi.org [Accessed 05/11/2010].

Baciu, G., Wong, W. S. K. & Sun, H. Q. 1999. RECODE: an image-based collision detection algorithm. *Journal of Visualization and Computer Animation,* 10**,** 181-192.

Bandi, S. & Thalmann, D. 1995. An Adaptive Spatial Subdivision of the Object Space for Fast Collision Detection of Animated Rigid Bodies. *Computer Graphics Forum,* 14**,** 259-270.

Baraff, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH Comput. Graph.,* 23**,** 223-232.

Baraff, D. 1991. Coping with friction for non-penetrating rigid body simulation. *Proceedings of the 18th annual conference on Computer graphics and interactive techniques.* ACM.

Baraff, D. 1993. Non-penetrating Rigid Body Simulation. *Eurographics '93 State of the Art Reports.* Barcelona.

Baraff, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques.* ACM.

Baraff, D. 1996. Linear-time dynamics using Lagrange multipliers. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques.* ACM.

Bergen, G. V. D. *SOLID: Software Library for Interference Detection* [Online]. Available: http://www.win.tue.nl/~gino/solid/ [Accessed 25/09/2010].

Bianchi, G., Harders, M. & Székely, G. 2003. Mesh Topology Identification for Mass-Spring Models. *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003.* Springer Berlin / Heidelberg.

Bianchi, G., Solenthaler, B., Szekely, G. & Harders, M. 2004. Simultaneous topology and stiffness identification for mass-spring models based on FEM reference deformations. *Medical Image Computing and Computer-Assisted Intervention - Miccai 2004, Pt 2, Proceedings,* 3217**,** 293-301.

Blankevoort, L., Kuiper, J. H., Huiskes, R. & Grootenboer, H. J. 1991. Articular contact in a three-dimensional model of the knee. *Journal of Biomechanics,* 24**,** 1019-1031.

Bobbert, M. F. & Schenau, G. J. V. 1990. Isokinetic Plantar Flexion - Experimental Results and Model-Calculations. *Journal of Biomechanics,* 23**,** 105-119.

Bonet, J. & Wood, R. D. 2008. *Nonlinear continuum mechanics for finite element analysis,* Cambridge, Cambridge University Press.

Bottlang, M., Madey, S. M., Steyers, C. M., Marsh, J. L. & Brown, T. D. 2000. Assessment of elbow joint kinematics in passive motion by electromagnetic motion tracking. *Journal of Orthopaedic Research,* 18**,** 195-202.

Chace, M. A. 1970. DAMN - a prototype program for the Dynamic Analysis of Mechanical Networks. *Proceedings of the 7th Design Automation Workshop.* San Francisco, California, United States: ACM.

Chace, M. A. & Angell, J. C. 1977. Interactive Simulation of Machinery with Friction and Impact Using Dram. *SAE Paper No. 770050.*

Chace, M. A. & Korybalski, K. E. 1970. Computer graphics in the schematic representation of nonlinear, constrained, multi-freedom mechanical systems. *In Proceedings of Computer Graphics 70 Conference.* Brunel University.

Chadwick, J. E., Haumann, D. R. & Parent, R. E. 1989. Layered construction for deformable animated characters. *Proceedings of the 16th annual conference on Computer graphics and interactive techniques.* ACM.

Chang, L. Y. & Pollard, N. S. 2007. Constrained least-squares optimization for robust estimation of center of rotation. *J Biomech,* 40**,** 1392-400.

Chao, E., Armiger, R., Yoshida, H., Lim, J. & Haraguchi, N. 2007. Virtual interactive musculoskeletal system (VIMS) in orthopaedic research, education and clinical patient care. *Journal of Orthopaedic Surgery and Research,* 2.

Chao, E. Y. 2003. Graphic-based musculoskeletal model for biomechanical analyses and animation. *Med Eng Phys,* 25**,** 201-12.

Cohen, J. D., Lin, M. C., Manocha, D. & Ponamgi, M. 1995. I-COLLIDE: an interactive and exact collision detection system for large-scale environments. *Proceedings of the 1995 symposium on Interactive 3D graphics.* Monterey, California, United States: ACM.

Cooper, J. E., Shwedyk, E., Quanbury, A. O., Miller, J. & Hildebrand, D. 1993. Elbow joint restriction: effect on functional upper limb motion during performance of three feeding activities. *Arch Phys Med Rehabil,* 74**,** 805-9.

Cutti, A. G., Giovanardi, A., Rocchi, L., Davalli, A. & Sacchetti, R. 2008. Ambulatory measurement of shoulder and elbow kinematics through inertial and magnetic sensors. *Medical & biological engineering & computing,* 46**,** 169-78.

Damsgaard, M., Rasmussen, J., Christensen, S. T., Surma, E. & De Zee, M. Year. AnyBody - a System for Biomechanical Analysis and Ergonomic Design. *In:* 2nd Max Planck Workshop on Engineering Design Optimization, 2001 Nyborg, Denmark.

Damsgaard, M., Rasmussen, J., Christensen, S. T., Surma, E. & De Zee, M. 2006. Analysis of musculoskeletal systems in the AnyBody Modeling System. *Simulation Modelling Practice and Theory,* 14**,** 1100-1111.

Dariush, B. 2003. Human motion analysis for biomechanics and biomedicine. *Mach. Vision Appl.,* 14**,** 202-205.

Davis, C. S. L., Mo, Us), , Hegde, M. S. L., Mo, Us), , Schmid, O. a. M., Wv, Us), , Maher, M. S. L., Mo, Us), & Bordes, J. P. S. C., Mo, Us). 2005. *System incorporating physics processing unit*. United States patent application 20050086040.

Davoodi, R. & Loeb, G. E. 2002. A software tool for faster development of complex models of musculoskeletal systems and sensorimotor controllers in Simulink (TM). *Journal of Applied Biomechanics,* 18**,** 357-365.

Delp, S. L. & Loan, J. P. 1995. A graphics-based software system to develop and analyze models of musculoskeletal structures. *Computers in Biology and Medicine,* 25**,** 21-34.

Delp, S. L. & Loan, J. P. 2000. A computational framework for simulating and analyzing human and animal movement. *Computing in Science & Engineering,* 2**,** 46-55.

Delp, S. L., Loan, J. P., Hoy, M. G., Zajac, F. E., Topp, E. L. & Rosen, J. M. 1990. An interactive graphics-based model of the lower extremity to study orthopaedic surgical procedures. *Biomedical Engineering, IEEE Transactions on,* 37**,** 757-767.

Denavit, J. & Hartenberg, R. S. 1955. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. of the ASME. Journal of Applied Mechanics,* 22**,** 215-221.

Derek, W. 2006. Exclusive: ASUS Debuts AGEIA PhysX Hardware. Available: http://www.anandtech.com/show/2001/4 [Accessed 03/11/2010].

Deussen, O., Kobbelt, L. & Tücke, P. Year. Using Simulated Annealing to Obtain Good Nodal Approximations of Deformable Bodies. *In:* In Sixth Eurographics Workshop on Simulation and Animation, 1995. Springer, 30-43.

Devmaster. 2010. *3D Engines Database* [Online]. Available: http://www.devmaster.net/engines/list.php [Accessed 29/11/2000].

Dobkin, D. P. & Kirkpatrick, D. G. 1990. Determining the separation of preprocessed polyhedra: a unified approach. *Proceedings of the seventeenth international colloquium on Automata, languages and programming.* Warwick University, England: Springer-Verlag New York, Inc.

Eberly, D. 2002. Dynamic Collision Detection using Oriented Bounding Boxes.

Ehrig, R. M., Taylor, W. R., Duda, G. N. & Heller, M. O. 2006. A survey of formal methods for determining the centre of rotation of ball joints. *J Biomech,* 39**,** 2798-809.

Engin, A. E. & Tumer, S. T. 1989. Three-Dimensional Kinematic Modelling of the Human Shoulder Complex---Part I: Physical Model and Determination of Joint Sinus Cones. *Journal of Biomechanical Engineering,* 111**,** 107-112.

Esat, Ii & Ozada, N. 2010. Articular human joint modelling. *Robotica,* 28**,** 321-339.

Eygendaal, D., Valstar, E. R., Söjbjerg, J. O. & Rozing, P. M. 2002. Biomechanical Evaluation of the Elbow Using Roentgen Stereophotogrammetric Analysis. *Clinical Orthopaedics and Related Research,* 396**,** 100-105.

Fabio Ganovelli , J. D. 2000. Improving collision detection between deformable objects. *In Proceedings of SCCG2000.*

Febio, F. E. F. B. Musculoskeletal Research Laboratories (MRL).

Flores, P. 2008. *Kinematics and dynamics of multibody systems with imperfect joints : models and case studies,* Berlin, Springer.

Floris, S., Olsen, B. S., Dalstra, M., Sojbjerg, J. O. & Sneppen, O. 1998. The medial collateral ligament of the elbow joint: Anatomy and kinematics. *Journal of Shoulder and Elbow Surgery,* 7**,** 345-351.

Gamage, S. S. & Lasenby, J. 2002. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *J Biomech,* 35**,** 87-93.

Garcia-Alonso, A., Nicol, Serrano, S. & Flaquer, J. 1994. Solving the Collision Detection Problem. *IEEE Comput. Graph. Appl.,* 14**,** 36-43.

Georgii, J. & Westermann, R. 2005. Mass-spring systems on the GPU. *Programmable Graphics Hardware,* 13**,** 693-702.

Goldsmith, J. S., J.;. Year. Automatic creation of object hierarchies for ray tracing. *In:* IEEE Computer Graphics and Applications 1987

Goldstine, H. H. 2001. *The Computer from Pascal to von Neumann,* Princeton University Press.

Gottschalk, S. 1996. Separating axis theorem: Technical Report TR96-024.

Gottschalk, S., Lin, M. C. & Manocha, D. 1996. OBBTree: a hierarchical structure for rapid interference detection. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques.* ACM.

