COMPUTER AIDED INTERTROCHANTERIC OSTEOTOMY PLANNING AND SURGERY SIMULATION

by

PATRICK J. LORD

S. B. in Mechanical Engineering, Massachusetts Institute of Technology (1987)

S. M. in Mechanical Engineering, Massachusetts Institute of Technology (1989)

Submitted to the
Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 18, 1994

Signature of Author _____
Department of Mechanical Engineering
May 18, 1994

Certified by _____
Professor Robert W. Mann
Thesis Supervisor

Certified by _____
Professor Ain A. Sonin
Chairman, Departmental Committee on Graduate Students

COMPUTER AIDED INTERTROCHANTERIC OSTEOTOMY PLANNING AND SURGERY SIMULATION

by

PATRICK J. LORD

Submitted to the Department of Mechanical Engineering
at the Massachusetts Institute of Technology
on May 18, 1994 in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in Mechanical Engineering

ABSTRACT

One of the most common degenerative joint diseases observed in both younger and older subjects is osteoarthritis. Very common in the hip, the disease leads to joint pain, typically resulting in mobility impairments. In the 80's, the favored solution became the Total Hip Replacement (THR) which replaces components of the joint with a mechanical prosthesis. While providing a comparatively simple surgical procedure and an almost certain success rate, THR's have now shown a mean time to failure of 10 to 12 years. Because of the bone stock damage incurred during the original surgery and the challenges presented by revision surgery in patients who are experiencing increases in life expectancy, alternative solutions as well as improvements to THR's are under investigation. Hip osteotomies are now looked upon as a possible alternative for indications of arthritic hip joints. The postponement of an ultimate THR can be obtained with an osteotomy procedure at the onset of the disease by altering the geometric orientation and mechanics of the hip joint and preserving valuable bone stock. However, the three-dimensional geometric complexity of the joint as well as the cartilage thicknesses and femoral head coverage are poorly visualized with planar X-rays, which are traditionally used for planning the necessary joint reorientation in intertrochanteric osteotomies.

The broad goal of this research is to assess the feasibility of a fully computerized predictive surgical planning tool for intertrochanteric osteotomies in the osteoarthritic hip joint. In addition to the development of better anatomical visual aid tools, a specific objective is to provide a computerized prognosis which presents predictive joint geometric and kinematic optimizations to calculate the "best" osteotomy wedge and consequent femoral head coverage. In effect, the computer quantifies outcomes and provides a prognostic scale as to whether an osteotomy is potentially a good alternative to a THR in the specific patient under consideration. To achieve this goal, the system ranges from imaging tools to process CT or MRI scans, including three-dimensional anatomy reconstruction, coordinate system assignment and joint geometry evaluation, to virtual environment databases describing anatomical features and joint kinematic information essential to the surgery simulation.

Results and conclusions drawn from this research contribute to a better understanding of the issues involved in the implementation of a computer aided intertrochanteric osteotomy planning system. Both cadaver and *in-vivo* studies provide a first assessment of the feasibility of such a system for presenting better prognoses that complement and augment clinician expertise and training. Finally, the qualitative and quantitative information derived from this research are relevant to the broader field of medical virtual environments determining the future of computerized surgery systems.

Thesis Supervisor: Robert W. Mann, Sc.D.
Title: Whitaker Professor Emeritus of Biomedical Engineering

MEMBERS OF THE COMMITTEE

Dr. Robert W. Mann, Sc.D.
Whitaker Professor Emeritus of Biomedical Engineering
Senior Lecturer
Department of Mechanical Engineering
Massachusetts Institute of Technology

Dr. David C. Gossard, Ph.D.
Professor
Department of Mechanical Engineering
Massachusetts Institute of Technology

Dr. Neville Hogan, Ph.D.
Professor
Department of Mechanical Engineering
Department of Brain & Cognitive Sciences
Massachusetts Institute of Technology

Dr. Derek Rowell, Ph.D.
Professor
Department of Mechanical Engineering
Massachusetts Institute of Technology

Dr. Gregory A. Brown, M.D. , Ph.D.
Lecturer
Department of Orthopaedic Surgery
University of Minnesota

ACKNOWLEDGMENTS

My thanks begin with Professor Robert W. Mann, my thesis advisor. I consider myself privileged to have been able to complete a Bachelor's, a Master's and a Ph.D. under his supervision. I am very grateful not only for his guidance, insight, support and financial assistance but also for the opportunity to work in this unique learning environment he created, the Newman Laboratory at MIT.

I offer my most sincere gratitude to Professor Neville Hogan for his advice, support and many suggestions during the development of this project. I am indebted to him for all of his constructive criticism.

Many thanks to Dr. Greg Brown for his expertise, time and patience for defining with me the main directions of this project and for providing me with all the advice and support critical to completing this degree from his own experience. Your connections and "network" were invaluable to the success of this thesis.

I would also like to thank the other members of my thesis committee. Professors David Gossard and Derek Rowell, who contributed in their own ways. Their participation, comments and suggestions were very helpful towards the completion of this thesis.

Dr. Slobodan Tepic and Dr. Stephen Bresina, from the AO/ASIF Research Institute in Davos Switzerland, deserve special thanks for the help they provided me with the manufacturing of stereolithography models. In this effort, my friend and colleague for now many years, Dr. Keita Ito, was instrumental. I thank you for your interest and friendship.

During these years working on my thesis, I believe I was able to keep my sanity thanks to friends, both within and outside the lab, too numerous to all be listed here. However, and at the risk of forgetting someone, I would like to mention some very special friends: Eric Twietmeyer, Mark Wang, Michael Murphy, Justin Won, Michael Goldfarb, Peter & Grace Mansfield, and Sylvain & Susan Lévesque. Thanks to you all.

Even though thanks do not even begin to repay the debt I have accumulated, much heartfelt thanks to you, Alisa, for your help, companionship and love which made the difference for the completion of this thesis and kept some sort of balance in my life.

Last, but not least, I wish to thank my Parents who, in a way, made it all possible and kept me going with their unconditional support, encouragements, efforts and contributions. I could not have done it without you, and best of all, I can tell you now that I have graduated: I'm finally done!

This research was performed in the Eric P. and Evelyn E. Newman Laboratory for Biomechanics and Human Rehabilitation at MIT. Funding was provided in part by the following: the MIT Mechanical Engineering Graduate Student Fellowship Program, the National Institutes of Health grant #5R01-AR40036-03, and the U.S. Dept. of Education, National Institute on Disability and Rehabilitation Research, Rehabilitation Engineering Center grant #133E80024.

To my Parents

For your love and support

CONTENTS

Title Page 1

Abstract 3

Acknowledgments 7

Contents 11

Chapter 1: Introduction 15

1.1 Issues 15

1.2 Objective 18

1.3 Background 19

1.4 Approach 23

1.5 Thesis Overview 24

Chapter 2: Anatomical Modeling 27

2.1 Patient Specific Anatomical Modeling 27

2.1.1 Computerized Tomography 28

2.1.2 Magnetic Resonance Imaging 29

2.1.3 Data Acquisition Protocol 31

2.1.4 Computing and Display Environment 33

2.2 Segmenting 35
2.2.1 Thresholding 35
2.2.2 Contouring: Edge Detecting 38
2.2.3 Results 41
2.2.4 Discussion 45
2.3 Three-Dimensional Surface Meshes 46
2.3.1 Introduction 47
2.3.2 Method 47
2.3.3 Results 50
2.3.4 discussion 51
2.4 Conclusions 54

Chapter 3: Principal Axes 57
3.1 Introduction 57
3.2 Method 59
3.2.1 Theory 61
3.2.2 Simulations 65
Sampled Nodes 66
Full Contours 71
Full Cross-Sections 76
3.3 Experimental Results 81
3.4 Discussion 82
3.5 Conclusion 83

Chapter 4: Cartilage Thickness 85
4.1 Introduction 85
4.2 Joint Geometry 86
4.2.1. Theory 86
4.2.2 Simulations 89
4.3 Cartilage Thickness Estimation 95
4.3.1. Theory 95
4.3.2 Simulations 96
4.4 Anatomical Results 97
4.4.1 CT Cadaver Data 98
4.4.2 MRI IN-vivo Data 105
4.4.3 Method Validation with MRI Information 107
4.5 Discussion 109
4.6 Conclusions 110

Chapter 5: Osteotomy Simulation 113
5.1 Introduction 113
5.2 Osteotomy Planning 113
5.2.1. Theory 114
5.2.2 Simulations 118
5.2.3 Anatomical Results 120
5.3 Surgery Simulation 122
5.4 Conclusions 125

Chapter 6: Conclusions 127
6.1 Review 127
6.2 Conclusions 128
6.2.1 Anatomical Modeling 128
6.2.2 Principal Axis Theory 129
6.2.3 Cartilage Thickness Estimation 130
6.2.4 Osteotomy Planning 131
6.3 Future Work 132

Bibliography 135

Appendix A: Anatomical Data 145
A.1 CT Cadaver Data 145
A.2 MRI In-Vivo Data 152

Appendix B: Segmentation Software 159

Appendix C: Stereolithography Software 167

Appendix D: Principal Axis Software 181

Appendix E: Cartilage Estimation Software 185
E.1 Joint Geometry Optimization 185
E.2 Cartilage Thickness Optimization 188

Appendix F: Osteotomy Simulation Software 195
F.1 Joint Reorientation Optimization 195
F.2 Osteotomy Wedge Calculation 199

CHAPTER
ONE

INTRODUCTION

1.1 ISSUES

Mankind has been fascinated by the study of human anatomy since ancient times. Even though one might casually say that healing remedies, or medicine as we know it today, started as early as the stone age, the advances in anatomical understanding achieved since the 19th century have been decisive. Only recently has technology reached a level providing more accurate and precise studies of the human anatomy, thus resulting in a better understanding of its complexity and leading to dramatic improvements in the effectiveness of medical treatment.

More precisely, the field of medical imaging has made several important contributions to society. The development of computer graphic capabilities in conjunction with tomography has opened a completely new range of fields of application for computers. Evidence of this new trend is certainly the progress achieved in medical imaging where three dimensional reconstruction and observation techniques are now providing clinicians with anatomical structures formerly unobtainable without physical dissection. This computer power has also been directed to speeding up the processing of information, making it thus readily available. Despite these advances, there are still limitations in the use of medical imaging. While widely used as a visualization tool to better display anatomical structures or isolate

pathological findings, only a few applications have seen it take the extra steps into diagnosing, predicting and suggesting a treatment procedure. Neurosurgery [42], craniofacial reconstruction [53, 82, 120], plastic surgery [35, 98] and radiation therapy are perhaps the medical fields that have benefited the most from computers in the treatment planning and implementation phases. In contrast, among all the medical disciplines, orthopedic surgery has perhaps seen least the effects of the computer age, aside from the improvements in prosthesis and surgical tool design resulting from engineering CAD/CAM, FEM and manufacturing processes [104] along with the implementations of custom implant design [47]. While computers now offer surgeons better visualization tools to pinpoint problems more accurately [2], they play a negligible role, if any, in the actual surgical procedures performed. These rely heavily on the skill of the surgeon as well as his/her ability to extrapolate from the virtual pictorial information to the actual patient anatomy during surgery. Most other medical fields also tend to limit computers to their data acquisition task and leave the diagnosis and prognosis to clinicians, this in part being the result of the failure to date of so called computer medical expert systems as well as the apprehension of taking the human factor out of the loop. In addition, due to lack of computer task flexibility and adaptation and often inadequately designed interfaces when presented with a previously unrecorded situation, clinical end-users ignore the systems due to time inefficiencies. Therefore, one seldom sees computers used beyond their data gathering and manipulation ability in the medical environment. Clinicians base their surgical decisions and prognosis on empirical data and personal observation without the use of potentially decision altering quantitative tools.

In the treatment of the osteoarthritic hip joint, where articular cartilage has been damaged, it was widely accepted that a reduction of the excessive joint pressure would stop the illness. To obtain this pressure reduction a reorientation of the joint geometry was necessary and achieved through a procedure better known as a hip osteotomy. Among several techniques, the intertrochanteric osteotomy emerged as a favorite based on the biomechanics of the hip and the regeneration of the cartilage as a result of the alterations of the joint mechanical conditions (load and stress) [39, 80, 96]. The intent of the surgical intervention through intertrochanteric osteotomy is to delay or prevent the progression of osteoarthritis by changing the geometric orientation and mechanics of the hip joint. While visually and conceptually simple, the actual geometric complexity of the three dimensional problem is often overlooked. The use of simple planar X-rays during preoperative planning does not provide the means to accurately describe the areas of damage in the osteoarthritic joint. Hence, the surgeon's experience as well as his/her understanding of the functional anatomy

and relationship between living cartilage and mechanical stresses become paramount to the quality of the procedure and the potential results. These problems, in conjunction with the advent of the Total Hip Replacement (THR), have contributed to the decline of the hip osteotomy in favor of the THR when faced with the treatment of osteoarthritic joints [34].

