uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Ding Cai

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Rapid Impingement Detection Systems for Ball-and-Socket Joints

TITRE DE LA THÈSE / TITLE OF THESIS

Wonsock Lee

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Chris Joslin

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

David Sankoff

Andy Adler

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Rapid Impingement Detection Systems for Ball-and-Socket Joints

Ding Cai

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

© Ding Cai, Ottawa, Canada, 2008

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-48592-7
Our file Notre référence
ISBN: 978-0-494-48592-7

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Detecting the position and the level of joint impingement is often a key to a computer-aided surgical plan to normalize joint kinematics. So far for ball-and-socket joints most of the existing impingement detection methods are not efficient or only consider the collided points as the detection result. In this thesis, we present two efficient and accurate collision detection systems to detect and evaluate the impingement on ball-and-socket joints.

Through the first system, we rapidly non-uniformly sample the object surface in the spherical coordinate system based on polygon rasterization and then globally check the distance differences between the objects to approximately estimate the impingement level; the second system has a similar design but is based on a uniform sampling method which can sample the object more adequately and reduce the memory cost by 20%.

We achieved the accuracy, the efficiency, and also the feature of providing overall impingement estimation and visualization on the joint surfaces. The first system is further applied in a real-time ROM observation simulation to flexibly check the location and the level of the impingement occurred during the motion.

Acknowledgments

First of all, I would like to thank my supervisors, Dr. Won-Sook Lee (School of Information Technology and Engineering, University of Ottawa) and Dr. Chris Joslin (School of Information Technology, Carleton University), for their understanding, guidance, support, and warm encouragement to complete my master studies.

I also appreciate Dr. Paul Beaulé (Division of Orthopaedic Surgery, Faculty of Medicine, University of Ottawa) for the comments on the medical aspects, Andrew Speirs (Division of Orthopaedic Surgery, Ottawa Hospital-General Campus) for the advices on CT data segmentation tool, Matt Kennedy (Biomedical Engineering, University of Ottawa) for the range of motion data, and Anna Conway (Division of Orthopaedic Surgery, Ottawa Hospital-General Campus) for coordinating CT data collection.

Finally, I would like to give my special thanks to my mother for her endless love and encouragement over these years.

Table of Contents

Figures

Tables

Glossary of Terms

2D	Two Dimensional
3D	Three Dimensional
DOF	Degrees of Freedom
LUT	Look-Up Table
ROM	Range of Motion
RSID	Rapid Spherical Impingement Detection
USID	Uniform Spherical Impingement Detection
CPU	Central Processing Unit
GPU	Graphics Processing Unit
CT	Computed Tomography
MRI	Magnetic Resonance Imaging

Chapter 1 Introduction

1.1 Motivation

"Femoro-Acetabular impingement (FAI) is a common cause of hip pain" due to "the morphologic abnormalities in the femoral head neck junction or the acetabulum" [GPB*03] [WG06]. FAI "limits the range of motion (ROM)" [GPB*03] [PMD*06]. We are motivated by the medical needs to reveal how the joint deformity leads to the impingement such as FAI. In this thesis, we focus on the problem of impingement detection for ball-and-socket joints, which is a part of a pre-operative arthroscopic surgical planning system (Figure 1) under development. Our impingement detection system, as a sub-system, is carried in the motion simulation step.

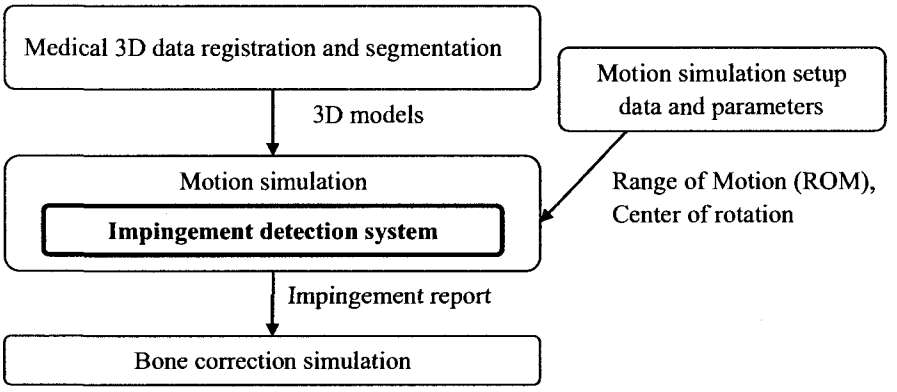


Figure 1: Process pipeline of .the pre-operative arthroscopic surgical planning system

Figure 1 illustrates the process pipeline of the pre-operative arthroscopic surgical planning system: before the "Motion simulation" step, 3D joint models are segmented and rebuilt from CT/MRI data in the modeling step "Medical 3D data registration and segmentation". The step "Motion simulation setup data and parameters" is to prepare the virtual motion simulation. The motion data, such as Range of Motion data, is

recorded from real scenes and is used to drive the virtual motion simulations. The simulation parameters, such as the rotation center of the femur bone, are used to setup models and the simulation scene; then during the "Motion simulation" step, our impingement detection system is carried as a function to detect the joint impingement; after the "Motion simulation" step, the impingement report can be used to evaluate the impingement. Further studies, such as the "Bone correction simulation", will be carried to simulate the surgery on the deformed joint.

The pre-operative arthroscopic surgical planning system is currently at the beginning stage. In this document, we only make research on the impingement detection system. So the other parts of the surgical planning system, such as modeling sub-system and motion simulation sub-system, are not in this thesis scope. These sub-systems are carried with simple scenarios at current research stage and will be studied in the future research for more scientific and accurate simulation results.

For our research, the efficiency and the accuracy of the impingement detection algorithm are considered the most important specifications. Estimating the overall impingement status on the joint surfaces, instead of just traditionally reporting a few collided points, increases the challenge.

1.2 Problem Statement

This thesis addresses the problems of developing algorithms to detect the impingement of ball-and-socket type joints.

There is one main issue we need to focus on:

- Impingement simulation: how to efficiently and accurately detect the impinged areas on the ball-and-socket joint surfaces.

Besides the main goals, we also consider:

- Surface-level impingement results: most of the existing impingement detection methods in Computer Graphics or in the medical area usually report point-level results which just mark the collided points on the models as the impingement detection results. In this traditional way, the user may locate the collided points but it is hard to estimate the impingement level and the status of the entire surface. The surface-level result can be more sufficient if all the points on the surface, the collided and the non-collided points, are reported with their distance information to the target.
- System practicability: how to ensure the overall efficiency of the entire system throughout the pipe line. The accuracy is an important goal of our research but the efficiency shall also be considered: the simulation shall be setup and changed at low prices; the simulation shall be processed in real time to help the user vividly and interactively understand the problem. The possible bottleneck may come from any algorithm of the system: efficiency on pre-processing step determines how fast the simulation can be setup and changed; efficiency on the simulation step determines the interactivity of the system to the user. Therefore, every step involved shall be efficient enough to make the entire system more friendly, more interactive, and more controllable to the users. At the beginning stage of the proposed long-term

research, it is valuable to make the fundamental algorithms more efficient before new algorithms and new functions are added in the coming years.

- System hardware requirement: how the system can be affordable to the low-end computers. One desired goal is to make the simulation can be carried not only on the high-end workstations in the research labs but also on the general computers as a cheap and flexible impingement estimation solution. Complicated simulations with a group of high resolution objects may require lots of memory. So memory cost shall be considered, especially at the first stage of this research.

1.3 Proposed Solution

This section lists the proposed solutions to the problems stated in the previous section. The solution to the major issue consists of the following techniques:

- Use collision detection technique dedicated to near-spherical objects to detect the impingement based on the sampled surface information.

The characteristics of the solution include the following approaches:

- Represent the model's geometry data in the spherical coordinate system rather than in the Cartesian coordinate system to obtain less computational complexity and memory cost. The femoral head and the acetabulum of the hip joint are both near-spherical objects so that the shape can be studied and sampled in a spherical coordinate system. In this way, we just need to use one distance parameter (the radius), instead of three traditional parameters (x,

y, and z) in the Cartesian coordinate system, to represent the near-spherical object surface in arbitrary orientation. We use very dense sampling covering all possible orientations to preserve the object geometry information.

- Uniform sampling to sample the near-spherical object more adequately and further reduce the memory cost. The representation of the object mentioned above is based on regular spherical coordinates, which is a non-uniform sampling and still uses unnecessary memory. Uniform sampling can further reduce the memory cost while the sampling precision is preserved.
- Look-up table data structure is used to save the sampled geometry data for instant by retrieving data in the simulation step. Quick accessing of the look-up table to get the required geometry data is an efficient way to speed up the impingement detection simulation. Different size of look-up table is used to reflect different sampling precision to represent the object geometry. For high-resolution simulation, a big look-up table can be configured to store the data; for low-resolution simulation, a small look-up table can be built.
- Surface-to-surface distance measurement is used to estimate the distance differences between the objects: The distances between the surfaces are measured on all collided and non-collided points. This feature provides more detailed global results which are particularly useful for an overall estimation. This surface-to-surface distance can be referred to as approximate strain between two objects.

Publications based on the work above are attached to the end of this thesis.

Chapter 2 Literature Review

In this section, the medical aspects of the hip joint are presented as the background study of the subject in our research. We also discuss the literature review on the collision detection methods in computer graphics and the medical area.

In the text and the figures of this document, we use anatomy directions to describe the viewing directions. The anatomy directions in the medical area are illustrated in Figure 2. The terms are described in Table 1 and are compared to the general directional terms used in Computer Graphics.

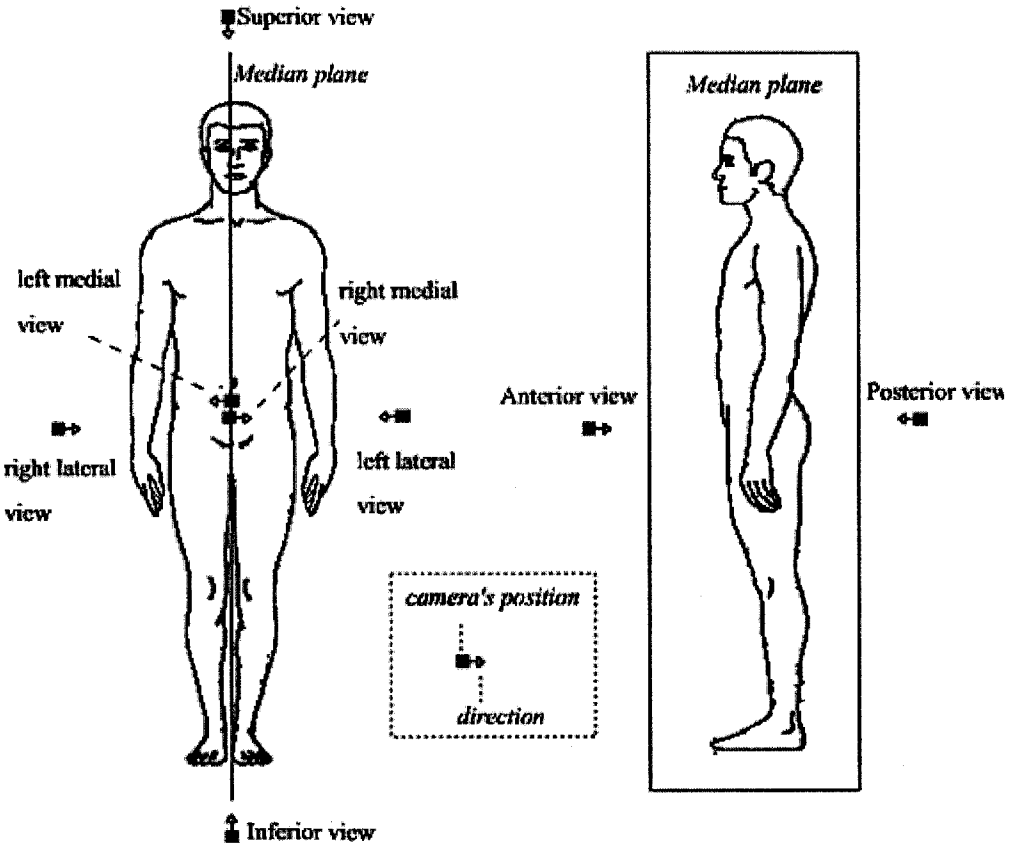


Figure 2: Anatomical directions (human body image reproduced from [SSS*05, p.39]).

Table 1: Equivalent directional terms used in human anatomy and Computer Graphics [SSS*05].

Directions used in human anatomy	**Location definition**	**Directions used in Computer Graphics**
Superior view	Upper or above	Top view
Inferior view	Lower or below	Bottom view
Anterior view	Toward the front	Front view
Posterior view	Toward the back	Back view
Left/Right lateral view	Away from the medial plane and toward the side	Left/Right side view
Left/Right medial view	Toward the median plane	N/A

2.1 The Hip Joint

2.1.1 Anatomy

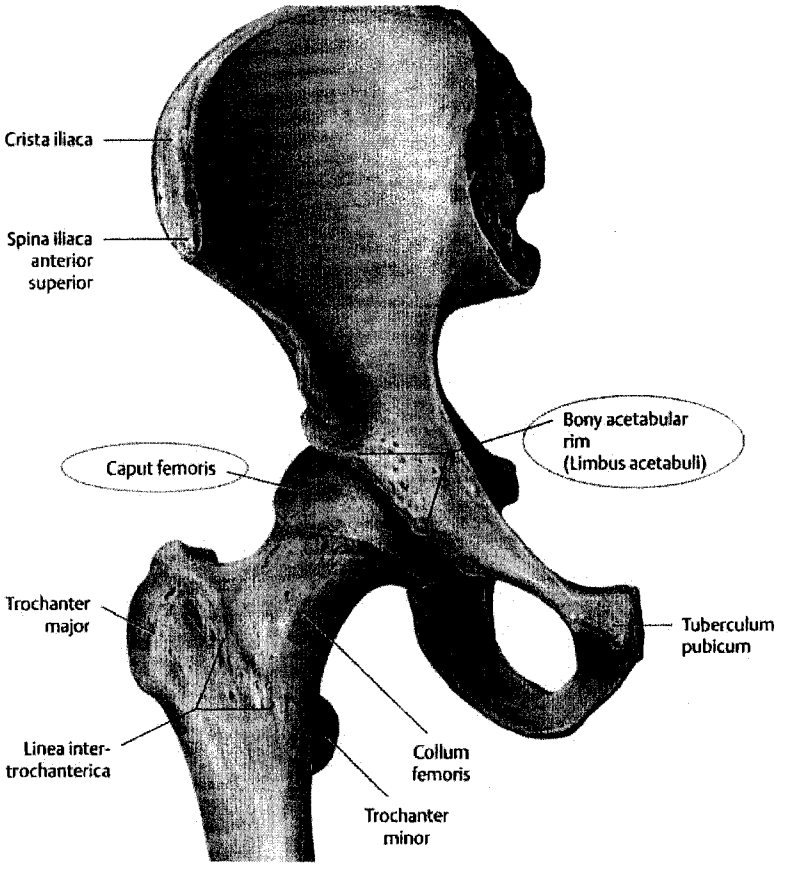


Figure 3: The right hip joint, anterior view (reproduced from [SSS*05, p.378]).

The Hip joint (Figure 3) is "the articulation between the pelvis and the femur" [Beh06, p.174]. It is classified as ball-and-socket joint. The joint's components are marked by ellipses in Figure 3: the grey *caput femoris* is commonly known as the head of the femur or femoral head; the *bony acetabular rim* is the outline of the acetabulum.

"The important structure of the pelvis regarding the hip joint is the anatomical area labeled the acetabulum" [Beh06, p.174]. The acetabulum is considered as the socket object. In Figure 4, the acetabulum is marked by the ellipse.

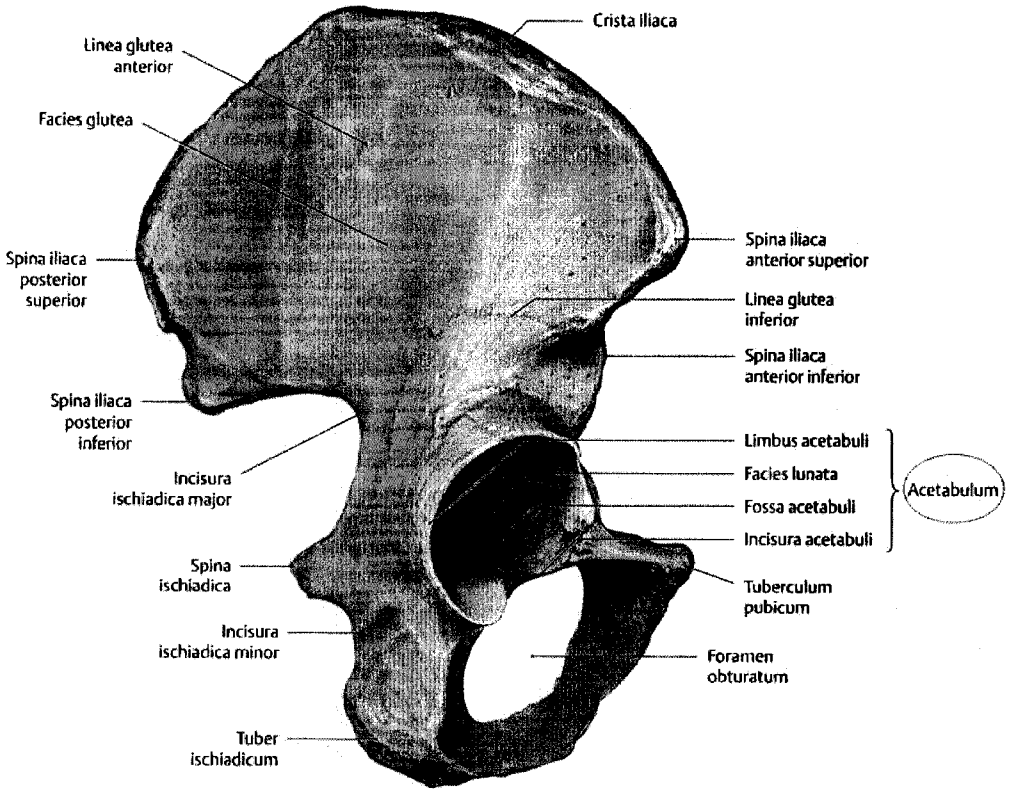


Figure 4: Human right pelvis, right lateral view (reproduced from [SSS*05, p.364]).

The femur is "the longest and largest bone in the body" [Beh06, p.174]. The head of the femur is "separated from the shaft the bone by the neck of the femur" [Beh06, p.175]. The femoral head is considered as the ball object. In Figure 5, the femoral head (also known as caput femoris) is marked by the ellipse.

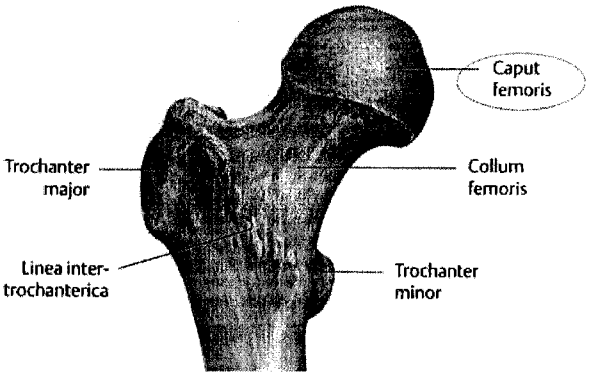


Figure 5: Upper extremity of human right femur, anterior view (reproduced from [SSS*05, p.366]).

The hip is "one of the most flexible and one of the most constricted joints of the body" [Jos06, p.70]. The other important joint in ball-and-socket joint category is the shoulder. "The hip joint is a much more stable joint than the shoulder because of the depth of the acetabulum" [Beh06, p.174].

Beside the bones, there are also tissues in the joint. The important tissue is the articular cartilages which smooth the movements between the bones. The femoral head and the acetabulum are both covered with a layer of the cartilage. According to [PBM83], the articular cartilage thickness of the hip joint is approximately 4.0 mm and "no sex or age difference among adults was found". The cartilages are illustrated as the blue layers covering the bones in Figure 6.

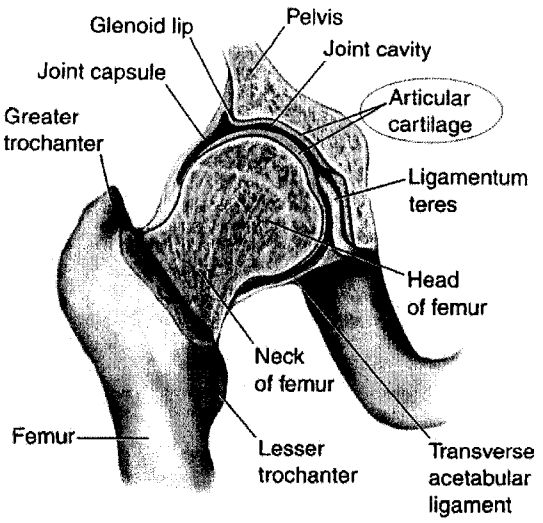


Figure 6: A longitudinal section of the hip joint (reproduced from [Beh06, p.176]).

2.1.2 Ball-Socket Joint and Its Function

Degree of Freedoms (DOF) defines "the possible movements of an object" [SSS*05, p.38]. There are three degrees of freedom in translation and three degrees of freedom in rotation. The hip joint has three degrees of freedom in rotation (Figure 7 a). "The main movement of the hip is rotation in any direction, albeit within restricted angles" and the hip "rotates around the nearly spherical head of the femur inside its socket" [Jos06, p.70]. As a comparison, the elbow joint, classified as a hinge type joint, has only one degree of freedom (Figure 7 b).

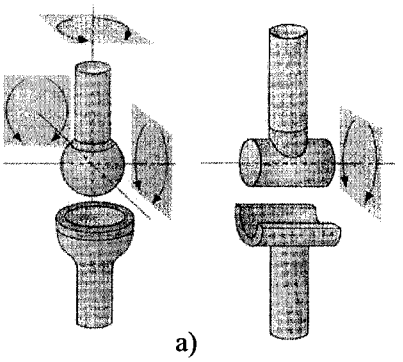


Figure 7: a) ball-and-socket joint's DOF; b) hinge joint's DOF as a comparison (reproduced from [SSS*05, p.38])

In kinetic anatomy, fundamental movements describe the movements "taking place in a plane (a flat surface) about an axis" [Beh06, p.28]. Similar to the 3 planes (*xy, yz,* and *zx*) used in 3D coordinate system, there are three planes (Figure 8) defined in human kinetics.

> "The Sagittal plane passes from the front through the back of the body, creating a left side and a right side of the body…the Frontal plane passes from one side of the body to the other, creating a front side and a back side of the body…the Transverse plane passes through the body horizontally to create top and bottom segments of the body" [Beh06, p.26-p.27].

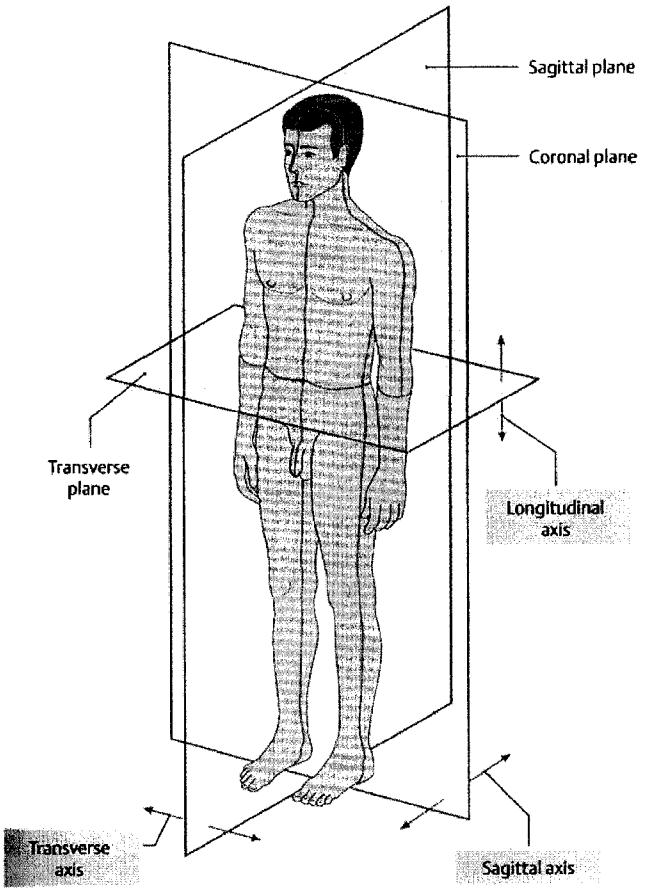


Figure 8: The axes of motion in the hip joint (reproduced from [SSS*05, p.25]).

Range of motion (ROM) is "the permitted relative motion of a joint" [BB00]. It is "characterized by the number of parameters that describe the motion space, and constrained by joint limits" [BB00]. Corresponding to the hip joint's movements on the three planes, three pairs of ROM are defined: range of flexion/extension on Sagittal plane (Figure 9 a); range of abduction/adduction on Frontal plane (Figure 9 b); range of internal rotation/external rotation on Transverse plane (Figure 9 c). The general average ROM values of a healthy hip joint are also presented in Figure 9.