Govindaraju, N. K., Kabul, I., Lin, M. C. & Manocha, D. 2007. Fast continuous collision detection among deformable models using graphics processors. *Computers & Graphics-Uk,* 31**,** 5-14.

Govindaraju, N. K., Lin, M. C. & Manocha, D. 2005. Quick-CULLIDE: Fast inter- and intra-object collision culling using graphics hardware. *Ieee Virtual Reality 2005, Conference Proceedings***,** 59.

Govindaraju, N. K., Lin, M. C. & Manocha, D. 2006. Efficient collision culling among deformable objects using graphics processors. *Presence-Teleoperators and Virtual Environments,* 15**,** 62-76.

Govindaraju, N. K., Redon, S., Lin, M. C. & Manocha, D. 2003. CULLIDE: interactive collision detection between complex models in large environments using graphics hardware. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware.* San Diego, California: Eurographics Association.

Hamada, K. H., Y. Year. Octree-based approach to real-time collision-free path planning for robot manipulator. *In:* AMC, 1996. 705 - 710

Hatzel, B., Horodyski, M., Kaminski, T. W., Meister, K., Powers, M. & Brunt, D. 2006. Measurement of glenohumeral joint laxity using the KT-2000 knee ligament arthrometer: Reliability analysis. *Physical Therapy in Sport*, 7**,** 137-143.

Haug, E. J. 1992. *Intermediate Dynamics*, Prentice Hall.

He, J., Li, D., Lu, B., Wang, Z. & Tao, Z. 2006. Custom fabrication of composite tibial hemi-knee joint combining CAD/CAE/CAM techniques. *Proc Inst Mech Eng H*, 220**,** 823-30.

Heidelberger, B., Teschner, M. & Gross, M. 2003. Real-time volumetric intersections of deforming object. *Proceedings of Vision, Modeling, and Visualization 2003.*

Heidelberger, B., Teschner, M. & Gross, M. 2004. Detection of Collisions and Self-collisions Using Image-space Techniques. *Journal of WSCG***,** 145--152.

Hodgins, J. K., Wooten, W. L., Brogan, D. C. & O'brien, J. F. 1995. Animating human athletics. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques.* ACM.

Hudson, T. C., Lin, M. C., Cohen, J., Gottschalk, S. & Manocha, D. 1997. V-COLLIDE: accelerated collision detection for VRML. *Proceedings of the second symposium on Virtual reality modeling language.* Monterey, California, United States: ACM.

Hutmacher, D. W. 2000. Scaffolds in tissue engineering bone and cartilage. *Biomaterials*, 21**,** 2529-43.

Intersense Inc. *InterSense Inc.* [Online]. USA. Available: http://www.intersense.com/ [Accessed 05/11/2010].

Jacob, H. A., Kunz, C. & Sennwald, G. 1992. [Biomechanics of the carpus--functional anatomy and movement analysis of the carpal bones]. *Orthopade*, 21**,** 81-7.

Jazar, R. N. 2010. *Theory of applied robotics : kinematics, dynamics, and control*, New York ; London, Springer.

Jensen, S. L., Olsen, B. S., Tyrdal, S., Sojbjerg, J. O. & Sneppen, O. 2005. Elbow joint laxity after experimental radial head excision and lateral collateral ligament rupture: Efficacy of prosthetic replacement and ligament repair. *Journal of Shoulder and Elbow Surgery*, 14**,** 78-84.

Joukhadar, A., Wabbi, A. & Laugier., C. 1996. Fast Contact Localization Between Deformable Polyhedra in Motion. *Proceedings of the Computer Animation.* IEEE Computer Society.

Katsuaki, K., Hiromasa, S. & Fumihiko, K. 1997. Simulation of Rigid Body Motion with Impulsive Friction Force. *Proceedings of IEEE International Symposium on Assembly and Task Planning*

Kenneth E. Hoff, I., Keyser, J., Lin, M., Manocha, D. & Culver, T. 1999. Fast computation of generalized Voronoi diagrams using graphics hardware. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co.

Kenneth E. Hoff, I., Zaferakis, A., Lin, M. & Manocha, D. 2001. Fast and simple 2D geometric proximity queries using graphics hardware. *Proceedings of the 2001 symposium on Interactive 3D graphics.* ACM.

King, G. J. W., Itoi, E., Risung, F., Niebur, G. L., Morrey, B. F. & An, K. N. 1993. Kinematics and Stability of the Norway Elbow - a Cadaveric Study. *Acta Orthopaedica Scandinavica,* 64**,** 657-663.

Kinzel, G. L., Hillberry, B. M., Hall Jr, A. S., Van Sickle, D. C. & Harvey, W. M. 1972. Measurement of the total motion between two body segments -- II Description of application. *Journal of Biomechanics,* 5**,** 283-284, IN9-IN11, 285-293.

Klosowski, J. T., Held, M., Mitchell, J. S. B., Sowizral, H. & Zikan, K. 1998. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics,* 4**,** 21-36.

Larsson, T. & Möller, A. T. Year. Collision detection for continuously deforming bodies. *In:* Eurographics, 2001. 325-333.

Lauterbur, P. C. & Mansfield, P. 2003. *The Nobel Prize in Physiology or Medicine* [Online]. The Nobel Prize. Available: nobelprize.org/nobel_prizes/medicine/laureates/2003/press.html [Accessed 25/02/2011].

Lee, J. W. 2001. *Procedural methods using physically based modeling.* The Gorge Washington University.

Lifemodeler, I. 2010. *LifeMOD™* [Online]. Available: http://www.lifemodeler.com [Accessed 15/12/2010].

Lin, M. C. C., J.F.;. Year. A fast algorithm for incremental distance calculation. *In:* IEEE International Conference on Robotics and Automation 1991.

Louchet, J., Provot, X. & Crochemore, D. 1995. Evolutionary identification of cloth animation models. *In:* DIMITRI, T. & DANIEL, T. (eds.) *Computer Animation and Simulation '95.* Maastricht, Netherlands: Springer-Verlag.

Magermans, D. J., Chadwick, E. K., Veeger, H. E. & Van Der Helm, F. C. 2005. Requirements for upper extremity motions during activities of daily living. *Clin Biomech (Bristol, Avon),* 20**,** 591-9.

Marks, S., Windsor, J., W\, B., \#252 & Nsche 2007. Evaluation of game engines for simulated surgical training. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia.* Perth, Australia: ACM.

Martin H. Weik. 1961. The ENIAC Story. Available: http://ftp.arl.mil/~mike/comphist/eniac-story.html [Accessed 05/11/2010].

Materialise. 2010. *Mimics* [Online]. Available: http://www.materialise.com/ [Accessed 23/02/2011].

Mathworks, T. *MATLAB* [Online]. The Mathworks, Inc. Available: http://www.mathworks.com/products/matlab/ [Accessed 20/02/2011].

Maurel, W. 1998. *3D modeling of the human upper limb including the biomechanics of joints, muscles and soft tissues.*

Maurel, W. & Thalmann, D. 2000. Human shoulder modeling including scapulo-thoracic constraint and joint sinus cones. *Computers & Graphics,* 24**,** 203-218.

Mcgurk, M., Amis, A. A., Potamianos, P. & Goodger, N. M. 1997. Rapid prototyping techniques for anatomical modelling in medicine. *Ann R Coll Surg Engl,* 79**,** 169-74.

Medis, M. I. S. *Medis* [Online]. The Netherlands
Leiden. Available: http://www.medis.nl/Products/index.htm [Accessed 21/03/2011].

Mezger, J., Kimmerle, S. & Etzmub, O. 2003. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG,* 11**,** 322--329.

Microstrain, I. *MicroStrain®* [Online]. USA. Available: http://www.microstrain.com/contact-us.asp [Accessed 21/03/2011].

Mirtich, B. 1996. Fast and accurate computation of polyhedral mass properties. *J. Graph. Tools,* 1**,** 31-50.

Mirtich, B. 1996. *Impulse-based Dynamic Simulation of Rigid Body Systems.* PhD, PhD thesis at University of California,.

Mirtich, B. 1998. V-Clip: fast and robust polyhedral collision detection. *ACM Trans. Graph.,* 17**,** 177-208.

Mirtich, B. & Canny, J. 1995. Impulse-based simulation of rigid bodies. *Proceedings of the 1995 symposium on Interactive 3D graphics.* Monterey, California, United States: ACM.

Möller, T. 1997. A fast triangle-triangle intersection test. *J. Graph. Tools,* 2**,** 25-30.

Mosegaard, J. & Sorensen, T. S. 2005. GPU Accelerated Surgical Simulators for Complex Morphology. *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality.* IEEE Computer Society.

Murray, W. M., Buchanan, T. S. & Delp, S. L. 2002. Scaling of peak moment arms of elbow muscles with upper extremity bone dimensions. *Journal of Biomechanics,* 35**,** 19-26.

Nealen, A., Mueller, M., Keiser, R., Boxerman, E. & Carlson, M. 2006. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum,* 25**,** 809-836.

Nvidia CUDA [online]. Available: http://www.nvidia.co.uk [Accessed 13/06/2010].

Olsen, B. S., Sojbjerg, J. O., Dalstra, M. & Sneppen, O. 1996. Kinematics of the lateral ligamentous constraints of the elbow joint. *J Shoulder Elbow Surg,* 5**,** 333-41.

Orlandea, N., Chace, M. A. And Calahan, D. A. Year. A sparsity-orientated approach to the dynamic analysis and design of mechanical systems. *In:* In Proceedings of ASME Mechanisms Conference, 1976 Montreal. Trans. ASME, 773-784.

Palmer, I. J. & Grimsdale, R. L. 1995. Collision Detection for Animation using Sphere-Trees. *Computer Graphics Forum,* 14**,** 105-116.

Pang, W.-M., Qin, J., Chui, Y.-P. & Heng, P.-A. 2010. Fast prototyping of virtual reality based surgical simulators with PhysX-enabled GPU. *In:* ZHIGENG, P., ADRIAN DAVID, C., WOLFGANG, M., LLER, XIAOPENG, Z. & KEVIN, W. (eds.) *Transactions on edutainment IV.* Springer-Verlag.