The demise of the intertrochanteric osteotomy can also be attributed to sociological and economical factors such as the hospitalization duration and costs, the relative ease of the THR procedure and its short term, comparatively high success rate, the associated royalty payments to surgeons by prosthesis manufacturers and the patient's perceived deference to medical implants. After a rapid expansion in the 80's, over 120,000 THR's are performed every year in North America, while during the same period intertrochanteric osteotomies have faded to less than 1% of all osteoarthritic hip surgery performed today [52]. THR's provide patients with a popular, quick, reliable and predictable fix to hip osteoarthritis. However, they now appear to have a limited lifetime thus precluding this solution for younger active patients. Due to mechanical property deficiencies of the prosthesis itself as well as biological reactions to the implant such as stress shielding, bone resorbtion, loosening, and wear debris (polyethylene macrophagic reactions), THR's offer a mean-time-to-failure of 10 to 12 years in most cases. As patient life expectancy increases, the likelihood of revision arthroplasty increases concordantly. The number of THR failures seen now is a reflection of the number of implants performed during the previous decade. As revision surgery cases continue to rise, with their inherent complications, more and more pressure will be applied to find alternative solutions to THR's especially for the younger patient population affected by early osteoarthritic symptoms.

Osteotomies are increasingly being considered as an alternative procedure to THR's because they are prevalently used with some success for indications of post traumatic malalignments and malunions, femoral neck pseudarthroses with viable head, congenital coxa vara, avascular necrosis, arthrodesis and femoral shortening and lengthening. This rebirth of osteotomies for indications of early osteoarthritis is in part due to their potential success and conceptual simplicity. They offer a solution to delay the course of osteoarthritis at the onset while preserving the bone stock and therefore do not preclude future arthroplasty procedures as the disease progresses. However, the limited experience of most orthopedists combined with the difficulty of imaging hip cartilage degeneration and predicting femoral head - acetabulum coverage throughout the gait cycle after geometric reorientation have caused orthopedic surgeons to choose THR over intertrochanteric osteotomy. They feel they can guarantee a better functional outcome with THR.

1.2 OBJECTIVE

While many reasons can be given as to why intertrochanteric osteotomies are not widely used by the orthopedic community to remedy osteoarthritis, very little has been done to improve the surgical planning and the procedure itself [29, 65, 121]. Unfortunately, even with improvements in the mechanical properties of the hip prosthesis and cement, available data indicate alternative treatments should be sought in specific cases. To correct hip joint disorders surgeons can select several osteotomy procedures in the femoral and pelvic regions. Although deemed important, acetabular osteotomies will not be addressed in this thesis. Among femoral procedures, intertrochanteric osteotomies in osteoarthritic joints will be investigated and be the focal point of this thesis because they are the most likely alternative to THR in the younger, active patient.

Quantitative clinical studies involving the use of conventional X-ray techniques for osteotomy planning have changed little since they first appeared over 50 years ago [22, 41, 56, 75, 76, 83, 84, 95, 124]. Most surgical planning is still performed using frontal and sagittal planar X-rays, or a variation thereof, in order to evaluate critical angles and axes defining the hip joint and the femoral head surface coverage. Then a trial correction angle wedge is estimated using a tracing of the preoperative X-ray film. The intrinsic two dimensional nature of the radiographs immediately emphasizes the limitations these techniques have in characterizing a problem that is inherently three dimensional, including issues of joint congruency, cartilage thickness, and femoral head coverage. Previous studies [24, 28, 30, 122] have attempted to use computers to help in the three dimensional visualization of the joint geometry, but they seldom go further than displaying and manipulating the anatomical information. The primary objective of this thesis will be to assess the feasibility of a fully computerized predictive surgical planning tool for intertrochanteric osteotomies in the osteoarthritic hip joint. Reasons for concentrating on intertrochanteric osteotomies of the osteoarthritic hip joint can be justified by the well defined issues that are presented by this specific problem, and the simplicity afforded by the joint geometry. Such simplification and focusing is indispensable to a preliminary implementation of a computer prognosis expert system and the first step towards envisioned computer aided surgery systems [77, 78, 114]. In addition, narrowing the field to intertrochanteric osteotomies also provides the unique opportunity to address not only the actual surgical procedure planning and outcome, but also the teaching and training of surgical skills.

Finally, the implementation of a computer aided surgery simulation will also present the

challenges of designing and studying the interactions between human, machine and computer virtual environments. Such a system must not only accommodate clinicians, traditionally more inclined towards physical examinations and observations rather than numerical analyses, but also the vast amounts of state of the art medical engineering and technology available today. While computers have the ability to easily handle three dimensional data sets, they are constrained traditionally and practically to display this information in two dimensions via the monitor. How to best present pictorially the information with this dimensional limitation becomes critical to the assessment and prognosis of the joint disease and the ultimate interaction and potential complement between clinicians and computers.

1.3 BACKGROUND

Osteotomies and THRs are not only different in regard of the operational technique, but also in the principle onset of treatment. Whereas the joint replacement is therapeutically at the end of the course of the disease, intertrochanteric osteotomies are joint preserving operations and therefore do not presuppose unsalvageable joints. While the only possible therapy in former times, the readjustment of the bone has been superseded by the joint replacement because of the difficulties in biomechanical planning and exact planned executions of osteotomies in all three dimensions. To better understand the complexity of such surgical procedures it is important to describe the anatomical concepts behind such methods as well as the planning techniques that have been traditionally used in clinical settings.

An intertrochanteric osteotomy is performed between the trochanters on the femur bone. These trochanters, referred to as the greater and the lesser trochanters, are prominent landmarks of bone that afford leverage to the muscles which rotate the thigh on its axis in the proximal area of the femur bone. The greater trochanter is situated on the outer superior side of the neck at the junction with the upper part of the shaft. Its anterior border provides attachment at the outer part to the *Gluteus Minimus*, while its inferior surface gives attachment to the *Vastus Externus* muscle. The summit of the lesser trochanter, located at the inferior base of the femoral neck, gives insertion to the tendon of the *Ilio-psoas*. While the shaft of the femur is comparable to a cylinder of compact tissue, referred to as cortical bone, hollowed out by a large medullary canal, the proximal end sees the separation of the bone layers into cancelli, which project into the medullary canal and finally obliterate it. This results in an intertrochanteric bone structure mostly consisting of cancellated tissue invested by a thin compact cortical layer and arranged in the direction of greatest stress to better support the pressure exerted by the weight of the body and the tensions created by the muscle

actuations. The interface between the femoral head and the acetabulum of the pelvis makes the hip joint. It mainly consists of two cartilage covered articular surfaces connected via the *Ligamentum Teres* and protected by the joint synovial fluid membrane.

To minimize the overlap of thin or damaged cartilage a reorientation of the femoral head position can be calculated and converted to a wedge of bone which once surgically removed will provide with the intended correction. To prevent avascular necrosis, the wedge is removed from the intertrochanteric region instead of the neck area where blood supplies to the femoral head are located. Figure 1.1 presents an example procedure where the corrections in abduction and flexion are achieved by removing a single biplanar wedge of the distal fragment.

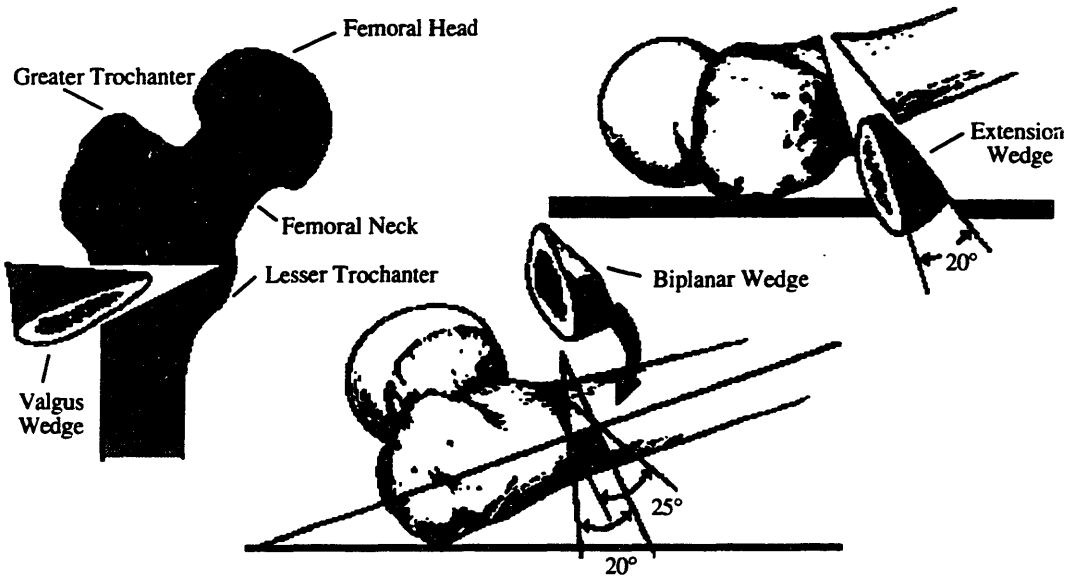


Figure 1.1: Valgus Intertrochanteric Osteotomy [99]

Intertrochanteric osteotomies most commonly fall into two distinct classes, *Valgus* and *Varus*, depending on the lateral direction of the reorientation achieved by the procedure. Figure 1.2 illustrates these two classes of osteotomies. In addition corrections are also performed with intertrochanteric osteotomies to compensate for reorientations in flexion-extension as well as internal-external rotation. Other osteotomies techniques approach the reorientation by performing a single cut along the bone shaft and rotating the proximal segment with respect to the distal one [107]. These methods have been particularly appropriate for the correction of long bone deformities.

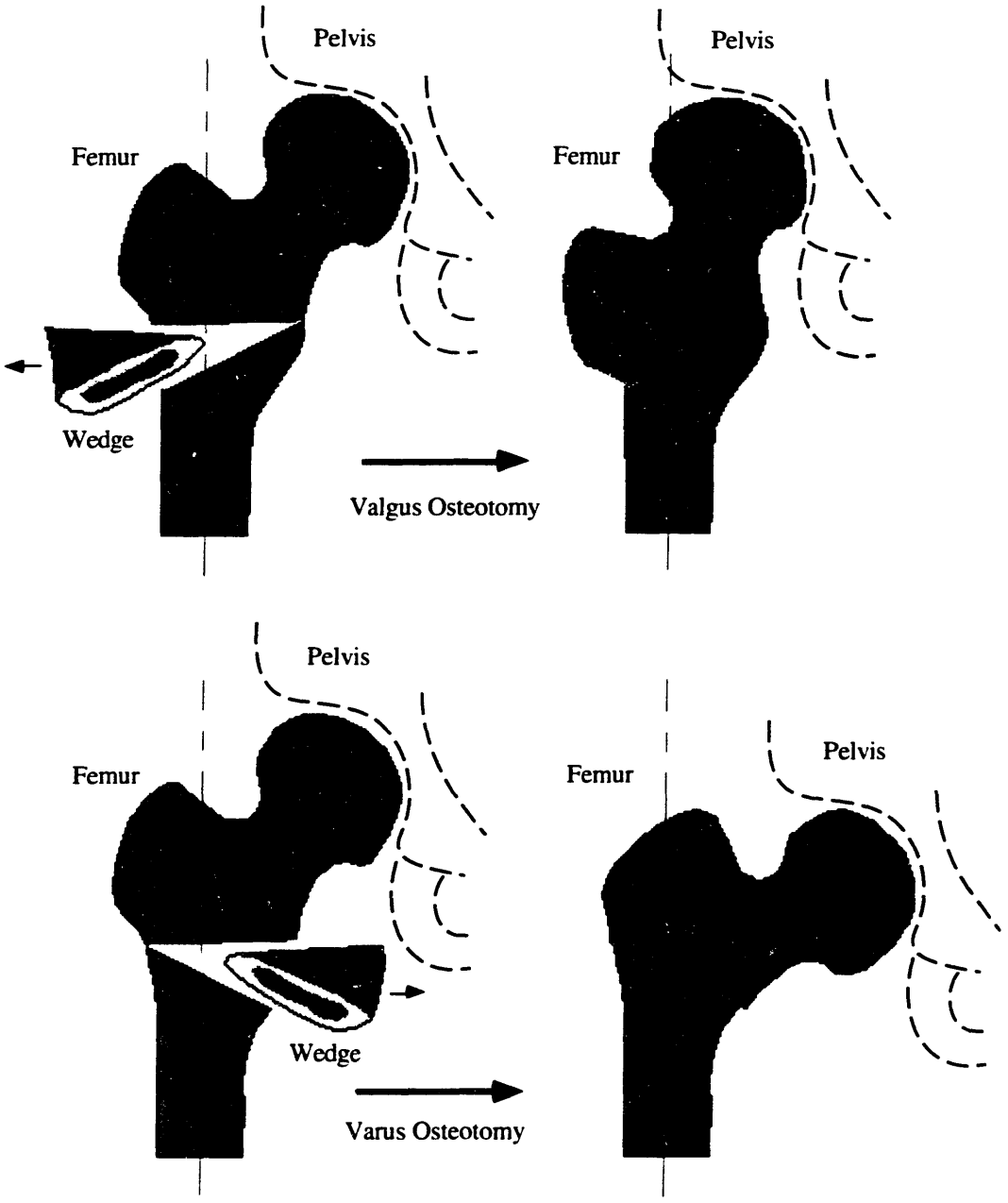


Figure 1.2: Valgus versus Varus Intertrochanteric Osteotomies

The most common osteotomy surgery planning technique is based on the acquisition of preoperative planar X-ray films and the extensive use of tracing paper. Figure 1.3 depicts the essence of the planning method. In (A), tracing paper is overlaid on top of the X-ray film to capture the outline of the femoral bone. The osteotomy site is chosen at the level shown (in between the trochanters) for two reasons: the site must be in a region of maximum cancellous bone and a strong medial buttress must be left on the distal fragment to insure proper and optimal healing. The tracing is cut along this line and the two pieces are rotated laterally

until they overlap at the chosen angle (B). The overlapping wedge is then cut away, leaving the femoral tracing corrected (C). If appropriate, a similar procedure can then be performed in the sagittal plane to compensate for flexion - extension and anteversion - retroversion reorientations [128]. Finally the overall osteotomic biplanar wedge can be inferred from the two correction wedges calculated.