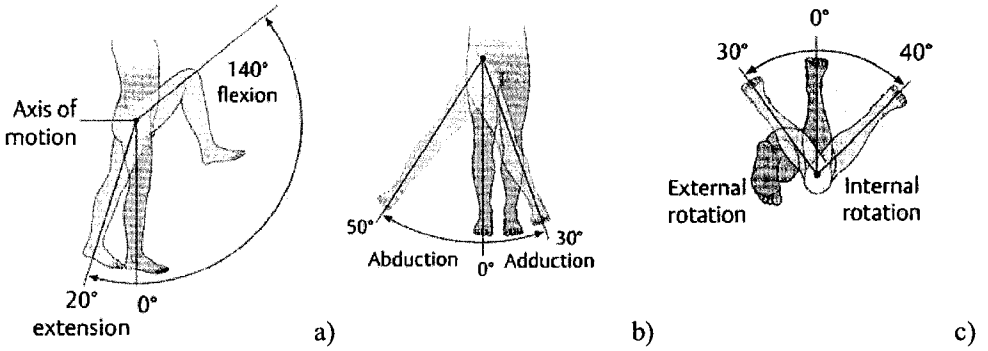


Figure 9: Three pairs of the hip joint' range of motion from the neutral (zero-degree) position (0°) (reproduced from [SSS*05, p.386]).

2.1.3 Femoro-Acetabular Impingement

Femoro-Acetabular impingement (FAI) is a primary disease of the hip joint and causes hip pain due to "the morphologic abnormalities in the femoral head neck junction or the acetabulum" [GPB*03] [WG06]. FAI "limits the range of motion (ROM)" due to the abnormal joint shape [GPB*03] [PMD*06] and can "instigate a progressive degenerative process and lead to early osteoarthritis of the hip" [WHC*03] [KDG91]. There are two types of FAI: Cam (Figure 10 b) and Pincer (Figure 10 c). The former has "excessive over coverage of the femoral head by the acetabulum" and the latter has "reduced femoral head and neck offset" [GPB*03] [LPB*04]. A mixed type (Figure 10 d) carries with Cam FAI and Pincer FAI.

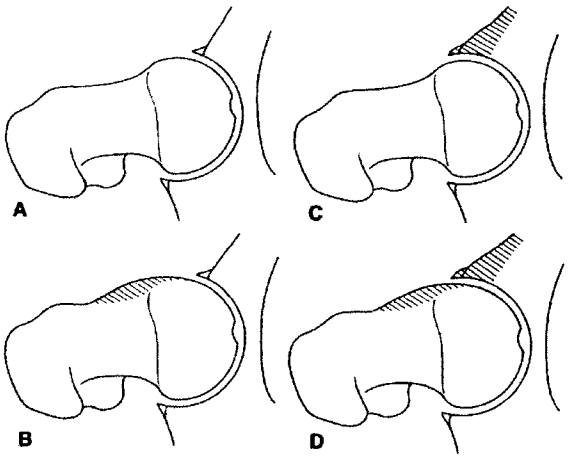


Figure 10: The factors causing FAI (reproduced from [LPB*04]); a) normal hip joint; b) Cam FAI; c) Pincer FAI; d) mixed type.

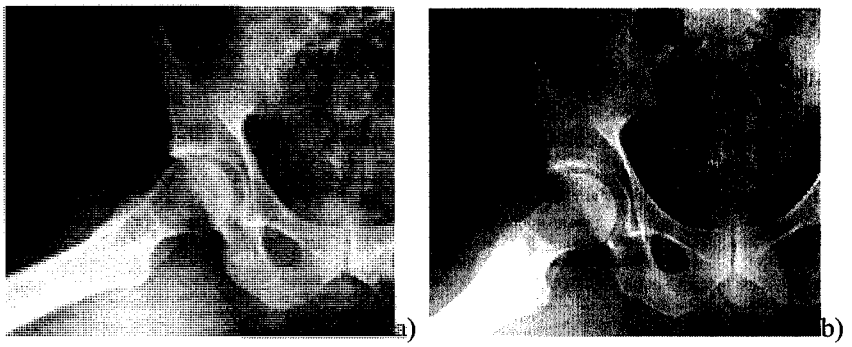


Figure 11: a) Oblique view of the right femur shows an abnormal femoral head–neck junction superiorly; b) Postoperative radiograph of the joint after femoral head–neck offset procedure (reproduced from [WG06]).

The surgical approach to treat FAI is to correct the abnormal shapes of the hip joint through resection osteoplasty (Figure 11) [WG06] [LPB*04]. The procedure is to "subluxate the femoral head, reshape the non-spherical femoral head, and excise the bony prominence of the femoral head" [LPB*04].

2.2 Collision Detection among 3D Objects

Collision detection is "also known as interference detection or impingement detection" [HLL*01]. The general difference between collision detection and

impingement is: the former usually describes the exact interferences between two surfaces; the latter may occur when two surfaces are close enough to produce strong forces on the surfaces. Generally in many of the current Computer-aided medical research, the impingement detection problems are solved as the collision detection problems. Collision detection is considered one major area in real-time simulations. The most basic and simplest collision detection solution is to run an exhaustive pair-wise testing between objects. Every primitive can be scanned and tested in this way but it is a naive approach with a great computation cost. Many methods are proposed to provide more affordable and more efficient solutions.

There are different ways to classify the existing collision detection methods. In [LG98], the methods are classified based on the approximations of the objects such as polygonal models against non-polygonal models.

In [TKH*05], classifications are based on the origins of the methods. The main categories are bounding volume hierarchies, distance fields, feature-based algorithms, and spatial subdivision.

The categories presented in [TKH*05] are further classified into two major groups in [KHI*07]. Kockara et al [KHI*07] used the terms of "broad-phase" and "narrow-phase" as higher categories which was first proposed by Hubbard [Hub93]. The two phases can be considered as two accuracy levels of collision examination. Broad-phase type collision detection methods process the algorithms either to just compute a coarse result or to filter out the potential collided objects which will be tested in narrow-phase. Spatial subdivision is classified in board-phase; Narrow-phase

type collision detection methods can further analyze the details. Bounding volume hierarchies, distance fields, and feature-based algorithms are classified in narrow-phase.

In the following sections, we will discuss the related existing collision detection methods, classified by bounding volume hierarchies, distance fields, feature-based algorithms, and spatial subdivision.

2.2.1 Bounding Volume Hierarchies (BVHs)

Bounding volume hierarchies (BVHs) are one type of data structure commonly used for general collision detection purpose either for rigid objects or for deformable objects. In this category, the methods commonly take advantages of tree structures' efficiency. BVHs decompose the object's surface into hierarchical partitions as trees (e.g. Figure 12). Collision detection is tested from the root (the entire body) to the leaves (the smallest partitioned elements). In the tree query, only the collided nodes are visited until the exact collided leaves are reported. The query time cost is reduced because the potential collided regions are examined. One common disadvantage of BVH approaches is that surface distance information is not provided.

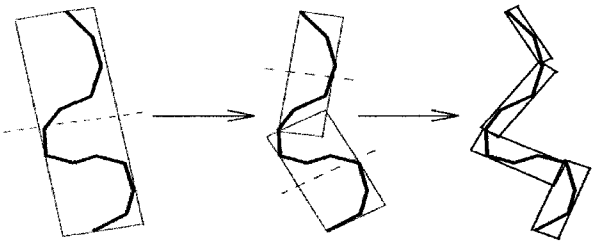


Figure 12: Building the BVH: recursively partition the bounded polygons and bound the resulting groups (reproduced from [GLM96]).

There two issues need to be considered to design a BVH based collision detection, one is the type of the bounding volume and the other is the tree design.

Bounding volume (BV) is a geometric description of an object. It can be a very simple description such as a sphere (or ellipse in 2D) or a box (a rectangle in 2D). The sphere based BVs have the advantage of completely independency of the orientation.

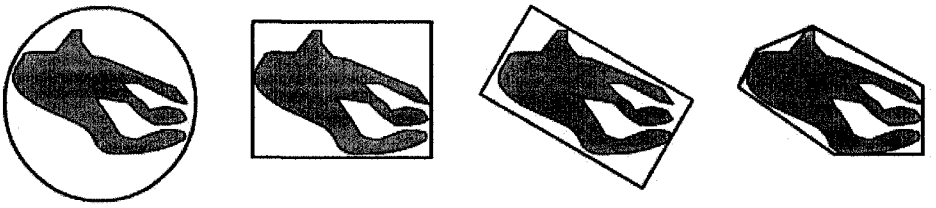


Figure 13: A variety of bounding volumes

Rather than assigning only one sphere/box to the object, BVHs describe the object as a hierarchical structure of many spheres/boxes. Based on sphere-tree design, the object's surface is partitioned and fitted with spheres as the bounding volumes [OD99] [KZ05].

Box based tree structures are more popular. The well-known methods are oriented bounding box tree (OBB-tree) and axis aligned bounding boxes (AABB-tree). Both methods have similar strategies to build the BVH for the object. AABB-tree builds AABB (Figure 13) bounding boxes aligned to the world's axes so it is fast to build; OBB-tree builds (Figure 13) bounding boxes aligned to the partitioned surface's local axes (Figure 12) so the tree is tighter and better fit the object.

In comparison to AABB-trees, the object-oriented property of OBB-trees provide tighter bounding volumes which are also independent from the world's axes. So

OBB-trees are more accurate while AABB-trees are simpler and easier to build. If the object is rotated, AABB-trees need to be recomputed to reflect the relative transformation between the world and the object.

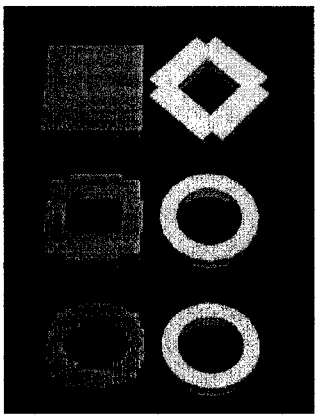


Figure 14: AABBs (left column) vs OBBs (right column) Approximation of a Torus: this shows OBBs converging to the shape of a torus more rapidly than AABBs (reproduced from [GLM96]).

The robust and accurate polygon interference detection library (RAPID) [GLM96] contains routines to build the oriented bounding box tree (OBB-Tree) data structure. It runs fast overlap tests between two OBB-Trees for robust and accurate collision detection, which has been used for virtual prototyping and simulation-based design applications [DJS96]. As a classic collision detection method, RAPID has been used in comparison in the recent surveys and publications.

Two OBB based collision detections are further developed from the work of RAPID. V-Collide introduced in [HLC*97] first filters out the potential contact pairs among the objects and then passes the selected objects to the RAPID algorithm for detailed collision results. [YM06] proposed a method to accelerate BVH access speed. Layouts of the BVH are computed, which "utilizes parent-child and spatial localities between the accessed nodes" [YM06]. During the simulation the access patterns of

the BVH can be predicted. The results on their selected models show a 26% -300% speed-up with respect to the RAPID algorithm can be performed. The limitation is "there is no guarantee that the layouts generated from a BVH always reduce the access time" [YM06] and assumptions are made where the access on a BVH starts.

One common disadvantage of RAPID and the further algorithms, like other BVH approaches, is that they do not solve the problem of computing the distance information between two rigid/deformable objects' surfaces. The collision detection is eventually performed by testing BVs' overlapping along the trees, not based on the distances between two objects' surfaces as Distance Fields (described in the next section 2.2.2) does. Researches related to distance computation with BVH have limited distance query problems such as the problem of the maximum length of the displacement between two objects [ZKM07]. Distance field type collision detection methods can easily provide the distance information between two objects comparing to BVH based methods, which will be introduced in next chapter.

2.2.2 Distance Fields

Collision detection using distance fields [FSG03] can be also efficiently processed by testing whether the target is inside or outside the pre-computed distance shield. Distance fields are also known as distance volumes [BMWM01] and distance functions [BMF03]. A nice feature is the distance information can be easily provided in a distance field while the methods in other collision detection categories cannot provide or require extra algorithms to compute. Computing a distance field can be

expensive, especially for a high resolution result. Once the distance field is generated, collision detection can be processed very efficiently so it is particularly useful for rigid objects in both interactive and non-interactive simulations. If the distance field requires frequent updates to catch up the changes of the object's surface such as deformation, current distance field algorithms are not fast enough to regenerate the fields for interactive simulations. In addition, usually distance fields are computed only in the common Cartesian coordinate system.

There are three major data structures used in the category of distance fields based collision detection: uniform 3D grids, octrees and BSP-trees.

Uniform 3D grids are simple and easy to construct in the 3D scene. The distance data is computed based on the grid point to the object's surface. The field's resolution is quite limited by the memory requirement. As an example, 8GB memory is required to store a 1024^3 grid in double precision 64-bit values. If a small feature on the object shall be captured, the entire 3D grid shall be globally adjusted to fit the resolution and memory waste is inevitable.

Figure 15: 3D Distance Field of Hugo Model (17k polygons): Distance to the surface is color coded, increasing from red to green to blue. The resolution is 73*45*128 (reproduced from [SOM04]).

Tree-structure and level-of-detail (LOD) techniques are introduced into distance fields to solve the drawbacks from the field generation. Octrees (one node has up to eight child nodes) based methods such as adaptively sampled distance fields (ADFs) [FPRJ00] and Binary Space Partitioning (BSP) trees based methods such as [WKE99] are proposed to provide sufficient details stored in the trees with an acceptable tree depth. Level-of-detail (LOD) describes the resolution level. Subdivisions rules are applied to determine the level of the region to be subdivided. Such processes reduce the memory cost, but the trade-off is the higher computational complexity and the higher time cost.

Scan-conversion and Voronoi diagrams are used to generate the multi-dimension distance fields [BMWM01] [SOM04] [SPG03]. The common method is to first cut the model into 2D slices. Then scan-conversion is used to determine the grid points which will be further used as the site points of Voronoi cells. Finally, Voronoi diagrams, as the distance fields to the primitives, are then generated on the slices to partition the corresponding regions to the primitives (point, edge, or face). Computing Voronoi diagram is a classic problem in Computational Geometry and usually takes an expensive computation cost. The methods are proposed to take advantages of the hardwares such as GPU to accelerate the process of scan-conversion and Voronoi diagram.

Determining the collisions based on pre-generated distance fields is generally fairly easy by comparing the target's location to the distance field. One important feature that distance fields can provide is the penetration depth computation

[TKH*05]. Since the distance fields are pre-generated, it is easy to compute the distance from the target to the corresponding surface to check how further the target has gone into or intersected with the object. This feature improves the reality and interactivity of the simulations and allows more collision detection feedbacks.

Distance fields have great efficiency and accuracy in the simulation step but require a time-consuming preprocessing step. This category is more suitable to the rigid objects than the deformable objects which require frequent updates. The current research of distance fields on deformable objects is still in progress, some of which are still only applicable to 2D problems [HZLM01].

2.2.3 Feature-based Algorithms

Feature-based algorithms are built based on the objects' features (the basic components of polygonal models) such as points, edges, and faces. While BVH methods describes the object as a set of bounding volumes and distance field methods generate distance shields as references to the object's surface, feature-based algorithms directly use the object's feature elements to locate the collided regions by searching for the closest features of two objects [LC91].

More complicated methods are developed to expand the concepts of the features such as [Mir98] which generates Voronoi regions for the features and detect collisions by testing a feature with others' Voronoi regions. Voronoi planes are generated based on the features to assign a corresponding space for the features. If model A's feature F_a and model B's feature F_b are both in each other's Voronoi region, then they can be

potential closest pair of features and further test on them will be carry on [Mir98]. For example in Figure 16, there are two models, X and Y. "Feature F(X) is edge E and feature F(Y) is vertex V. If P(X) is in Voronoi region of Y and P(Y) is in the Voronoi region of X, the F(X) and F(Y) are closest pair of features" [Mir98].

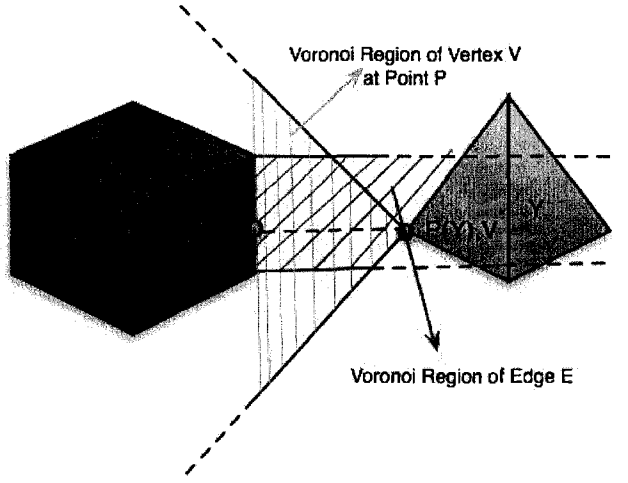


Figure 16: Closest pair of features for Vertex-Edge feature pair (reproduced from [KHI*07]).

Feature-based algorithms have many limitations besides the computation cost. They only work for closed objects and rigid objects and may fail on the penetrating polyhedra [KHI*07] so that they are not considerable for dynamic simulations.

2.2.4 Spatial Subdivision

Most of the current collision detection methods are object-oriented. As a quite different approach, spatial subdivision type methods focus on space-oriented examination. Similar to the object-oriented methods mentioned in the previous sections, there different ways to partition the space such as uniform 3D grids [Lev66] and trees [BT95] [Mel00].

The basic idea is to recursively cut the space into certain number of cells. The collisions can then be detected based on the components contained in the cells. Uniform 3D grids, as discussed already, can take huge memory to capture a small feature on the object with an $O(n^3)$ space complexity. Since many methods in this category are object-independent, the geometric complexity of the object in the space does not influence the efficiency of the algorithms. The other advantages of the spatial subdivision technique are that it is a simple and fast approach and is "independent of topology changes of objects" [TKH*05]. So it can be applied to both rigid and deformable objects. Furthermore, "the performance of spatial subdivision is independent of the number of objects" [TKH*05] in the target space; however, this technique requires a much more computation cost and memory cost since it is space-oriented. A common problem of spatial subdivision type methods is that it is hard to determine the size of the cell or the data structure design [TKH*05].

2.2.5 Other methods

In Computer Graphics, there exist innumerous collision detection methods and the family is still expanding.

Image-Space based algorithms are implemented and accelerated by the GPU such as the ray-casting algorithm [KP03] (Figure 17). The ray-casting algorithm takes advantages of a Computational Geometry theory that a point must locates inside a (closed) object if and only if the ray from the points hit the object's boundary odd times [KP03].

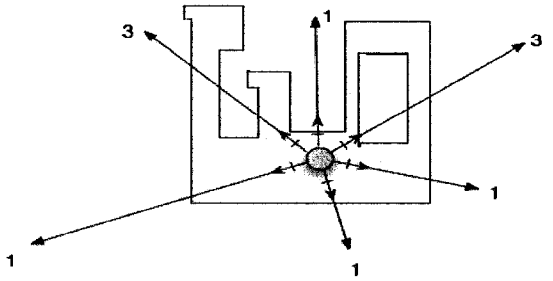


Figure 17: Ray-casting algorithm (reproduced from [KP03]).

Stochastic Methods have an interesting physically-plausible property. Since the common models used in Computer Graphics are polygonal models, the models are just approximation of the real geometry [TKH*05]. In addition, humans are easily confused between "the physically-correct and physically-plausible behavior of the objects" [BHW96]. For example, the geometry of the two objects (balls) illustrated in Figure 18 are approximated by a group of random sample points (on which the collisions are tested). Collision results may not be very precise but can provide approximated information. In this category, probabilistic methods and stochastic sampling methods are proposed and provide a well-balanced trade-off between the speed and quality [TKH*05]. It is more accurate than the simple bounding volume based methods and has a reasonable computation complexity.

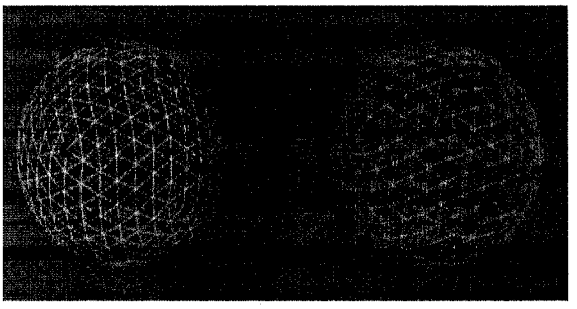


Figure 18: Stochastic collision detection between two volumetric bodies (reproduced from [TKH*05]).

2.3 Impingement Detection Methods for Medical Simulations

In the medical area, impingement detection is generally considered a type of collision detection when the interference occurs between the subjects in the body. The methods in Computer Graphics mentioned in the previous sections can also be adapted to the medical simulations. Many of the existing impingement detection methods consider the term of impingement is equivalent to the term of collision though the impingement may occur when two surfaces are not yet touched. The collision detection methods for deformable objects are suitable to solve the problems of soft tissues such as skins and muscles; the collision detection methods for rigid objects can be applied to the bones. Depending on the interactivity needs of the simulation, the methods can be designed to balance between speed and accuracy. For the interactive simulations such as real-time surgery simulations, time cost is more critical since such simulations require instant collision responses. For the non-interactive simulations, the accuracy of the results may be more important so that the computations or animations can be processed off-line. In the following sections, we will discuss about the general medical simulations and then the specific simulations for ball-and-socket joints.

2.3.1 General Methods for Medical Simulations

For the general medical simulations, the collision detection methods in Computer Graphics are also applicable. There are also specific methods proposed based on the

model format used in the medical area or further developed for medical purposes. In addition, collision response is a research direction that produces medical analysis results based on the collision results generated from the collision detection step. Mass-spring system (MSS) and finite element method (FEM) are commonly used to simulate and analyze deformable objects such as soft tissues [SMVT04]. MSS is suitable for real-time simulations and deformation simulations while FEM is used in non-interactive simulations. Both methods are embedded into the object and applied to respond to the collisions. MSS is easy to construct and easy to adapt to the object's arbitrary shape. It decomposes the object into sets of mass points that are located on the features of the object; "in FEM, the continuum (object) is divided into elements joined at discrete node points" [SMVT04]. FEM provides "a more realistic simulation than mass-spring methods but are less computationally efficient". [SMVT04].

In [BMG99], the author solved collision detection by testing the probe with a list of the tetrahedrons that are the basic components of the mass-spring network (Figure 19). The list is kept updated by tracing the neighboring tetrahedrons which may intersect with the probe so that no all the tetrahedrons need to be tested every frame.

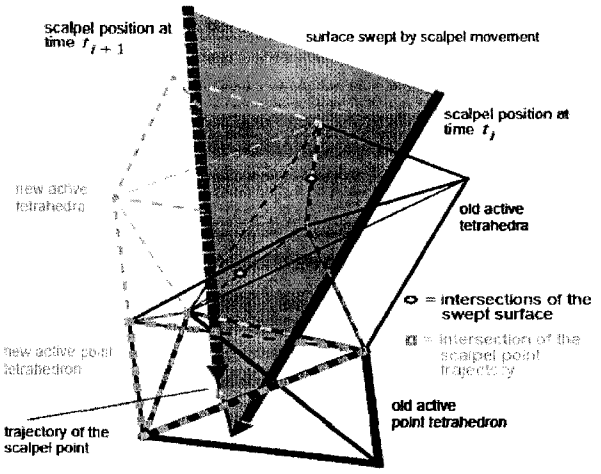


Figure 19: Local collision detection using a list of active tetrahedral (reproduced from [BMG99]).

In [LCN99], the authors addressed the problem of real-time collision detection for virtual surgery. They proposed a method based on graphics hardware. The camera's viewing volume (Figure 20) is considered as the probe's volume. In Computer Graphics, camera's viewing volume defines the 3D volume that is visible to the user. Elements outside the viewing volume will not be captured by the camera and thus will not be rendered on the screen. Lombardo et al. [LCN99] uses the visibility to detect the collision. If there are certain triangles that are visible to the camera (appear in the viewing volume), then these triangles must touch with the extremity of the probe so that the collision is then detected. The idea was implemented with OpenGL and is compared to RAPID [GLM96] and is five times faster than RAPID on a 1k-triangle liver model [LCN99]. Since the viewing volume is applied as the probe's volume, this method is limited within a small set of problems, such as regular box volume testing. It can only represent one single box object to

collide with the other objects and is not suitable on arbitrary shapes for collision detection tests.

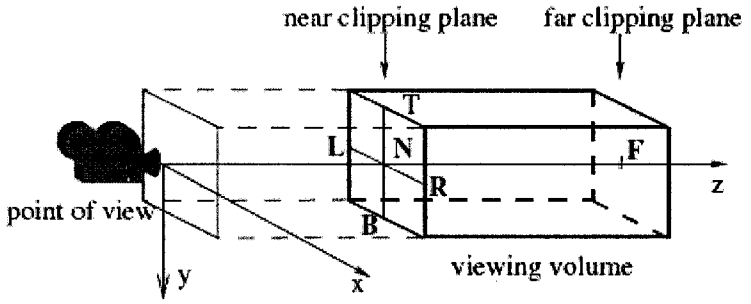


Figure 20: The OpenGL orthographic camera (reproduced from [LCN99]). The viewing volume is a box.

In [DDCB01], an adaptive mass-spring system is proposed. The basic collision detection method was inherited from [LCN99]. The collision response is enhanced with multi levels of the mass-spring system adapted to the liver. As an example, Figure 21 illustrates the mass-spring system based method with a level-of-detail (LOD) design. Several mass-spring systems with different resolutions are generated as the proximities of the same object. Each mass point from the lower resolution system is the parent point of a few mass points from the higher resolution system in the same region of the object body. In this way, a coarse mass-spring system takes the first collision response and a child system with a higher resolution is triggered in that local region to generate the detailed collision feedback. The non-collided areas will not be examined. The collision response is then more realistic and faster.