Pang, W.-M., Qin, J., Chui, Y.-P., Wong, T.-T., Leung, K.-S. & Heng, P.-A. 2007. Orthopedics surgery trainer with PPU-accelerated blood and tissue simulation. *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention.* Brisbane, Australia: Springer-Verlag.

Petuskey, K., Bagley, A., Abdala, E., James, M. A. & Rab, G. 2007. Upper extremity kinematics during functional activities: three-dimensional studies in a normal pediatric population. *Gait Posture,* 25**,** 573-9.

Pomianowski, S., O'driscoll, S. W., Neale, P. G., Park, M. J., Morrey, B. F. & An, K. N. 2001. The effect of forearm rotation on laxity and stability of the elbow. *Clinical Biomechanics,* 16**,** 401-407.

Qualisys. 2010. *Motion Capturing Systems* [Online]. Sweden. Available: http://www.qualisys.com/ [Accessed 21/03/2011].

Rab, G., Petuskey, K. & Bagley, A. 2002. A method for determination of upper extremity kinematics. *Gait & Posture,* 15**,** 113-119.

Redon, S., Kheddar, A. & Coquillart, S. 2002. Fast continuous collision detection between rigid bodies. *Computer Graphics Forum,* 21**,** 279.

Roehl, B. 1998. Specifcation for a standard vrml humanoid.

S. Singare, W. P., X. Guanghui 2010. The Application of Rapid Prototyping and Manufacturing for Anatomical Modelling in Medicine. *Journal of Biomimetics, Biomaterials, and Tissue Engineering,* 6**,** 57-65.

Saladin, K. S. 2010. *Anatomy & physiology : the unity of form and function,* Maidenhead, McGraw-Hill Higher Education.

Samet, H. & Webber, R. E. 1988. Hierarchical Data Structures and Algorithms for Computer Graphics. Part I. *IEEE Comput. Graph. Appl.,* 8**,** 48-68.

Sauers, E. L., Borsa, P. A., Herling, D. E. & Stanley, R. D. 2001. Instrumented measurement of glenohumeral joint laxity reliability and normative data. *Knee Surgery Sports Traumatology Arthroscopy,* 9**,** 34-41.

Shinya, M. & Forgue, M. C. 1991. Interference detection through rasterization. *The Journal of Visualization and Computer Animation,* 2**,** 132-134.

Sirkett, D., Mullineux, G., Giddins, G. & Miles, A. 2004. A kinematic model of the wrist based on maximization of joint contact area. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine,* 218**,** 349-359.

Stavlas, P., Jensen, S. L. & Sojbjerg, J. O. 2007. Kinematics of the ligamentous unstable elbow joint after application of a hinged external fixation device: A cadaveric study. *Journal of Shoulder and Elbow Surgery,* 16**,** 491-496.

Sun, W., Starly, B., Darling, A. & Gomez, C. 2004. Computer-aided tissue engineering: application to biomimetic modelling and design of tissue scaffolds. *Biotechnol Appl Biochem,* 39**,** 49-58.

Tanaka, S., An, K.-N. & Morrey, B. F. 1998. Kinematics and Laxity of Ulnohumeral Joint Under Valgus-Varus Stress. *Journal of Musculoskeletal Research (JMR),* 2**,** 45-54.

Tao, Y., Hu, H. & Zhou, H. 2007. Integration of Vision and Inertial Sensors for 3D Arm Motion Tracking in Home-based Rehabilitation. *Int. J. Rob. Res.,* 26**,** 607-624.

Taylor, Z. A., Cheng, M. & Ourselin, S. 2008. High-speed nonlinear finite element analysis for surgical simulation using graphics processing units. *IEEE transactions on medical imaging,* 27**,** 650-63.

Terzopoulos, D., Platt, J., Barr, A. & Fleischer, K. 1987. Elastically deformable models. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques.* ACM.

Terzopoulos, D. & Witkin, A. 1988. Physically Based Models with Rigid and Deformable Components. *Ieee Computer Graphics and Applications,* 8**,** 41-51.

Tortora, G. J. & Grabowski, S. R. 2007. *Introduction to the human body : the essentials of anatomy and physiology,* New York, N.Y. ; [Chichester], Wiley.

Xsens. The Netherlands Available: http://www.xsens.com/ [Accessed 21/03/2011].

Zatsiorsky, V. M. 1998. *Kinematics of human motion,* Champaign, Illinois: Human Kinetics Publishers.

APPENDICES

Appendix A1

A1.1: Scripting code of the physics based modelling using DX Studio game engine
Flexion script

```
var flex
function onUpdate()
{
var y1 =scenes.scene_1.objects.Marker_angle1.posWorld.y
var x1 =scenes.scene_1.objects.Marker_angle1.posWorld.z

var y2 =scenes.scene_1.objects.Marker_angle2.posWorld.y
var x2 =scenes.scene_1.objects.Marker_angle2.posWorld.z

var y3 =y1
var x3 =x2

var power1= Math.pow((x1-x2),2) + Math.pow((y1-y2), 2)
var a1 = Math.sqrt(power1)

var power2= Math.pow((x1-x3),2) + Math.pow((y1-y3), 2)
var b1 = Math.sqrt(power2)

var power3= Math.pow((x2-x3),2) + Math.pow((y2-y3), 2)
var c1 = Math.sqrt(power3)

var Angle = Math.acos(b1/a1)*180/3.1416
// for first triangle;

if (y2<y1)
object.text = 90+Angle*-1

else
object.text = 90+Angle
flex = object.text
}
```

Laxity script

```
var laxity
function onUpdate()
{
var z1  =  scenes. scene_1. objects. Marker_angle_valgus1. posWorld. z
var x1  = scenes. scene_1. objects. Marker_angle_valgus1. posWorld. x
var z2  = scenes. scene_1. objects. Marker_angle_valgus2. posWorld. z
var x2  = scenes. scene_1. objects. Marker_angle_valgus2. posWorld. x

var z3  = z2
var x3  = x1
var power1 =  Math. pow((x1 − x2),2)  +  Math. pow((z1 − z2), 2)
var a1  =  Math. sqrt(power1)

var power2 =  Math. pow((x1 − x3),2)  +  Math. pow((z1 − z3), 2)
var b1  =  Math. sqrt(power2)
var power3 = Math.pow((x2 − x3),2) + Math.pow((z2 − z3), 2)
var c1 = Math.sqrt(power3)


var Angle = Math.acos(c1/b1) ∗ 180/3.1416

// for first triangle;
object.text = 90 − Angle
laxity = object.text }
```

Real time data plotting script

```
var flexArray = new Array();
var LaxityArray = new Array();
var flexlaxArray = new Array();
var width = 1;//Width of lines
var vScale = 1;// Vertical scale factor for lines
var spacing = 0;// Vertical distance between lines

function onInit()
{
// Set up lineArray as a 2D array. Is there a better
flexArray = new Array;
LaxityArray = new Array;
flexlaxArray = new Array;

for (var i = 0; i < 100; i + +)
flexlaxArray[i] = new Array();

for (line = 0; line < 100; line + +) //Fill array with zeroes

{for (point = 1; point < 2; point + +) //You want the lines to start flatlined
flexlaxArray[line][point] = 0;
} //Fill array with zeroes
for (point = 1; point < 101; point + +)//You want the lines to start flatlined
{
LaxityArray [point] = 0;
flexArray [point] = 0;
}
//then come to life as data hits them.
object.setTimer("lineUpdate", 0.2);//Set initial timer − update every 0.1 seconds.
}
function onTimer(lineUpdate)
{

object.shapeRemoveAll();// Need to remove all existing lines first.

var y = .5;// height of lowest line

// Cycle through each line
{

for (var point = 1; point < 101; point + +)// Cycle through each point
{
var x = point ∗ (width / 200);
object. shapeAddLine(new Vector(x, flexArray[point]  +  y, 0), new Vector(x
                        +  (width / 200), flexArray[point + 1]  +  y, 0), new Color(1,1,0));

flexArray[point − 1]  =  flexArray[point];

object. shapeAddLine(new Vector(x, LaxityArray[point]  +  y, 0), new Vector(x
                        +  (width / 200), LaxityArray[point + 1]  +  y, 0), new Color(1,0,0));

LaxityArray[point − 1]  =  LaxityArray [point];

flexlaxArray[point − 1][0] = LaxityArray[point − 1];

flexlaxArray[point − 1][1] = flexArray[point − 1];

}

flexArray[100]  =  (objects. flexion. script. flex) ∗ vScale;

LaxityArray[100]  =  (objects. laxity. script. laxity) ∗ vScale;//Or display random values
y +=  spacing;

}
object. setTimer("lineUpdate", .2);// Call this function again in 0.1 seconds

}
```

Position recording script

```
function onClick()
{
var saveposition  =  system. file. openWrite("C:\\Users\\pc\\Desktop\\flexion. txt");

saveposition. writeStringArray(objects. effectplane_2. script. flexArray);

var saveforce  =  system. file. openWrite("C:\\Users\\pc\\Desktop\\laxity. txt");

saveforce. writeStringArray(objects. effectplane_2. script. LaxityArray);

var flexlax  =  system. file. openWrite("C:\\Users\\pc\\Desktop\\flexlaxity. txt");

flexlax. writeStringArray(objects. effectplane_2. script. flexlaxArray);

}
```

Muscle activate and release script (biceps)

```
function onClick()

{
scenes. scene_1. objects. FixedUp_15. physics. enable = false
scenes. scene_1. objects. FixedUp_16. physics. enable = false

}
```

```
function onClick()
{
scenes.scene_1.objects.FixedUp_15.physics.enable = true
scenes.scene_1.objects.FixedUp_16.physics.enable = true

}
```