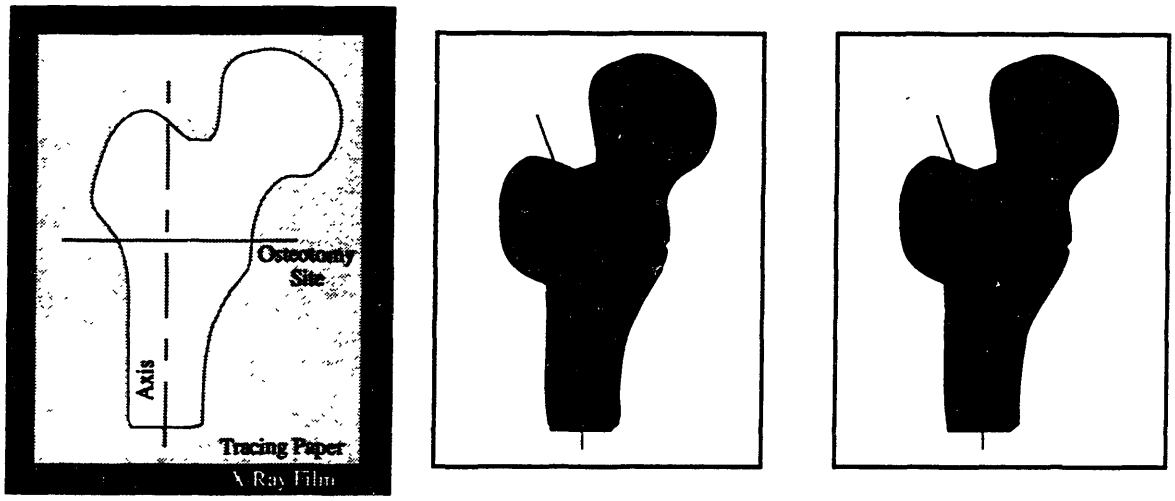


Figure 1.3: Traditional Intertrochanteric Osteotomy Planning

One can readily see that this preoperative technique relies heavily on the knowledge of the three dimensional nature and quality of the cartilage at the joint and the computation of the appropriate reorientation that minimizes "poor" cartilage overlap. Neither are available from the two dimensional X-ray films which explains the difficulties encountered by orthopedists during the surgery planning phase as well as the poor success rates of such procedures. Although many positive results had been obtained through a better understanding of the mechanisms by which osteotomies work [10, 36, 37, 66, 72, 93, 97, 123], this procedure has gone out of favor and been replaced by the THR. However, since osteotomies work well in selected cases [12, 95, 108], since healing of the diseased articular surfaces has been demonstrated in selected cases [7, 22, 95], and since the limitations of THRs have become apparent, it seems reasonable to re-evaluate the role of osteotomies in the treatment of younger persons with osteoarthritis of the hip [81].

The availability of high speed digital computers now allows us to implement models to simulate and better understand the dynamics involved through the repositioning of the femoral head. With the help of such computer modeling, many research efforts [30, 34, 59, 85, 90] have lead to a better understanding of the mechanical changes associated with surgical procedures which alter the relationships of muscles relative to joints. However, at

the present time, these models have shown their predictive limitations. Most often based on the osteotomic redistribution of forces at the articular surface advocated by Pauwels [95], these models fail to capture the ability of muscles to adapt to their new length over time. Therefore, the assumption that an osteotomy will not alter the ability of muscles to generate forces is arguable. In addition, other investigations [15] also suggest that proximal femoral osteotomies do not have much effect on the forces on the articular surface and conclude that a change in load at the hip is unlikely to be one of the reasons for the beneficial effects when the results of the operations are good. This comes in direct contrast with Pauwels' work which states that a varus osteotomy of 30 degrees reduces the resultant hip contact force by 25% by increasing the abductor muscle moment arm and that a valgus osteotomy of 30 degrees increases the resultant hip contact force by 25% by reducing the abductor muscle moment arm.

Intertrochanteric osteotomies have a number of mechanical and biological effects on the hip joint which make the determination of whether an osteotomy succeeds almost impossible and therefore preclude a clear indication for the procedure in a specific patient case. By focusing our modeling on the cause and the reason of the hip joint problem, the pain resulting from the interarticular cartilage damage, we believe we can obtain a better predictor of surgery recommendation and outcome. Armed with the assumption that the osteotomy planning should be based on compensating for the "poor" cartilage quality overlap, the approach in this investigation concentrates on identifying the three dimensional hip joint geometry and cartilage quality and implementing the analytical and computational tools necessary to calculate the optimal osteotomic wedge.

1.4 APPROACH

Computerized Tomography (CT) and Magnetic Resonance Imaging (MRI) have become popular and common processes in hospital settings. These techniques offer the *in vivo* measurement of the anatomical geometry by permitting cross sectional scans of the pathological joint. The approach taken in implementing the surgery simulation virtual environment database will require a series of scans, allowing inputs from either imaging system to describe patient specific joint anatomy. With experimental data collected with the finest resolutions of CT or MRI both in cadaver and *in vivo* studies, a method has been devised to extract the quantitative data necessary to characterize the pathology of the joint. Three dimensional models can then be reconstructed [71] from the serial scans to present qualitative displays of the bone structure and thus an important visual aid in comprehending

not only the geometric but also the kinematic relationships in the joint. To this effect, surface skin kinematic data, is collected with a Selspot optoelectronic camera system and processed with the MIT Newman Laboratory TRACK software [4, 61, 73, 74, 79, 88, 89]. Next, automatic evaluation of femoral and acetabular cartilage thickness is performed. Joint congruency and cartilage maps with areas of damage are then assessed to evaluate the feasibility and potential success outcome of an intertrochanteric osteotomy. For example, recommendations are provided to clinicians as to whether an osteotomy is a potentially good alternative to a THR in the specific patient case under consideration. Finally predictive geometric and kinematic optimization algorithms have been implemented to quantify results and ascertain an optimal osteotomic wedge.

1.5 THESIS OVERVIEW

This section contains a brief summary of the remaining chapters. The chapters describe separate investigations which highlight the necessary steps that have been combined to create our computer aided intertrochanteric osteotomy planning system. Each chapter has been written to present the approach and material of a specific issue or method and includes its own introduction, analysis, simulations, results and conclusions.

Chapter 2 pertains to the anatomical data acquisition and the three dimensional reconstruction of the patient specific skeletal information. Two different scanning imaging techniques are presented along with their respective advantages and the protocols adopted. The techniques implemented for processing the biological images and extracting the pertinent information about the hip joint are then described. Finally, *in-vivo* and cadaver experimental results are shown and computer algorithms and techniques used to visualize and handle the reconstructed information are depicted. Chapter 3 concentrates on the development of a systematic anatomical coordinate system selection. A method was implemented to define a coordinate system for the anatomical data that is independent of the coordinate system of the data acquisition machine used. This allows us to consistently compare anatomical data sets collected with different systems as well as at different time intervals. This technique also offers a systematic approach to compute the realignment of data necessary to compensate for misorientations in the scanner systems used.

Chapters 4 investigates techniques to evaluate hip joint kinematic congruency and cartilage estimation. The congruency of the hip joint needs to be assessed prior to any osteotomy procedure recommendation. This analysis is achieved through the fitting of optimal known

geometries to the anatomical information. Then, using optimization methods, the cartilage thickness both on the femoral head and inside the acetabulum is estimated. Since this information is not readily available from the scanned information, it was necessary to develop a procedure that allows areas of damaged cartilage to be clearly identified and the joint kinematic center calculated. The algorithms implemented are verified with simulated data and the results presented. Issues of three dimensional cartilage data visualization are also addressed. Chapter 5 pertains to the reorientation of the femoral head inside the acetabulum to maximize the "good" joint cartilage overlap. The optimization algorithm used to compute this angular correction is presented along with test results from simulations. These results are then translated to define optimal osteotomic wedges through an interactive approach that incorporates clinician expertise if so desired. Finally, the approach used to provide clinicians with a qualitative assessment of the potential osteotomy is described.

Overall conclusions are presented in Chapter 6 and summarize the results obtained from Chapters 2 to 5. The integration of each part into a complete computer-aided surgery planning system is discussed. The significant contributions of this research work are identified along with the specific issues described in section 1.1 that have been addressed. Finally, future research directions and recommendations are provided at the end of this chapter.

CHAPTER
TWO

ANATOMICAL MODELING

2.1 PATIENT SPECIFIC ANATOMICAL MODELING

In connection with surgical reconstructions and simulations, the skeletal surfaces of human anatomy have no known equation permitting their direct duplication and use by computer methods. It is necessary to measure anatomical features at a discrete number of points and to develop methods to build analytical representations of the data. In addition the difficulty in applying a uniform scaling process to a single data set to fit the description of any patient (we all have legs of different length, shapes and aspect ratios), forces the acquisition of patient specific anatomical information and its transformation into a usable three-dimensional database.

Biological structures are very different from a typical engineering object, which often has smooth analytical surfaces and constant material properties. For bone, the geometry problem is more general than simply describing the surface of an unusually shaped object. A cross-section of bone is not uniform, but has large variations in material density and strength. In addition, the non homogeneous distribution of material density within that object can also be of concern. Fortunately, research and clinical practice has begun to make use of remote sensing modalities such as computerized tomography (CT) and magnetic resonance imaging (MRI) which can provide this information.

The medical need to see inside the human body from the outside has been met for many decades by recording the differential absorption of X-rays. A major deficiency of the standard method of radiography is its inability to discriminate among overlapping structures. This deficiency was remedied by the development of X-ray computerized tomography. CT records X-ray data from many different directions and reconstructs mathematically the information to yield cross-sectional views of any part of the body. Although CT scanning is an extremely useful diagnostic tool, its use of X-rays, even in small doses, carries a risk of doing physiological harm to sensitive organs (i.e.: reproductive system). More recently MRI, a new technique for obtaining cross-sectional pictures through the human body without exposing the patient to ionizing radiation has gained popularity not only because it provides information comparable to CT but also because it discriminates more sensitively between healthy and diseased tissue. Both imaging techniques are readily available within the clinical setting and are the basis today of much medical diagnoses through the two-dimensional images they provide of a patient specific cross section.

Since it is important for the reader to have a general understanding of the basic concepts behind imaging techniques such as CT and MRI, brief overviews are presented in the following sections.

2.1.1 COMPUTERIZED TOMOGRAPHY

Wilhem Konrad Röntgen discovered X-rays in 1895, and his imaging technique has been used since then to image tissues areas with large density differences. For example, a shadow projection of a bone can be clearly distinguished from its surrounding soft tissue structures which do exhibit density differences sufficient enough to allow for distinct identification. Moreover, and as noted above, superimposed areas produce an image that is the result from multiple shadows. In 1963, A. M. Cormack conceived the idea of X-ray transmission along lines parallel to a large number of different directions to produce a sequence of X-ray transmission profile and proposed its use in the field of medical imaging [26]. However, it is only in 1976 that his work was adapted by Robert S. Ledley to an Automatic Computerized Transverse Axial scanner [70]. ACTA was the first CT scanner with the capacity to produce clear and detailed cross sectional images over the entire human body and was immediately adopted to complement and help physicians with their analyses and diagnoses.

CT scanning can be achieved in different ways. The most common involves both translation along the longitudinal axis of the body and rotation around the cross section of the body that is being examined. For each image at a specific translation coordinate, the mechanical

scanner rotates 180° in 1° or 2° increments to allow for highly collimated X-ray beams to pass through the body section at each position. The next image is generated similarly by acquiring data at a new translational location. While some of the X-rays are absorbed by the body, others pass through and are detected by a scintillation crystal. The intensity of the beam is measured and recorded to create an intensity profile. The absorption pattern projection depends on the sum of the absorption coefficients of the tissues through which the beam passes. By combining many such absorption patterns along the 180° rotation of the scanner the computer can then reconstruct the contribution of sub volume areas inside the cross section being studied. Then, by assigning a value from a gray scale gradient to the intensity of absorbed X-ray transmission by a part of the body, an image can be created which visually identifies the structures within the cross section. Subtle variations in density such as in muscles, ligaments, fat and derm layers are difficult to discriminate while bone provides a high density contrast which can be clearly visualized. It is important to also note that for the particular concern of this investigation, CT does not allow for clear imaging of the cartilage surfaces which because of their particular nature are not easily distinguishable from the surrounding soft tissue.

While it has been argued that conventional X-rays expose a patient to higher dose of radiation than CT scans do, the use of numerous thin, contiguous CT scans needed to produce three-dimensional anatomical reconstructions raises concerns regarding its invasiveness to patient health. However, it is true that the high collimation of the X-ray beam does not result in much scatter and only exposes a specific and localized area of the body. In addition the cross sectional slices are usually taken far apart to prevent the cumulative radiation exposure. Nonetheless, in cases of younger patients and in areas of high concentration of sensitive organs it is preferable to minimize, if not suppress, the potentially harmful effects of this imaging technique.