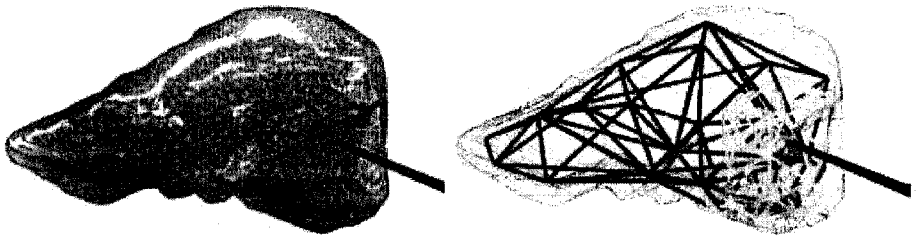


Figure 21: Real-time multi-resolution model allows for a wide range of applications, from virtual surgery simulators (liver laparoscopic operation) (reproduced from [DDCB01]).

2.3.2 Methods for Ball-and-Socket Joints

Detecting collision and measuring the stress/strain as the collision responses for the ball-and-socket joint has been researched with different approaches in the medical area.

Several mathematical approaches to compute stress and strain such as the methods proposed by Bartos et al. [BK95] and Genda et al. [GIL*01] were developed to compute the forces between the bones. Yoshida et al. [YFW*06] used Discrete Element Analysis (DEA) to measure the "pressure distribution of the hip joint during activities of daily living". Those studies are either theoretical research on ideal cases or only applied to coarse 2D models.

More flexible methods for 3D virtual operations on hip joint have been developed recently. In [MBT03] a mass-spring system is introduced to simulate the soft tissue (e.g. cartilage) behaviors. Spheres, called as "molecules" in [MBT03], are used as sensor units which respond to the external forces and the internal forces. Springs connect the molecules and a network of collision sensors is formed. The entire system works under the traditional mass-spring system's rules. A uniform 3D grid (Figure 22

a) is used to represent the idea by the authors. It is another form of sphere bounding volume hierarchies representation discussed in the previous collision detection sections. Based on the ideas of [MBT03], Sarni et al. [SMVT04] developed a molecular model based system to represent the cartilages to detect the impingements and measure the forces inside the hip joint. The precision of the molecules-based system is the bottleneck. With a small number of molecules, the cartilages' surfaces are approximated in very coarse forms; with a great number of molecules, the efficiency of the system is reduced. One more problem is the molecules as spheres, however, cannot form and represent the smooth surface, which causes errors to detect the impingement.

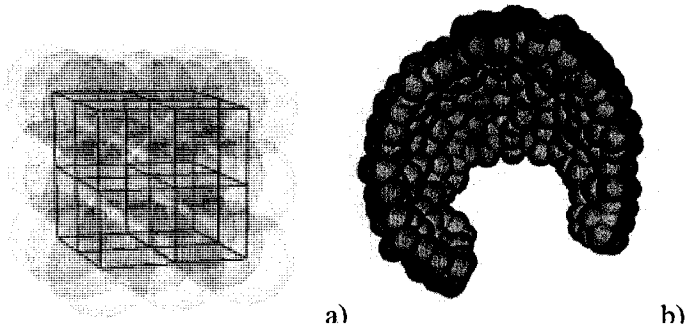


Figure 22: a) Molecules-based system with 27 molecules in uniform 3D grid (reproduced from [MBT03]); b) The cartilage cap of the acetabular cup discretized into a set of spherical regions (reproduced from [SMVT04]).

Hu et al. [HLL*01] developed a spatial based impingement detection algorithm by using a look-up table (LUT) and a linear transform. As a reference, this LUT based impingement detection system design inspired us. First they assume the model is voxelized model which is the common 3D form of the models used in the medical area. Voxel is a 3D volume unit like the pixel unit in 2D image. Then based on the uniform 3D world space, each voxel can be assigned with a unique index (*id*) number.

A 1D LUT is created to store and manage the voxels according to the voxels' index numbers. The indexing requires 3 parameters (x, y, and z coordinates), which refers the 3D space. The quality depends on the resolution of the voxelized object. Each unit space has the same size as the given voxel and is referred to an LUT element so that all the coordinates inside a unit space correspond to one LUT element.

In the preprocessing step, each object's voxel is scanned. An *id* (Eq. 1) is generated based on the voxel's coordinates. The corresponding LUT element under that id will be marked with a binary value of 1 if the object occupies that unit space; on the other hand, the binary value 0 will be marked if that unit space is not occupied by any of the object's voxels.

$$id(x,y,z) = x - Min_A X + (y - Min_A Y) \cdot (Dim_A X + 1) + (z - Min_A Z) \cdot (Dim_A X + 1) \cdot (Dim_A Y + 1) \quad \textbf{Eq. 1}$$

where Dim_A is the 3D extent of object A and Min_A is the extreme position

During the simulation step, the model B's voxels will be scanned and checked with the pre-computed LUT. The id of each B' voxel is generated with the same equation. If the corresponding LUT shows it is occupied by model A's voxel, then an impingement must occur right at that unit space [HLL*01].

The following equations show each combination of the (x, y, z) coordinates has one and only one corresponding id to access the LUT.

$$z = \text{int}\left(\frac{id}{(Dim_A X + 1) \cdot (Dim_A Y + 1)}\right) + Min_A Z \quad \textbf{Eq. 2}$$

$$y = \text{int}\left(\frac{id - (z - Min_A Z) \cdot (Dim_A X + 1) \cdot (Dim_A Y + 1)}{Dim_A X + 1}\right) + Min_A Y \quad \textbf{Eq. 3}$$

$$x = id - (z - Min_A Z) \cdot (Dim_A X + 1) \cdot (Dim_A Y + 1) - (y - Min_A Y) \cdot (Dim_A X + 1) + Min_A X \quad \textbf{Eq. 4}$$

The algorithm is implemented and demonstrated on a hip joint reconstructed from the CT scan. The only problem of the system is to balance between the performance and the precision. Since the voxels are organized on 3D grid, higher precision can simultaneously increase the extents on x, y, and z axes simultaneously under $O(n^3)$ space complexity. So the size of the required LUT increases by three orders of magnitude. Both of the sampling in the preprocessing step and the memory cost of storing the uniform 3D grid can be the bottleneck of the system.

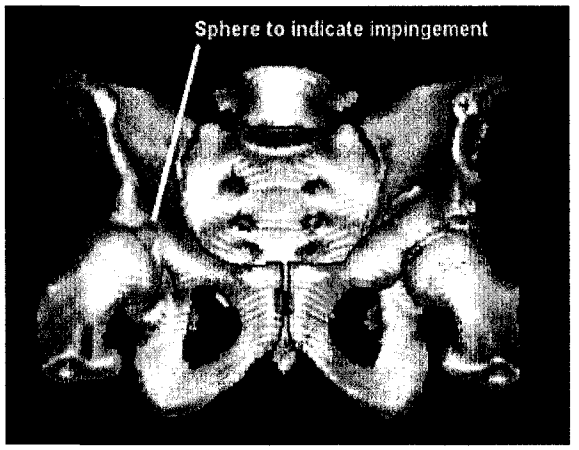


Figure 23: The impingement detection algorithm of Hu et al. (reproduced from [HLL*01]). The CT scan is 3-mm thickness with a resolution of 256*256.

Kubiak-Langer et al. [KTM*07] [TKL*07] further used the algorithm proposed in [HLL*01] to compute the range of motion (ROM) in one anterior FAI. The femur rotates about a fixed rotation center and stops once an impinged point is detected. The resulting limitations of the motions are then considered as the ROMs. The impinged point during the motion is marked by a circle or a sphere as the impingement report.

Very recently Maciel et al. [MBT07] used a spherical sliding method which indexes the triangles in the partitioned space and detects impingement by testing the triangles in each unit space. Model *A* represents the ball model and model B represents the socket model. In the sampling step, a look-up table (LUT) is created corresponding to the spherical orientations. A group of sampling rays are used to sample the model A's surface in each orientation. If there is a triangle hit by a ray, then the triangle is referred by an LUT element corresponding to that ray's orientation (Figure 24). The LUT is filled by the references of the sampled triangles.

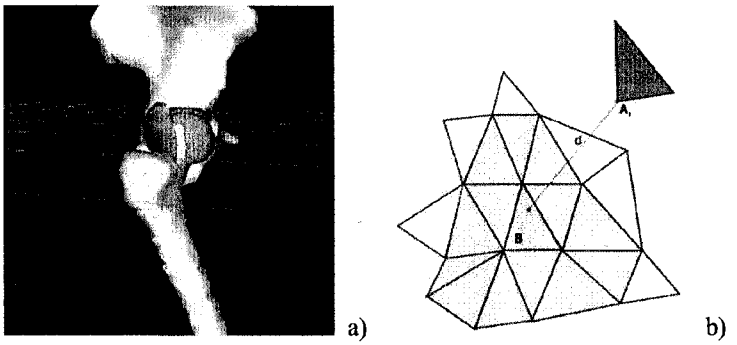


Figure 24: a) hip joint impingement simulation by Maciel et al. [MBT07]; b) the sampled triangles are referred from LUT data structure in the simulation to detect the collision (reproduced from [MBT07]).

Then during the collision detection step, the other model B's vertex is checked with model A's LUT. If that vertex of B happens to hit or penetrate the triangle of A in the same orientation, then there is an impingement. The method proposed in [MBT07] is efficient in run-time calculation, but it requires an expensive $O(n^2)$ sampling method to test m sampling rays with the n triangles in pre-processing. As a reference, this is the most similar to ours and we will compare this method with ours in Chapter 3.

Chapter 3 Rapid Spherical Impingement Detection

Like many of the methods discussed in the Literature Review section, before the simulation we need to save the object's surface geometry information into a data bank and then in the simulation we use the sampled geometry data to detect the impingement by checking whether the target is inside or outside of the surface. Based on the idea, we designed a look-up table (LUT) based data structure for near-spherical objects and developed a **Rapid Spherical Impingement Detection** system (**RSID** system). To summarize our system it contains two major algorithms:

(i) Sampling algorithm: scans and stores the models' geometry data into LUT, which is the major part of the preprocessing step.

(ii) Impingement detection algorithm: detects the impingements between the models based on the sampled data, which is the major part of the simulation step.

First in the sampling, we input triangulated models. In Computer Graphics and in the medical area, triangulated models are common for simulation purpose. The sample points (e.g. Figure 25 b) on the model surface are rapidly generated through rasterization in the spherical coordinate system. The sample points are used as the proximity of the object (e.g. Figure 25 c). We use a large size LUT, indexed by spherical coordinates, to refer the sample points and store the sampled geometry distance data gained from the sample points.

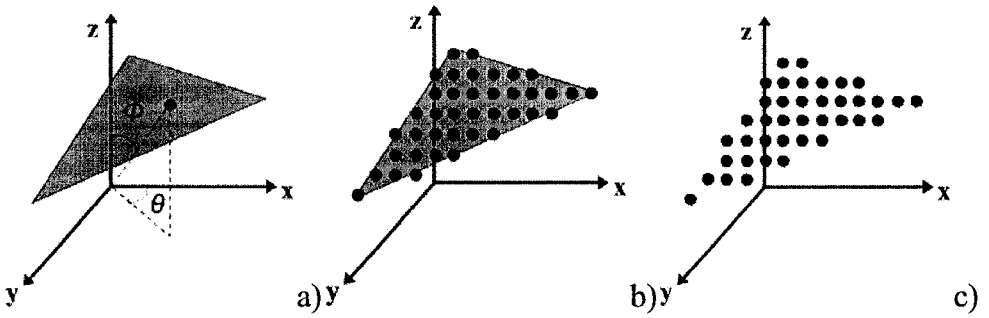


Figure 25: a) A triangle in 3D space and an arbitrary sample point on the triangle with its spherical coordinates; b) the sample points generated on the triangle; c) the proximity of the triangle, formed by the sample points.

Then in the simulation, we use the sampled geometry data stored in the LUT as the reference of the object surface and check the distance difference between the target and the object surface in the orientation of the target. The impingement can be quickly detected when the target is on or inside the object surface (e.g. Figure 26 c).

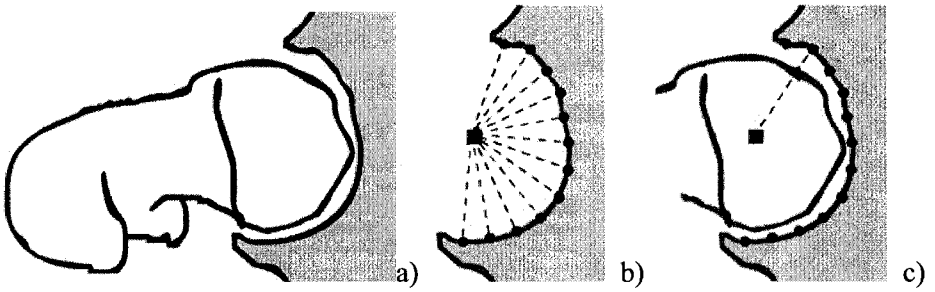


Figure 26: 2D example: a) the hip joint; b) sampling step. Rectangle is the origin of the spherical coordinate system. Dots are the sample points. Dash lines are the distances from the sample points to the origin; c) simulation step. The vertex on the ball object is tested against the sampled data in the orientation of that vertex.

A flowchart (Figure 27) is available to help understand the system pipeline. We input two models, sample them, and detect impingement between them. The details of the impingement detection system will be introduced in section 3.1, section 3.2, and section 3.3.

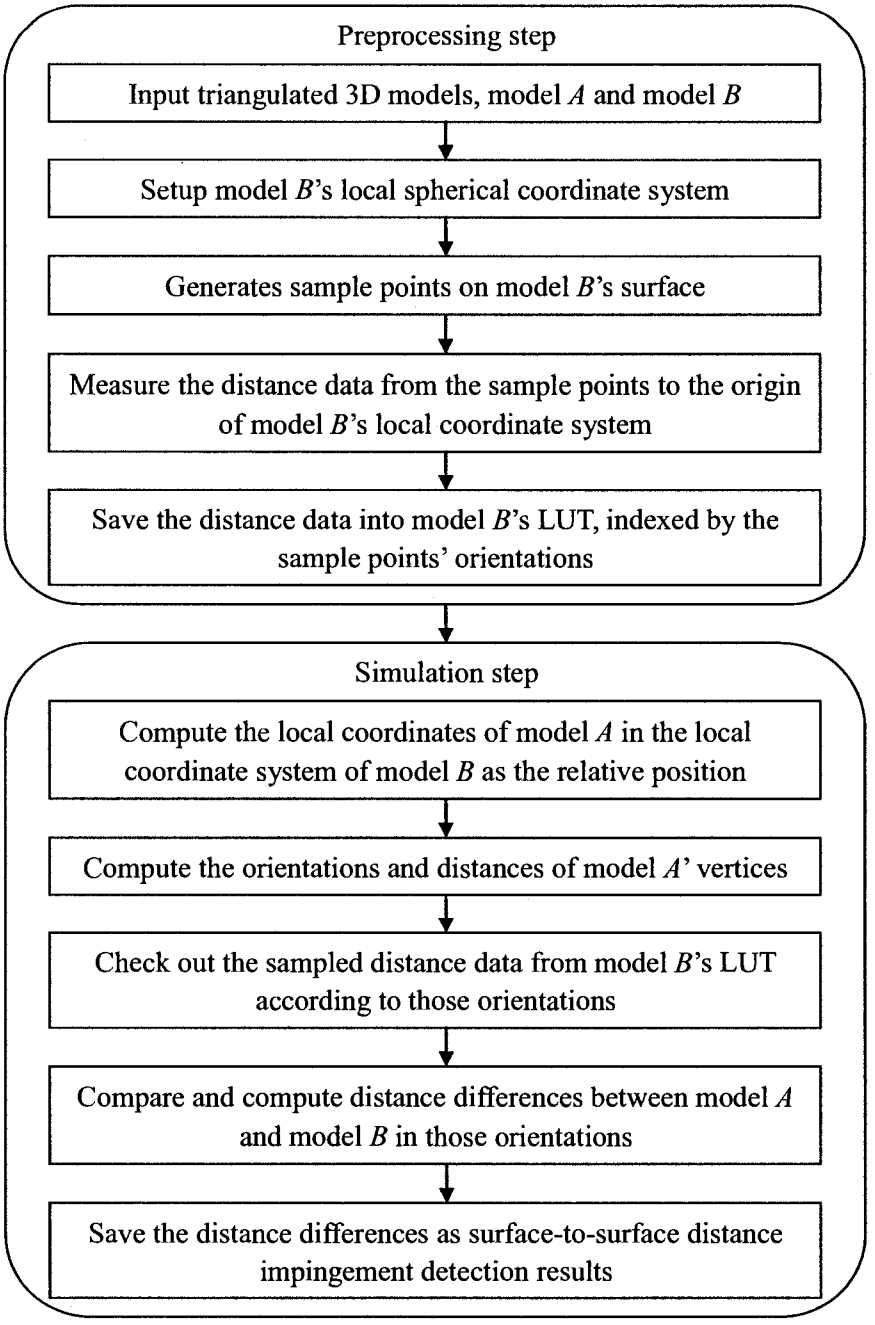


Figure 27: Flowchart of the Rapid Spherical Impingement Detection System. The example here is testing model *A*'s vertices against model *B*'s LUT.

This solution can also provide a rapid surface-to-surface distance measurement feature by saving the distance differences between two surfaces in all orientations, which BVH based collision detection methods cannot provide and distance field

based collision detection methods need high pre-processing cost to accomplish. Due to the complex structure of the hip joint, the impingement may not just occur when two bones physically touch each other but occurs when the distances between the two bones are small enough to cause pain by strong strains on the tissues such as the cartilages between the bones. Therefore, the level of impingement also depends on the distances between the two bones. The feature is used in our system to represent the approximate strain measurement on the surfaces of the ball-and-socket joint since the impingement detection is based on distance data directly.

3.1 Look-Up Table for a Geometric Object

Look-up table (LUT) is the data structure we use in our system to store the model's geometry information. Each model has a corresponding LUT. The LUT is configured as a 2D array with 2 indexing parameters ϕ and θ.

ϕ and θ are the two angle parameters used in the standard spherical coordinate system where there is one more parameter known as the distance d (or generally called the radius r).

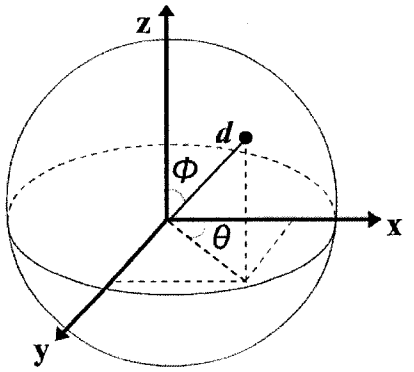


Figure 28: Spherical coordinate system

Each combination of the two angle parameters ϕ and θ in the spherical coordinate system produces a unique ray in 3D space. With one more parameter, the distance *d*, it produces a one-to-one correspondence to a vertex *(x, y, z)* in the Cartesian coordinate system:

$$x = d\sin\phi\cos\theta\ ;\ y = d\sin\phi\sin\theta\ ;\ z = d\cos\phi \qquad \textbf{Eq. 5}$$

where $d \in [0,\infty), \theta \in [0°,360°)$ *and* $\phi \in [0°,180°]$

$$d = \sqrt{x^2 + y^2 + z^2}\ ;\ \theta = \tan^{-1}\left(\frac{y}{x}\right);\ \phi = \cos^{-1}\left(\frac{z}{d}\right) \qquad \textbf{Eq. 6}$$

where $x \in (-\infty,\infty), y \in (-\infty,\infty)$, *and* $z \in (-\infty,\infty)$

In this document, the ray configured with ϕ and θ is generally called **sampling ray.** The ray is used to measure the distance from the origin of the ray to the model surface in the orientation of the ray. The origin of the sampling ray is set to the origin of the spherical coordinate system; the intersection point, which is generated by the intersection of the ray and the model surface, is generally called **sample point**.

The way to index the sample point into the LUT is using the ϕ value as the 1st index (row) and the θ value as the 2nd index (column): $LUT[\phi][\theta] = d$ where *d* is the distance from the origin to the sample point (if there is an intersection). In other words, one LUT element stores the distance data sampled from the object surface in the orientation of (ϕ, θ).

Besides the distance data *d*, it is also possible to choose the object's features (vertices or triangles) as the data stored in LUT type data structures. But it is hard to organize and index such data since the object's vertices or triangles are usually not well distributed in the space. Indexing one orientation with one vertex/triangle of the

object has a vertex/triangle capturing error as discussed in [MBT07]. The problem impacts the accuracy and reliability of the simulation system. So rather than poorly indexing the vertices/triangles as the proximity of the object, we generate one sample point in one orientation and save the sampled distance data into LUT.

The suitable objects for this data structure are spherical objects and near-spherical objects, which does not have to come with a perfect spherical surface as long as the points on the surface can be configured with unique spherical coordinates. In this document, we use the term **near-spherical object** to classify the suitable objects for our system. The shapes of near-spherical objects (Figure 29 b) are close to a sphere. A more general suitable object type is the star-shape objects (Figure 29 c) if inside the object there is a point visible to all the points on the object surface. But for the ball-and-socket joint in our research, star shape is not realistic so that near-spherical object is the type we aim on in this document.

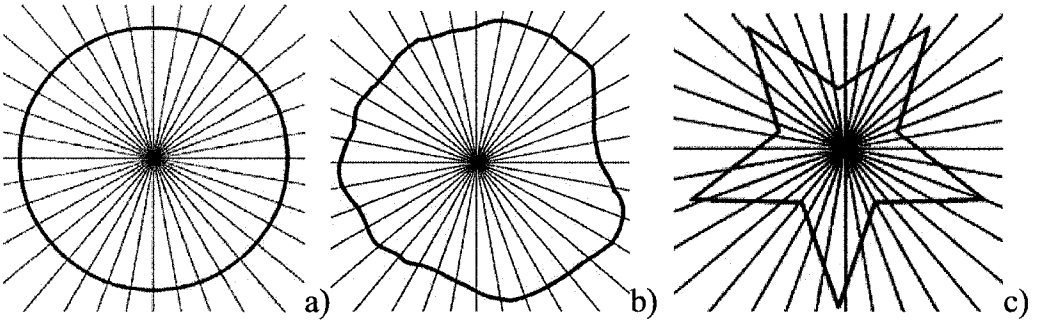


Figure 29: a) spherical object; b) near-spherical object; c) star-shape object

The ball-and-socket joint, such as hip joint, is usually represented by near-spherical objects. In our design, we configure the sample points on the near-spherical object using their spherical coordinates. So we just need one parameter, the distance (or generally called the radius), to represent the surface geometry

information in each orientation. In the traditional sampling way using Cartesian coordinates, we usually need to save three parameters (x, y, and z) to refer the surface. Sampling the near-spherical object in the spherical coordinate system can reduce the space complexity and the memory cost.

3.2 Rapid Spherical Sampling Method

Any vertex v on the model mesh is located on a corresponding ray of (ϕ, θ) and the distance value d can determine the exact position of v on this ray. In order to get the distance data from the generated intersection points (sample points), one common method is to scan and test each sampling ray against points inside area of all triangles to find out the exact interfered triangle with the ray and then sample the distance, which usually is a naive *O(m*n)* algorithm to test m sampling rays against n triangles.

The naive sampling method may be a basic solution but is definitely too inefficient. The reason we also need to improve the performance of the sampling algorithm (the major part of preprocessing step) is this algorithm determines how efficiently and accurately the model geometry information is referred in the simulation step to detect the impingement. With a coarse sampling method, the geometry information would be poorly saved with great errors and would influence the accuracy of the simulation results; with an inefficient sampling method, it will take a long time to setup or change the simulation settings every time so that the practicability of the system would be low. Therefore, the geometry information of the model needs to be rapidly and accurately sampled.

In our algorithm, we sample the distance data in a different approach. Rather than using the naive $O(n^2)$ exhaustive pair-wise testing on models with n vertices, we developed an $O(n)$ method to rapidly interpolate the distance data and then save the data into LUT.

First, a 2D spherical coordinate grid is created to guide the sampling and filling the LUT. The idea is similar to create a 2D texture of Earth on the 2D latitude and longitude grid. We treat the spherical coordinate ϕ as the abscissa on a 2D grid and treat the spherical coordinate θ as the ordinate. In this way, a 3D mesh in the Cartesian coordinate system (Figure 30 a) is mapped to the 2D spherical coordinate grid (Figure 30 b). The dimension of the problem is now converted from 3D to 2D.

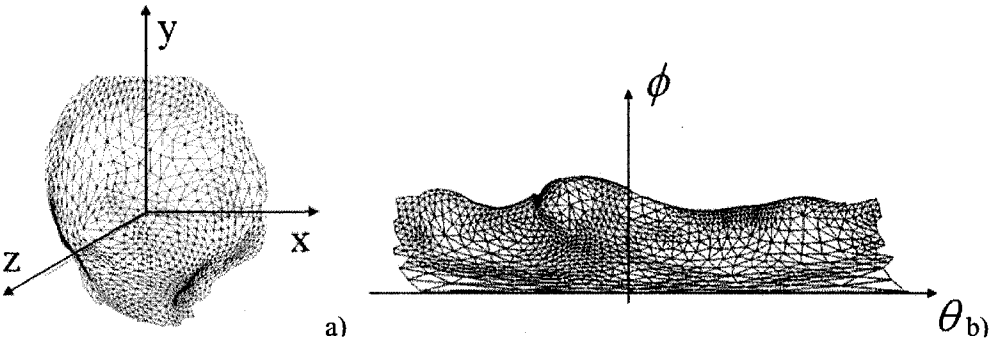


Figure 30: Mapping mesh step in the sampling step: a spherical mesh in 3D Cartesian space is mapped to the 2D spherical coordinate grid

Next, we treat the distance value d of each vertex computed in the 3D Cartesian space as the grey value of the vertex. Every mapped vertex is assigned a grey value representing the distance value (Figure 31 a). Now given a specified orientation, we can gain the corresponding distance value from this grid through interpolation. Since we know the spherical coordinates of all three vertices of a triangle and also have the vertices' distance values, we can compute the distance value of an arbitrary sample

point through triangle interpolation based on the sample point's spherical coordinates. The method we choose is the rasterization technique to rapidly interpolate the distance data corresponding to the orientations of the sample points within each triangle's region.