Instant centre of rotation script

```
var arrayY = new Array();
var arrayX = new Array();
var arrayZ = new Array();
var Allarray  =  new Array();
var Farray  =  new Array();
var M11  =  new Array();
var M12  =  new Array();
var M13  =  new Array();
var M14  =  new Array();
var M15  =  new Array();

var point
var ma;  var ma2;
var mb;  var mb2;
var mc;
var X1;  var X2;  var X3;  var X4;
var Y1;  var Y2;  var Y3; var Y4;
var Z1;  var Z2;  var Z3; var Z4;
var Xc;
var Yc;
var Zc;
var arrayXc  =  new Array();
var arrayYc  =  new Array();
var arrayZc  =  new Array();

var arraycor  =  new Array();
var angel  =  new Array();
var XYZ1;
var XYZ2;
var XYZ3;
var XYZ4;

function onInit()
{
{
for (var i  =  0; i  <  100; i++)
Allarray[i]  =  new Array;

for (var i  =  0; i  <  100; i++)
Farray[i]  =  new Array;

for (var j  =  0; j  <  4; j++)
M11[j]  =  new Array;

for (var j  =  0; j  <  4; j++)

M12[j]  =  new Array;
```

```
for (var j = 0; j < 4; j + +)
M13[j] = new Array;

for (var j = 0; j < 4; j + +)
M14[j] = new Array;

for (var j = 0; j < 4; j + +)
M15[j] = new Array;



for (line = 0; line < 100; line + +)//Fill array with zeroes – only needed if
for (point = 1; point < 3; point + +)//You want the lines to start flatlined, and
Allarray[line][point] = 0;

for (line = 0; line < 100; line + +)//Fill array with zeroes – only needed if
for (point = 1; point < 4; point + +)
Farray[line][point] = 0


}


object. setTimer("posupdate", .2);

}

function onTimer(posupdate)

{
{
for (var point = 1; point < 101; point + +)// Cycle through each point
{

arrayX[point – 1] = arrayX[point];
arrayY[point – 1] = arrayY[point];
arrayZ[point – 1] = arrayZ[point];
angel[point – 1] = angel[point];

Allarray[point – 1][0] = arrayX[point]
Allarray[point – 1][1] = arrayY[point]
Allarray[point – 1][2] = arrayZ[point]

Farray[point – 1][0] = angel[point – 1];
Farray[point – 1][1] = arrayXc[point – 1];
Farray[point – 1][2] = arrayYc[point – 1];
Farray[point – 1][3] = arrayZc[point – 1];

// Recording the temporary four successive points
X1 = arrayX[point]; X2 = arrayX[point – 2]; X3 = arrayX[point – 4]; X4
                    = arrayX[point – 6];
Y1 = arrayY[point]; Y2 = arrayY[point – 2]; Y3 = arrayY[point – 4]; Y4
                    = arrayY[point – 6];
Z1 = arrayZ[point]; Z2 = arrayZ[point – 2]; Z3 = arrayZ[point – 4]; Z4
                    = arrayZ[point – 6];
XYZ1  =  (X1 * X1) + (Y1 * Y1) + (Z1 * Z1);
XYZ2  =  (X2 * X2) + (Y2 * Y2) + (Z2 * Z2);
XYZ3  =  (X3 * X3) + (Y3 * Y3) + (Z3 * Z3);
XYZ4  =  (X4 * X4) + (Y4 * Y4) + (Z4 * Z4);

// Building temp M11

M11[0][0] = X1; M11[0][1] = Y1; M11[0][2] = Z1; M11[0][3] = 1;
M11[1][0] = X2; M11[1][1] = Y2; M11[1][2] = Z2; M11[1][3] = 1;
M11[2][0] = X3; M11[2][1] = Y3; M11[2][2] = Z3; M11[2][3] = 1;
M11[3][0] = X4; M11[3][1] = Y4; M11[3][2] = Z4; M11[3][3] = 1;

// Building temp M12

M12[0][0] = XYZ1; M12[0][1] = Y1; M12[0][2] = Z1; M12[0][3] = 1;
M12[1][0] = XYZ2; M12[1][1] = Y2; M12[1][2] = Z2; M12[1][3] = 1;
M12[2][0] = XYZ3; M12[2][1] = Y3; M12[2][2] = Z3; M12[2][3] = 1;
M12[3][0] = XYZ4; M12[3][1] = Y4; M12[3][2] = Z4; M12[3][3] = 1;

// Building temp M13

M13[0][0] = XYZ1; M13[0][1] = X1; M13[0][2] = Z1; M13[0][3] = 1;
M13[1][0] = XYZ2; M13[1][1] = X2; M13[1][2] = Z2; M13[1][3] = 1;
M13[2][0] = XYZ3; M13[2][1] = X3; M13[2][2] = Z3; M13[2][3] = 1;
M13[3][0] = XYZ4; M13[3][1] = X4; M13[3][2] = Z4; M13[3][3] = 1;

// Building temp M14

M14[0][0] = XYZ1; M14[0][1] = X1; M14[0][2] = Y1; M14[0][3] = 1;
M14[1][0] = XYZ2; M14[1][1] = X2; M14[1][2] = Y2; M14[1][3] = 1;
M14[2][0] = XYZ3; M14[2][1] = X3; M14[2][2] = Y3; M14[2][3] = 1;
M14[3][0] = XYZ4; M14[3][1] = X4; M14[3][2] = Y4; M14[3][3] = 1;

// Building temp M15

M15[0][0] = XYZ1; M15[0][1] = X1; M15[0][2] = Y1; M15[0][3] = Z1;
M15[1][0] = XYZ2; M15[1][1] = X2; M15[1][2] = Y2; M15[1][3] = Z2;
M15[2][0] = XYZ3; M15[2][1] = X3; M15[2][2] = Y3; M15[2][3] = Z3;
M15[3][0] = XYZ4; M15[3][1] = X4; M15[3][2] = Y4; M15[3][3] = Z4;

// Instant Center of Rotation, (Xc, Yc, Zc) is then found by:

Xc  =  0.5 * M12 * M11;
Yc  =  0.5 * M13/M11;
Zc  =  0.5 * M14/M11;

arrayXc[point] = Xc;
arrayYc[point] = Yc;
arrayZc[point] = Zc;
}
arrayX[100] = scenes.scene_1.objects.FixedDown_16.pos.x
arrayY[100] = scenes.scene_1.objects.FixedDown_16.pos.y
arrayZ[100] = scenes.scene_1.objects.FixedDown_16.pos.z

angel[100] = scenes.scene_2.objects.flexion.script.flex

}
```

```
object. setTimer("posupdate", .2);
}
```

```
var Rxarray  =  new Array();
var Ryarray  =  new Array();
var Rzarray  =  new Array();
var carray  =  new Array();
var arrayY = new Array();
var arrayX = new Array();
var arrayZ = new Array();
var angel  =  new Array();
var MoA  =  new Array();

function onInit()

{

object. setTimer("posupdate", .1);


for (var i  =  0; i  <  100; i + +)// Set up lineArray as a 2D array.
carray[i]  =  new Array;


for (line  =  0; line  <  100; line + +)//Fill array with zeroes
for (point  =  1; point  <  5; point + +)//You want the lines to start flat

carray[line][point]  =  0;

object. setTimer("posupdate", .1);
}

function onTimer(posupdate)
{

for (var point  =  1; point  <  101; point + +)// Cycle through each point
{

arrayX[point − 1] = arrayX[point];
arrayY[point − 1] = arrayY[point];
arrayZ[point − 1] = arrayZ[point];
angel[point − 1] = angel[point];
MoA[point − 1] = MoA[point];

carray[point − 1][0] = angel[point];
carray[point − 1][1] = arrayX[point];
carray[point − 1][2] = arrayY[point];
carray[point − 1][3] = arrayZ[point];
carray[point − 1][4] = MoA[point];

var Vx =  scenes. scene_1. objects. FixedDown_16. posVelocity. x
var Vy =  scenes. scene_1. objects. FixedDown_16. posVelocity. y
var Vz =  scenes. scene_1. objects. FixedDown_16. posVelocity. z
var w  =  scenes. scene_1. objects. FixedDown_16. rotVelocity. w
arrayX[point]  =  Vx/w;
arrayY[point]  =  Vy/w;
arrayZ[point]  =  Vz/w;
angel[point] = scenes. scene_2. objects. flexion. script. flex;

MoA[point] =  scenes. scene_2. objects. moment. script. Fa * 13.5;

object. text =  carray[point]

}

object. setTimer("posupdate", .1);
}
```

Hold position script

```
function onClick()
{
// for muscle segment 1

Xu1  =  scenes. scene_1. objects. FixedUp_15. pos. x
Yu1  =  scenes. scene_1. objects. FixedUp_15. pos. y
Zu1  =  scenes. scene_1. objects. FixedUp_15. pos. z

Xd1  =  scenes. scene_1. objects. FixedDown_15. pos. x
Yd1  =  scenes. scene_1. objects. FixedDown_15. pos. y
Zd1  =  scenes. scene_1. objects. FixedDown_15. pos. z

Fx1  =  (Xu1 − Xd1) * 200
Fy1  =  (Yu1 − Yd1) * 200
Fz1  =  (Zu1 − Zd1) * 200

scenes. scene_1. objects. FixedDown_15. physics. applyCentralForce(Fx1, Fy1, Fz1);

// for muscle segment 2
Xu2  =  scenes. scene_1. objects. FixedUp_16. pos. x
Yu2  =  scenes. scene_1. objects. FixedUp_16. pos. y
Zu2  =  scenes. scene_1. objects. FixedUp_16. pos. z

Xd2  =  scenes. scene_1. objects. FixedDown_16. pos. x
Yd2  =  scenes. scene_1. objects. FixedDown_16. pos. y
Zd2  =  scenes. scene_1. objects. FixedDown_16. pos. z

Fx2  =  (Xu2 − Xd2) * 200
Fy2  =  (Yu2 − Yd2) * 200
Fz2  =  (Zu2 − Zd2) * 200

scenes. scene_1. objects. FixedDown_16. physics. applyCentralForce(Fx2, Fy2, Fz2);


}
```