2.1.2 MAGNETIC RESONANCE IMAGING

The experimental foundations of nuclear magnetic resonance (NMR) were first discovered by Felix Bloch [9] and Edward Purcell [101] in 1946 and resulted in a Nobel prize award in 1952. While traditionally used in physics and chemistry to investigate molecular composition or monitor metabolic reactions, it is not until recently that NMR was introduced to the medical imaging field and renamed magnetic resonance imaging (MRI). The potential offered by MRI to distinguish between benign and malignant tissue was discovered in 1971 by Damadian and represents the initial application of NMR to a medical field [27]. Lauterbur then develop an improved MRI technique which used magnetic field gradients and

tomographic reconstruction techniques to produce images [69]. His work was the basis for the implementation of modern MRI systems which yielded the first cross sectional images of human anatomy in 1977.

Since MRI uses magnetic fields as the source for its imaging method, it does not expose patients to any ionizing radiation and is often considered a very safe alternative to CT. This assumption is commonly accepted in clinical settings and studies have demonstrated that the health risks associated with MRI are minimal [14, 19, 20, 126]. The sources of potentially harmful health effect have been identified and limited to three: static magnetic fields which are less than 2 Tesla (20,000 gauss), changing magnetic fields of 5 mT (50 gauss) amplitude and less than 100 Hz frequency, and radio frequency heating up to 4 W/kg. For most MRI investigations, the exposure to these effects have been found to be well below the known thresholds for health effects. However, while both MRI and CT produce cross sectional images of human anatomy, there are some fundamental differences between the two imaging techniques. Primarily, the signal measured in MRI is related to the density and relaxation times of the atoms being exited and not to the mass density of the tissue as in CT. The most common magnetic resonance imaging is based on proton nuclear magnetic resonance and involves the study of the density of hydrogen atoms in the water molecule because of their intrinsic NMR sensitivity and high concentrations in biological material. As a result, MRI measures several parameters that, once combined, define the intensity of the image pixel at the particular location of the molecule being examined. These parameters are the longitudinal or spin-lattice relaxation time, the transverse or spin-spin relaxation time, and the density of the magnetic nuclei [13, 32]. By combining these parameters with different and suitable weighting techniques, it is possible to create an image that provides the contrast necessary to highlight specific tissue areas proportionately to the molecular environment being imaged.

The principles behind MRI imaging rely on the use of a static magnetic field to align atoms with naturally occurring magnetic dipoles together with a short pulse of a radio frequency (RF) magnetic field that perturbs the atoms [102]. The maximum signal can be observed immediately after the initialization of the RF field where the magnetic moments of the excited nuclei are all lined up in their new orientation along the direction of the superimposed RF magnetic field. This signal then decays over time in two ways. The first one corresponds to a slow down of the spin system until thermal equilibrium is reached within the lattice structure and is known as the spin-lattice relaxation. The second one represents the dephasing or misalignment of the nuclear magnetic moments which create a local magnetic field and is referred to as the exponential decay of the spin-spin relaxation. The sum of the

static magnetic field and the surrounding fluctuating fields determines the total field at a nucleus. Finally, the greater the number of magnetic nuclei present, the larger the NMR signal. This density parameter determines the contrast of the NMR image.

The fact that MRI images can be most easily generated from the resonance of hydrogen nuclei is fortunate because the human body is 70% water. It also indicates that MRI has the ability to image soft tissues and distinguish among different ones such as muscles, tendons, ligaments and cartilage with a higher contrast than CT can and explains why most MRI scans are now used for brain and spinal cord studies to assess the existence and location of tumors. MRI is therefore well suited to the visualization and determination of body segment mass and inertial properties [17]. It should be noted though that since bone material holds little water content, and thus fewer hydrogen atoms, its imaging via MRI does not have the accuracy that is offered by CT. However, and most importantly, MRI does not expose subjects to harmful radiation and therefore lessens the potential health risks when compared to computerized tomography.

2.1.3 DATA ACQUISITION PROTOCOL

Since both CT and MRI are readily available in clinical settings, this investigation implemented an approach to three-dimensional anatomical modeling of the hip joint that is independent from the data acquisition method chosen. Since each imaging modalities has unique advantages and limitations, this approach allows clinicians to choose the most appropriate imaging technique depending on a compromise between the patient's age, health risks and benefits, and the accuracy sought for the three-dimensional reconstruction. However, it is the belief of this investigation that MRI should be the preferred *in-vivo* method over CT because of the necessity to acquire multiple thin slices in an area of the human anatomy that would be most sensitive to X-ray over-exposure. Even though today CT provides higher resolution and more accurate images of skeletal information than MRI, future developments in MRI technologies related to extensive testing for its applications throughout the body will overcome present drawbacks. A discussion of the protocol used for CT on cadaver material and for MR imaging in *in-vivo* experiments follows.

A protocol was designed to acquire data with the highest possible resolution and density. Standard GE transaxial CT and MRI scans were obtained on a cadaver leg and subjects respectively. The resolution achieved with CT was 310 μm with a 512 x 512 pixel image. The MRI resolution was 930 μm and resulted in a 256 x 256 image. The scanners were recalibrated prior to any experiment with the phantoms provided by the manufacturers. The

body areas under interest were positioned in the center of the scanner field to minimize the effects of nonlinearities particularly evident at the edges of the viewing field. Figure 2.1, the scout view from a CT scan on a cadaver leg, illustrates the "slicing" protocol adopted. Slices with the thinnest thickness (1 mm for CT and 2 mm for MRI) were taken at the hip joint with the shortest distance apart (1.5 mm for CT and 3 mm for MRI) while still preventing signal coupling among slices during image reconstruction. For MRI, the pulses associated with the relaxation times T1 and T2 were chosen to maximize the imaging of bone structures. Finally, slices set further apart were obtained only for qualitative three-dimensional modeling along the distal area of the femur and in the upper part of the iliac crest. More specifically, the data acquisition protocol specifies for 5 mm contiguous slices from the crest of the ilium to a point just superior to the acetabulum, 1.5 mm contiguous slices (3 mm for MRI) through the femoral head until the inferior part of the lesser trochanter, and then 10 mm contiguous slices down the femoral shaft to a point at least 15 cm distal to the lesser trochanter. In the area of highest slice density, an exposure to around 2.5 rads can be expected with CT scanning.

While using image enhancers has become very common to highlight specific anatomical features during medical imaging, its invasiveness in the case of hip joint imaging makes it impractical as a standard approach. Radio and magneto opaque dyes, such as gadolinium, monosodium urate, or calcium pyrophosphate, would have to be injected in the joint capsule to promote the detection of the inter-articular gap by the imaging technique considered. Studies [8] have shown that such procedures not only present health risks for patients but also do not guarantee better visualization of the joint space. This is in part due to the poor infiltration of the dye inside the joint capsule, and its poor diffusion through the synovial fluid. Therefore, this research project opted to collect anatomical information without requiring the use of image enhancing dyes. This makes for much simpler and safer clinical protocols that limits themselves to the health hazards associated with medical scanning systems. At most hospital medical imaging centers, a data acquisition session can be completed in about one hour and at cost of about $900 for CT and $1300 for MRI. The scanner output can be displayed on the system console where it can be manipulated to adjust for brightness and contrast. The resulting image can then be presented on film, similar to X-ray radiographs, as well as stored digitally on magnetic media for further processing. Appendix A presents the different data sets collected for this study with CT and MRI both on cadaver material and patients.

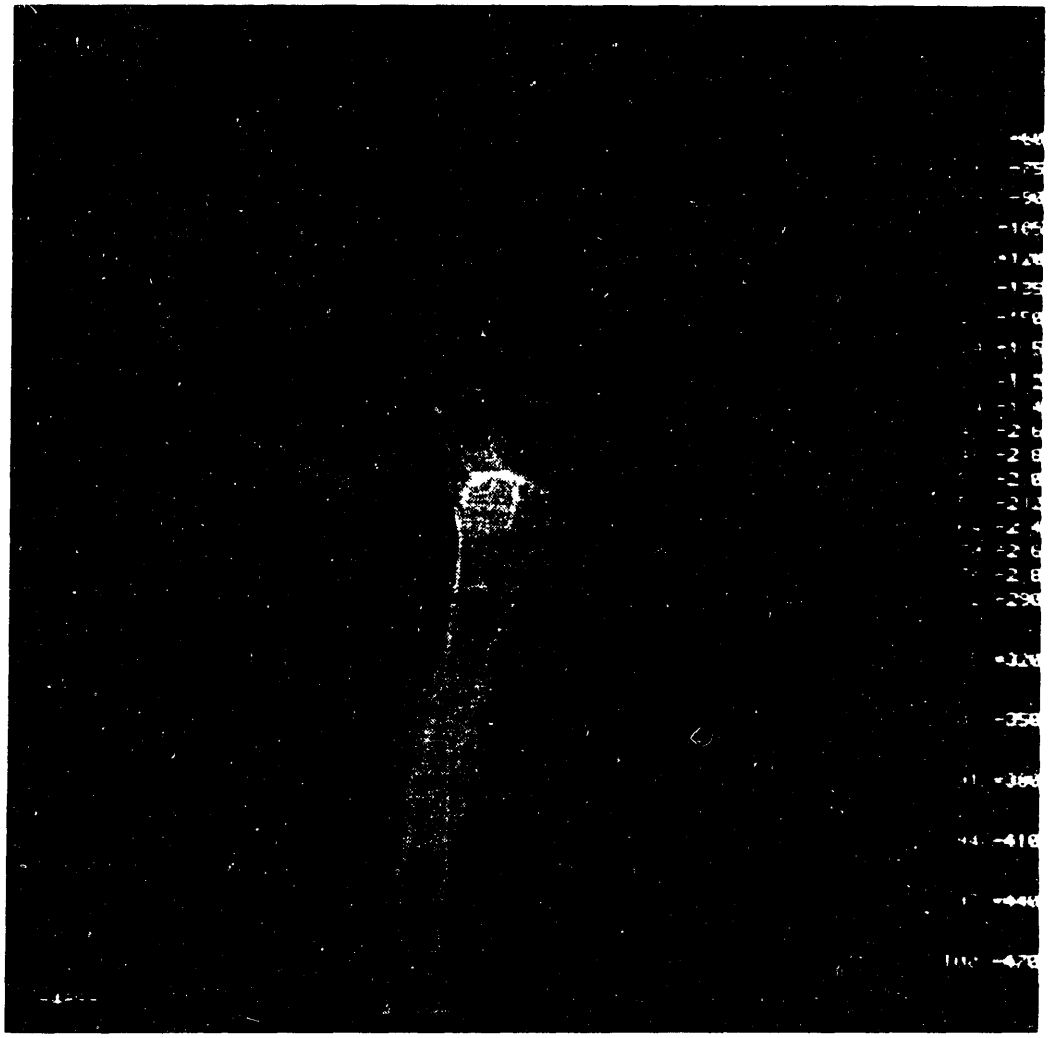


Figure 2.1: Scanner Slicing Protocol (Cadaver CT Scout View)

Standard experiments result in more than 200 images with a storage capacity need of over 100 Megabytes. The images gathered were stored on magnetic tapes and transferred to the workstation system hard disk used for the image manipulation and computer-aided surgery system implementation. Proprietary header information was stripped from the image files which were then stored as binary data matrices in either 512 x 512 or 256 x 256 grids.

2.1.4 COMPUTING AND DISPLAY ENVIRONMENT

A Silicon Graphics Personal IRIS 4D35TG workstation was used to store the raw and converted scan data and generate the 2-D contour data and the 3-D surface mesh reconstruction. The computer, based on a 33 MHz MIPS R4000 RISC microprocessor, provides an integer performance of 35 million instructions per second (MIPS) and 6 million floating point operations per second (Mflops). In addition this computer system offers for

display purposes a graphics raster subsystem based on a high-performance, high-resolution color three-dimensional geometry pipeline engine to handle display transformations (translations, rotations, scaling, and viewing transforms of points, vectors and polygons), matrix manipulations, lighting, material properties, shading, color and texture mapping. The computer operated under IRIX™, a UNIX™ operating system compliant with the AT&T System V Release 4 standard. While a confounding variety of graphics computer hardware commercially available today and supported by different software would have most likely produced the same results and accomplished the same goals, the balance of real-time simulation, computation power, three-dimensional graphics technology, and interactive device I/O offered by SGI workstations was found invaluable for this research project. In addition, the recent standardization of OpenGL™, the three-dimensional graphics programming library pioneered by SGI, among multiple hardware vendors makes the implementation of this computer-aided surgery system cross compatible with other workstations (HP, Sun, IBM, ...).