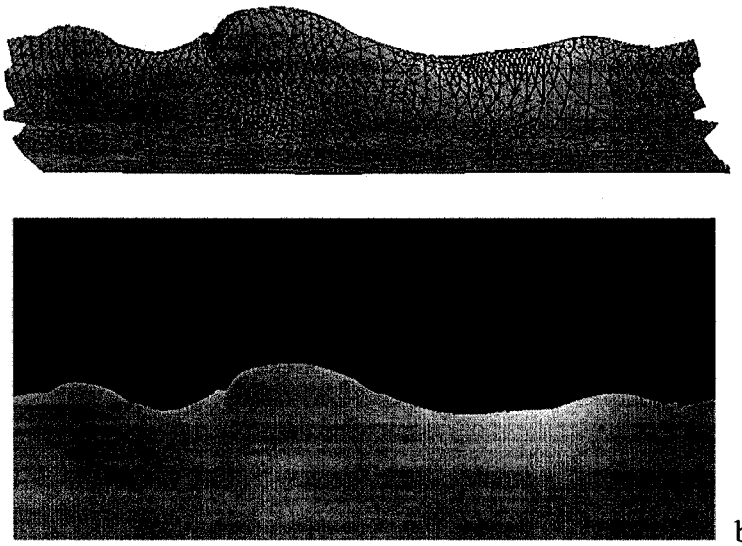


Figure 31: Filling LUT in the sampling step: a) the mapped socket model on the 2D spherical coordinate grid where ϕ is y axis and θ is x axis. Grey level represents the normalized distance data; b) the 2D LUT at precision=0.1° rasterized from the mapped model.

In imaging processing, rasterization is used to convert a 2D object into raster image which consists of pixels in rows and columns. The size of the pixels determines the final quality of the image. With smaller pixels, the image has a higher resolution (Figure 32).

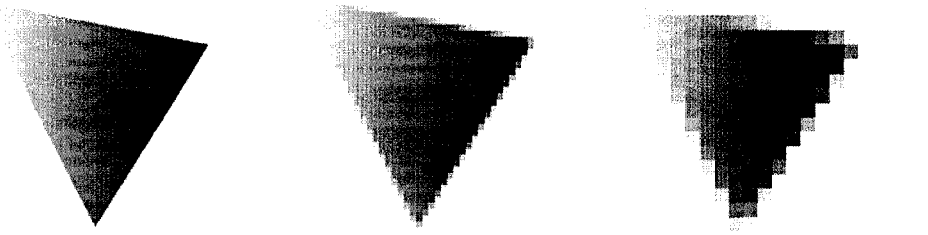


Figure 32: Triangle rasterization: a) high resolution with small pixel size; b) middle resolution with middle pixel size; c) low resolution with large pixel size

Since we have mapped the triangles on the 2D grid and treated the vertices' distance data as "grey values" (Figure 31 a), we can consider the grid cells as the "pixels". The "grey values" of these "pixels" within a triangle can be rapidly interpolated from the same rasterization method.

Through triangle rasterization, the distance value d in a certain orientation (ϕ_i, θ_i) can be quickly solved by rasterizing the corresponding triangle. The distance values interpolated from all triangles on the 2D spherical coordinate grid can be visualized as a grey scale distance map, which covers all spherical coordinates the object occupies. This 2D grey scale distance map can be understood as an LUT while each pixel is an LUT element. In other words, the LUT can be efficiently filled by solving this 2D grey scale distance map based on the triangle rasterization (Figure 31 b).

In this sampling design, there are three exceptional cases in the mapping step. The common issue of these exceptional cases is the error mapping of the triangles containing the poles.

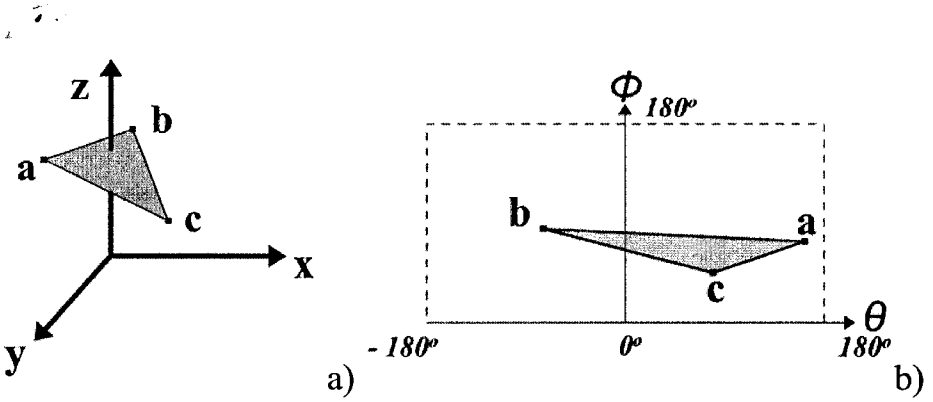


Figure 33: Case 3: a) the triangle *abc* in 3D Cartesian space; b) the direct mapping does not correctly cover the corresponding orientations on 2D spherical coordinate grid.

For example, the triangle *abc* in Figure 33 a) contains the pole (here is the intersection point with z axis) and shall contains all the sample points with spherical coordiante $\phi = 0°$. But the directly mapped *abc* in Figure 33 b) does not cover the area where $\phi = 0°$. So the sample points with spherical coordinate $\phi = 0°$ cannot be properly interpolated in this example.

The three cases are categorized by three appearances of the pole: as a vertex, on an edge, or inside a triangle:

Case 1 (called "vertex case"): the pole appears as an endpoint shared by one or more triangles.

Case 2 (called "edge case"): the pole appears on an edge shared by at most two triangles.

Case 3 (called "face case"): the pole appears inside the region of at most one triangle.

The solution to case 1 is: for each endpoint (except the endpoint as the pole) of the involved triangles on the 2D spherical coordinate grid, it is vertically projected to

the line ϕ=0° (or ϕ=180°) on the grid to form the corresponding polygon regions. Then those polygons can be rasterized.

The solution to case 2 and case 3 is to first locate the pole, subdivide the involved triangle(s) at the pole position, and then apply the solution to case 1. For example, Figure 34 illustrates the solution to case 3. The pole is located inside the triangle *abc* in 3D Cartesian space. When the triangle *abc* is mapped to the 2D spherical coordinate grid, it will be subdivide into three quadrilaterals: (b c p_1 p_0), (c a p_2 p_1), and (a b p_0 p_2), where p_0, p_1, and p_2 represent the pole p (ϕ=0°) and are the projections of *b,c, and a*. Then these polygons form the correct mapped region of the triangle *abc* on the grid and can be rasterized to generate the sampled data as usual.

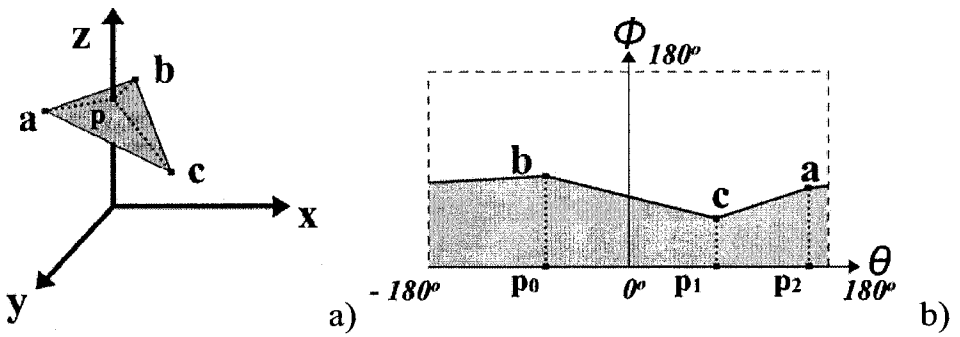


Figure 34: solution to case 3 and the correct mapped triangle region. p is the pole mapped as p_0, p_1, and p_2 which are also the projections of *a, b*, and *c*.

The complexity of our sampling method is *O(n)*: first, the three exceptional cases are handled by scanning n triangles for at most two vertices (for vertex case) and at most four triangles (for edge case and triangle case); then the rest of triangles, up to n triangles, are mapped to the 2D spherical coordinate grid and rasterized to fill the LUT.

Algorithm 1: the RSID system, sampling algorithm for near-spherical object

Handle the three extreme cases

For i=0; i <number of model's triangles; i++ {
    For each vertices of ith triangle {
        Convert mesh vertex (x, y, z) to (θ, ϕ, d)

        Treat θ as abscissa and ϕ as ordinate

        Map that vertex to the 2D spherical coordinate grid
        Convert d as the grey value of that vertex
    }
    Rasterize that mapped triangle and fill LUT with interpolated distance values
        from barycentric coordinates
}

3.3 Rapid Spherical Impingement Detection and Surface-to-Surface Distance Measurement

In the preprocess step, we need to create one LUT for the ball model and one LUT for the socket model. In our ball-and-socket joint simulation, we need to process impingement detection twice to avoid inter-penetration artifacts: the ball's vertices shall be tested against the socket's LUT as the first scan; the socket's vertices shall be tested against the ball's LUT as the second scan. The two impingement detection scans carry the same algorithm so that in the following paragraphs we only go through one scan (e.g. the ball's vertices against the socket's LUT) to explain the idea of impingement detection algorithm.

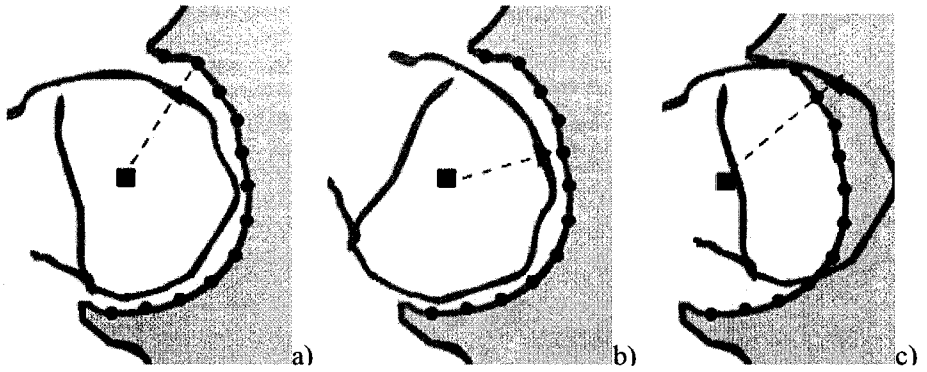


Figure 35: Examples in impingement detection: a) comparing a vertex (black triangle) of the ball object against the sample points (black dots) on the socket; b) the ball object has rotated so that the orientation of the vertex has changed; c) the ball object interfered with the socket object

As an example, we process the test of the ball's vertices against the socket's LUT (Figure 35 a). In the previous sampling step, the socket object was sampled through generating the sample points (black dots in Figure 35) on the socket. The distances (dash lines in Figure 35 a) were measured from the sample points to the system origin (black rectangle as the socket's local system origin in Figure 35). Then the distance data were saved in LUT as the proximity information of the socket object. Now in the simulation step, we are given a target vertex v on the ball object. We find the coordinates of v in the socket's local coordinate system and then the orientation of v is computed. In the orientation of v, we check out the corresponding sampled distance data from the socket's LUT to compute the distance difference (Figure 35). If the ball object rotates or moves (Figure 35 a), we simply find the new orientation of the vertex v and do the same query procedure. If the distance difference shows the vertex is on the surface or has interfered with the socket surface (Figure 35 c) as an extreme theoretical example), then there must be a penetration in that orientation. In this way,

we can efficiently detect the impingement based on the distance difference of two objects in the same orientation.

The reason we need to do the computation in socket's local coordinate system is that we need to find the relative positions between the objects. So we do not need to sample the socket again and update the socket's LUT once the socket moves or rotates in the world coordinate system. For example, when the entire joint is moving, the relate positions between the ball and the socket do not change.

So the above example of processing the test of ball's vertices against socket's LUT can be concluded in theory: we use Eq. 5 to compute the indices (i.e. ϕ_i, and θ_i) of a target vertex v on the ball, check out the socket distance d_i from LUT[ϕ_i][θ_i], and compute the distance d_v of the vertex v to the same spherical coordinate system origin. The impingement detection can be done instantly by comparing the difference between d_i and d_v to check whether the vertex is inside or outside of the socket surface. The complexity of our impingement detection algorithm is $O(n)$ where n is the number of the ball' vertices to be tested.

Since the impingement detection is processed based on the distance difference, it is easy to save the distance differences of all vertices for the surface-level estimation. We can check how close two surfaces are in all orientations instead of just getting the traditional report of a few collided points. The distance difference can be further referred to as the approximate strain between the surfaces.

Algorithm 2: the RSID system, impingement detection algorithm for near-spherical objects (per iteration)

For i =0; i <number of model B's vertices; i ++ {

Convert vertex B_i into model A's local spherical coordinate system;

Compute distance data d_i from B_i to the system origin;

Compute spherical coordinates $[\phi_i, \theta_i]$ of B_i;

Compute interpolated distance data d from A's LUT at $[\phi_i, \theta_i]$;

if $d <= d_i$ {

Impingement detected at that vertex;

$\Delta d = d - d_i$;

Save Δd as the surface-to-surface distance in the orientation of $[\phi_i, \theta_i]$;

}

else {

There is no impingement;

}

Notice that, in order to avoid inter-penetration artifacts, Algorithm 2 must also be applied to test model A's vertices against model B's LUT. For instance, we need to also process a test of socket's vertices against ball's LUT after the above example of processing the test of ball's vertices against socket's LUT.

3.4 Validation and Results

All tests were conducted using an AMD Opteron Processor 252 at 2.6 GHz with 2GB of memory.

The visualization system **Joint3D** was developed as the implementation to demonstrate our collision detection/proximity calculation system introduced in this thesis. The visualization was implemented with OpenGL® (Open Graphics Library). The user interface and the controllers are implemented with MFC (Microsoft® Foundation Classes). The programming language used is C++.

The basic function of Joint3D is to control the motion of the hip joint, to detect the impingement, and to visualize the surface-to-surface distance measurement.

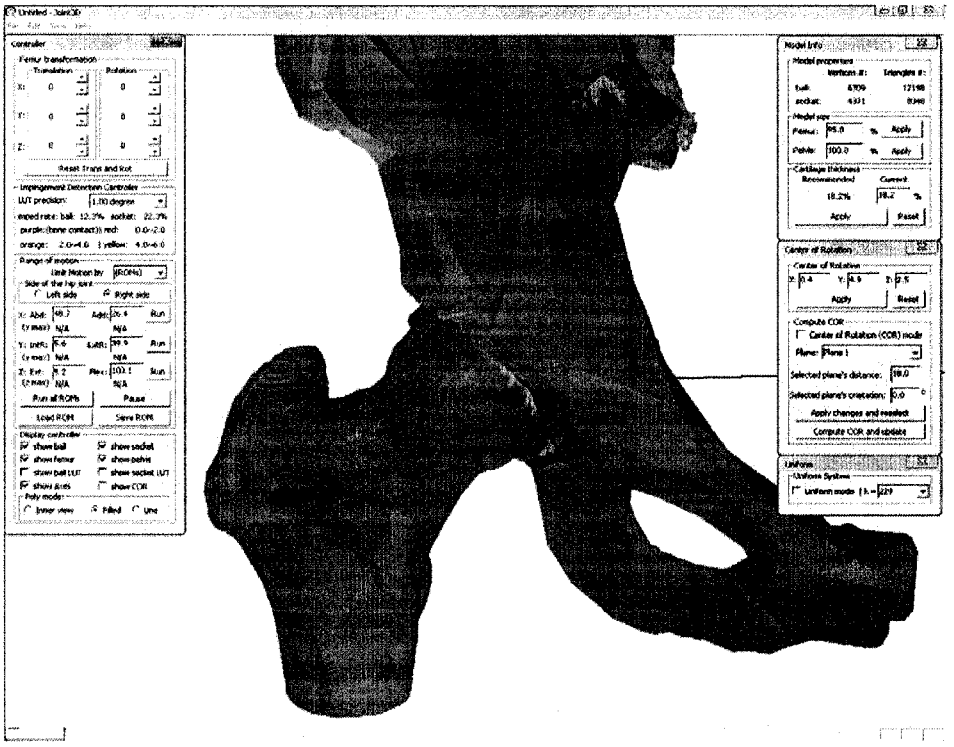


Figure 36: Joint3D: A right hip joint is displayed from anterior view.

The RSID system is first applied to several sets of low-resolution generic models from 300-triangle level (each model has about 300 triangles) to 3k-triangle level (each model has about 3k triangles). The original generic models were gained from TurboSquid, an online 3D model library [TurboSquid].

The RSID system is then further applied to one set of real models at 12k-triangle level (each model has about 12k triangles) (Figure 37) to validate the practicability and reliability. The models are segmented from the CT data of an FAI case. The patient is a male, around 40 years old, and has FAI on his right hip joint. As presented in the introduction section (Figure 1), there is a modeling step which models the

proper 3D models from CT data. Since the research on the modeling is not made at current research stage, we have very limited modeling solutions and currently use a 3rd party segmentation tool called ITK-SNAP [ITK-SNAP] to process the modeling. The exported 3D models are in triangulated mesh form. The current modeling method is not flexible and the resolution is not controllable to the user. The problems of how to flexibly rebuild the 3D models and how to control the resolutions are the problems to be solved in the future research.

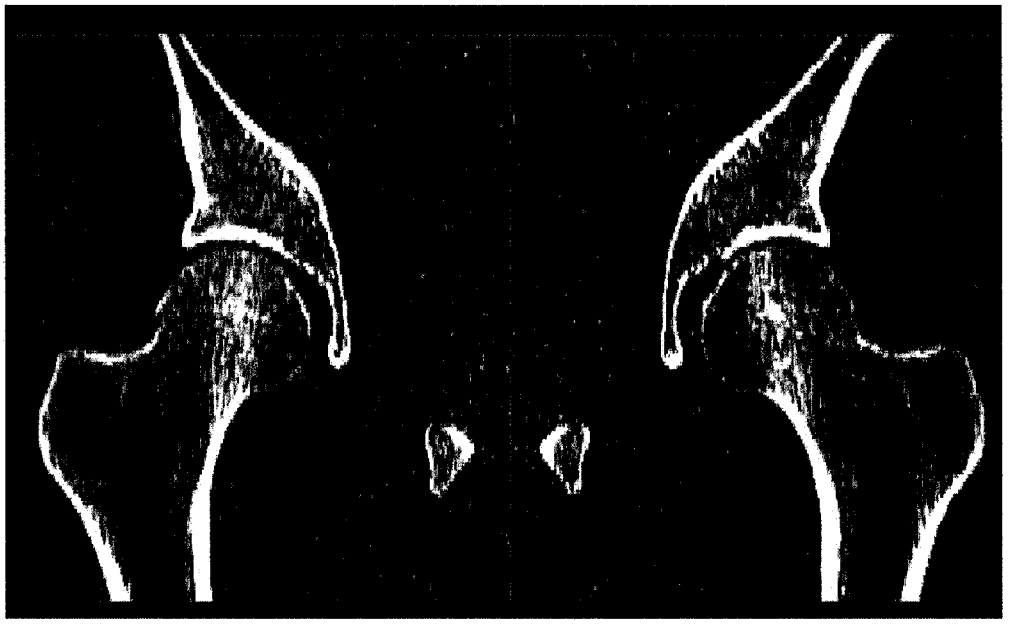


Figure 37: The patient's pelvis region is scanned into a set of 147 CT slices (1.25 mm thickness, resolution of 512*512), anterior view.

For the models built from CT data, each of the femoral head region and the acetabulum region has about 12k triangles for simulation purpose. The rest part of the femur and the pelvis are roughly rebuilt for visualization purposes only. The models were setup based on their original relative coordinates in the CT data. The origin of the RSID system to build the Cartesian coordinate system and the spherical coordinate system is set at the **center of rotation** (COR) of the femoral head. As presented in the

introduction section (Figure 1), COR is a research to be made as one of the parameters to setup the motion simulations. In current community, many researches consider COR as the center of the femoral head and treat COR as a fixed point during the joint motion. In this thesis, we do not further research or analyze the COR but consider COR as the geometric center of the femoral head model as the common solution at current research stage. It is gained from the statistical analysis of the femoral head's vertices based on the least squares fitting: first, the most spherical region on the femoral head is selected (Figure 38 a).We manually use several planes to define the selection volume and select the most spherical femoral region inside the volume; then, the least squares fitting algorithm is applied to compute the geometric center and the radius of the femoral head (Figure 38 b and c). In Figure 39, a color scale is used to visualize the difference between the model surface and the fitted sphere.

The Least Squares Fitting algorithm [Eberly99] we choose is to fit a sphere to 3D points by minimizing an energy function:

> "Given a set of points $\{(x_i, y_i, z_i)\}_1^n, n \geq 4$, fit them with a sphere $(x-a)^2 + (y-b)^2 + (z-c)^2 = r^2$ where (a, b, c) is the sphere center and r is the sphere radius. An assumption of this algorithm is that not all the points are coplanar. The energy function to be minimized is $E(a,b,c,r) = \sum_{i=1}^{m} (L_i - r)^2$." [Eberly99]

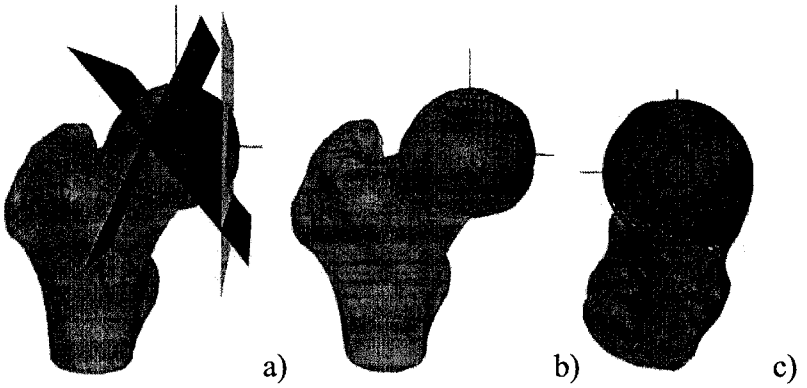


Figure 38: Fitting the femur head with sphere to locate the geometric center: a) use planes (in brown) to select the spherical region (in purple) to fit; b) the fitted sphere illustrated as black wires, anterior view; c) the fitted sphere, superior view

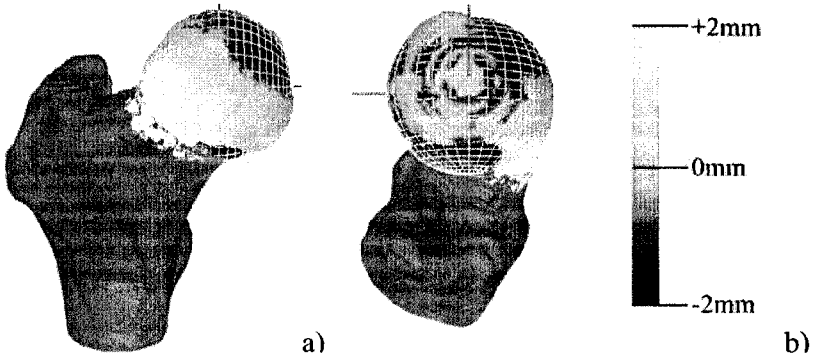


Figure 39: Color scale of the difference from the model surface to the fitted sphere: a) the fitted sphere as white wires, anterior view; b) the fitted sphere, superior view.

Four sampling precision levels are set up from 1° to 0.05°. In theory, our sampling method can reach much higher sampling precision but in practice the hardware is not affordable to display the large number of sample points more than 26 million (at 0.05° sampling precision). The LUT sizes and the corresponding memory requirements for each sampling precision in the RSID system are listed in Table 2.

Table 2: The LUT sizes and the memory requirements of the RSID system (in double) under 4 sampling precision settings.

Precision	Size of 2D LUT in the RSID system	Memory requirement
1°	(180/1+1)*(360/1)=65,160	0.497MB
0.5°	(180/0.5+1)*(360/0.5)=259,920	1.98MB
0.1°	(180/0.1+1)*(360/0.1)=6,483,600	49.5MB
0.05°	(180/0.05+1)*(360/0.05)=25,927,200	197.8MB

Our sampling method has an efficiency of $O(n)$ and makes the preprocessing step much shorter. Unlike the traditional sampling methods, it generates sample points directly on the surface and then gains the desired surface distance data (Figure 40). Compared to the existing methods such as the $O(n^2)$ sampling algorithm, our method can reach a much higher sampling precision with a much lower computation time cost and is more accurate (Table 3).