Moment arm script

```
var Xa;
var Ya;
var Za;
var Xb;
var Yb;
var Zb;
var L;
var Fa

function onUpdate()
{

Xa  =  Math.abs(scenes.scene_1.objects.FixedUp_16.pos.x
                        − scenes.scene_1.objects.FixedDown_16.pos.x)

Ya  =  Math.abs(scenes.scene_1.objects.FixedUp_16.pos.y
                        − scenes.scene_1.objects.FixedDown_16.pos.y)

Za  =  Math.abs(scenes.scene_1.objects.FixedUp_16.pos.z
                        − scenes.scene_1.objects.FixedDown_16.pos.z)

L  =  Math.sqrt(Xa ∗ Xa + Ya ∗ Ya + Za ∗ Za);

Fa  =  L ∗ 200;
}
```

Appendix A2

A2.1:The Visual Basic code for the image based system (Window PC)

```
Imports System.Runtime.InteropServices
Imports System.Windows.Forms

Public Class Form1
  Inherits System.Windows.Forms.Form

  Const WM_CAP As Short = &H400S
  Const WM_CAP_DRIVER_CONNECT As Integer = WM_CAP + 10
  Const WM_CAP_DRIVER_DISCONNECT As Integer = WM_CAP + 11
  Const WM_CAP_EDIT_COPY As Integer = WM_CAP + 30
  Const WM_CAP_SET_PREVIEW As Integer = WM_CAP + 50
  Const WM_CAP_SET_PREVIEWRATE As Integer = WM_CAP + 52
  Const WM_CAP_SET_SCALE As Integer = WM_CAP + 53
  Const WS_CHILD As Integer = &H40000000
  Const WS_VISIBLE As Integer = &H10000000
  Const SWP_NOMOVE As Short = &H2S
  Const SWP_NOSIZE As Short = 1
  Const SWP_NOZORDER As Short = &H4S
  Const HWND_BOTTOM As Short = 1

  Dim iDevice As Integer = 0 ' Current device ID
  Dim hHwnd As Integer ' Handle to preview window


  Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hwnd As Integer, ByVal wMsg As Integer, ByVal wParam As Integer, _
    <MarshalAs(UnmanagedType.AsAny)> ByVal lParam As Object) As Integer


  Declare Function SetWindowPos Lib "user32" Alias "SetWindowPos" (ByVal hwnd As Integer, _
    ByVal hWndInsertAfter As Integer, ByVal x As Integer, ByVal y As Integer, _
    ByVal cx As Integer, ByVal cy As Integer, ByVal wFlags As Integer) As Integer


  Declare Function DestroyWindow Lib "user32" (ByVal hndw As Integer) As Boolean


  Declare Function capCreateCaptureWindowA Lib "avicap32.dll" _
    (ByVal lpszWindowName As String, ByVal dwStyle As Integer, _
    ByVal x As Integer, ByVal y As Integer, ByVal nWidth As Integer, _
    ByVal nHeight As Short, ByVal hWndParent As Integer, _
    ByVal nID As Integer) As Integer


  Declare Function capGetDriverDescriptionA Lib "avicap32.dll" (ByVal wDriver As Short, _
    ByVal lpszName As String, ByVal cbName As Integer, ByVal lpszVer As String, _
    ByVal cbVer As Integer) As Boolean


 Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
    LoadDeviceList()
    If lstDevices.Items.Count > 0 Then
        Play1.Enabled = True
        lstDevices.SelectedIndex = 0
        Play1.Enabled = True
    Else
        lstDevices.Items.Add("No Capture Device")
        Play1.Enabled = False
    End If

    Still1.Enabled = False
    Save1.Enabled = False
    Still2.Enabled = False
    Save2.Enabled = False

    PictureBox3.SizeMode = PictureBoxSizeMode.StretchImage
End Sub

Private Sub LoadDeviceList()
    Dim strName As String = Space(100)
    Dim strVer As String = Space(100)
    Dim bReturn As Boolean
    Dim x As Integer = 0

    ' Load name of all avialable devices into the lstDevices

    Do
        '   Get Driver name and version

        bReturn = capGetDriverDescriptionA(x, strName, 100, strVer, 100)

        ' If there was a device add device name to the list

        If bReturn Then lstDevices.Items.Add(strName.Trim)
        x += 1
    Loop Until bReturn = False
End Sub

Private Sub OpenPreviewWindow()
    Dim iHeight As Integer = PictureBox3.Height
    Dim iWidth As Integer = PictureBox3.Width

    ' Open Preview window in picturebox

    hHwnd = capCreateCaptureWindowA(iDevice, WS_VISIBLE Or WS_CHILD, 0, 0, 640, _
        480, PictureBox3.Handle.ToInt32, 0)

    ' Connect to device


    If SendMessage(hHwnd, WM_CAP_DRIVER_CONNECT, iDevice, 0) Then
        '
        'Set the preview scale
        '
        SendMessage(hHwnd, WM_CAP_SET_SCALE, True, 0)

        '
        'Set the preview rate in milliseconds
        '
        SendMessage(hHwnd, WM_CAP_SET_PREVIEWRATE, 66, 0)
        '
        'Start previewing the image from the camera
        '
        SendMessage(hHwnd, WM_CAP_SET_PREVIEW, True, 0)

        '
        ' Resize window to fit in picturebox
        '
        SetWindowPos(hHwnd, HWND_BOTTOM, 0, 0, PictureBox3.Width, PictureBox3.Height, _
            SWP_NOMOVE Or SWP_NOZORDER)

        Save1.Enabled = False
        Still1.Enabled = True
        Save2.Enabled = False
        Still2.Enabled = True
        Play1.Enabled = False
      Else
        '
        ' Error connecting to device close window
        '
        DestroyWindow(hHwnd)


      End If
    End Sub

    Private Sub play1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Play1.Click
      iDevice = lstDevices.SelectedIndex
      OpenPreviewWindow()
    End Sub

    Private Sub Still1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Still1.Click
      Dim data As IDataObject
      Dim bmap As Image
      Data = Clipboard.GetDataObject()
      SendMessage(hHwnd, WM_CAP_EDIT_COPY, 0, 0)
      ' Get image from clipboard and convert it to a bitmap
      '
      data = Clipboard.GetDataObject()
      If data.GetDataPresent(GetType(System.Drawing.Bitmap)) Then
        bmap = CType(data.GetData(GetType(System.Drawing.Bitmap)), Image)
        PictureBox1.Image = bmap

        Save1.Enabled = True
        Still1.Enabled = True
        Play1.Enabled = False
      Else
      End If

    End Sub

    Private Sub Save1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Save1.Click
      Dim bmap As Image
```

```
        Dim savePath As String

        bmap = PictureBox1.Image
        With SaveFile
            .Title = "Save Image File!"
            .Filter = "Bitmap (bmp)|*.bmp| (Jpeg)|*.jpg"
            .InitialDirectory = "c:\Users\KW\Desktop\Laxity"
            .ShowDialog()
            savePath = .FileName
        End With
        PictureBox1.Image.Save("savePath")
    End Sub

    Private Sub Still2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Still2.Click
        Dim data As IDataObject
        Dim bmap As Image
        data = Clipboard.GetDataObject()
        SendMessage(hHwnd, WM_CAP_EDIT_COPY, 0, 0)
        ' Get image from clipboard and convert it to a bitmap
        '
        data = Clipboard.GetDataObject()
        If data.GetDataPresent(GetType(System.Drawing.Bitmap)) Then
            bmap = CType(data.GetData(GetType(System.Drawing.Bitmap)), Image)
            PictureBox2.Image = bmap

            Save2.Enabled = True
            Still2.Enabled = True
            Play1.Enabled = False
        Else
        End If

    End Sub

    Private Sub Save2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Save2.Click
        Dim bmap As Image

        Dim savePath As String

        bmap = picturebox2.Image
        With SaveFile
            .Title = "Save Image File!"
            .Filter = "Bitmap (bmp)|*.bmp| (Jpeg)|*.jpg"
            .InitialDirectory = "c:\Users\KW\Desktop\Laxity"
            .ShowDialog()
            savePath = .FileName
        End With
        picturebox2.Image.Save("savePath")
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

        Dim open As OpenFileDialog = New OpenFileDialog()
        Dim imagFilter As String = "Image files (*.jpg,*.jpeg,*.bmp,*.gif,*.png)|*.BMP;*.PNG;*.JPG;
*.JPEG;*.GIF"
        Dim Thresh As Integer = TextBox1.Text
```

```
      open.Filter = imagFilter

      If open.ShowDialog() = DialogResult.OK Then

        Dim imag2 As Image = Image.FromFile(open.FileName)
        PictureBox1.Image = imag2.GetThumbnailImage(PictureBox1.Width, PictureBox1.Height,
 Nothing, Nothing)

      End If
      Dim btm As New Bitmap(PictureBox1.Image)
      Dim source_btm1 As New Bitmap(btm)


      PictureBox1.Image = source_btm1

      btm.Dispose()
      PictureBox1.Image = source_btm1
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click

      Dim open As OpenFileDialog = New OpenFileDialog()
      Dim imagFilter As String = "Image files (*.jpg,*.jpeg,*.bmp,*.gif,*.png)|*.BMP;*.PNG;*.JPG;
*.JPEG;*.GIF"