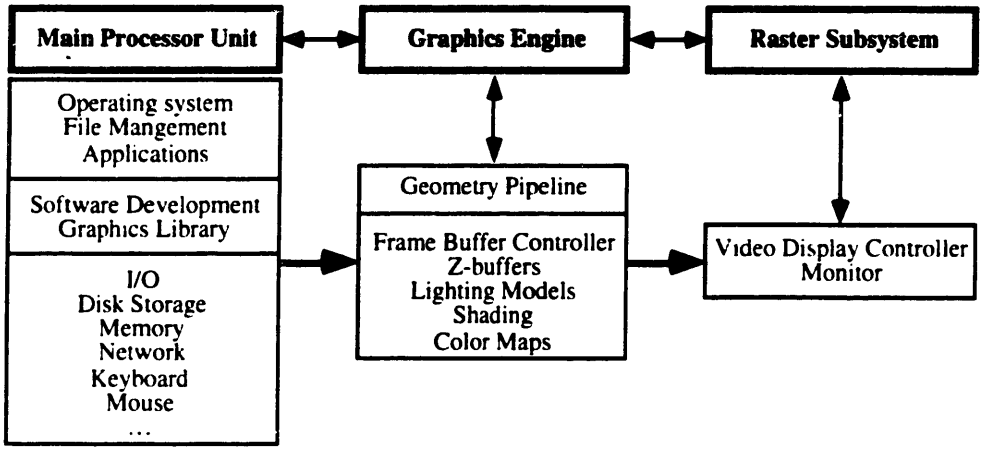


Figure 2.2: Silicon Graphics Computer System Environment

The programming language adopted for the software implementation was C [63]. This decision was due largely because of four major lines. First, the block structure provided by C allows developers to divide large programs into smaller files and functions and to create their own flexible data structures. Second, C is a pointer, or address, based language that, as opposed to value oriented languages, permits the fast and efficient manipulation of large data structures not uncommon to biological databases. In addition, this feature permits the dynamic memory allocation necessary to handle data sets of unknown and variable sizes and optimizes the sharing of computer system memory resources. Third, by logically connecting input devices (mouse, tablet, trackball, ...) to the display transformations, C simplifies the development of code that allows the user to interact in real-time with the on screen

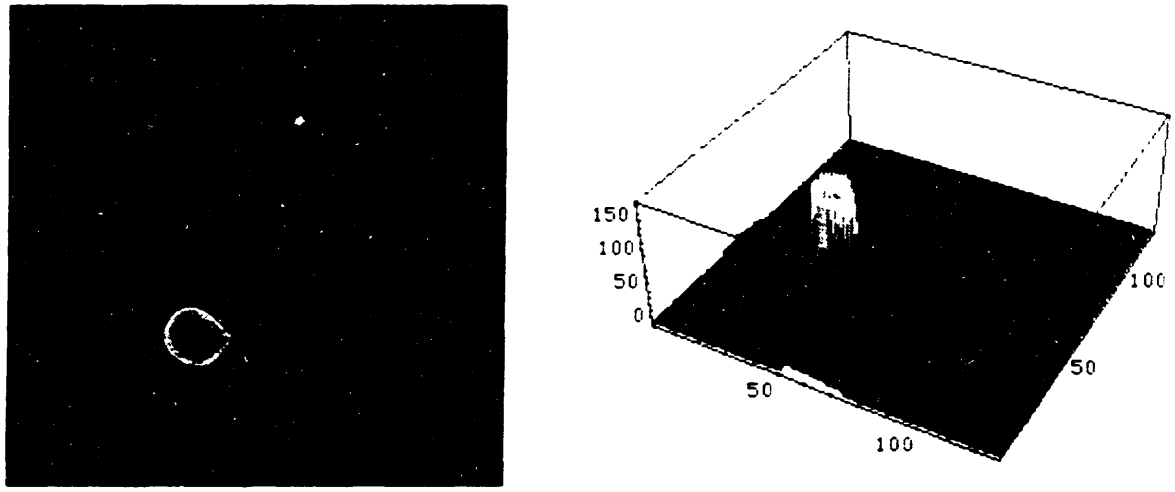
information. Finally, the versatility and recent popularity of C made it the programming language of choice to plan for further future development of the project. Most of the software developed as part of this research is presented in the appendices: the following sections and chapters will direct the reader to the appropriate appendix to use as a reference.

2.2 SEGMENTING

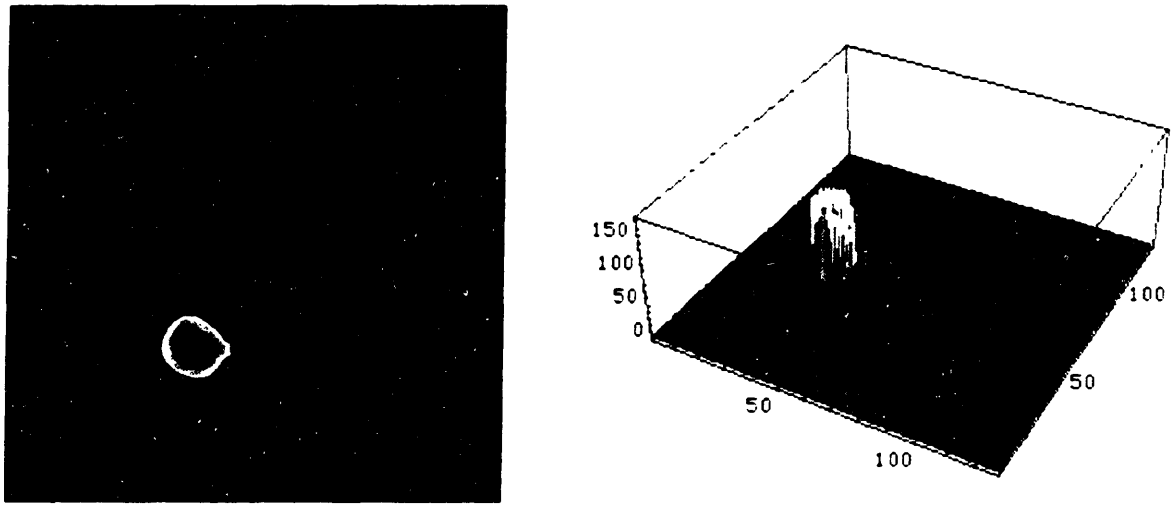
CT and MR imaging provide clinicians with two-dimensional cross-sectional images of the human anatomy that contain vast amounts on information. There is a need to reconstruct this information in three dimensions, but also to extract the information needed out of the images selected. For example, if interested in the skeletal data, one should be capable of discarding the soft tissue information and vice-versa. To this effect, a program to generate viewable anatomical structures from multiple serial CT or MRI scans has been developed as part of this project. While other studies have developed similar techniques, our goal was to identify the soft tissues and bones, isolate the bones, and segment them into individual databases with as little human intervention as possible. If the concept of computer aided surgical systems is to be successful in the clinical setting, anatomical reconstruction software will have to be used by paramedical personnel and accomplished as quickly and as effortlessly as possible. First, we focus on the extraction of skeletal features from scanner images.

2.2.1 THRESHOLDING

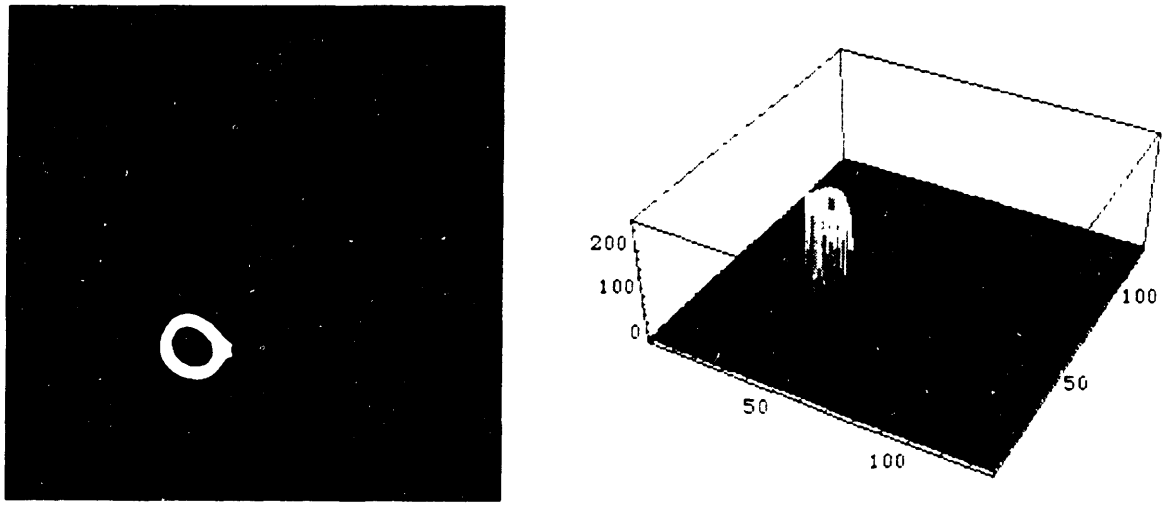
For each image slice, the bone structure was isolated and extracted from the rest of the anatomical information. In the case of CT, each image pixel represents a Hounsfield CT number. Thus, to separate bone tissue from the other tissue densities, a single Hounsfield number, or a range of CT numbers was chosen. Even though MRI contrasts human anatomy differently, the methods used to process these images were inherently the same with some adjustments to the MRI parameters chosen. The main difference results from the fact that some soft tissue structures in MRI images can have pixel intensity levels very close to bone and sometimes even higher. Bone material can then be comprised in a very narrow band of intensities which makes it necessary to eliminate information that can be not only darker but also brighter than bone. However, this process can be done rapidly with some trial and error. Figure 2.3 shows a three-dimensional representation of a femur cross-section image before (a) and after thresholding (b). Finally, the image is transformed with a binary filter to identify bone in white and the rest of the information in the image in black (c).



(a) Raw Medical Image



(b) Image Thresholded to Delete Soft Tissue Information.



(c) Binary Image : bone is in white, anything else in black.

Figure 2.3: Thresholding Process

One will note that, as expected, the CT image presented in figure 2.3 does not discriminate well among soft tissues (the waves that can be observed in the image background are signals corresponding to the presence of the plastic bag used to contain the cadaver specimen). Bone tissues stand out and can easily be identify and extracted by choosing an appropriate CT number. This will vary between experiments, but a Hounsfield unit of 176 was found to be a good thresholding number for bone tissue. The thresholding value in the case of MR images greatly varies with the parameters chosen for the magnetic relaxation times and the radio frequency decay of a particular scan set. Therefore, for each MRI experiment, the images were analyzed to derive the best thresholds.

Several studies have investigated the accuracy and precision of CT and MRI systems. For CT, numerous factors, including beam hardening, partial volume effects, medullary fat content and image display parameters can affect quantitative analyses of the reconstructed images. While thickness measurements of soft tissues can result in errors as high as 30%, Sumner *et al.* [112, 113] found that periosteal bone diameters were accurate to ±1.0% with endosteal diameters only accurate to ±4.6% because of the consistent over-estimation of the medullary dimensions. In addition, Woolson *et al.* [127] confirmed these findings by showing that external linear dimensions (perimeters) of CT based femoral shaft models were within 3 mm (±1.5%) of actual measurements while medullary canal models were consistently smaller than the actual specimen. Finally, Smith *et al.* [111] demonstrated accuracies of 99.0% for CT and 97.5% for MRI based reconstruction of femoral bones by comparing Bridgeport milled slices from image data with caliper measurements directly over the carefully positioned specimen with and without the overlying soft tissues.

Table 2.1: CT and MRI Accuracy Sensitivity

	CT		MRI	
	Elasticity	Variation	Elasticity	Variation
Perimeter	0.019	±0.95%	0.035	±1.76%
Area	0.116	±5.82%	0.179	±8.93%

In this study, we investigated the elasticity or the ratio of percentage change of bone area and perimeter variations over the percentage change in image threshold value since we were interested in the external dimensions of the reconstructed information. Our results, presented in table 2.1, confirm previous study findings. Elasticities were found to be small suggesting that only large changes in threshold values result in small changes in bone perimeters and areas. As expected, results are better for CT than for MRI and area measurements are more sensitive than perimeter dimensions to threshold variations, this being mostly due to larger

errors in bone marrow evaluation. All calculations were performed in pixel base units.

Three-dimensional reconstructions of femur anatomy based on CT or MRI are spatially accurate and unaffected by the presence or absence of soft tissues. Oblique slices can accurately and reproducibly be used to reconstruct and characterize articular geometry. CT provides better assessment of calcification, ossification, and periosteal reaction while MRI represents the most accurate imaging modality for evaluating intramedullary and soft tissue extent. In addition, although CT offers better resolution and accuracy, MRI remains the imaging modality of choice for obtaining *in-vivo* patient specific joint anatomy information due to the reduced health hazards.

2.2.2 CONTOURING: EDGE DETECTING

To further the reduction in the amount of information to be manipulated by the computer, the thresholded images were contoured to extract the edges of bone information. The three-dimensional tile representation provided by stacking all two-dimensional thresholded images provides a clue to the three-dimensional nature of the data and has been extensively used by many investigators [6, 33]. However, the amount of data does not permit real-time manipulation of the anatomical information even with specialized three-dimensional graphic hardware. The adoption of contour representation allows the operator to reduce the information to the essential of the investigation. Other studies [129, 130] have adopted this technique which is often referred to as the ring-stack representation. These one dimensional unit based approaches consist of delineating the surfaces of an object which intersect with a set of slice. This results in a stack of borders that represent the surfaces of the object and the object itself. As in all feature extraction problems the main theme is which are the best cues for extracting the features. This depends on the definition of the feature to be extracted; by previously thresholding the images, the nature of the feature to be extracted involves directly large intensity variations between a region and another one. The obvious cue will be the intensity difference, or a weighted difference of intensities within a specified neighborhood. This approach has indeed been followed for many years by a wide number of authors in the feature extraction imaging field [109].