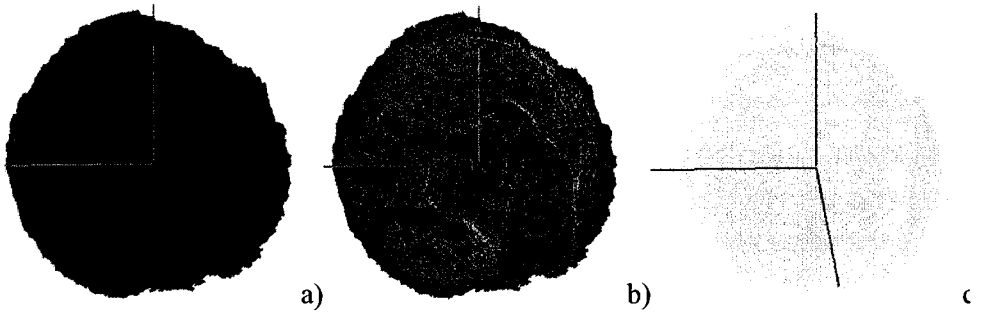


Figure 40: Directly generated sample points (in green): a) the socket (in blue), right lateral view; b) the sample points and the socket; c) the sample points.

Table 3: Benchmark of 2 sampling methods on 3k-triangle level generic models only (in millisecond)

Precision	**Pair-wise testing** $O(n^2)$	**Our sampling method** $O(n)$
precision=1°	176,328 ms	15 ms
precision=0.5°	700,844 ms	16 ms
precision=0.1°	17,722,156 ms	62 ms
precision=0.05°	N/A	187 ms

Our impingement detection method has an efficiency of $O(n)$. In the hip impingement detection tests from 300-triangle level to 3k-triangle level in Table 4, our method is faster than the well-known RAPID [GLM96] which is an efficient OBB-tree based collision detection method. In high precision simulations, our method is slower than RAPID but can be processed in real-time. Notice that this comparison is a speed-only test against the methods (RAPID as the chosen example here) aimed at speed, which is used to show the efficiency level of our algorithm. Besides the speed, RAPID is not able to provide distance information between two surfaces while the benchmark of our method also includes the computation cost of the distance information. More comparisons against RAPID and related methods are in the following comparison section.

Table 4: Benchmark of 3 impingement detection methods on 4 different triangle levels (in milliseconds).

Triangle level (per model)	**Pair-wise testing** $O(n^2)$	**RAPID** [GLM96]	**Our impingement detection method** $O(n)$
Generic model (300-triangle)	30.7 ms	1.41 ms	0.06 ms
Generic model (1k-triangle)	522 ms	1.41 ms	0.18 ms
Generic model (3k-triangle)	2,831 ms	1.43 ms	0.48 ms
Real model from CT (12k-triangle)	N/A	1.5 ms	5.7 ms

Distance information between two surfaces in RSID is further referred to as the approximate strain. We use color representation to show different level of the impingement and the strain distributed on the model surface. Here we apply a uniform distance (4mm) as a virtual offset to each bone's surface to represent the thickness of the cartilages. If the distances between the bones are greater than the cartilage thickness, then we say there is no impingement and use green (used by the ball) or blue (used by the socket) to indicate that area; otherwise, we set four levels of colors (yellow, orange, red, and purple) to indicate different levels of the distance between the bones to represent the penetration depth between two virtual cartilage layers. If the distance is zero (physical touching between two bones), purple is used to indicate that region. So the five color levels are used to specify the distances and represent the approximate strain distribution.

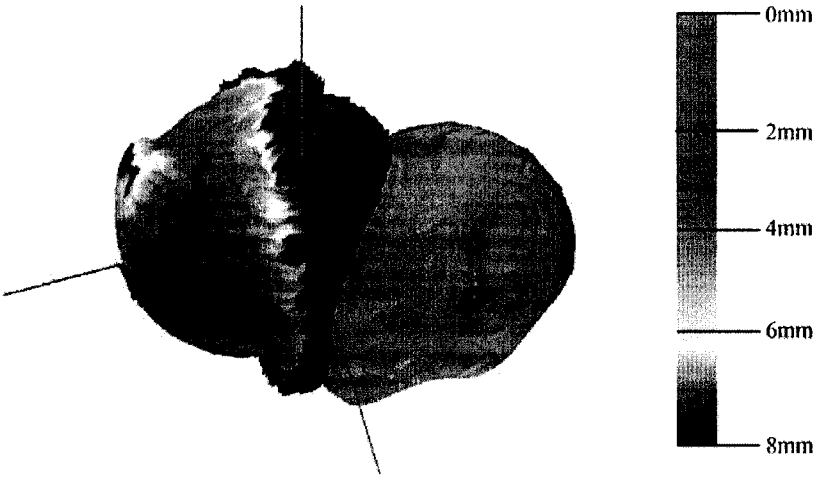


Figure 41: Color representation of the surface-to-surface distances, which is used to represent approximate impingement level. Notice the femoral head here is purposely moved forward to intersect with socket to demonstrate the colors.

The impingement detection and the strain distribution from the RSID system are applied to the real model segmented from the CT data. The time cost of impingement detection is listed in Table 4, which proves the efficiency. The result is illustrated in Figure 42.

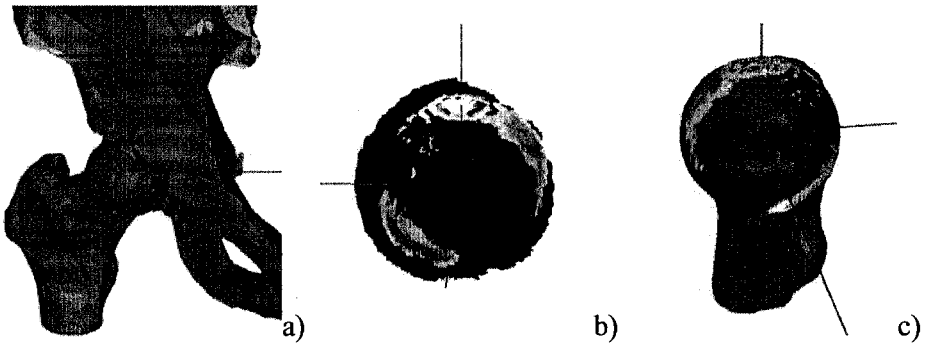


Figure 42: The strain distribution as the impingement detection result from the RSID system is drawn on both models: a) the right hip joint, anterior view; b) the socket, right side view (right lateral view); c) the femur, left medial view.

The impingement detection result can also be validated by being compared to the results from other research groups. According to the research of Kubiak-Langer et al. [KTM*07], "the impingement zones were localized in the anterosuperior quadrant of

the acetabulum for the impingement groups" and "did not differ when comparing the control with the femoroacetabular impingement group" (Figure 43). The result from our RSID system illustrated in Figure 42 shows that impinged regions are at 12~4 o'clock and 8~9 o'clock, which matches with their conclusion.

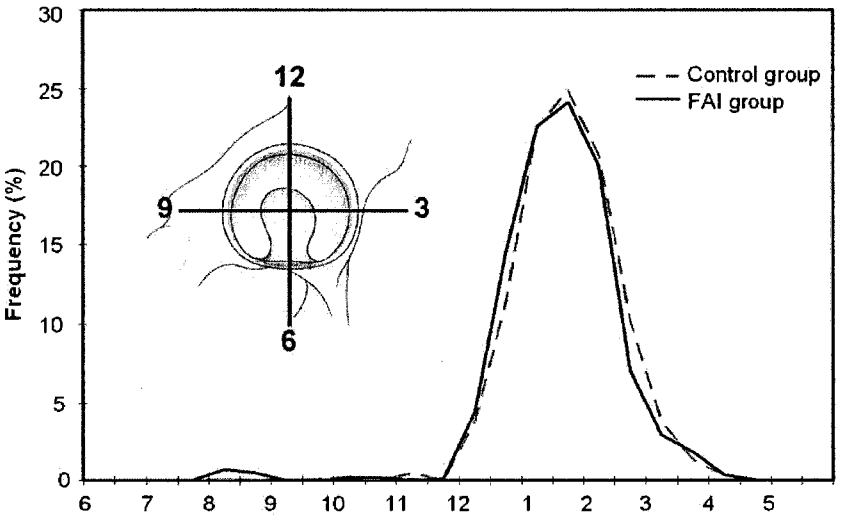


Figure 43: Similar distribution of impingement zones between the normal and the impingement groups is shown (reproduced from [KTM*07]). Both were localized in the anterosuperior quadrant of the acetabulum.

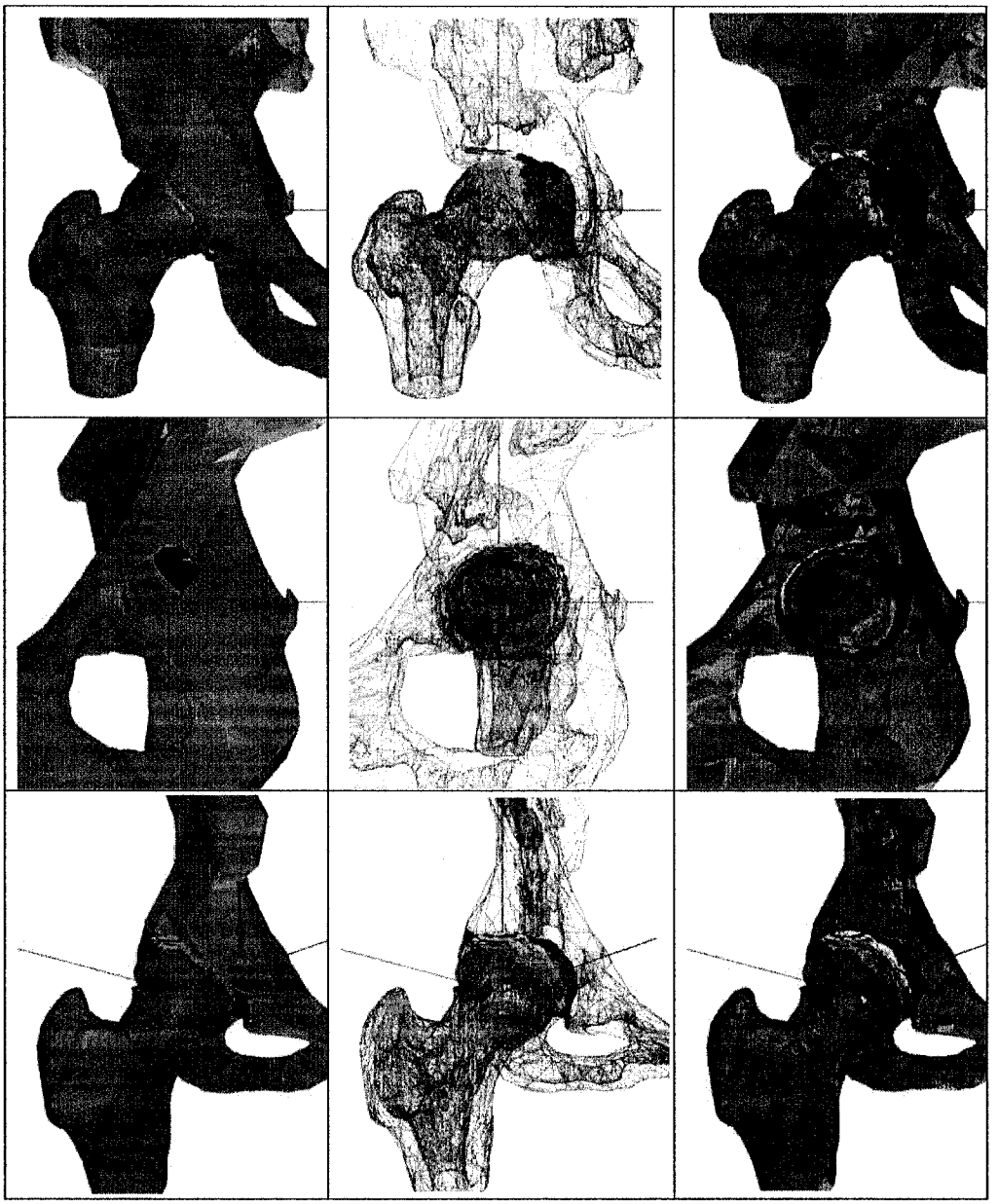


Figure 44: Different visualizations and views to observe the models and the impingement results. Three columns from left to right list three visualization options: filled polygons, lines, and blending. Three rows from top to bottom list three views: anterior view, left medial view, and customized oblique view.

3.5 Sampling Error Analysis

In the following sections, we will discuss about the sampling error of our sampling method. In section 3.5.1, we consider the hip joint as a mathematical model;

in section 3.5.2, we consider the hip joint as a sliced model which is a common medical data format such as CT data format and MRI data format.

The discussion here is more about theoretical discussion on selecting the sampling precisions directly based on the hip joint data analysis because in theory and in practice actually there is a modeling step (see Figure 1 for the process pipeline) before the CT data can be used in our impingement detection system. Our system doesn't directly input the original scan data but reads in refined specified 3D models from the modeling step, which is a common procedure for medical simulations in current community. Therefore more scientific and realistic analysis on the sampling error can be made after the modeling system is properly researched and solved in the future. Our sampling precisions can be flexibly adjusted and defined according to the real output from the modeling system. More discussions on the modeling are in the Discussion section.

3.5.1. Sampling error analysis based on mathematical model

In this section we analyze the sampling error based on the general mathematical model of the target object, without assuming the actual data format of the object to be saved (e.g. CT voxelized data or triangulated meshes). The target object here is theoretically considered as a mathematical continuum, such as a sphere. The sample points generated in the sampling step form the proximity of the original mesh. Such proximity based on sample points (also known as point cloud) is a common approach in the modeling area. The sampling of RSID system is non-uniform so that the

sampling density is lowest at equator and is highest at the poles. The maximum (absolute) sampling error is related to the perimeter of the hip joint's equator:

$$Maximum\ sampling\ error = \frac{2\pi r}{\#of\ samplePoint} * \frac{\sqrt{2}}{2} = \frac{\sqrt{2}\pi r}{\#of\ samplePoint}$$

Eq. 7

The maximum sampling error shows a linear increase with the radius but its relative sampling error to the radius is constant:

$$Relative\ error = \frac{Maximum\ error}{r} = \frac{\sqrt{2}\pi}{\#of\ samplePoint}$$

Eq. 8

As an example, in our simulation we use the hip joint as the target object. The number of the sample points at the equator is from 360 to 3600 and the femoral head's diameter is around *44 mm* [SRLR07]. We can compute the maximum sampling error and relative sampling error for each precision case.

Table 5: maximum sampling errors of RSID system under 4 precision settings

Precision	**Maximum Error**	**Relative error (against radius)**
1°	0.272 mm	1.23 %
0.5°	0.136 mm	0.62%
0.1°	0.027 mm	0.12%
0.05°	0.014 mm	0.06%

In this section, we made an analysis of sampling error based on mathematical model. As an example, we studied the hip joint with its mathematical shape and the precisions equal to or smaller than 0.1° give acceptable sampling errors in theory.

3.5.2. Sampling error analysis for sliced data format

In common medical data format, the object data is usually scanned in slices. First, we setup the coordinate system in the object's space: the x-y plane of the coordinate system (Figure 28) is set to the middle plane of the space containing the object and the slices are perpendicular in z-axis (Figure 45 a). The origin is set to the center of the space. The slice ID increases when the slice is farther from the middle plane. A 2D view (Figure 45 b) can help to understand the setup and the analysis. Since the object data is saved on each slice, we need to adjust the sampling orientation in order to capture the data on each slice. The thickness of the slices is ΔT. As illustrated in Figure 45 b), for ith slice we find ϕ_i as the sampling orientation with $\phi_i = \arccos(i * \Delta T / d)$, where d is the radius and $i * \Delta T$ is the distance of ith slice to the middle plane.

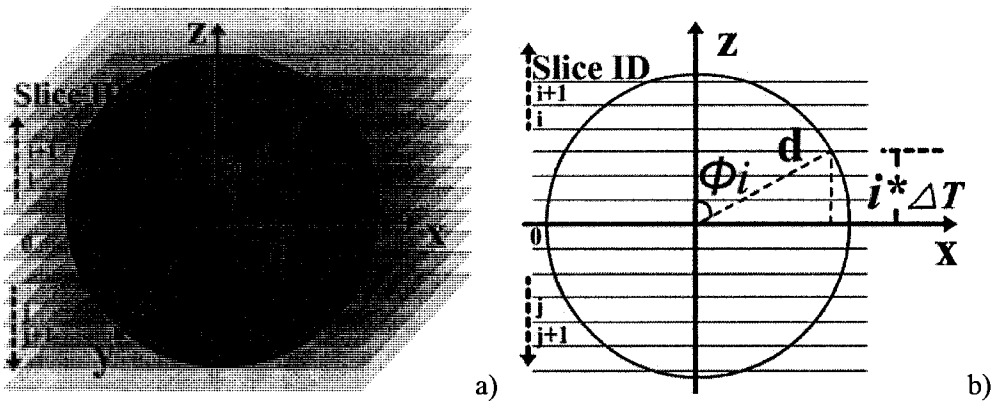


Figure 45: a) a sphere contained in a sliced space; b) a sphere contained in a sliced space, 2D view.

The sampling precision is determined by the smallest orientation gap $\Delta\phi$ where $\Delta\phi = \phi_{i+1} - \phi_i$. As illustrated in Figure 46 a), the tangent of the arccosine curve is increasing, which means $\Delta\phi$ increases for the slices farther from the middle.

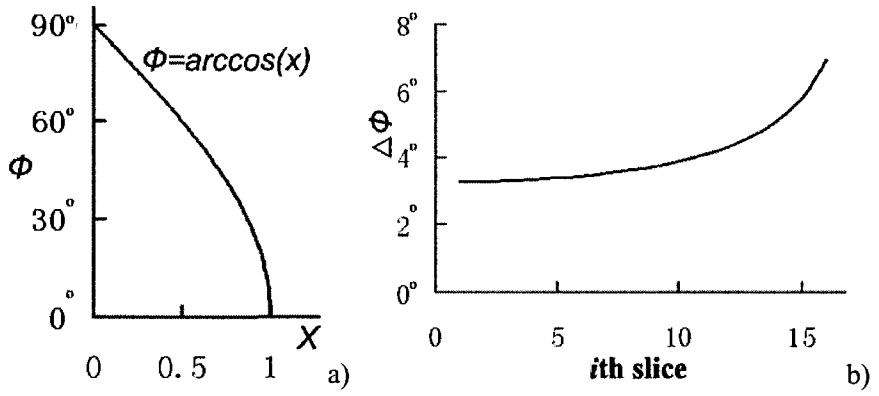


Figure 46: a) the arccosine curve; b) $\Delta\phi$ increases when the slices are further from the middle.

As an example, we analyze the CT data used in our simulation. The slice thickness of the CT data is 1.25 mm. Since the femoral head's diameter is around *44 mm* [SRLR07], there are around 35 slices. So there are 17 slices above the middle plane and 17 slices below. $\Delta\phi$ between every two slices is computed and is listed in Figure 46 b). We can see that $\Delta\phi$ increases when the slices are farther from the middle plane as predicted in Figure 46 a). The smallest $\Delta\phi$ is between the 1st slice and the middle plan and is about 3°, which is a reference to choose the sampling precision.

For this particular analysis on sliced object in this section, we may use Shannon sampling theorem to determine the final sampling precision. Shannon sampling theorem indicates that "if the function is sampled at a rate equal to or greater than twice its highest frequency, it is possible to recover completely the original function from its samples" [RGR02].

So after we know the smallest $\Delta\phi$ is 3°, we need to double the sampling frequency. So the sampling precision need to be equal to or smaller than 1.5°.

In this section, we made an analysis of choosing a sampling precision for the sliced data format. As an example, a sampling precision of 1.5° or smaller shall be chosen for a CT data with 1.25mm slice thickness.

3.6 Comparison

This section presents some discussions and comparisons to the existing methods to show the features of our system. Notice that, the collision detection survey [TKH*05] states exact comparisons among the existing methods are not possible and "there exist no general or optimal approach" due to many reasons: different input/output data requirements, designed for different purposes, restricted to specific software/hardware support requirement, or the availability of the source codes [TKH*05]. Therefore, the comparisons here are intended to provide a theoretical discussion on the computational complexities.

3.6.1. Comparing to BVH Based Approaches

BVH based collision detection methods have good computational efficiency. RAPID [GLM96], a classic BVH based collision detection method, is compared with our method in the experiments. The results show BVH has constant complexity regardless the number of triangles of the model and can get better collision detection performance than ours when the model contains a great number of triangles. To determine the existence of collision on the surface, BVH is a better choice for high resolution models; however, current BVH based approaches do not provide the

important distance information we are interested in between two surfaces, while our system can solve the problem without increasing the overall computation cost.

As explained in the Literature Review, the efficient BVH based collision detection is actually a fast search process for collided points. Through the paths of collided nodes along the BVH tree, the collided points at the leaves can be efficiently marked. The problem domain for BVH methods is not to compute distance information between objects but to efficiently locate the collided points. The current research on BVH regarding distance information is limited in scope, such as the computing of minimum or maximum distances between two objects [ZKM07]. The other collision detection category, distance fields, can better handle the problems than BVH. To compute the surface-to-surface distance information based on BVH, some rules of matching points from two objects must be applied; otherwise, the point from one object can be matched to any point from the other object to compute some kind of distance. For example, in our strategy we selected (or interpolated) points that appear in the same orientation (have the same spherical coordinates) as the matching rule to compute the distance between two near-spherical objects' surfaces. BVH methods need to search throughout the tree every time to figure out a corresponding point on the other object, while we simply check out the corresponding point from LUT in one step. Therefore, the computation cost of BVH to compute the n-point object's distance information is $O(nlogn)$ while our method is $O(n)$, which shows BVH based methods are inefficient and not suitable for our research.

Distance information in our research has been considered an important direction to study the impingement level between the bones' surfaces. So, BVH may be faster to examine the existence of collisions but is much slower to provide the necessary distance information in which we are interested.

3.6.2. Comparing to Distance Field Based Approaches

As mentioned in the previous sections, our methods belong to the category of distance field based approaches: the surface of the near-spherical object is converted into a distance shield in the spherical coordinate system. The common distance field based approaches usually have an expensive algorithm to generate the fields as the sampling step, which usually limits the fields in small resolutions due to the high computation and huge memory cost. Also, the methods generate the fields only in Cartesian space.

In comparison to the common distance field with an $O(n^3)$ space complexity, our methods generate the field on a 2D spherical coordinate grid with an $O(n^2)$ space complexity. Since the complexity of the problem is decreased with one order of magnitude, our systems can generate higher resolution fields with lower computation and less memory cost. Therefore, the advantage of our systems, in comparison with the general distance field based approaches, is mainly the rapid sampling step to generate the distance fields. The collision detection is done in the similar way by testing whether the target is inside or outside the field. The efficiency of collision detection algorithm has little difference.

3.6.3. Comparing to Existing Hip Joint Simulation Approaches

Hu et al. [HLL*01] and Kubiak-Langer et al. [KTM*07] [TKL*07] use LUT to accelerate the collision detection process in hip joint impingement simulation. Their method creates an LUT to represent a 3D space and only consider the collided points as the detection result. In comparison to their approaches, our systems have a lighter LUT structure based on a 2D grid, which requires less memory and provides higher sampling resolution. Organizing the geometry information based on spherical coordinates is more suitable to near-spherical objects. Also our systems provide surface-to-surface distance information as approximate strain for more realistic impingement simulations than their collided-points-only results.

The design of the impingement detection step in the RSID system has some similarity to the collision detection method proposed by Maciel et al. [MBT07] based on spherical distances; however, they did not solve the overall practicability of their system with inefficient and inaccurate sampling methods and data structure. The authors only tested with small-size LUTs (low sampling precisions) from 10*10 (precision of 1.8°) to 400*400 (precision of 0.45°) from off-line computations while our tests can easily reach 7201*3601 (precision of 0.05°) in real-time and is able to reach much higher precision without any limit on the real-time requirement. Their inefficient sampling is not the only problem of their system design. Referring one triangle to one orientation is not a good sampling strategy. For a sampling precision of 0.45°, the range of 0.45° is considered the unit orientation (unit area) corresponding to

a sampling ray. Many triangles may appear in the same unit orientation. Therefore, even in their highest precision 400*400 test, there are many triangles unable to be sampled. In this case, only one triangle may be referred and the other triangles in the same unit sampling orientation have to be ignored. This triangle capturing error they recognized shall be considered all the time, while our method samples all the triangles through the 2D grid to generate sample points without missing a single triangle. Their inefficient and inaccurate sampling method then results poor model geometry information in addition to a high computation cost.

In the impingement detection step, the author used triangle interpolation to do a very accurate vertex projection on the target triangle. But with a very low sampling precision and a poor sampling strategy, such effort on the accuracy of the impingement result becomes meaningless. Also, their impingement detection method claims to be very efficient; however, spending lots of time on a very expensive $O(n^2)$ sampling method to prepare a fast impingement detection deducts the overall efficiency of the system. In addition, their impingement detection still considers a few physically collided points between two objects as the results, which belongs to the traditional strategy as discussed above.

The defects in the sampling and the impingement detection directly impact the practicability, the accuracy and the reliability of their entire system.

Therefore, comparing to [MBT07] our RSID system solves the sampling step rapidly and accurately as well as the simulation step. Our system can reach much higher precision within a lower computation cost, which benefits from the overall

design of the system. Furthermore, our impingement detection algorithm is more accurate and also provides useful surface-to-surface distance measurement to represent the approximate strain. In addition, our further-developed uniform version, introduced in the next chapter, improved our ideas.