      open.Filter = imagFilter

      If open.ShowDialog() = DialogResult.OK Then

        Dim imag2 As Image = Image.FromFile(open.FileName)
        picturebox2.Image = imag2.GetThumbnailImage(picturebox2.Width, picturebox2.Height,
Nothing, Nothing)

      End If
      Dim btm As New Bitmap(picturebox2.Image)
      Dim source_btm2 As New Bitmap(btm)
      picturebox2.Image = source_btm2

      btm.Dispose()
      picturebox2.Image = source_btm2
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click

      Dim Thresh As Integer = TextBox1.Text
      Dim dest_gray1 As New Bitmap(PictureBox1.Image)
      Dim dest_gray2 As New Bitmap(picturebox2.Image)
      Dim dest_thresh As New Bitmap(dest_gray1)
      Dim dest_thresh2 As New Bitmap(dest_gray2)

      ' Process the image's

      For y As Integer = 0 To dest_gray1.Height - 1
        For x As Integer = 0 To dest_gray1.Width - 1
          ' Get this pixel's color.
        Dim clr1 As Color = dest_gray1.GetPixel(x, y)
        Dim clr2 As Color = dest_thresh.GetPixel(x, y)

        Dim Nr, Ng, Nb, Nrgb As Double

        Nr = clr1.R
        Ng = clr1.G
        Nb = clr1.B
        Nrgb = (Nr + Ng + Nb) / 3

        clr1 = Color.FromArgb(255, Nrgb, Nrgb, Nrgb)

        If Nrgb < Thresh Then
          Nrgb = 0
        Else : Nrgb = 255
        End If

        clr2 = Color.FromArgb(255, Nrgb, Nrgb, Nrgb)

        ' set the new pixel colors

        dest_gray1.SetPixel(x, y, clr1)
        dest_thresh.SetPixel(x, y, clr2)

      Next x
    Next y

    picdest.Image = dest_gray1
    picdestbw.Image = dest_thresh

    For y As Integer = 0 To dest_gray2.Height - 1
      For x As Integer = 0 To dest_gray2.Width - 1

        Dim clr1 As Color = dest_gray2.GetPixel(x, y)
        Dim clr2 As Color = dest_thresh2.GetPixel(x, y)

        Dim Nr, Ng, Nb, Nrgb As Double

        Nr = clr1.R
        Ng = clr1.G
        Nb = clr1.B
        Nrgb = (Nr + Ng + Nb) / 3

        clr1 = Color.FromArgb(255, Nrgb, Nrgb, Nrgb)

        If Nrgb < Thresh Then
          Nrgb = 0
        Else : Nrgb = 255
        End If

        clr2 = Color.FromArgb(255, Nrgb, Nrgb, Nrgb)

        ' set the new pixel colors
        dest_gray2.SetPixel(x, y, clr1)
        dest_thresh2.SetPixel(x, y, clr2)

      Next x
      Next y

      picdest2.Image = dest_gray2
      picdestbw2.Image = dest_thresh2

   End Sub

   Private Sub picturebox1_Mouseclick(ByVal sender As Object, ByVal
e As System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseClick
      Dim bmp As New Bitmap(PictureBox1.Image)


      If txtX11.Text = Nothing Then
         txtX11.Text = e.X.ToString
         txtY11.Text = e.Y.ToString
         txtRGB11.Text = bmp.GetPixel(e.X, e.Y).R.ToString & ","
& bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString

      ElseIf txtX12.Text = Nothing Then

         txtX12.Text = e.X.ToString
         txtY12.Text = e.Y.ToString
         txtRGB12.Text = bmp.GetPixel(e.X, e.Y).R.ToString & ","
& bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString



      Else
         txtX13.Text = e.X.ToString
         txtY13.Text = e.Y.ToString
         txtRGB13.Text = bmp.GetPixel(e.X, e.Y).R.ToString & ","
& bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString


      End If


   End Sub

   Private Sub picturebox2_Mouseclick(ByVal sender As Object, ByVal
e As System.Windows.Forms.MouseEventArgs) Handles picturebox2.MouseClick

      Dim bmp As New Bitmap(picturebox2.Image)

      If txtX21.Text = Nothing Then
         txtX21.Text = e.X.ToString
         txtY21.Text = e.Y.ToString
         txtRGB21.Text = bmp.GetPixel(e.X, e.Y).R.ToString &
"," & bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString

      ElseIf txtX22.Text = Nothing Then
         txtX22.Text = e.X.ToString
         txtY22.Text = e.Y.ToString

 txtRGB22.Text = bmp.GetPixel(e.X, e.Y).R.ToString & "," & bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel

      Else
         txtX23.Text = e.X.ToString
```

```
        txtY23.Text = e.Y.ToString
        txtRGB23.Text = bmp.GetPixel(e.X, e.Y).R.ToString & ","
& bmp.GetPixel(e.X, e.Y).G.ToString & "," & bmp.GetPixel(e.X, e.Y).B.ToString

      End If
   End Sub

   Private Sub Draw1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Draw1.Click

      Dim f As Pen = New Pen(Color.Red, 2)

      Dim X11, X12, X13 As Integer
      Dim Y11, Y12, Y13 As Integer
      X11 = txtX11.Text
      Y11 = txtY11.Text
      X12 = txtX12.Text
      Y12 = txtY12.Text
      X13 = txtX13.Text
      Y13 = txtY13.Text
      PictureBox1.CreateGraphics.DrawLine(f, X11, Y11, X12, Y12)
      PictureBox1.CreateGraphics.DrawLine(f, X12, Y12, X13, Y13)
      PictureBox1.CreateGraphics.DrawLine(f, X13, Y13, X11, Y11)
   End Sub



   Private Sub Draw2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Draw2.Click

      Dim f As Pen = New Pen(Color.Red, 2)

      Dim X21, X22, X23 As Integer
      Dim Y21, Y22, Y23 As Integer
      X21 = txtX21.Text
      Y21 = txtY21.Text
      X22 = txtX22.Text
      Y22 = txtY22.Text
      X23 = txtX23.Text
      Y23 = txtY23.Text
      picturebox2.CreateGraphics.DrawLine(f, X21, Y21, X22, Y22)
      picturebox2.CreateGraphics.DrawLine(f, X22, Y22, X23, Y23)
      picturebox2.CreateGraphics.DrawLine(f, X23, Y23, X21, Y21)

   End Sub

   Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click

      ' for the first picture

      Dim X11, X12, X13 As Integer
      Dim Y11, Y12, Y13 As Integer



' for the second picture
```

```
Dim X21, X22, X23 As Integer
Dim Y21, Y22, Y23 As Integer

' Sides and angles of triangle 1 and triangle 2

Dim a1, b1, c1 As Double

Dim a2, c2, b2 As Double

Dim area1, area2 As Double

Dim alpha1, beta1, gama1 As Double

Dim alpha2, beta2, gama2 As Double

X11 = txtX11.Text
X12 = txtX12.Text
X13 = txtX13.Text

Y11 = txtY11.Text
Y12 = txtY12.Text
Y13 = txtY13.Text

X21 = txtX21.Text
X22 = txtX22.Text
X23 = txtX23.Text

Y21 = txtY21.Text
Y22 = txtY22.Text
Y23 = txtY23.Text

' for picture 1

a1 = ((Y11 - Y13) ^ 2 + (X11 - X13) ^ 2) ^ 0.5
b1 = ((Y11 - Y12) ^ 2 + (X11 - X12) ^ 2) ^ 0.5
c1 = ((Y12 - Y13) ^ 2 + (X12 - X13) ^ 2) ^ 0.5


alpha1 = Math.Acos((-a1 ^ 2 + b1 ^ 2 + c1 ^ 2) / (2 * b1 * c1))
beta1 = Math.Acos((a1 ^ 2 - b1 ^ 2 + c1 ^ 2) / (2 * a1 * c1))
gama1 = Math.Acos((a1 ^ 2 + b1 ^ 2 - c1 ^ 2) / (2 * a1 * b1))

' for picture 2

a2 = ((Y21 - Y23) ^ 2 + (X21 - X23) ^ 2) ^ 0.5
b2 = ((Y21 - Y22) ^ 2 + (X21 - X22) ^ 2) ^ 0.5
c2 = ((Y22 - Y23) ^ 2 + (X22 - X23) ^ 2) ^ 0.5

alpha2 = Math.Acos((-a2 ^ 2 + b2 ^ 2 + c2 ^ 2) / (2 * b2 * c2))

beta2 = Math.Acos((a2 ^ 2 - b2 ^ 2 + c2 ^ 2) / (2 * a2 * c2))

gama2 = Math.Acos((a2 ^ 2 + b2 ^ 2 - c2 ^ 2) / (2 * a2 * b2))


area1 = a1 * b1 * Math.Sin(gama1) / 2

area2 = a2 * b2 * Math.Sin(gama2) / 2
```

```
    AreaChange.Text = (area1 / area2) * 100 - 100

    txtalpha1.Text = alpha1 * 180 / 3.1416
    txtalpha2.Text = alpha2 * 180 / 3.1416
    distorsionAlpha.Text = alpha2 * 180 / 3.1416 - alpha1 * 180 / 3.1416

    txtBeta1.Text = beta1 * 180 / 3.1416
    txtBeta2.Text = beta2 * 180 / 3.1416
    DistorsionBeta.Text = beta2 * 180 / 3.1416 - beta1 * 180 / 3.1416

    txtGama1.Text = gama1 * 180 / 3.1416
    txtGama2.Text = gama2 * 180 / 3.1416
    DistorsionGama.Text = gama2 * 180 / 3.1416 - gama1 * 180 / 3.1416

  End Sub

  Private Sub Clear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Clear.Click

    ' Clear the fields on the form
    AreaChange.ResetText()
    txtX11.ResetText()
    txtY11.ResetText()
    txtX12.ResetText()
    txtY12.ResetText()
    txtX13.ResetText()
    txtY13.ResetText()
    txtX21.ResetText()
    txtY21.ResetText()
    txtX22.ResetText()
    txtY22.ResetText()
    txtY23.ResetText()
    txtX23.ResetText()
    txtRGB11.ResetText()
    txtRGB12.ResetText()
    txtRGB13.ResetText()
    txtRGB21.ResetText()
    txtRGB22.ResetText()
    txtRGB23.ResetText()
    txtalpha1.ResetText()
    txtalpha2.ResetText()
    txtBeta1.ResetText()
    txtBeta2.ResetText()
    txtGama1.ResetText()
    txtGama2.ResetText()
    distorsionAlpha.ResetText()
    DistorsionBeta.ResetText()
    DistorsionGama.ResetText()

  End Sub

End Class
```