The edge detection algorithm implemented in this study is mostly based on the first and most intuitive approach which is to extract the pixels belonging to an edge, thus using a gradient transformation on the digital image so as to detect local intensity variations [57]. In addition, gradient based methods, while very sensitive to noise in the image, provide a very good criteria for edge detection in the case of a thresholded binary image, where the background

has been normalized to zero. For continuous two-dimensional functions, the gradient is given by:

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}\hat{i} + \frac{\partial f}{\partial y}\hat{j} \right) \quad \text{(Eq. 2.1)}$$

with its magnitude defined as:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \text{(Eq. 2.2)}$$

and the orientation of the gradient vector:

$$\alpha = Tan^{-1}\frac{\left(\frac{\partial f}{\partial x}\right)}{\left(\frac{\partial f}{\partial y}\right)} \quad \text{(Eq. 2.3)}$$

But when operating on a discrete, or digital, image, x,y and f(x,y) are non negative integer numbers so that the partial derivatives must be approximated with finite difference along the two orthogonal directions x and y, thus resulting in:

$$\nabla_x f(x,y) = f(x,y) - f(x-1,y) \quad \text{(Eq. 2.4)}$$

$$\nabla_y f(x,y) = f(x,y) - f(x,y-1) \quad \text{(Eq. 2.5)}$$

where x and y are pixel positions and f(x,y) is the pixel intensity value. Then, for any orientation α we have:

$$\nabla f(x,y) = f(x,y)\cos(\alpha) + f(x,y)\sin(\alpha) \quad \text{(Eq. 2.6)}$$

Thus the digital approximation to the magnitude of the gradient of f(x,y) can be given as

$$|\nabla f(x,y)| = \sqrt{\nabla_x f(x,y)^2 + \nabla_y f(x,y)^2} \quad \text{(Eq. 2.7)}$$

To simplify this expression, and speed up computation, it generally happens that the digital gradient magnitude is approximated to be either the sum of the absolute values of the two-directional increments or the maximum between these same two increments:

$$|\nabla f(x,y)| \approx |\nabla_x f(x,y)| + |\nabla_y f(x,y)| \quad \text{(Eq. 2.8)}$$

or

$$|\nabla f(x,y)| \approx MAX\left(\left|\nabla_x f(x,y)\right|, \left|\nabla_y f(x,y)\right|\right) \qquad \text{(Eq. 2.9)}$$

These approximation are dependent on orientation as investigated by Deutsch and Fram [31]. The process then involves the evaluation of the digital gradient at the eight pixel neighbors using:

$$f_{l,m}(x,y) = MAX(f(x,y) - f(l,m)) \qquad \text{(Eq. 2.10)}$$

where l and m are the coordinates of the eight neighbors of the pixel located at x, y and depicted in Figure 2.4.

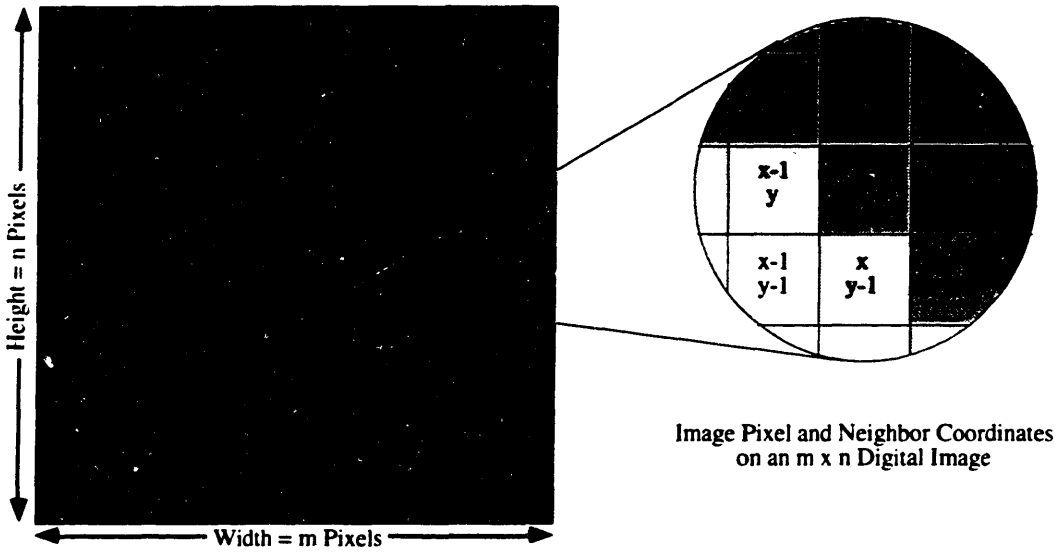
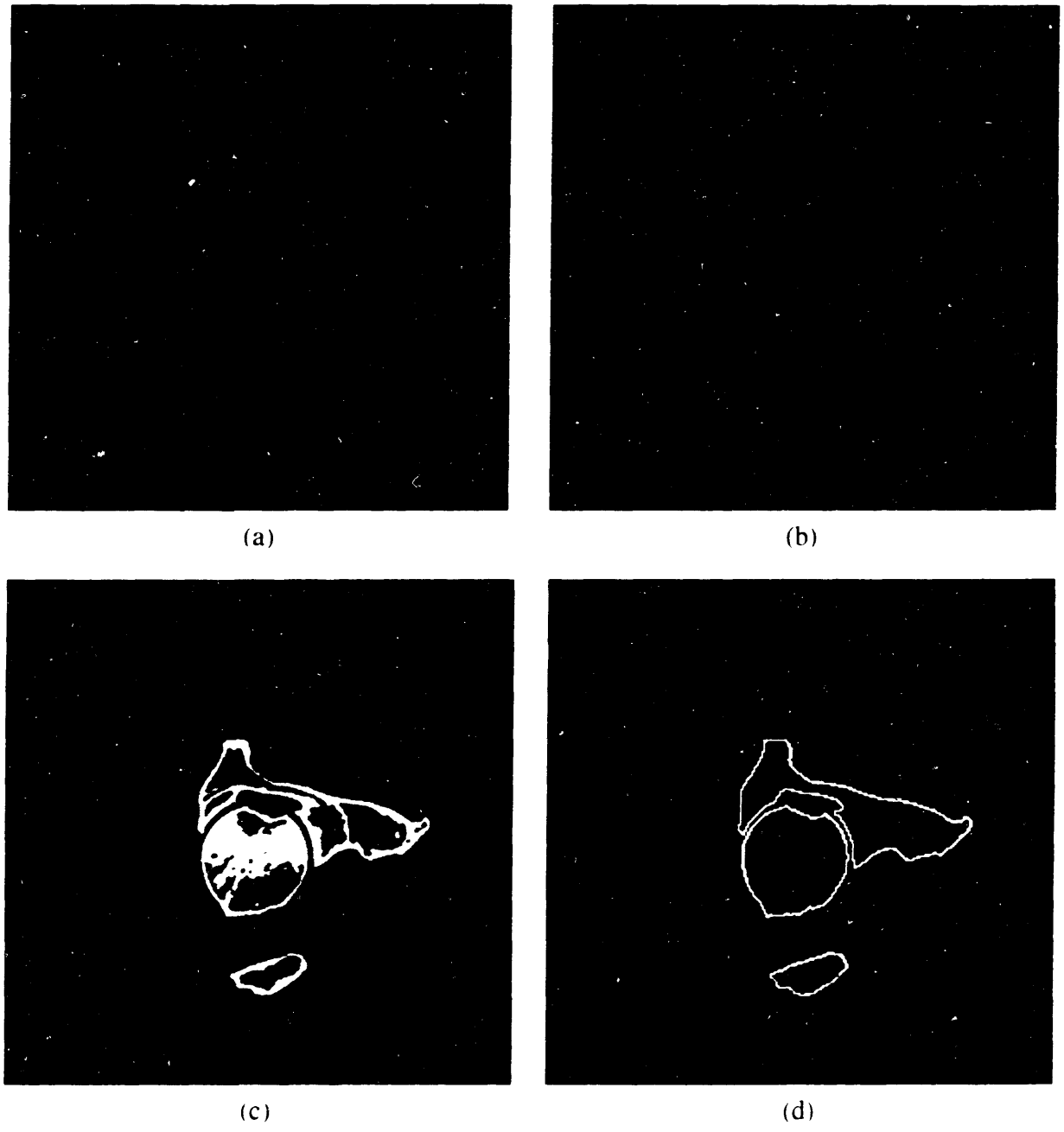


Figure 2.4: Definition of Image Pixel and Neighbors

The orientation of the gradient α corresponds to the gradient magnitude that is found to be the maximum. Then, the process continues by following in this direction the path, pixel by pixel, until a loop is closed thus resulting in a contour. One will notice that the identification of the initial pixel can be critical in the contour extraction process. Several studies have tried to fully automate this process [116], however, this project opted for operator intervention to select the initial point and thus optimize performance. This user action permits not only the selection of internal startup pixels to produce contours of the endosteal area of the femoral shaft but also the identification of contours of different bone structures in the joint area (i.e.: contours defining the femoral head as opposed to the ones defining the pelvis acetabulum.) thus providing bone segmentation with relative ease. In addition, this manual selection, performed by operators directly on the screen with mouse control, limits the evaluation of contours to the surfaces of interest and does not result in the production of a multitude of

contours, a definite drawback of fully automatic contouring algorithms in the presence of complex anatomical shapes.

2.2.3 RESULTS



(a) (b)

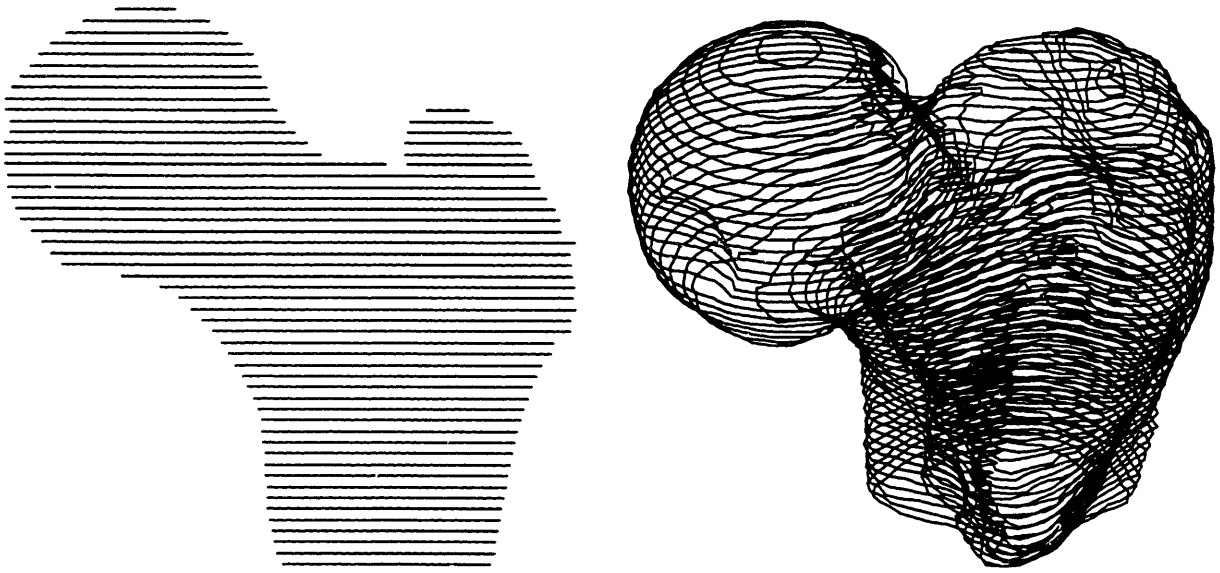
(c) (d)

Figure 2.5: Image Processing from Raw Image (a) to Segmented Contour (d)

The contouring algorithm is an integral part of the overall program used for image viewing and thresholding. For reference purposes, this software is presented in its entirety in Appendix B. The process of segmenting a complete data set can be performed by an operator