As a general conclusion to the comparisons in this area, most of the existing up-to-date impingement detection methods are not as efficient as ours or only consider the collided points as the detection results. Marking collided points as impingement are actually not scientific by ignoring other components (such as the tissues appearing between the bones) and assuming the bones can physically collide with each other. Also, a few scattered, collided points cannot show the global impingement information on the joint surfaces. In comparison to those joint impingement simulation methods, our system is able to check the entire surface's status besides the collided points as the global impingement examination of the joint at surface level.

Chapter 4 Rapid Spherical Impingement Detection with Uniform Sampling

The sampling method of our RSID system described in the previous chapter generates non-uniform sample points because we use regular spherical coordinate grid to define the orientations of the sampling rays in 3D Cartesian space. The non-uniform sampling method is easy to configure and construct but the drawback is that the sampling density is not consistent. The sampling density increases at the poles; the sampling density decreases at the equator. One consequence of the non-uniform sampling is the exceptional cases discussed in the previous section. There are many sample points squeezed at the pole while only one sample point is actually needed for the pole location. Non-uniform sampling still takes some unnecessary memory to store the geometry information.

The RSID system is then further developed for uniform sample point distribution in 3D Cartesian space, which optimizes the density of the sample points and reduces the memory cost. Both the sampling method and the impingement detection method of RSID need to be redesigned due to a different LUT data structure. The redesigned simulation system is called **Uniform Spherical Impingement Detection** system (**USID** system) with a uniform sampling algorithm and a uniform impingement simulation algorithm.

The overall system structure is similar to RSID system. So the general idea of USID system can be checked in Figure 26 in the chapter of RSID system and the flowchart of USID system can be checked in Figure 27 in the chapter of RSID system.

The design of sample points in USID system is based on several factors inherited from the RSID system design such as:

(i) Traceable property of the points to generate the sample points in the sampling step and save the sampled distance data into LUT. For example, in the RSID system, each sample point has unique spherical coordinates as the LUT indices. The spherical coordinates are the traceable property of a sample point and are used to mark the corresponding position in the LUT to save the sampled data.

(ii) Traceable property of the points to access the LUT in the simulation step and check out the distance data corresponding to a given vertex. For example, in the RSID system, the given vertex also has its spherical coordinates and the spherical coordinates are the traceable property to locate the neighboring sample points to check out the sampled distance data from the LUT.

(iii) Efficiency and accuracy on (i) and (ii). For example, in the RSID system, one LUT query (check-in or check-out) only takes *O(1)* time cost.

(iv) Multiple sampling precisions. For example, in the RSID system, ϕ and θ can be easily configured for different sampling precision.

Nishio et al. [NAKK06] showed that an arbitrary number of points can be approximately uniformly distributed on a unit sphere by configuring a spherical helix curve. The spherical helix curve formula [NAKK06] is:

$\theta = 2k\phi\,(0 \le \phi \le \pi)$ *where* ϕ and θ *are spherical coordinates, k is the number of helix turns* **Eq. 9**

Nishio et al. states that the spherical helix points specified by Eq. 9 can be approximately uniformly distributed on the surface of a unit sphere when $k = \sqrt{w}$, where k is the number of helix turns in Eq. 9 and w is the total number of the points to be distributed. Once k is defined, the helix curve is defined by Eq. 9.

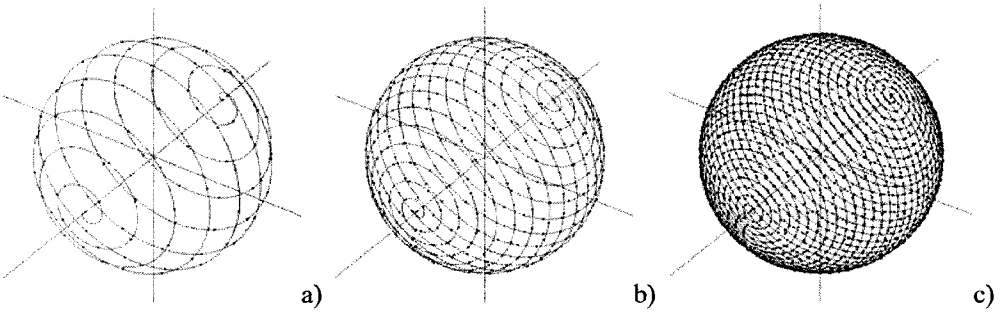


Figure 47: k-turn spherical helix curve on a unit sphere: a) k=10, w=100; b) k=20, w=400; c) k=40, w=1600.

4.1 Spherical Helix Based Look-Up Table

The LUT design for uniform sampling is inherited from the original LUT design presented in RSID. It is constructed as a 1D array to represent a single k-turn spherical helix curve. The size of the LUT is determined by the number of the points on the spherical helix. The data saved in LUT is still the distance data sampled in the corresponding orientations. Based on the characteristics of spherical helix, we developed the new LUT indexing methods and will introduce the methods in the following sections.

4.2 Uniform Spherical Sampling Method

4.2.1. Original Uniform Spherical Sampling Method

Specified spherical helix points on a unit sphere are considered another form of sampling orientation configuration. The spherical helix curve can define arbitrary number of sample points, which makes multiple sampling resolutions possible by configuring the curve formula Eq. 9. One example is illustrated in Figure 47, which shows three different sampling resolutions with three different configurations of the curve formula Eq. 9.

We call the points defined by Eq. 9 as **helix orientation points,** which are used to define the sampling rays' orientations. Each *helix orientation point* has a corresponding LUT element. Sliding along the spherical helix curve, one sampling ray is created at each specified *helix orientation point*. The surface distance data is then sampled and checked into the corresponding position of the 1D LUT in order. In this way, the LUT is filled after the spherical helix curve is swept. Originally this is an *O(m*n)* sampling by testing m sampling ray against n triangles.

4.2.2. Accelerated Uniform Spherical Sampling Method

Since the pattern of the *helix orientation point*s appearing on 2D spherical grid is no longer regular or uniform (due to the irregular spherical coordinates generated by Eq. 9), rasterization technique is not helpful to directly fill the 1D spherical helix based LUT; however, we can still take advantage of the rasterization design

introduced in our RSID system to accelerate the uniform sampling. First, the non-uniform sampling algorithm of RSID is applied. The distance data is rasterized and fills a 2D temporary LUT as a 2D distance map in *O(n)* where *n* is the number of the triangles; then, we can easily sample this 2D distance map with the predefined *helix orientation point*s' coordinates and fill our spherical helix based LUT in *O(n)* where *n* is the number of the spherical helix points. The 2D temporary LUT can be reused for multiple objects in the scene. It is released from the memory after the sampling step. Since, in this case, the samplings are based on interpolation, the accuracy of the distance data information is maximally persevered. The time complexity of our accelerated uniform spherical sampling method is then improved to be *O(n)*.

4.3 Uniform Spherical Impingement Detection and Surface-to-Surface Distance Measurement

During our ball-and-socket joint simulation, we can use Eq. 5 to compute the indices (i.e. ϕ_i and θ_i) of a target vertex v on the ball, compute the interpolated distance d_i from the neighboring *helix orientation point*s on the spherical helix, and compute the ball vertex's distance d_v to the same spherical coordinate system origin. The impingement detection can be instantly done by comparing the difference between d_i and d_v to find whether the vertex is inside or outside of the socket surface.

We need to locate the *helix orientation point*s neighboring to the target vertex v to get the sampled distance data from the LUT (Figure 48 a). There are two spherical helix turns that neighbor to v and four *helix orientation point*s on these turns that have

the closest orientations to this vertex. Two neighboring *helix orientation point*s are located in the turn above and two are in the turn below. We can find the neighboring *helix orientation point*s through the following three steps:

a) Given the spherical coordinate ϕ_v of *v*, we can first easily locate the two neighboring turns based on Eq. 9.

b) Then given spherical coordinate θ_v of v, we can find two intersection points where the line $\theta = \theta_v$ intersects with the two turns;

c) Finally, in each turn two *helix orientation point*s closest to the intersection point can be located.

For example, in Figure 48 b) P_a is *v*'s intersection point with the turn above while P_b is *v*'s intersection point with the turn below. P_0 and P_1 are the *helix orientation point*s neighboring to P_a while P_2 and P_3 are the *helix orientation point*s neighboring to P_b. Therefore P_0, P_1, P_2, and P_3 are the *helix orientation point*s neighboring to *v*. There is one exceptional case when the vertex is located inside the region of the first turn or the last turn. We just need to find three neighbors.

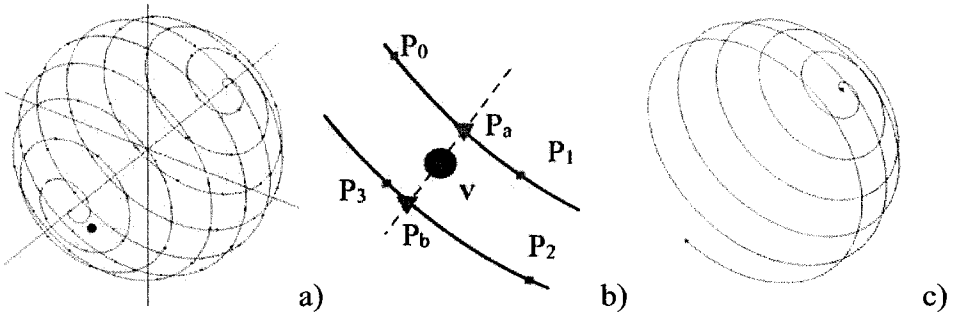


Figure 48: locating the *helix orientation point*s close to the orientation of vertex v (in blue): a) global view of the spherical helix curve (k=10); b) local view. c) one example of traveled distance. The black curve indicates Pb's traveled path from the start point of the spherical helix curve.

Step a) and step b) can be easily done. The key step is step c). Efficiently locating the two *helix orientation point*s neighboring each intersection point can be done by comparing the spherical helix points' **Traveled Distance** (Figure 48 c):

$$SHPointTraveledDistance = \frac{(4*k+1)*(-1*\cos(\phi)+1)}{2} \qquad \textbf{Eq. 10}$$

where ϕ *is the spherical coordinate of the point and k is the number of turns in the helix*

Eq. 10 is gained from data fitting based on Eq. 9 for the relationship between *k*-turn spherical helix point's segment length and the point's coordinate ϕ. The *traveled distance*s of spherical helix points (including the predefined *helix orientation point*s) on the curve are used to examine the points' neighbor relationships on the spherical helix curve.

With the intersection point's ϕ coordinate, the intersection point's *traveled distance T* can be instantly computed from Eq. 10. Since all *helix orientation point*s are equally distributed along the curve with a distance Δx, the LUT index numbers of the two neighboring points to the intersection point are $(\ floor(\frac{T}{\Delta x})\ )$ and $(\ floor(\frac{T}{\Delta x})+1\ )$.

After all four neighboring points are located, the interpolated distance data *d* in *v*'s orientation can then be computed based on these neighboring *helix orientation point*s. The impingement detection can be done by examining the difference of sampled distance *d* and the vertex's distance d_v. This distance difference can be saved as the surface-to-surface distance in *v*'s orientation and can be further referred to as the approximate strain in the medical simulation.

Algorithm 3: the USID system, impingement detection algorithm (per iteration)

For i=0; i<number of model B's vertices; i++

Convert vertex B_i into model A's local spherical coordinate system;

Compute distance d_i from B_i to the system origin;

Compute spherical coordinates $[\phi_i, \theta_i]$;

Locate the neighboring turns of the helix curve based on ϕ_i;

Locate 2 neighboring *helix orientation point*s in each turn based on θ_i;

Compute interpolated distance d from A's LUT base on $[\phi_i, \theta_i]$;

if $d <= d_i$

Impingement detected at that vertex;

$\Delta d = d - d_i$;

Save Δd as the surface-to-surface distance in the orientation of $[\phi_i, \theta_i]$;

else

There is no impingement;

The complexity of uniform impingement detection is *O(n)* where n is the number of the model B's vertices to be tested against model A's LUT.

4.4 Validation and Results

All tests were conducted using an AMD Opteron Processor 252 at 2.6 GHz with 2GB of memory.

The uniform spherical impingement detection system, USID, is first applied to several sets of generic models from 300-triangle level to 3k-triangle level to prove the design and compare with other methods. The small number of triangles is affordable to the $O(n^2)$ *and O(nlogn)* algorithms we use to compare with our *O(n)* system.

The USID system is also compared with our RSID system on the efficiency and the memory cost. Before the tests, we need to match the USID system with the RSID system to produce comparable sampling precisions and results. For RSID, we chose four sampling precision levels from 1° to 0.05° shown in Table 2; for USID, the same

sampling precision can be configured by a proper k: for each certain precision used for our non-uniform sampling method, the number of the turns, k, can be determined by matching the number of the sample points at the equator generated by non-uniform sampling. The equator has lowest sample point density among all ϕ levels (or latitudes) in non-uniform sampling, which means the sample point density at the equator is the bottom line when determining determine the sampling accuracy or the sampling error. In Table 6, the non-uniform sampling precisions and their corresponding k are matched. Table 2 lists the memory cost of our non-uniform RSID system. Table 7 lists the memory cost of our USID system.

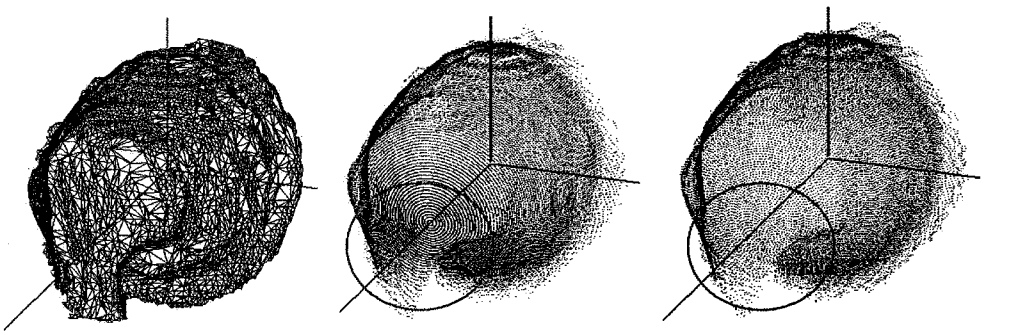
Table 6: 4 sampling precisions of non-uniform sampling version (from Chapter 3) and their corresponding k-turn of uniform sampling version (from this Chapter).

Precision of non-uniform sampling method	Turns of helix of uniform sampling
precision=1° (360 points at equator)	k=229 (360 points at equator)
precision=0.5° (720 points at equator)	k=458 (720 points at equator)
precision=0.1° (3,600 points at equator)	k =2,292 (3,601 points at equator)
precision=0.05° (7,200 points at equator)	k=4,584 (7,201 points at equator)

Table 7: The LUT size and the memory requirement of spherical helix based the USID system (in double) under 4 sampling precision settings, comparing to memory requirement of the non-uniform sampling methods

Precision	Size of 1D LUT in the USID system	USID Memory requirement	memory requirement of non-uniform sampling
1°	w= k^2=52,441	0.40MB	0.497MB
0.5°	w= k^2=209,764	1.60MB	1.98MB
0.1°	w= k^2=5,253,264	40.07MB	49.5MB
0.05°	w= k^2=21,013,056	160.32MB	197.8MB

The comparisons show USID can reduce about 20% sample points and the memory cost. In Figure 49, the global density of the sample points generated by our non-uniform sampling and uniform sampling methods is compared. The local densities of the sample points, especially at the pole (ϕ=0°) marked by red circles, are compared in Figure 49. We observed that in the area around the pole the sample point density has a clear reduction as desired.



a) b) c)

Figure 49: The comparison of sample point density generated by our two sampling methods at same sampling precision=1°: a) the original socket model; b) its sample points (65,160 points) generated by our non-uniform sampling; c) its sample points (52,441 points) generated by our uniform sampling.

Our non-uniform sampling method has an efficiency of *O(n)* and makes the preprocess step much shorter. The uniform version of sampling takes more

computations for a fair memory cost reduction but the sampling speed is still fast when it is accelerated by the sampling algorithm designed for the RSID system as introduced in section 4.2.

Table 8: benchmark of 3 sampling methods on 3k-triangle level generic models only (in millisecond)

Precision	Pair-wise testing $O(n^2)$	Sampling method in RSID $O(n)$	Sampling method in USID (accelerated) $O(n)$
precision=1°	176,328 ms	15 ms	31 ms
precision=0.5°	700,844 ms	16 ms	109 ms
precision=0.1°	17,722,156 ms	62 ms	2,781 ms
precision=0.05°	N/A	187 ms	11,172 ms

The impingement detection algorithm in USID requires higher computation cost than RSID because it takes more steps analyzing the spherical helix curve to locate the neighboring sample points on the curve. The benchmarks in Table 9 shows that the impingement detection algorithm in USID takes more computations than the algorithm in RSID; however the impingement detection algorithm in the USID system is still an $O(n)$ algorithm and the efficiency shown in Table 9 is sufficient for real-time simulations.. The USID system is further applied to one set of real models at 12k-triangle level to validate the practicability and reliability. The models are segmented from the CT data. In Figure 50, the impingement result with approximate strain computation from USID (Figure 50 b) is illustrated and compared with the

result from RSID (Figure 50 a). The USID system produces the same result as well with a reduced number on sample points. The trade-off is the increased computation time cost on both sampling and impingement detection than the RSID system.

Table 9: Benchmark of 4 impingement detection methods on 4 different triangle levels (in milliseconds).

Triangle level (per model)	Pair-wise testing $O(n^2)$	RAPID [GLM96]	Impingement detection method in RSID $O(n)$	Impingement detection method in USID $O(n)$
Generic model (300-triangle)	30.7 ms	1.41 ms	0.06 ms	0.07 ms
Generic model (1k-triangle)	522 ms	1.41 ms	0.18 ms	0.65 ms
Generic model (3k-triangle)	2,831 ms	1.43 ms	0.48 ms	1.8 ms
Real model from CT (12k-triangle)	N/A	1.45 ms	5.7 ms	12 ms

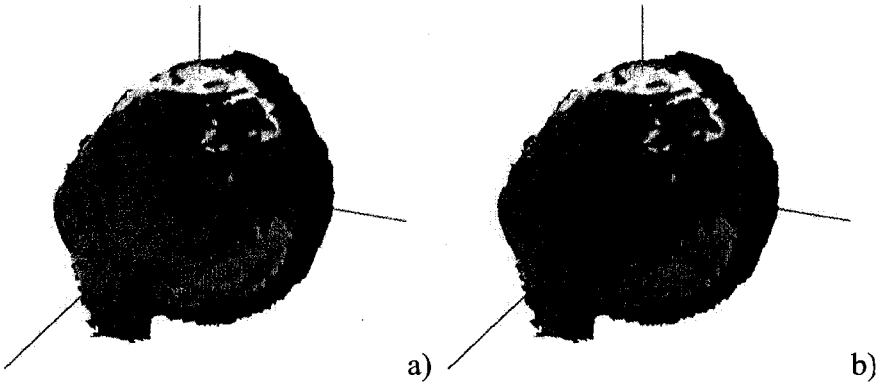


Figure 50: RSID compared to USID. Impingement detection results on the socket with sample points (in green), left posterior view: a) results from RSID; b) results from USID

4.5 Comparison

In this section, we compare our uniform sampling method in the USID system to the existing methods that uniformly sample on a sphere. The method of uniformly distributing sample points determined the designs of other parts in the USID system: the corresponding LUT data structure and the impingement detection algorithm. After all, the new feature of USID comparing to RSID is its uniform spherical sampling feature. The impingement detection method in the USID system has similar features and results to the RSID system. So the comparison between impingement detection of USID to the existing collision detection methods can be checked in the chapter of RSID, section 3.5.2.

In the USID system, we use spherical helix curve formula from [NAKK06] to equally distribute n points on the unit sphere to define the sampling orientations. It is a uniform deterministic sampling solution. Uniform random/non-deterministic sampling is more popular in scientific analysis but the irregular point distribution property is not suitable for traceable LUT indexing and the same pattern of point distribution is not repeatable. For the deterministic sampling, there are different ways to equally distributing n points on the sphere, such as subdividing spherical polygons and mathematical approaches.

Subdividing icosahedron is one way to approximating a spherical shape with well-distributed triangles/vertices which is usually used in the modeling problems. In a recent publication of uniform sampling [PH03], Praun et al. samples the spherical domain using uniformly subdivided polyhedral domains, namely the tetrahedron,

octahedron, and cube. However, their approach is aimed at the modeling problems and the computational complexity of their sampling is $O(n^2)$ without any concern for the speed requirement. Furthermore, their approach only focuses on sampling a single model. It is not designed to provide efficient data structure for collision detection problems between multiple objects. Given a vertex v it requires a domain search to figure out which partition of the subdivided polyhedron corresponds to v and then which vertex in that partition is closest to v. Such computational complexity can be $O(nlogn)$ for n vertices' queries for a hierarchical structure. The domain or range information of each partition shall also be frequently updated when the model is rotating or translating, which will further increase the complexity of locating problems. In comparison to their sampling method, we considered the speed besides the accuracy since we focus on rapid collision detection problems among multiple models. Our method has an $O(n)$ computational complexity for n vertices' queries and does not require any updates for rigid objects.

Mathematical approaches such as spiral points [SK97] or spherical helix points [NAKK06] are one deterministic way to distribute points on the unit sphere. The advantage of mathematical approaches is that they usually have clear math formulas to compute the points' locations. This advantage makes it possible to find the pattern to trace the generated points. Since distributing n points on a unit sphere is still considered as an on-going mathematical problem, there are always some limitations in the current approaches. The common limitation of deterministic mathematical approaches, including the recent method [NAKK06] we use, is that they are still

approximations to the equally distribution. For [SK97], another limitation is that the number of points is limited to 12,000, which is not suitable when used in a high-precision sampling scenario.

Chapter 5 Related Research for Medical Simulations

In this thesis, the major technical contributions are the RSID system and the USID system as the solutions to detect the impingement of the ball-and-socket type joints. Since the beginning, we have tried different experiments and extended the major research to study some interesting hip joint problems. In this chapter, two experiments are selected as the extension studies: one is the weighted spherical sampling for hip joint's conchoid representation and the other is the impingement observation in a range of motion simulation.

5.1 Weighted Sampling for Conchoid Representation

Before RSID and USID were designed, sampling rays were the major method to sample object geometry data, which was generally called *spherical sampling* method. Since hip joint was our subject of interest, the research was made to adjust the sampling rays' orientations to better fit the hip joint's shape. Weighted sampling is introduced in this section. In the following paragraphs, sampling ray testing is the major way to perform the sampling. Sample point is the intersection point of the sampling ray and the surface.

5.1.1 Conchoid Representation

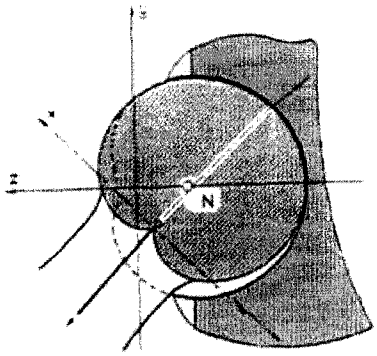
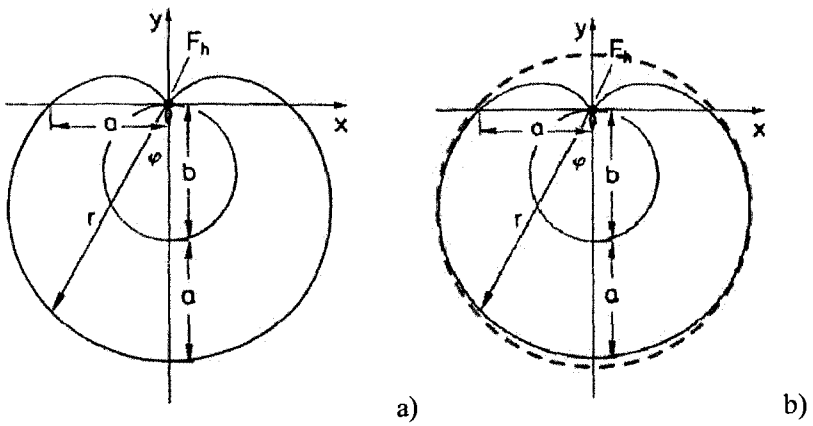


Figure 51: The conchoid representation for the hip joint (reproduced from [Men97]).

According to Menschik [Men97], the hip joint, both femoral head and the acetabulum, can be better represented by conchoid shape (Figure 51) (Figure 52 a) which is also known as one type of the Limacon of Pascal curves. Eq. 11 is the conchoid form:

$$r = a + b\cos\varphi \qquad \textbf{Eq. 11}$$



a) b)

Figure 52: a) conchoid representation (reproduced from [Men97]); b) comparing to the spherical representation (dashed line).

Figure 52 b) shows the difference between the original spherical representation and the conchoid representation. Spherical sampling method for general near-spherical objects uses equally distributed sampling rays to sample the surface. As

a conchoid is like a deformed sphere, general spherical sampling method does not equally sample the conchoid shape surface. Our experimental measurement and also the theoretical proof (based on the Eq. 8) showed the problem that on each 2D conchoid slice of the 3D hip joint about the the general spherical sampling is non-uniform sampling: if we place the origin of the spherical coordinate system at the center of the model, the segments in the middle are up to 20% longer than the bottom segments (Figure 53 a); if we place the origin exactly at the origin of the conchoid system, the segments at the bottom are up to 30% longer than the segments at the top (Figure 53 b). Therefore we need to adjust the original spherical sampling method to better fit the conchoid system so that the surface can be equally segmented and sampled.