A2.1: The Visual Basic code for the Windows Mobile version of the image based analysis

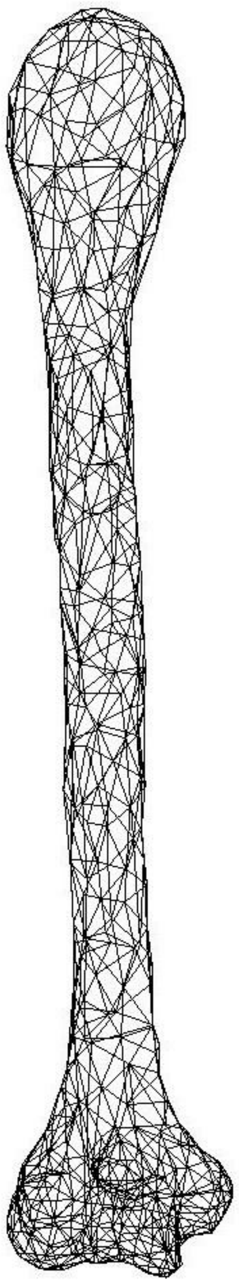
```
Public Class Form1

  Inherits System.Windows.Forms.Form

  Const WM_CAP As Short = &H400S
  Const WM_CAP_DRIVER_CONNECT As Integer = WM_CAP + 10
  Const WM_CAP_DRIVER_DISCONNECT As Integer = WM_CAP + 11
  Const WM_CAP_EDIT_COPY As Integer = WM_CAP + 30
  Const WM_CAP_SET_PREVIEW As Integer = WM_CAP + 50
  Const WM_CAP_SET_PREVIEWRATE As Integer = WM_CAP + 52
  Const WM_CAP_SET_SCALE As Integer = WM_CAP + 53
  Const WS_CHILD As Integer = &H40000000
  Const WS_VISIBLE As Integer = &H10000000
  Const SWP_NOMOVE As Short = &H2S
  Const SWP_NOSIZE As Short = 1
  Const SWP_NOZORDER As Short = &H4S
  Const HWND_BOTTOM As Short = 1

  Dim iDevice As Integer = 0 ' Current device ID

  Dim hHwnd As Integer ' Handle to preview window

  Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
     (ByVal hwnd As Integer, ByVal wMsg As Integer, ByVal wParam As Integer,
<Runtime.InteropServices.MarshalAs(Runtime.InteropServices.UnmanagedType.AsAny)>
ByVal lParam As Object) As Integer

  Declare Function SetWindowPos Lib "user32" Alias "SetWindowPos" (ByVal hwnd As Integer, _
     ByVal hWndInsertAfter As Integer, ByVal x As Integer, ByVal y As Integer, _
     ByVal cx As Integer, ByVal cy As Integer, ByVal wFlags As Integer) As Integer

  Declare Function DestroyWindow Lib "user32" (ByVal hndw As Integer) As Boolean

  Declare Function capCreateCaptureWindowA Lib "avicap32.dll" _
     (ByVal lpszWindowName As String, ByVal dwStyle As Integer, _
     ByVal x As Integer, ByVal y As Integer, ByVal nWidth As Integer, _
     ByVal nHeight As Short, ByVal hWndParent As Integer, _
     ByVal nID As Integer) As Integer

  Declare Function capGetDriverDescriptionA Lib "avicap32.dll" (ByVal wDriver As Short, _
     ByVal lpszName As String, ByVal cbName As Integer, ByVal lpszVer As String, _
     ByVal cbVer As Integer) As Boolean


  Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

     If lstDevices.Items.Count > 0 Then
        Play1.Enabled = True
        lstDevices.SelectedIndex = 0
        Play1.Enabled = True
```

```
    Else
      lstDevices.Items.Add("No Capture Device")
      Play1.Enabled = False
    End If

    Still1.Enabled = False
    Save1.Enabled = False
    Still2.Enabled = False
    Save2.Enabled = False

    PictureBox3.SizeMode = PictureBoxSizeMode.StretchImage
  End Sub

  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

    With OpenFileDialog1

      .Filter = "All Files|*.*|Bitmap Files (*)|*.bmp;*.gif;*.jpg"
      .FilterIndex = 2
      If .ShowDialog = DialogResult.OK Then
    ' Load the specified file into a PictureBox control.
        PictureBox1.Image = New Bitmap(OpenFileDialog1.FileName)

      End If
    End With
  End Sub

  Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click

    With OpenFileDialog1

      .Filter = "All Files|*.*|Bitmap Files (*)|*.bmp;*.gif;*.jpg"
      .FilterIndex = 2
      If .ShowDialog = DialogResult.OK Then
' Load the specified file into a PictureBox control.
        PictureBox2.Image = New Bitmap(OpenFileDialog1.FileName)
      End If

    End With

  End Sub

  Private Sub picturebox1_mouseclick(ByVal sender As System.Object, ByVal
e As System.Windows.Forms.MouseEventArgs) Handles Picturebox1.MouseDown

    Dim bmp1 As Bitmap = Picturebox1.Image

    If TextBox1.Text = Nothing Then
      TextBox1.Text = e.X.ToString
      TextBox2.Text = e.Y.ToString
      TextBox3.Text = bmp1.GetPixel(e.X, e.Y).R.ToString &
"," & bmp1.GetPixel(e.X, e.Y).G.ToString & "," & bmp1.GetPixel(e.X, e.Y).B.ToString

    ElseIf TextBox4.Text = Nothing Then
      TextBox4.Text = e.X.ToString
      TextBox5.Text = e.Y.ToString
  TextBox6.Text = bmp1.GetPixel(e.X, e.Y).R.ToString &
"," & bmp1.GetPixel(e.X, e.Y).G.ToString & "," & bmp1.GetPixel(e.X, e.Y).B.ToString

    Else
      TextBox7.Text = e.X.ToString
      TextBox8.Text = e.Y.ToString
      TextBox9.Text = bmp1.GetPixel(e.X, e.Y).R.ToString &
"," & bmp1.GetPixel(e.X, e.Y).G.ToString & "," & bmp1.GetPixel(e.X, e.Y).B.ToString

    End If
  End Sub
  Private Sub picturebox2_mouseclick(ByVal sender As System.Object, ByVal
 e As System.Windows.Forms.MouseEventArgs) Handles PictureBox2.MouseDown

    Dim bmp2 As Bitmap = Picturebox1.Image

    If TextBox10.Text = Nothing Then
      TextBox10.Text = e.X.ToString
      TextBox11.Text = e.Y.ToString
      TextBox12.Text = bmp2.GetPixel(e.X, e.Y).R.ToString &
"," & bmp2.GetPixel(e.X, e.Y).G.ToString & "," & bmp2.GetPixel(e.X, e.Y).B.ToString

    ElseIf TextBox13.Text = Nothing Then
      TextBox13.Text = e.Y.ToString
      TextBox14.Text = e.X.ToString
      TextBox15.Text = bmp2.GetPixel(e.X, e.Y).R.ToString &
"," & bmp2.GetPixel(e.X, e.Y).G.ToString & "," & bmp2.GetPixel(e.X, e.Y).B.ToString

    Else
      TextBox16.Text = e.X.ToString
      TextBox17.Text = e.Y.ToString
      TextBox18.Text = bmp2.GetPixel(e.X, e.Y).R.ToString &
"," & bmp2.GetPixel(e.X, e.Y).G.ToString & "," & bmp2.GetPixel(e.X, e.Y).B.ToString

    End If
  End Sub
  Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click

    ' for the first picture
    Dim X11 As Integer
    Dim X12 As Integer
    Dim X13 As Integer
    Dim Y11 As Integer
    Dim Y12 As Integer
    Dim Y13 As Integer

    ' for the second picture

    Dim X21 As Integer
    Dim X22 As Integer
    Dim X23 As Integer
    Dim Y21 As Integer
    Dim Y22 As Integer
    Dim Y23 As Integer
    ' To calculate the area of the triangle 1
    Dim L112 As Integer
    Dim L113 As Integer
    Dim L123 As Integer

    Dim L212 As Integer
    Dim L213 As Integer
    Dim L223 As Integer

    Dim alpha1 As Double
    Dim alpha2 As Double

    Dim S1 As Double
    Dim S2 As Double

    Dim A1 As Double
    Dim A2 As Double

    Y11 = TextBox2.Text
    Y12 = TextBox5.Text
    Y13 = TextBox8.Text
    X11 = TextBox1.Text
    X12 = TextBox4.Text
    X13 = TextBox7.Text

    Y21 = TextBox11.Text
    Y22 = TextBox14.Text
    Y23 = TextBox17.Text
    X21 = TextBox10.Text
    X22 = TextBox13.Text
    X23 = TextBox16.Text

    ' for picture 1
    L113 = ((Y11 - Y13) ^ 2 + (X11 - X13) ^ 2) ^ 0.5
    L112 = ((Y11 - Y12) ^ 2 + (X11 - X12) ^ 2) ^ 0.5
    L123 = ((Y12 - Y13) ^ 2 + (X12 - X13) ^ 2) ^ 0.5

    S1 = 1 / 2 * (L112 + L113 + L123)

    alpha1 = Math.Asin(2 / (L112 * L123) * (S1 * (S1 - L113) * (S1 - L112)
* (S1 - L123)) ^ 0.5) * 180 / 3.1416

    ' for picture 2
    L213 = ((Y21 - Y23) ^ 2 + (X21 - X23) ^ 2) ^ 0.5
    L212 = ((Y21 - Y22) ^ 2 + (X21 - X22) ^ 2) ^ 0.5
    L223 = ((Y22 - Y23) ^ 2 + (X22 - X23) ^ 2) ^ 0.5

    S2 = 1 / 2 * (L212 + L213 + L223)

    alpha2 = Math.Asin(2 / (L212 * L223) * (S2 * (S2 - L213) * (S2 - L212)
* (S2 - L223)) ^ 0.5) * 180 / 3.1416

    A1 = (S1 * (S1 - L113) * (S1 - L112) * (S1 - L123)) ^ 0.5
    A2 = (S2 * (S2 - L213) * (S2 - L212) * (S2 - L223)) ^ 0.5

    TextBox19.Text = (1 - (A1 / A2)) * 100

    TextBox20.Text = alpha1
    TextBox21.Text = alpha2
    TextBox22.Text = alpha1 - alpha2

  End Sub

  Private Sub Button4_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click

    ' Clear the fields on the form

    Dim clear As String
    clear = ""
    TextBox1.Text = ""
    TextBox2.Text = ""
    TextBox3.Text = ""
    TextBox4.Text = ""
    TextBox5.Text = ""
    TextBox6.Text = ""
    TextBox7.Text = ""
    TextBox8.Text = ""
    TextBox9.Text = ""
    TextBox10.Text = ""
    TextBox11.Text = ""
    TextBox12.Text = ""
    TextBox13.Text = ""
    TextBox14.Text = ""
    TextBox15.Text = ""
    TextBox16.Text = ""
    TextBox17.Text = ""
    TextBox18.Text = ""
    TextBox19.Text = ""
    TextBox20.Text = ""
    TextBox21.Text = ""
    TextBox22.Text = ""


  End Sub

  Private Sub Still1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

  End Sub

  Private Sub Save1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

  End Sub

  Private Sub Still2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

  End Sub

  Private Sub Save2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

  End Sub
```