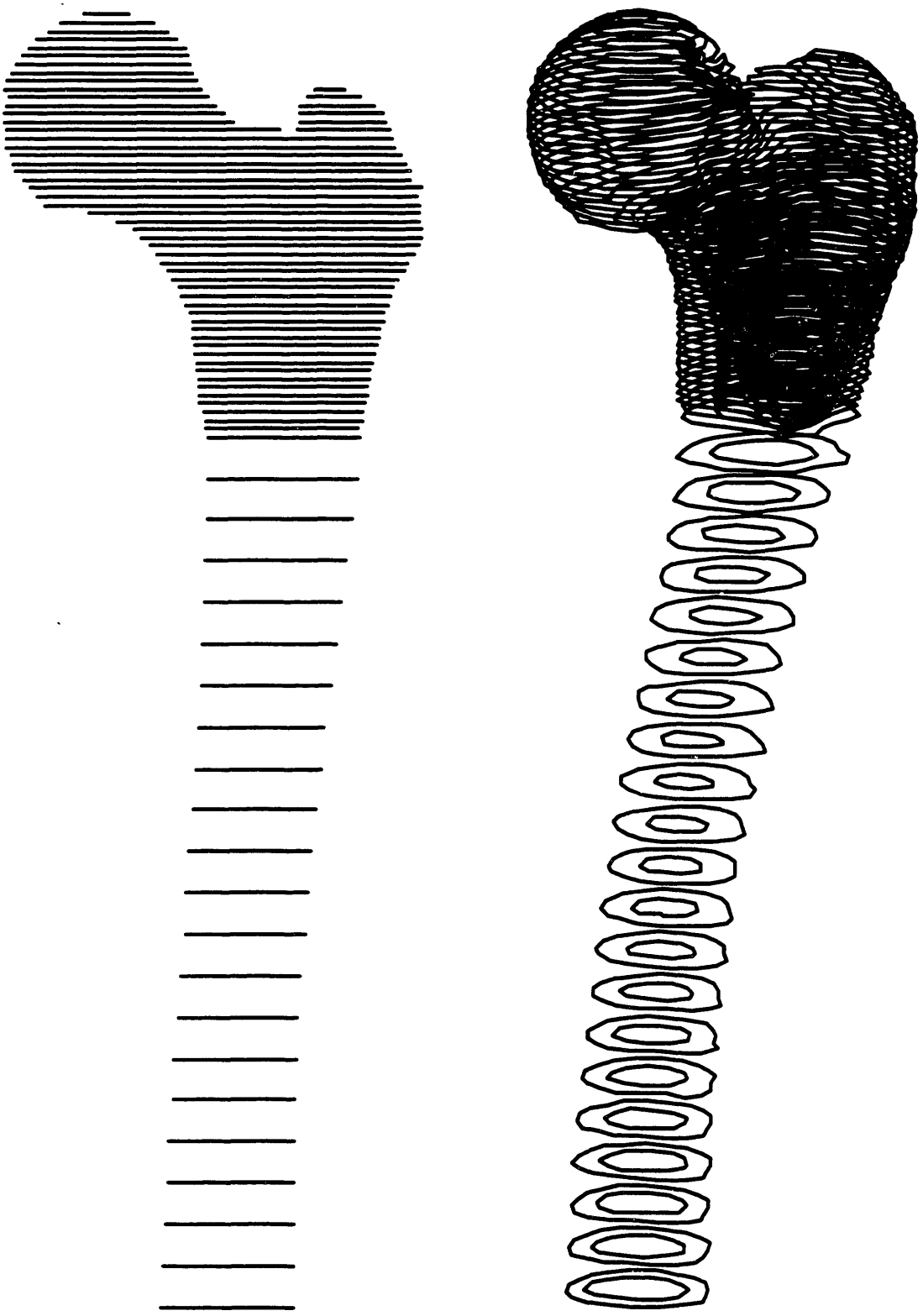
in a matter of few minutes to one hour depending on the size of the data set. The final output, is a set of ASCII character files that describe bone contours for either the femur or the pelvis. As presented in Figure 2.5, a user can with a few keystrokes select a scanner image (a), fine tune the thresholding values to eliminate soft tissues (b), transform the image into a normalized binary display clearly identifying bone (c) and finally contour this bone information and segment the skeletal structures (d). As can be seen in Figure 2.5(d), the data has been reduced to the essential characterization of the bone surface edge information on each image and that pelvic and femoral information can be independently isolated. Through this process, the data set is reduced by a factor of approximately 35, which permits the manipulation of the bone information in real time.

Displaying all successive contours of a scanned data set by adjusting the third dimension for their respective translational scanner position can give a feel for the three-dimensional nature of the information. Figure 2.6 shows contours of the proximal end of a femur and illustrates how a small rotation in the display of the information can result in improved visual understanding of the three-dimensional measurements. It should also become apparent that the distribution of the data is not uniform in all directions. Figure 2.6(a) displays the slices stacked up in the vertical direction and emphasizes the coarser resolution in the translational axis of the scanner. Figure 2.6(b) demonstrates how a simple orthographic rotation allows us to extrapolate the information in 3 dimensions a obtain a better perspective on the data.



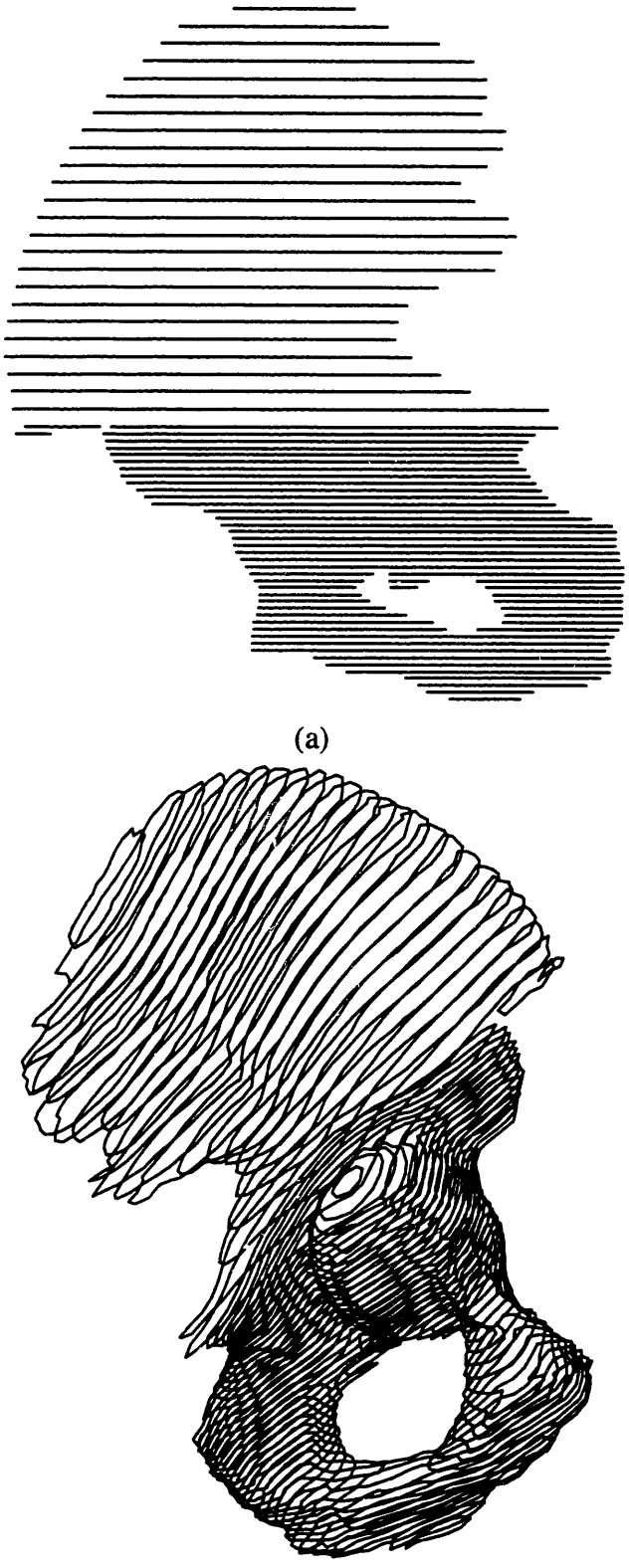
(a) (b)

Figure 2.6: Ring-Stack Display of Proximal Femur in Neutral (a) and Rotated (b) Positions.



(a) (b)

Figure 2.7: Ring-Stack Display of Femur in Neutral (a) and Rotated (b) Scanner Positions with Coarse Shaft Data.



(a)

(b)

Figure 2.8: Ring-Stack Display of Pelvis in Neutral (a) and Rotated (b) Scanner Positions.

Other computer graphic options are available for viewing such contour files. For example, the bone contour outlines may be shaded and then stacked to give a more solid appearing representation. This method is often termed a shaded slice image, and while it has been a favored alternative to more elegant 3-D representations, because of its simplicity it fails to adequately represent the smooth surface of the objects it displays in cases of large inter slice scan intervals. Another option is to use depth cueing where the depth of an object is represented by varying degrees of intensity: objects intended to appear closer to the viewer are displayed at higher intensity. The eye's intensity resolution is poorer than its spatial resolution, so that intensity cueing cannot be used to depict small differences in distance from the viewer. It is quite effective, however, for depicting large differences. Depth cueing has some parallels in real vision: distant objects appear dimmer than closer objects, especially on a hazy day. In large part, though, the viewer's response to depth cueing in graphics seems learned rather than intuitive.

This can be illustrated by Figure 2.7 where stacking scan slice contours can be separated by as much as 10 mm along the femoral shaft. In such cases, using a finer resolution would not provide the user with invaluable quantitative information and only result in longer data collection processes and inefficient data storage techniques. Figure 2.7(b) illustrates that even with substantial rotations, the data still appears very two-dimensional, and errors in perspective parallax cannot be avoided due to visual perception illusions: this being very similar to looking at a wire mesh cube and trying to decide whether it is inside-out or outside-in. Figure 2.8 presents the contoured information for a hemi-pelvis and emphasizes in (b) the difficulty in assessing if the image displays the acetabulum from a medial or a lateral position and thus whether it is a right or a left hemi-pelvis. Nonetheless, ring stack representations offer an adequate tool to verify the quality of segmented anatomical data and allow users to rapidly identify problem areas such as discontinuities.

2.2.4 DISCUSSION

While a number of different methods for edge extraction and anatomical data segmentation have been developed, the solution adopted by this research seems to provide the right balance between computer automation and user intervention. It is common to look upon manual intervention in any digitization process as a disadvantage. However, the inability of fully automatic data segmentation systems to extract just the anatomical information of interest out of a scanned image coupled with their tendency to emphasize the existence of artifacts as anatomical features have rendered them cumbersome. The limited use of an operator as an expert system identifying the relevant anatomical structures, especially under extreme shape

variations, combined with the quantitative ability of computers calculating the geometric descriptions of such features provides a fast and efficient approach to creating three-dimensional anatomical databases. As opposed to fully automatic feature extraction systems that are computer intensive and require careful sorting and editing of the features extracted, the approach presented above can segment most CT and MRI data sets in only a few minutes using a computer under the guidance of a clinical operator. In addition, the implementation of the system has been made very general so as to also provide users with the ability to extract soft tissue information.

CT and MRI were found to allow for accurate measurements of bone structure dimensions, the primary concern of this project. While this method provides invaluable quantitative information, it was however found that simple three-dimensional ring stack representations that are formed by stacking scan slice contours do not provide the necessary visual feedback for clear interpretation of the data. More elegant 3D representations need to be developed as visual aids and should include the use of depth perception, material properties, lighting models, shading techniques and real-time manipulation encoding to best render the anatomy under investigation. To this effect, a description of the approach implemented for this project follows.

2.3 THREE-DIMENSIONAL SURFACE MESHES

While ring stack approaches offer semblance of three-dimensionality, it should be noted that they remain essentially two-dimensional. Each contour is displayed independently from the next one which does not promote the use of three-dimensional operations on the entire object, especially in an area located in between two slices. Basically, there are two classes of approaches to the display of three-dimensional information of human anatomy which are commonly referred to as the two-dimensional unit-based or slice by slice and the true three-dimensional or volumetric rendering approaches. While the first technique discards the density information and produces a surface display, the other one creates space filling images in the form of continuous distribution of densities.

The more clinically "realistic" images produced by volumetric rendering have made it the approach of choice to reconstruct anatomical information in three dimension. It has also been claimed that defects and abnormalities less than a pixel wide can be detected from these images which significantly enhances the planning of surgical procedures. Objects in such images are represented as a set of cubes, often referred to as cuberilles or voxels [55, 119].

Cuberilles of bone are created by identifying the voxels on all CT or MRI digital images that have the intensity of bone. Other tissue structures can similarly be identified. Then using either surface rendering or volumetric rendering [38, 44], the surfaces of the identified tissue can be displayed. In surface rendering, a surface detection algorithm is applied to the objects and the surface is made opaque. While only the surface of each object is drawn, the objects are not segmented and thus separate entries in computer memory. Volumetric rendering uses the intensity of each voxel and assigns it a color and opacity. In a volumetric rendered image, all the information in the initial CT or MRI scans is still present and one can "peel" at will the different layers to reveal hidden information. This representation gives a feeling of thickness especially since surfaces do not have to be opaque. However, the manipulation, editing and modification of such images is computer intensive. Handling of the images such as rotations cannot be performed in real-time which therefore precludes interactive input from clinicians. More often, movies are created frame by frame and then played back to enhance visual perception. With this technique, it is also difficult to differentiate structures, such as the femoral head from the acetabulum as they are not represented by two separate databases. Because of these limitations, this project opted for a different approach that specifically addresses the segmentation and real-time issues.

2.3.1 INTRODUCTION

When only position, shape and geometry of anatomical structures are critical to the visualization, diagnosis and treatment of a particular medical concern, a surface triangulation display method, based on the ring stack information, seems to be most appropriate [92, 119]. This is the case for our intertrochanteric computer aided surgery system, where only the surface geometry of the skeletal information is relevant. The technique depends significantly on prior work by Lévesque [71] and is described below.

2.3.2 METHOD

As in the one dimensional unit based approach, objects are represented by a stack of borders. The next step is to create a surface for the object based on this information. The approach simply consists of tiling in between a pair of borders using triangles. It is critical to represent the borders with carefully selected nodes. This is accomplished by using Fourier descriptors, the coefficient of a Fourier series, and resampling the borders into contours that approximate the original shape. The number of nodes in a contour is defined by some error criteria to the original shape. For example, a complex shape, with high changes in curvature will require a higher number of nodes in its contour representation to best describe it. In addition, it is

often necessary to interpolate contours to achieve an even spacing in the translational scanner axis direction. Figure 2.7 illustrates the issue by showing the difference in vertical spacing along the femoral shaft as opposed to the spacing through the femoral head. Again, using Fourier descriptors, one can easily interpolate and sample a contour to fill a gap.