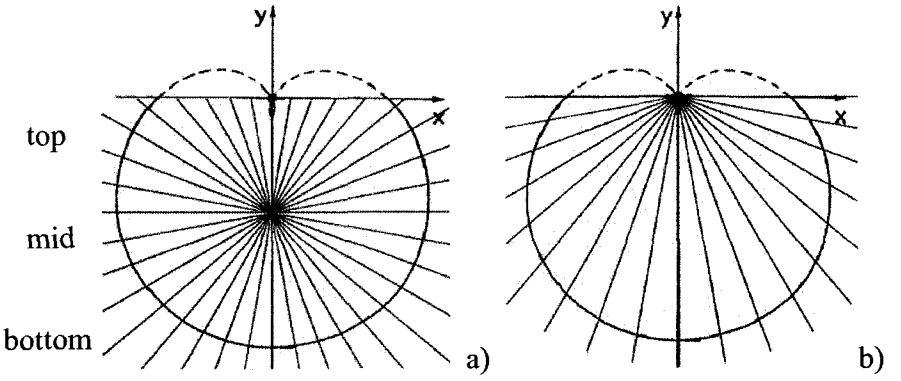


Figure 53: Adapt spherical sampling to the shapes by setting the origin to a) the center of the object; b) the conchoid's origin.

5.1.2 Adjusted Spherical Sampling for Conchoid Representation

In order to fully adapt the spherical sampling method to the conchoid system of the hip joint (Figure 53), the origin is set exactly to the origin of the conchoid system.

In [Men97], the 3D hip joint can be described as a 3D conchoid (Figure 54 a), formed by the 2D conchoid curve rotated about a rotation axis (Figure 54 b). Each slice about the rotation axis is a 2D conchoid (Figure 54 c).

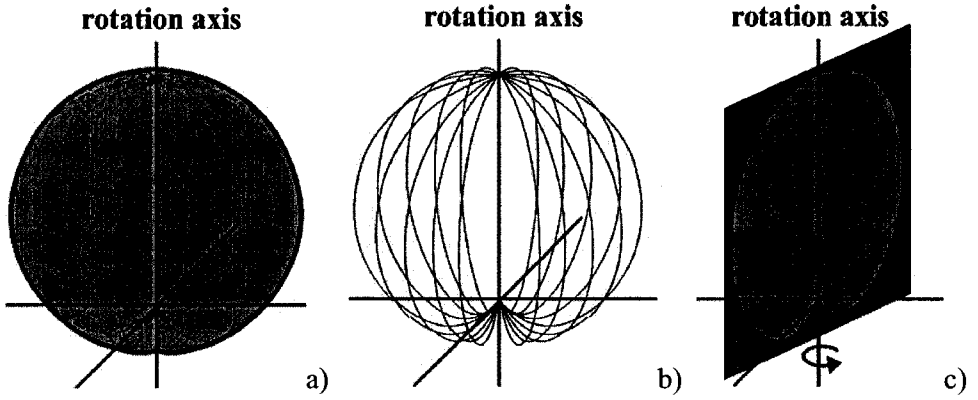


Figure 54: Conchoid representation of hip joint in 3D: a) 3D conchoid; b) 2D conchoid curve rotated about the rotation axis; c) each slice is a 2D conchoid.

The spherical coordinate θ can be used as the rotation angle of a 2D conchoid slice about the rotation axis (but there is no need to define and study this θ for this research as discussed in the following sections). The spherical coordinate ϕ is the conchoid φ angle of a 2D conchoid curve.

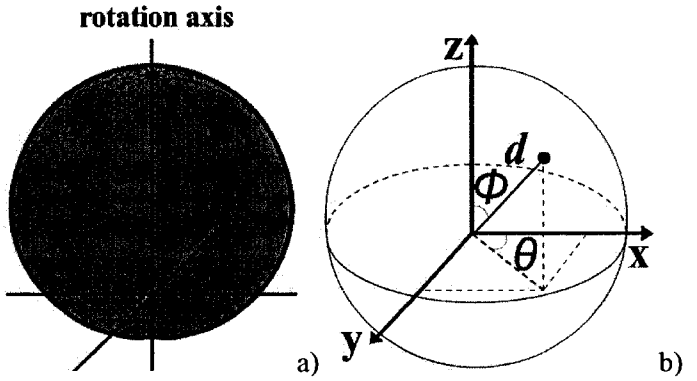
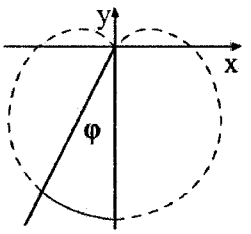


Figure 55: Setup the coordinate system in a conchoid representation of 3D hip joint a) the proximity of 3D hip joint with rotated conchoid curves; b) spherical coordinate system

On each 2D conchoid slice, the conchoid φ angle can be converted to spherical coordinate ϕ by $\phi = \varphi$ from the conchoid system to spherical system. The other spherical coordinate θ is only used in 3D as the rotation angles of the conchoid slices about the rotation axis. The problem we want to solve here is how to adjust the conchoid φ angle on each 2D conchoid slice so that θ is not concerned and we do not need to adjust θ. The way we partition the conchoid surface is first to analyze its curve length formula. Given a conchoid φ angle, we can get the curve length covered by the angle with Eq. 12.

$$p = \int_0^{\varphi} \sqrt{\left(\frac{dr}{d\varphi}\right)^2 + r^2}\, d\varphi$$

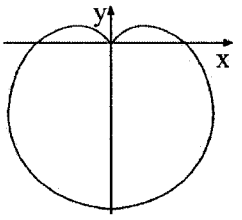
Eq. 12



The conchoid curve is fully covered when $\varphi = \Pi$. The formula for the hipint is Eq. 13 and the covered curve is illustrated beside the formula.

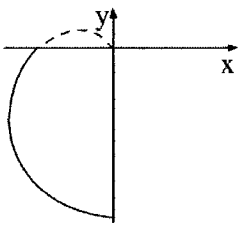
$$p = 2 * \left(\int_0^{\Pi} \sqrt{\left(\frac{dr}{d\varphi}\right)^2 + r^2}\, d\varphi\right)$$

Eq. 13



However, the conchoid representation of the hip joint requires has the φ angle from 0° to 90° (or $\frac{\Pi}{2}$). In addition, only one side of the rotation axis is required and selected for analysis as the shape is mirrored in the rotation axis (y axis in the figure). The perimeter of the selected curve (illustrated besides the formula) is:

$$p = \int_0^{\frac{\Pi}{2}} \sqrt{\left(\frac{dr}{d\varphi}\right)^2 + r^2}\, d\varphi$$



Eq. 14

Now, we need to evaluate the integral to find a simplified perimeter formula. According to [Men97], b = *17.843mm* and a = *20.501mm* for the acetabulum's conchoid curve formula (Eq. 11). We then have a = *(20.501/17.843)*b=1.15b* for a scalable hip joint model with only one parameter b left to help evaluate the equation. The perimeter p of the selected curve in Eq. 14 is calculated as follows:

$$p = \int_0^{\frac{\Pi}{2}} \sqrt{(-b\sin\varphi)^2 + (a + b\cos\varphi)^2}\, d\varphi$$

$$= \int_0^{\frac{\Pi}{2}} \sqrt{b^2\sin^2\varphi + a^2 + 2ab\cos\varphi + b^2\cos^2\varphi}\, d\varphi$$

$$= \int_0^{\frac{\Pi}{2}} \sqrt{b^2 + a^2 + 2ab\cos\varphi}\, d\varphi$$

$$= \int_0^{\frac{\Pi}{2}} \sqrt{2.3b^2 + 2.3b^2\cos\varphi}\, d\varphi$$

$$= \sqrt{2.3}b \int_0^{\frac{\Pi}{2}} \sqrt{1 + \cos\varphi}\, d\varphi$$

$$= \sqrt{2.3}b * \sqrt{2} \int_0^{\frac{\Pi}{2}} \cos\frac{\varphi}{2}\, d\varphi$$

$$= \sqrt{4.6}b * \left[2\sin\frac{\varphi}{2}\right]_0^{\frac{\pi}{2}}$$

$$= (4.29b * \sin((\pi/2)/2) - 0)$$

$$= 4.29b * \sin((\pi/2)/2) = 3.03b$$

The length of the selected conchoid curve in Eq. 14 is *3.03b*. Since the goal of this research is to cut the curve into equally-partitioned segments with n sampling rays. Each segment will have the same length of *3.03b/n*. For example, the conchoid curve in Figure 56 is sampled by 4 sampling rays which cut the curve into 3 segments. Each curve segment will have the same length of $\Delta L = 3.03b/4$. The curve length

covered by 1st ray is ΔL, the curve length covered by 2nd ray is $2*\Delta L$, and the curve length covered by 2nd ray shall be $3*\Delta L$. So we need to figure out how to define the φ angle of each sampling ray in order to cut the curve into these equally-partitioned segments.

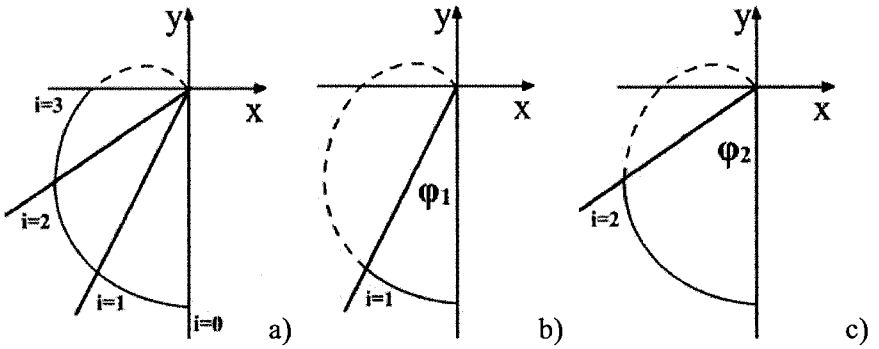


Figure 56: a) the conchoid curve sampled by 4 rays; b) the curve length covered by 1st ray; c) the curve length covered by 2nd ray.

The conchoid curve length covered by ith φ angle shall be the segment length multiplied by i as shown on the right side of Eq. 15, which evolves into the formula of computing the adjusted φ angle (Eq. 16). The new φ angles of the sampling rays will then produce the equally-partitioned segments.

$$\because \left(4.29b\sin(\frac{\varphi_i}{2})-0\right)=\frac{3.03b}{\#\text{of sampling rays}-1}*i \qquad \textbf{Eq. 15}$$

where i is the index of the current i^{th} sampling ray we look at.

$$\therefore \varphi_i = 2*\sin^{-1}(\frac{3.03*i}{(\#\ \text{of sampling rays}-1)*4.29}) \qquad \textbf{Eq. 16}$$

For example, using 91 sampling rays (precision=1°) to partition the curve shown in Eq. 14 into 90 segments, the adjusted angle formula for ith φ angle is obtained as follows:

$$\varphi_i = 2*\sin^{-1}(\frac{3.03*i}{90*4.29})=2*\sin^{-1}(0.00785*i)$$

5.1.3 Results

Using Eq. 9, we first tried a simple example to partition the interested curve into 3 segments by 4 sampling rays as shown in Table 10. The sampling precision is 30°.

Table 10: An example of adjusting sampling rays. The precision is 30° and 4 sampling rays with their original φ from 0° to 90° are listed.

i	Original φ	Adjusted φ	Adjusted rate on φ	Adjusted rate on ϕ by $\phi = \varphi$
i=0	0°	0°	0°	0°
i=1	30°	27°	-3°	-3°
i=2	60°	56°	-4°	-4°
i=3	90°	90°	0°	0°

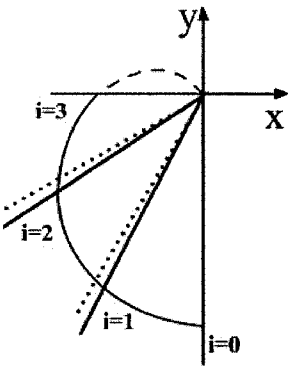


Figure 57: Sampling rays: original (in dash lines) and adjusted (in solid lines).

Figure 57 shows the adjustments on the sampling rays. We provide two comparing charts under a higher precision of 0.1° which contains 901 sampling rays for the interested curve (Figure 58). With a larger number of sampling rays, we get a curve of the adjusted ϕ angle rates (Figure 59).

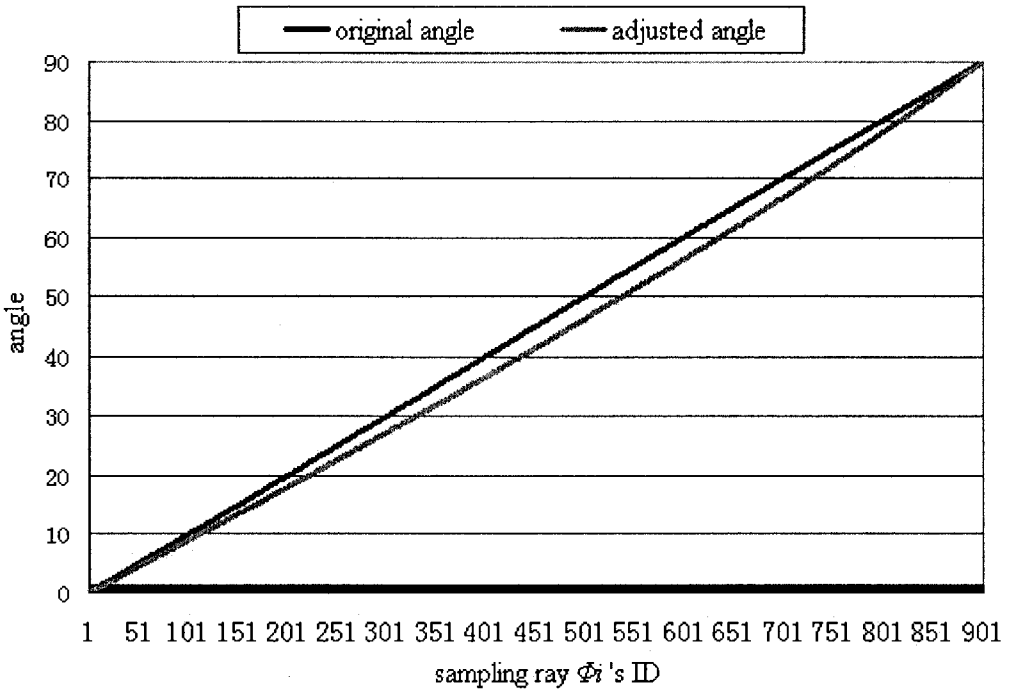


Figure 58: Adjusted ϕ angle at precision=0.1°: the original angles using spherical representation (blue) and the adjusted angles using conchoid representation (purple)

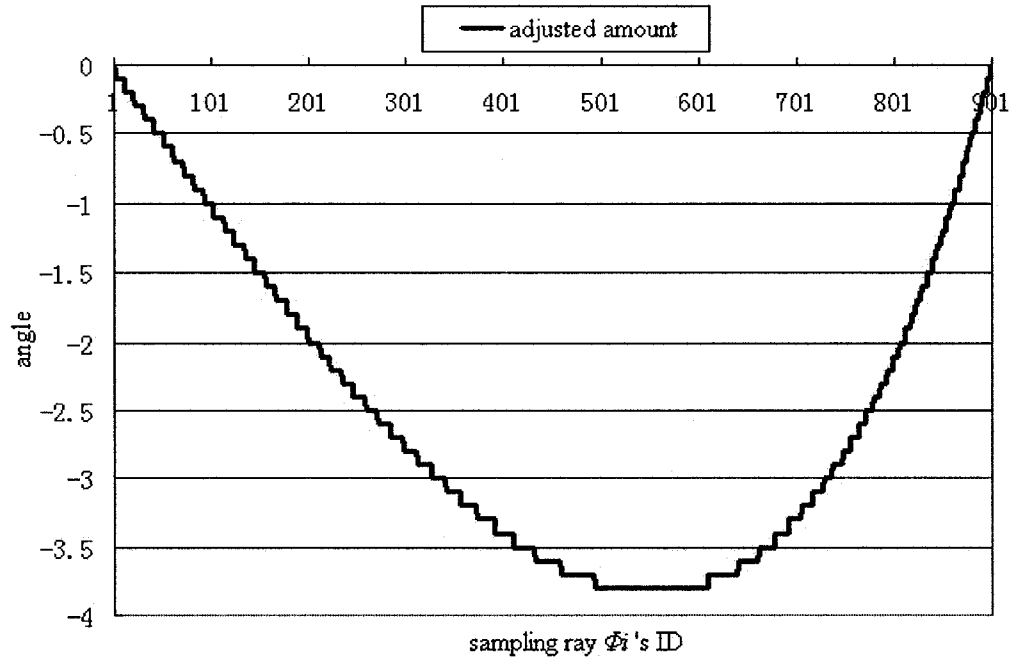


Figure 59: Adjusted ϕ angle at precision=0.1°: the amount of the angles adjusted.

The purpose of this research is to find the adjustment of the sampling rays in order to equally sample the conchoid shape of the hip joint. The results suggest that the sampling orientations shall be adjusted according to the amount illustrated in Figure 59. The maximum rate is -4° for the rays in the middle. Further discussion on the weighted spherical sampling is in the Discussion section of this document.

5.2 Impingement Observation with Range of Motion

Based on the RSID system, we made a simple range of motion (ROM) simulation to rotate the femur model under the ROM values and observe the impingement results during the motions. In section 1.1, we present the process pipeline of a pre-operative arthroscopic surgical planning system (Figure 1) which is used to evaluate the impingement during the motion and then simulate the bone correction. Here the motion simulation is a simple ROM simulation. The concept of ROM is presented in section 2.1.2. In the ROM simulation the femur model rotates about the center of rotation (COR) on the three planes: Sagittal plane, Frontal plane, and Transverse plane. We use the ROM values to set the ranges of the rotations. So the femur rotates on each plane and stops when its rotation angle reaches the corresponding ROM value.

Here we segmented two sets of models from CT data which are identified as patient A and patient B. Patient A is a male around 40 years old and has FAI on the right hip joint. Patient B is a male around 30 years old and has FAI on the left hip joint. The resolution of the CT data is 512*512 and the thickness of each slice is 1.25mm. In Figure 60, the excessive region on patient A's hip joint (marked by a red

circle) can be specified based on the color scale (also introduced in Figure 39). Patient A's joint is compared to a similar case of FAI on the right hip joint from [WG06]. This indicates that the impingement will most likely occur at the marked excessive region; in Figure 61, the excessive region on patient B's hip joint (marked by a red circle) is specified based on the color scale. Patient B's joint is compared to a similar case of FAI on the left hip joint from [WG06].

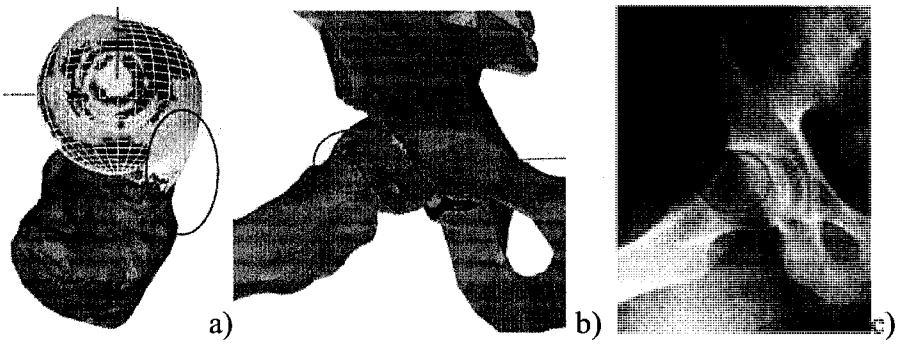


Figure 60: Patient A: a) the right femoral head, superior view; b) the right hip joint, oblique view; c) a similar case of FAI on right hip joint, oblique view (reproduced from [WG06]).

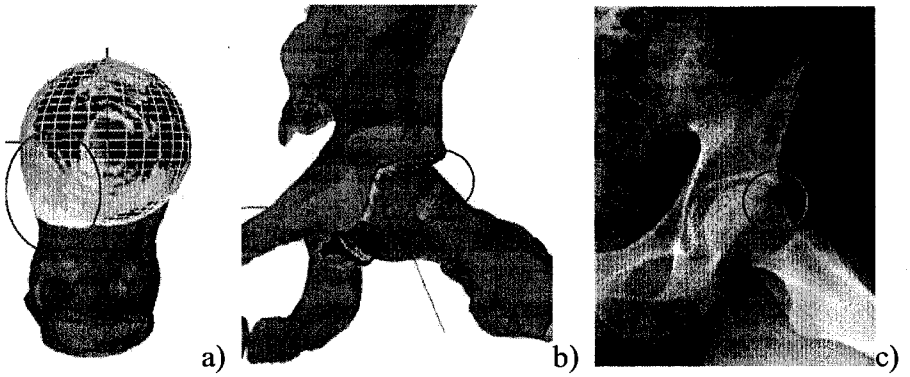


Figure 61: Patient B: a) the left femoral head, superior view; b) the left hip joint, oblique view; c) a similar case of FAI on left hip joint, oblique view (reproduced from [WG06]).

The ROM data (Table 11) is provided by School of Human Kinetics, University of Ottawa. The motion analysis data using a 3D Vicon analysis were collected from a

patient suffering from FAI. For the motion analysis, seven VICON MX-13 cameras at 200 Hz with retroreflective markers placed on anatomical landmarks was used. The data provider, Kennedy M. J., presented the procedure of capturing and analyzing the ROM data in [KLB08]:

> "...Participants were asked to put on a skin-tight suit and performed a stretching warm-up routine. After the stretching warm-up, participants completed a sit and reach flexibility test, had retro-reflective markers placed on anatomical landmarks, and underwent anthropometric measurements. Thereafter participants performed a series of maximal dynamic hip ROM trials. Participants stood upright holding onto a support frame while they maximally swung their leg back and forth in five times in each plane at a self selected speed..."[KLB08]

In our ROM simulation, the femur model rotates about its center of rotation (COR) and the simulated virtual motion is limited within ranges of the recorded ROM data. The setup method of the COR is discussed in the section of the RSID system and is illustrated in Figure 38: for each patient's femur model, we fitted a sphere to the femoral head and found the COR. At the extremities of the motions represented by the ROM values, the computed impingement results are pictured.

Table 11: ROM data of Patient A and Patient B

ROM	Patient A (right hip)	Patient B (left hip)
Abduction	48.7°	30.3°
Adduction	26.4°	22.7°
External rotation	39.9°	21.0°
Internal rotation	5.6°	12.2°
Extension	5.2°	14.5°
Flexion	103.1°	115.8°

The right hip joint of patient A is visualized form left medial view. The pelvis (except the socket region) is not visualized. The setup of the simulation scene is illustrated in Figure 62 a). The femoral head is behind the socket and is then not visible to the camera from left medial view. The socket surface is observed for the impingement results.

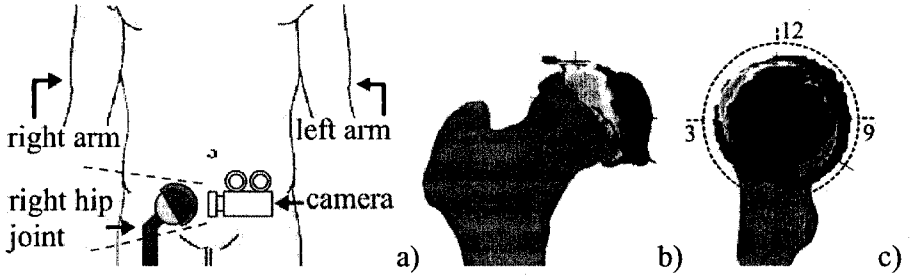
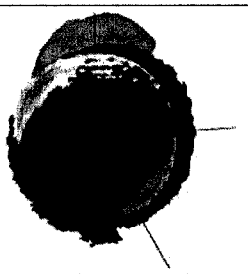
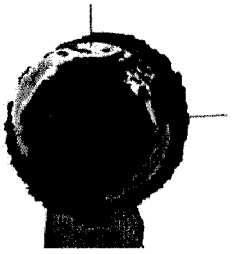
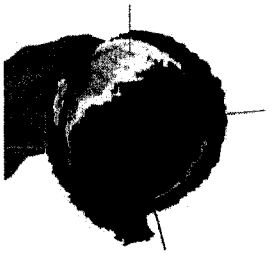


Figure 62: a) the setup of the scene for patient A (human body image reproduced from [SSS*05, p39]); b) hip joint, anterior view of the scene; c) hip joint with clock system, left medial view from the camera.

The clock system for the socket shown in Figure 43 is used to guide the observation. In this case, it is horizontally reversed when visualized from left medial view (Figure 62 c). According to the results listed in Table 12, we can observe the red zones appearing frequently at 1~2 o'clock. The red shows that the distance between the two bones' surfaces is smaller than 2mm while the blue indicates a distance greater than 6mm. Impingement most likely occurs at 1~2 o'clock. In the motions

Abduction, Internal rotation, and Flexion, the regions have the smallest gap between the bones so that the strain and the impingement are greater at the extremities of these motions. The red zones match the excessive regions of the femoral head illustrated in Figure 60 and causes impingement during the motions.

Table 12: Impingement results under the ROMs of patient A. It is the right hip joint from left medial view. The pelvis (except the blue socket region) is not visualized.