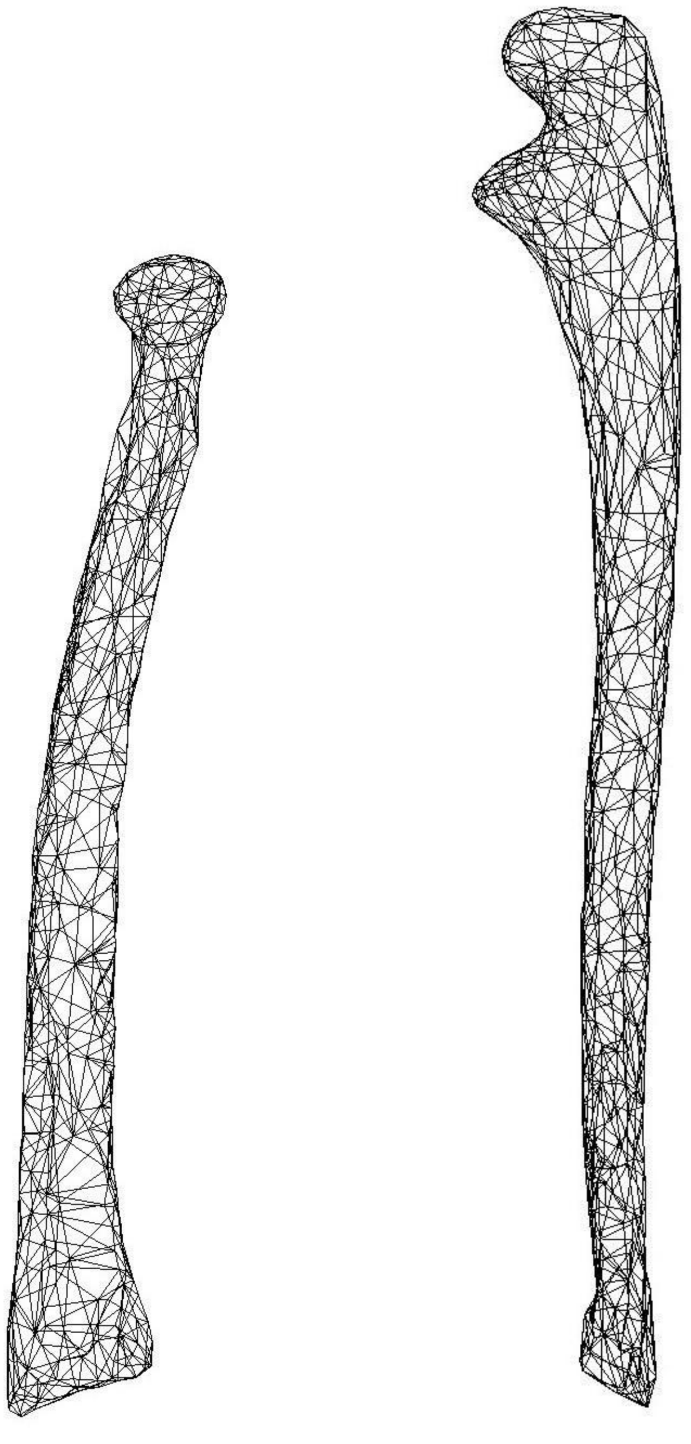
```
    Private Sub Draw1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

    End Sub


    Private Sub Draw2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

    End Sub

    Private Sub play1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

    End Sub
End Class
```

Appendix A3: Human Elbow Joint

A3.1: The bone triangular mesh files exported from Geomagic to gaming platform, each with 1000 triangle.



a)



b) c)

Figure A3.1: Triangular mesh files for the right hand elbow joint
a) Humerus, b) Radius, and c) Ulna

Table A3.1 Origin and insertion points of elbow joint ligaments

Elbow joint ligaments	Origin Point	Insertion Point
Elbow_Annular_Ligament (AL)	Unla_Supinator_Crest (approximate)	Ulna_Sublime_Tubercle (approximate)
Elbow_Radial_Collateral_Ligament (RCL) (part of LCL)	Humerus_Lateral_Epicondyle	Radius_Head and AL
Elbow_Lateral_Ulnar_Collateral_Ligament (LUCL) (part of LCL)	Humerus_Lateral_Epicondyle	Unla_Supinator_Crest
Elbow_Anterior_Medial_Collateral_Ligament (part of MCL) (also called Anterior Oblique)	Humerus_Medial_Epicondyle	Ulna_Sublime_Tubercle
Elbow_Posterior_Medial_Collateral_Ligament (part of MCL) (also called Posterior Oblique)	Humerus_Medial_Epicondyle	Ulna_Olecranon
Elbow_Transverse_Medial_Collateral_Ligament (part of MCL)	Ulna	Ulna
Elbow_Oblique_Ligament	Ulna	Radius

Table A3.2 Stiffness and damping parameters of elbow joint ligaments (Regan et al., 1991)

Regan et al., 1991	AL Stiffness (N/mm)	RCL Stiffness (N/mm)	LUCL Stiffness (N/mm)	Anterior MCL Stiffness (N/mm)	Posterior MCL Stiffness (N/mm)	Transverse MCL Stiffness (N/mm)	Oblique Ligament Stiffness (N/mm)
	28.5	15.5	57.0	72.3	52.2		

Table A3.3 Valgus and varus movement range of elbow joint relative to flexion angles

O`Driscoll et al. 1992 (Through comparison with literature relatively lower varus-valgus laxity in implanted joint has been corresponded in to muscle force absorption due to semi constrained hinge implant)	Normal (Intact) elbow joint maximum varus valgus laxity during active flexion (degrees)	Implanted (loose hinge) elbow joint maximum varus-valgus laxity during active flexion (degrees)	Normal elbow joint maximum varus-valgus Laxity during active muscle flexion (degrees)	Implanted (loose-hinge) elbow joint maximum varus-valgus Laxity during active muscle flexion (degrees)
Loose-hinge (linked)	$2.7^0 \pm 1.5^0$	$3.8^0 \pm 1.4^0$	$6.9^0 \pm 3.7^0$	$10.8^0 \pm 1.8^0$

An , 2005	Normal elbow Joint maximum varus-valgus laxity during flexion (degrees)	Implanted elbow joint maximum varus-valgus laxity during flexion (degrees)	Normal elbow joint maximum varus-valgus laxity with muscle loading during flexion (degrees)	Implanted elbow joint maximum varus-valgus laxity with muscle loading during flexion (degrees)	Implanted elbow joint maximum varus-valgus laxity with ligament release during flexion (degrees)	Elbow joint maximum varus-valgus laxity with radial head excision during flexion (degrees)	Elbow joint maximum varus-valgus laxity with MCL incision and radial head excision during flexion (degrees)
Coonrad Morrey semiconstrained TEA	$2.7^0 \pm 1.5^0$	$3.8^0 \pm 1.4^0$	$6.9^0 \pm 3.7^0$	$10.8^0 \pm 1.8^0$			
GBS III	$5.8^0 \pm 2.2^0$	$9.6^0 \pm 2.5^0$			$12.8^0 \pm 2.9^0$		
Norway TEA							
Ewald Capitellocondylar TEA				$4.3^0 \pm 2.4^0$			

Souter Strathclyde TEA			$4.3^0 \pm 2.3^0$ $3.5^0 \pm 1.7^0$ (with doubled muscle load)	$6.5^0 \pm 1.5^0$ $5.5^0 \pm 1.6^0$ (with doubled muscle load)			
Sorbie Questor (ulinked)				$8.6^0 \pm 4.0^0$		$13.3^0 \pm 5.5^0$	
Additional assessment	$3.4^0 \pm 1.6^0$						$11.1^0 \pm 5.6^0$

Ahmad, Park & ElAttrache, 2004 (Full tear of MUCL shows maximum valgus range of movement)	Load (N.m)	Change in valgus angle with partial tear of MUCL (medial ulnar collateral ligament) relative to intact MUCL (degrees)	Change in valgus angle with full tear of MUCL relative to intact MUCL (degrees)
Elbow flexion angle (30^0)	1.25	$2.30^0 \pm 0.71^0$	$6.59^0 \pm 2.76^0$
Elbow flexion angle (90^0)	1.25	$0.71^0 \pm 1.69^0$	$3.64^0 \pm 2.59^0$
Elbow flexion angle (30^0)	2.00	$2.21^0 \pm 0.81^0$	$7.37^0 \pm 3.45^0$
Elbow flexion angle (90^0)	2.00	$0.05^0 \pm 2.00^0$	$4.26^0 \pm 3.10^0$