		
Abduction	Internal rotation	Extension
		
Adduction	External rotation	Flexion

The left hip joint of patient B is visualized from right medial view. The pelvis (except the socket region) is not visualized. The setup of the simulation scene is illustrated in Figure 63 a). The femoral head is behind the socket and is not visible to the camera from right medial view. The clock system for the left hip joint is illustrated in Figure 63 c). The socket surface is observed for the impingement results.

According to the results in Table 13, we can observe red zones appearing frequently at 1~3 o'clock and 7 o'clock. In the motions Abduction, Adduction, Internal rotation, and Flexion, the regions have the smallest gap between the bones so that the strain and the impingement are greater at the extremities of these motions.

The red zones match the excessive regions of the femoral head illustrated in Figure 61 and causes impingement during the motions.

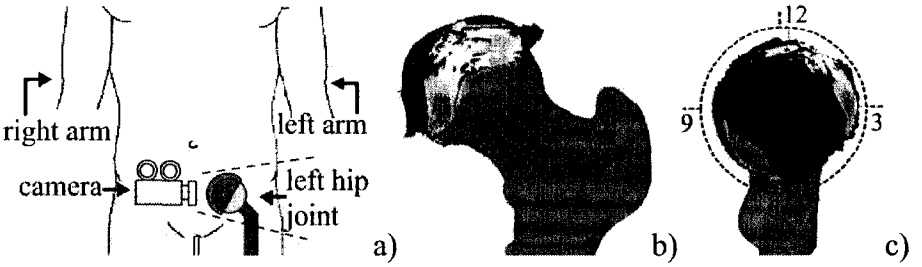
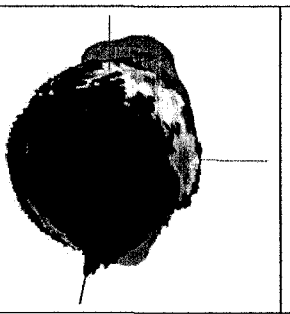
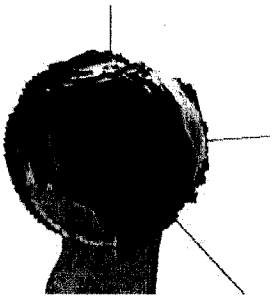
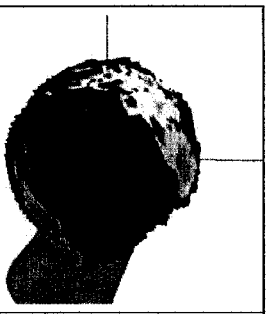
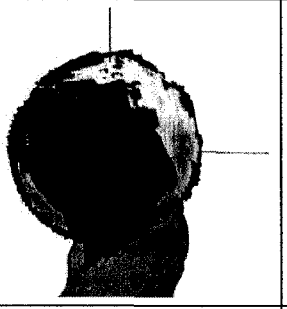
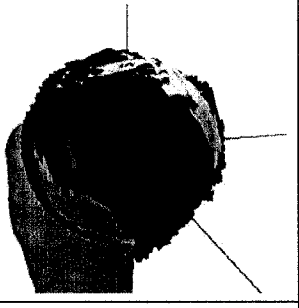
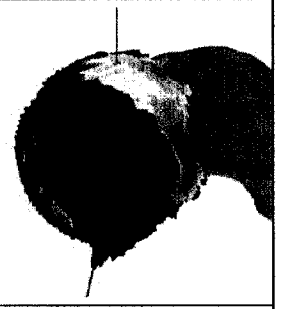


Figure 63: a) the setup of the scene for patient B (human body image reproduced from [SSS*05, p39]); b) hip joint, anterior view of the scene; c) hip joint with clock system, right medial view from the camera.

Table 13: Impingement results under the ROMs of patient B. It is the left hip joint from right medial view. The pelvis (except the socket region) is not visualized.

		
Abduction	Internal rotation	Extension
		
Adduction	External rotation	Flexion

Further discussion on this ROM observation function is in the Discussion section of this document.

Chapter 6 Conclusion

In this thesis, we first introduced a rapid impingement detection system. It is designed for the purpose of ball-and-socket joint impingement detection with a rapid sampling algorithm and a rapid impingement detection algorithm. Surface-to-surface distance measurement for ball-and-socket joints is another major contribution of our system where the distance between the ball and the socket is measured instantly during the kinematic simulation. The second system, as a uniform sampling version, is introduced to sample the space more adequately and efficiently.

Our systems are suitable to all ball-and-socket type joints and near-spherical objects. Star-shape objects are also suitable as long as a point inside the object there is visible to all the points on the object's surface.

The systems are very rapid for geometric shape representation in the spherical coordinate system to process the impingement detection. They can be extended to adapt to a deformable model such as cartridge for impingement detection because updating the sampled data of a deformed surface is very efficient. Self-collision of a deformed surface can also be efficiently detected if, in each sampling step, sample points from different triangles (except neighboring triangles) are overlapped with the same spherical coordinates and same distance values.

Since our rasterization based sampling method of the RSID system directly process all the triangles to generate the sample points through rasterization, it is more accurate

and much faster than most of the existing sampling methods to sample the near-spherical objects.

Our impingement detection systems, RSID and USID, belong to the category of distance fields. The LUT data structure is a type of distance map. Since the subjects we study commonly have one spherical layer/surface affected by the impingement response, we only consider one layer of distance field as a simple demonstration of the general idea. Multiple layers can be accomplished by applying distance offsets. For example, in the extension research of range of motion observation we use a distance offset to represent the thickness of the cartilage. In comparison to the common distance fields generated using uniform 3D Cartesian grids, our RSID and USID methods have one order of magnitude less on the computational complexity and memory requirement; in comparison to the BVH based methods aimed at the speed, our impingement detection methods are a bit slower for high-resolution models but are still fast enough for real-time simulations at millisecond level computing. Most importantly, our methods are able to provide surface-to-surface distance information between the objects without increasing the time cost while BVH based methods are unable to provide such collision information; in comparison to most of the existing joint impingement simulation methods, our overall system designs are very efficient and can provide a global impingement estimation of the joint at surface level rather than simply reporting a few collided points. Our research is going in a meaningful and scientific direction to study the joint impingement based on the surface-to-surface distance information.

Our uniform sampling is a deterministic sampling which is traceable for LUT indexing purposes and can be easily configured for different levels of the sampling precision. Our USID system shows a 20% memory cost reduction than our RSID system while the system is still efficient for real-time simulations. The memory efficiency provided by our study is more significant when multiple pairs of the models are tested in one scene.

Besides the major research on the RSID system and the USID system, two researches related to the medical simulations are selected as the studies on the hip joint problems: the first study is the weighted sampling for hip joint's conchoid representation. It was a theoretical study on equally distributing sample points on the hip joint surface. The result shows a maximum adjustment of -4° will be applied to the ϕ angle of the sampling rays. Based on the idea of equally sampling, the more formal USID research on uniform sampling was made for general near-spherical objects; the second study is the impingement observation in a range of motion simulation. Based on the RSID system, the impinged areas of the hip joint can be observed in real-time during the motions.

6.1 Contributions

The following contributions were made in this thesis:

- Developed two efficient and accurate impingement detection systems for ball-and-socket joints with the valuable surface-to-surface distance

measurement feature. The second system is based on a uniform sampling method which can further reduce the memory cost.

- Designed a visualization system for ball-and-socket joints to estimate the joint impingement level and compute the surface-to-surface distance information as the approximate strain distributions.
- Weighted spherical sampling method is theoretically studied based on the conchoid representation for the hip joint.

6.2 Discussion

The research on weighted spherical sampling started before the first impingement detection system, the RSID system, when the exhaustive pair-wise testing was used to generate the sample points by sampling rays. The weighted sampling is theoretical studied as a different way to sample the hip joint. The original idea was to produce a uniform sampling on the hip joint; however, this weighted sampling method is uniform only on each 2D conchoid slice but is non-uniform in 3D with no reduction of the number of sample points. To better solve the issue, the more specific USID system was then developed in order to provide a uniform sampling in 3D Cartesian space for general near-spherical objects.

The RSID system is the faster solution equipped with a rapid sampling method and a rapid impingement detection method. The USID system is an alternative solution that provides a memory efficient feature. For a simple simulation scene

containing only one pair of models, the RSID system has a better performance than the USID system because the extra memory cost is not that significant.

At the current research stage, the ROM simulation discussed in this thesis only works as a very simple observation tool built on the RSID system. It is not a formal ROM research because of the current limitations of the modeling and the depth of the ROM research. Due to the difficulties of modeling the abnormal surfaces in FAI, the modeling step requires lots of manual work which causes certain manual errors. Also, the traditional voxelized models used in the medical area have errors to represent the smooth shapes. The accuracy of the RSID system and the USID system heavily relies on the input models. Therefore, due to the limitations from the current available modeling methods to segment FAI bones, ROM research, such as computing ROMs based on impingement detection, are not very reliable at this stage. Besides the modeling, there are also many other affecting factors shall be systematically studied and validated in the future before any reliable ROM simulations can be designed and built, such as the models' setup, the parameters researched from real ROM scenes, and the medical study of the patient cases. Even the current research on ROM simulations in the medical community has many assumptions, such as considering physical collisions as the impingement and considering a fixed rotation center.

6.3 Future Research

This research on impingement detection discussed in this thesis is the beginning of a long-term medical study on the hip joint impingement. Comparing to the existing

solutions, there are many new and valuable features created in our systems; however, many of the features are not yet the scientific solutions such as the approximate strain computation and the ROM observations simulations. Further studies on modeling, ROM studies, stress/strain computation, cartilage studies, virtual surgery simulation, etc., shall be systematically carried to provide more reliable and scientific solutions while these problems are still being studied in the medical community.

The accuracy of the systems heavily depends on the inputs. How to efficiently and accurately model a proper hip joint from the CT data shall be researched. Most of the current modeling methods are not specially designed for the hip joint and have more difficulties rebuilding abnormal hip joints. The available segmentation methods are not yet flexible or advanced enough to identify the desired details from the scanned data, which usually leads to heavy manual work to solve the problem. After all, the voxelized models produced from segmentation are not suitable to represent the smooth objects. In order to produce more accurate impingement results from our systems, the modeling efficiency and quality shall be studied and improved. Also, besides the bone modeling, soft tissue modeling shall be studied. In this document, we use a uniform distance offset to represent the cartilage thickness. In a human hip joint, the tissues, such as the muscles and the cartilages, are more complicated structures. The data set we use is CT data, which is not convenient to gain the tissue information. Modeling the tissues, especially the tissues on the abnormal bones (e.g. FAI cases), can be a challenging research topic.

The impingement (impingement location and level) need to be further studied to reveal the causes of hip pain. Most of the current simulation studies being carried by others only consider very simple scenarios to determine impingement. In their research on ROM, they compute the limits of ROM by detecting one or several collided points on the surfaces during the motion; however, direct collisions between bones are not realistic without concerning the tissues between the bones. Also, the internal forces may be the right reasons to limit ROMs instead of the physical contact between the bones. In this document, the strain distribution in ROM observation simulation is represented by the surface distance information as the impingement results. It now can give people a general idea of the surface-level impingement results but it requires further research to really understand the problems. Associated with the further medical studies and material science on the bones and tissues, more scientific and accurate stress/strain distribution on the hip joint surfaces can be simulated to reveal the causes of the hip pain and the influences on the ROMs.

ROM is worth a systematic research to design a reliable virtual ROM simulation. The modeling problem mentioned above influences the accuracy of the ROM computation. There are more problems to be studied: first the methods of recording the patient's movement determine the reliability of the ROM data analysis. For example, what are the errors of placing the tracking markers on the skin to track the bones and what are the relative positions of the bones during the motions? A study shall verify the causes of the ROM reduction (e.g. stopped by physical contact between the bones or stopped by the pain level felt by the patient) in order to properly

simulate ROM. The level of the hip pain and the level of the stress/strain during the motion must be analyzed to define the proper levels of impingement so that the ROM simulation can be more reliable and more realistic. Many other issues influence ROMs, such as the individual patient's age, gender, case history, exercise skills, and movement capabilities. With all validated parameters from the systematic research of ROM, a reliable ROM simulation can be built.

References

[BB00] Baerlocher P., Boulic R.: Parameterization and range of motion of the ball-and-socket joint. DEFORM/AVATARS 2000: 180-190

[Beh06] Behnke R. S.: *Kinetic Anatomy*. Human Kinetics Publishers. 2nd edition. January 30, 2006

[BHW96] Barzel R., Hughes J., Wood D. N.: Plausible motion simulation for computer graphics animation. *In Proceedings of the Eurographics Workshop Computer Animation and Simulation (1996)*, Springer, pp. 183–197.

[BK95] Bartos M., Kestranek Z.: Numerical solution of the contact problem. Application to a simple model of the human hip joint. *Journal of Computational and Applied Mathematics*, Volume 63, Number 1, 20 November 1995 , pp. 439-447(9).

[BMF03] Bridson R., Marino S., Fedkiw R.: Simulation of clothing with folds and wrinkles. *In Proc. ACM/Eurographics Symposium on Computer Animation*, pp. 28–36, 2003.

[BMG99] Bielser D., Maiwald V. A., Gross M. H.: Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum*, 1999, 18(3):31～38.

[BMWM01] Breen D. E., Mauch S., Whitaker R. T., Mao J.: 3d metamorphosis between different types of geometric models. *Eurographics 2001 Proceedings* 20(3), 2001, pp. 36–48.

[BT95] Bandi S., Thalmann D.: An adaptive spatial subdivision of the object space for fast collision detection of animating rigid bodies. *In Eurographics'95* (1995), pp. 259–270.

[DDCB01] Debunne G., Desbrun M., Cani M.-P., Barr A. H.: Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. *Proceedings of ACM SIGGRAPH 2001*. pp. 31-36, 2001.

[DJS96] Dinesh M., Jonathan D. C., Stefan G.: Collision Detection: Algorithms and Applications. *Algorithms for Robot Motion and Manipulation*, pp. 129-142, 1996

[Eberly99] Eberly D.: Least Squares Fitting of Data. 1999 Available on: http://www.sci.utah.edu/~balling/FEtools/doc_files/LeastSquaresFitting.pdf

[FPRJ00] Frisken S. F., Perry R. N., Rockwood A. P., Jones T. R.: Adaptively sampled distance fields: A general representation of shape for computer graphics. *SIGGRAPH 2000, Computer Graphics Proceedings (2000)*, 249– 254.

[FSG03] Fuhrmann A., Sobottka G., Groß C.: Distance Fields for Rapid Collision Detection in Physically Based Modeling. *Proc. GraphiCon, Keldysh Inst. of Applied Mathematics*, 2003, pp. 58-65.

[GIL*01] Genda E., Iwasaki N., Li G., Macwilliams B. A., Barrance P. J., Chao E. Y.: Normal hip joint contact pressure distribution in single-leg standing-effect of gender and anatomic parameters. *Journal of Biomech*, 2001, 34(7): 895~905.

[GLM96] Gottschalk S., Lin M., Manocha D.: Obb-tree: A hierarchical structure for rapid interference detection. *In Computer Grajphics (SIGGRAPH '96 Proceedings)*, pages 171--180, 1996.

[GPB*03] Ganz R., Parvizi J., Beck M., Leunig M., Notzli H., Siebenrock K. A.: Femoroacetabular Impingement: A cause for osteoarthritis of the hip. *Clin Orthop Relat Res* 417:1–9, 2003.

[HLL*01] Hu Q., Langlotz U., Lawrence J., Langlotz F., Nolte L.-P.: A Fast Impingement Detection Algorithm for Computer-Aided Orthopedic Surgery. *Journal Computer Aided Surgery*, Vol.6, No.2, pp. 104-110, 2001.

[Hub93] Hubbard P. M.: Interactive Collision Detection. *In Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality*, 1993, pp.24-32.

[HLC*97] Hudson T. C., Lin M.C., Cohen J., Gottschalk S., Manocha D.: V-Collide: Accelerated Collision Detection for VRML. *Proc. Second Symp. Virtual Reality Modeling Language (VRML '97)*, pp. 119-125, 1997.

[HZLM01] Hoff III K. E., Zaferakis A., Lin M. C.: Manocha D., Fast and simple 2D geometric proximity queries using graphics hardware. *In Symposium on Interactive 3D Graphics* (2001), pp. 145—148

[ITK-SNAP] Itk-snap. Available at http://www.itksnap.org/

[Jos06] Joseph. C.: *Pocket Anatomy: A Complete Guide to the Human Body for Artists & Students*. Barron's Educational Series. May 1, 2006

[KDG91] Klaue K., Durnin C. W., Ganz R.: The acetabular rim syndrome: A clinical presentation of dysplasia of the hip. *Journal of Bone and Joint Surgery - British Volume*, 1991; Vol 73-B, Issue 3, 423-429

[KHI*07] Kockara S., Halic T., Iqbal K., Bayrak C., Rowe R.: Collision detection: A survey. *Systems, Man and Cybernetics, ISIC. IEEE.* 2007, pp. 4046-4051

[KLB08] Kennedy M. J., Lamontagne M., Beaule P. E.: The Effect of Cam Femoroacetabular Impingement on Hip Maximal Dynamic Range of Motion, accepted to Journal of Orthopedics, 2008

[KP03] Knott D., Pai D.: CInDeR: Collision and Interference Detection in Real–time Using Graphics Hardware. *In Proc. of Graphics Interface.* pp. 73-80, 2003.

[KTM*07] Kubiak-Langer M., Tannast M., Murphy S. B., Siebenrock K. A., Langlotz F.: Range of motion in anterior femoroacetabular impingement. *Clinical Orthopedics and Related Research.* 2007 May; 458:117-24.

[KZ05] Kavan L., Zara J.: Fast Collision Detection for Skeletally Deformable Models; *In Computer Graphics Forum.* 2005, vol. 24, no. 3, p. 363-372. ISSN 0167-7055.

[LC91] Lin M., Canny J.: A fast Algorithm for Incremental Distance Calculation. *Proc. of the 1991 IEEE International Conference on Robotics and Automation*, pp. 1008-1014, 1991.

[LCN99] Lombardo J. C., Cani M.-P., Neyret F.: Real-Time Collision Detection for Virtual Surgery. *Proc. Computer Animation '99*, pp. 82-91, 1999.

[Lev66] Levinthal C.: Molecular model-building by computer. *Scientific American*, 214 (June 1966), 42–52.

[LG98] Lin M., Gottschalk S.: Collision Detection between Geometric Models: A Survey. *In Proceedings of IMA Conference on Mathematics of Surfaces,* pages 37--56, 1998.

[LPB*04] Lavigne M., Parvizi J., Beck M., Siebenrock K. A., Ganz R., Leunig M.: Anterior Femoroacetabular Impingement: part I. Techniques of joint preserving surgery. *Clin Orthop Relat Res*, 2004, (418):61-66.

[MBT03] Maciel A., Boulic R., Thalmann D.: Deformable Tissue Parameterized by Properties of Real Biological Tissue. *In. International Symposium on Surgery Simulation and Soft Tissue Modeling*, 2003, pp.76-89.

[MBT07] Maciel A., Boulic R., Thalmann D.: Efficient Collision Detection within Deforming Spherical Sliding Contact. *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 518-529, May/Jun, 2007.

[Mel00] Melax S.: Dynamic plane shifting bsp traversal. *In Proc. of Graphics Interface '00* (2000), pp. 213–220.

[Men97] Menschik F.: The hip joint as a conchoid shape. *Journal of Biomechanics*. 1997; 30:971-973

[Mir98] B. Mirtich: V-Clip: Fast and Robust Polyhedral Collision Detection. *ACM Transactions on Graphics*, vol. 17(3), pp. 177-208, 1998.

[NAKK06] Nishio H., Altaf-Ul-Amin Md., Kurokawa K., Kanaya S.: Spherical SOM and arrangement of neurons using helix on sphere. *IPSJ Transactions on Mathematical Modeling and its Applications*, Vol.47, 56-60, 2006.

[OD99] O'Sullivan C., Dingliana, J.: Real-Time Collision Detection and Response Using Sphere-Trees. *Proceedings of the Spring Conference in Computer Graphics*, Bratislava April 28-May 1st 1999 pp 83-92.

[PBM83] Pogrund H., Bloom R., Mogle P.: The normal width of the adult hip joint: the relationship to age, sex, and obesity. *Skeletal Radiol.* 1983; 10(1):10–12.

[PH03] Praun E., Hoppe H.: Spherical parametrizaztion and remeshing, *in ACM SIGGRAPH 2003 Papers, ACM Press*, New York, NY, USA, 340–349, 2003.

[PMD*06] Pfirrmann C.W., Mengiardi B., Dora C., Kalberer F., Zanetti M,. Hodler J.: Cam and pincer femoroacetabular impingement: characteristic MR arthrographic findings in 50 patients. *Radiology* 2006;240(3):778-85.

[RGR02] Rafael C. Gonzalez, Richard E.: *Digital Image Processing*, Prentice Hall, 2nd edition, 2002

[SK97] Saff, E.B., Kuijlaars, A.B.J.: Distributing many points on a sphere, *Mathematical Intelligencer*, v 19 #1 (1997) 5-11

[SMVT04] Sarni S., Maciel A., Boulic R., Thalmann D.: Evaluation and Visualization of Stress and Strain on Soft Biological Tissues in Contact. *Proceedings of International Conference on Shape Modeling and Applications, p. 255-262. Los Alamitos, CA IEEE Computer Society Press, 2004.*

[SOM04] Sud A., Otaduy M. A., Manocha D.: DiFi: Fast 3D Distance Field Computation Using Graphics Hardware. *Computer Graphics Forum (Proc. Eurographics)* 23, 3, 557–566. 2004.

[SPG03] Sigg C., Peikert R., Gross M.: Signed Distance Transform Using Graphics Hardware. *Proceedings of IEEE Visualization 2003, IEEE Computer Society Press*, pp. 83-90

[SRLR07] Sproul R.C., Reynolds H. M., Lotz J. C., Ries Michael D.: Relationship Between Femoral Head Size and Distance to Lesser Trochanter. *Clinical Orthopaedics & Related Research.* 461:122-124, August 2007.

[SSS*05] Schuenke M., Schulte E., Schumacher U., Lamperti E. D., Ross L. M., Wesker K. H.: *General Anatomy and the Musculoskeletal System (THIEME Atlas of Anatomy)*. Thieme Medical Publishers; 1 edition. July 1, 2005

[TKH*05] Teschner M., Kimmerle S., Heidelberger B., Zachmann G., Raghupathi L., Fuhrmann A., Cani M.-P., Faure F., Magnenat-Thalmann N., Strasser W., Volino P.: Collision detection for deformable objects. *Computer Graphics forum*, vol. 24, no. 1, pp. 61–81, 2005.

[TKL*07] Tannast, M., Kubiak-Langer, M., Langlotz, F., Puls, M., Murphy, S. B., Siebenrock, K.: Noninvasive Three-Dimensional Assessment of Femoroacetabular Impingement, 2007, *Journal of Orthopaedic Research*, vol. 25(1), p. 122-131

[TurboSquid]Turbo Squid. Available at http://www.turbosquid.com

[WG06] Wisniewski S. J., Grogg. B.: Femoroacetabular Impingement: An Overlooked Cause of Hip Pain. *American Journal of Physical Medicine and Rehabilitation*, 2006, vol 85; numb 6, pages 546-549.

[WHC*03] Wagner S., Hofstetter W., Chiquet M., Mainil-Varlet P., Stauffer E., Ganz R. Siebenrock K. A.: Early osteoarthritic changes of human femoral head cartilage subsequent to femoro-acetabular impingement. *Osteoarthritis Cartilage* 2003; 11: 508–518.

[WKE99] Westermann R., Kobbelt L., Ertl T.: Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer* 15, 2 (1999), 100–111.

[YFW*06] Yoshida H., Faust A., Wilckens J., Kitagawa M., Fetto J., Chao E.: Three-dimensional dynamic hip contact area and pressure distribution during activities of daily living. *Journal of Biomechanics*, 2006, Volume 39, Issue 11, Pages 1996-2004.

[YM06] Yoon S.-E., Manocha D.: Cache-Efficient Layouts of Bounding Volume Hierarchies. *Eurographics Computer graphics forum*, volume 25, issue 3, 2006

[ZKM07] Zhang L. J., Kim Y. J., Manocha D.: C-Dist: Efficient Distance Computation for Rigid and Articulated Models in Configuration Space, ACM

Related Publications by the Author

- Chris Joslin, Ding Cai, Won-Sook Lee and Paul E. Beaule. Rapid Collision Detection Technique in the Evaluation of Femoroacetabular Impingement, accepted to Orthopaedic Research Society 55th Annual Meeting, 2009.
- Ding Cai, Won-Sook Lee, Chris Joslin, Paul Beaule: Rapid Impingement Detection and Surface Distance Measurement System for Real-Time Ball-and-Socket Joint Motion Simulation, accepted to the special issue on International Journal of Advanced Media and Communications, 2008.
- Ding Cai, Won-Sook Lee, Chris Joslin, Paul Beaule: Rapid Impingement Detection System with Uniform Sampling for Ball-and-Socket Joint", accepted to the workshop on 3D Physiological Human, in Zermatt, Switzerland on December 1-4, 2008.
- Ding Cai, Won-Sook Lee and Chris Joslin. Rapid Ball-and-Socket Joint Collision Detection, IEEE International Workshop on Medical Measurements and Applications (MeMeA), pp.25-28, 2008
- Ding Cai, Won-Sook Lee and Chris Joslin. Physically-based dynamic cell representation with consistent cell neighbor relationships. Canadian Conference on Computer Science & Software Engineering (C3S2E-08P), in ACM library, pp.125-128, 2008