Generally speaking, the problem of triangulation can be stated as the evaluation of a set of triangular patches which determine the most reasonable model for the three-dimensional surface approximated by a given set of points in space and specified on the surface of the object. The number of possible triangle permutations can be very large, even for relatively few data points. The complexity is greatly reduced if it is assumed that surfaces are obtainable by considering a pair of successive contours at a time instead of all the sample points that define the surface of the object. Defining a precise criterion for the calculation of an acceptable surface is however a difficult process and has only been achieved by either optimization criteria [40, 62] or heuristic approaches [25, 43]. Figure 2.9 illustrates the triangulation process on a contour pair.

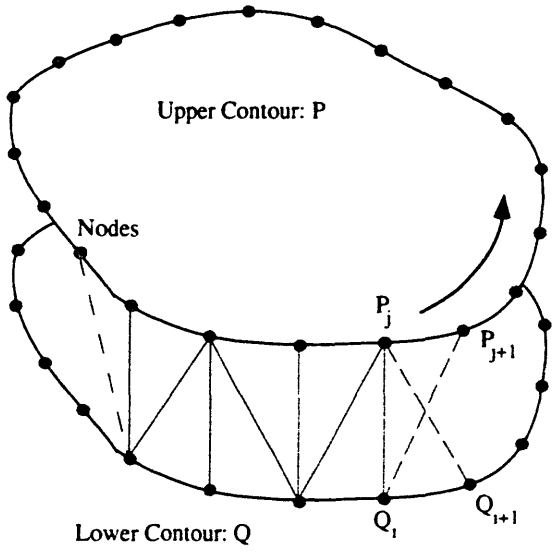


Figure 2.9: Triangulation Concept Process

Keppel [62] proposed to solve the combinatorial problem of triangulation with known graph-theoretical techniques. The problem of finding the best arrangement of triangles is treated by assuming an adequate objective function which optimizes the solution by maximizing the volume enclosed by the tiled surface. This technique works best on contour pairs of similar geometry but fails when two contours present unmatched concave or convex perimeter segments. To avoid these complications, Fuchs *et al.* [40] developed a method that uses a more sophisticated directed graph and defines the optimal surface such that the surface area is minimal. Both of these optimization techniques are not only computer intensive but also do

not guarantee that reconstructed surfaces will be mathematically accurate and esthetically pleasing. In such cases, heuristic methods can be considered as more efficient alternatives.

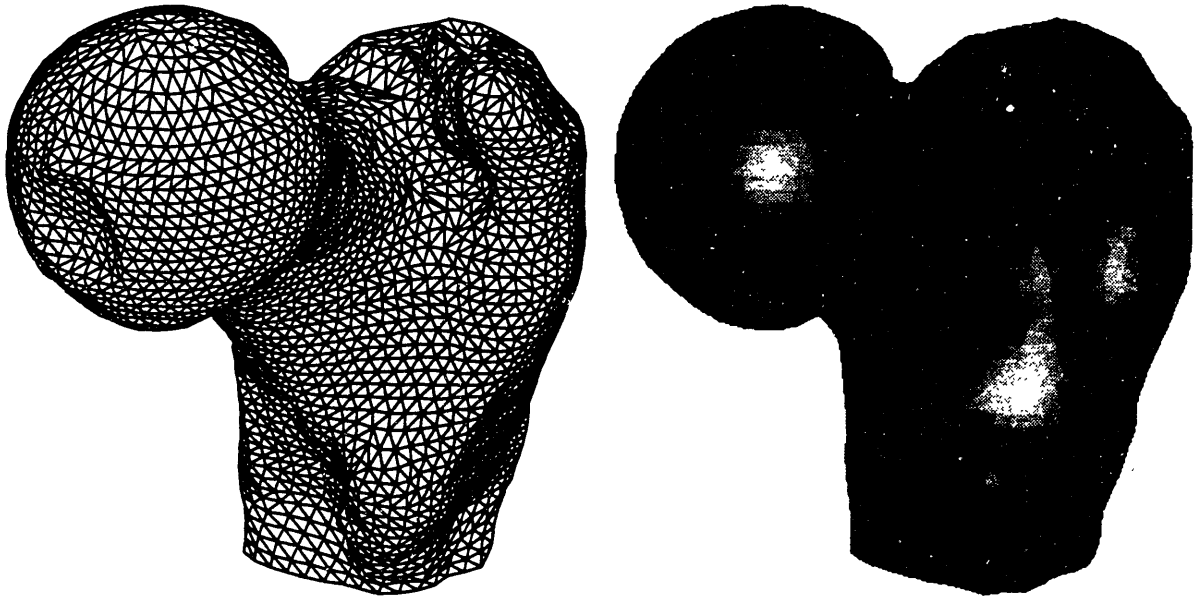
Christiansen and Sederberg [25] developed a set of heuristic methods that perform best on contour pairs which are similar in size and shape and are mutually centered. This is similar to the set of contours that describe a femoral shaft for example. Their approach maps adjacent contours on the same unit square, connects the nodes of one contour to their nearest neighbors on the other contour and results in only $O(m+n)$ operations where m and n are the number of points on each contour respectively. This yields performance results far superior to the two previous method which require $O((m \times n)^2)$ and $O((\log_2 m) \times (mn))$ operations respectively. Ganapathy and Dennehy [43] have proposed another heuristic scheme where each contour is transformed such that the length of its perimeter is normalized to 1. Tiles are then added to the surface in such a way that the absolute difference between the sum of the upper normalized perimeter and the sum of the lower normalized perimeter is minimized at all times. The choice of the pair of points from which to begin the process can be selected by considering the two points that have minimum distance between them. This method eliminates the constraint of having contours mutually centered or even parallel to one another. Finally, in addition to relaxed geometrical constraints, the algorithm only requires $O(m+n)$ operations which makes it attractive for the surface computation of large anatomical structures.

The method adopted for this project uses as a basis the criteria developed by Ganapathy and takes advantage of the fact that the anatomical information has already been segmented, contoured and stored in ring-stack format. The objective is to build topologically acceptable surfaces and to construct triangles as close as possible to equilateral, so that the triangulated surfaces can be used for real-time display of the anatomical structures. However, because of the complexity of some anatomical contours, Ganapathy's method can yield ill-defined triangles that are inappropriate for display purposes, especially if lighting and shading models are to be used. To address this issue, improvements provided by Lévesque [71] compensate by interpolating along part of a contour in order to obtain well-defined triangles. In addition, his work provides a solution to the problem of triangulating across bifurcating structures, a common occurrence in three-dimensional pelvic and femoral skeletal reconstructions. This triangulation of bifurcation is one of the major difficulties of surface representation techniques. A bifurcation occurs on a structure when the number of contours from one slice to the other increases by one. It has been found very difficult to automate the triangulation and to interpolate at a bifurcation mainly because the contour description contains very little information about how a structure bifurcates. Using human operator intervention to help the

computer define the surface of the bifurcation can speed up the process dramatically. This interactive approach relies on the user's knowledge of the structure modeled to design the bifurcation while minimizing the loss in automation. Further details as well as a comprehensive description of the algorithms used can be found in Lévesque [71].

2.3.3 RESULTS

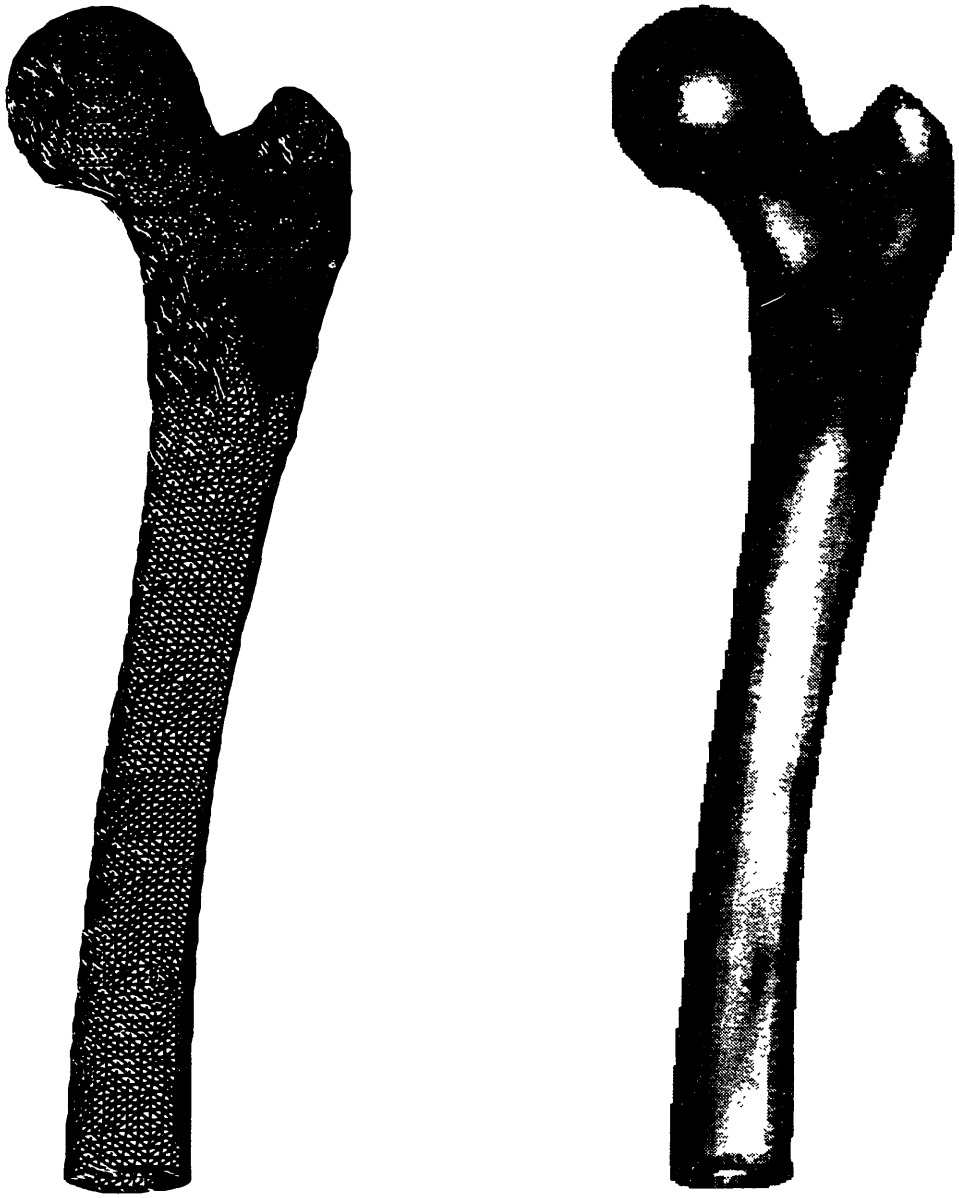
Figure 2.10 to 2.12 present three-dimensional femoral and pelvic reconstructions. Algorithms were implemented to display the information in real-time on the computer CRT. They are listed in Appendix B. One will notice the complexity of the reconstruction which for a simple data set can yield over 18,000 nodes and 35,000 triangular patches. Simple hidden-line removal and Gouraud shading algorithms were implemented to provide a more realistic display with most parameters of normals to patches and material properties pre-computed.



(a) (b)

Figure 2.10: Triangulated Proximal Femur (a) Hidden-Line (b) Gouraud Shading.

The surface structures presented are closed surfaces which is of particular importance when calculating and evaluating the surface of the medullary canal inside the femoral shaft (Figure 2.11). The computer data structures used to describe the triangular surface reconstruction are a simple list of nodes with their respective three-dimensional coordinates and list of patches defined by three nodes. Issues of capping the end of structures, such as the top of the femoral head, are also automatically handled by the software.

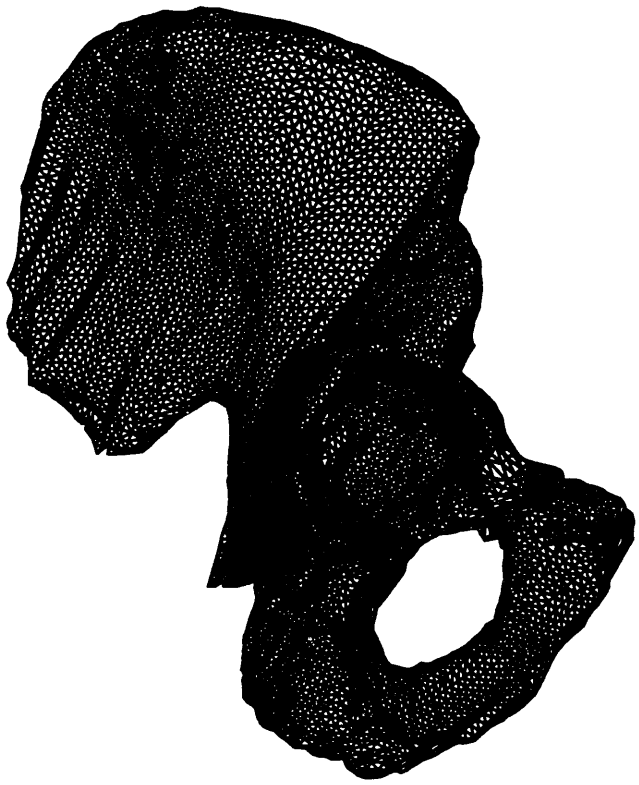


(a) (b)

Figure 2.11: Triangulated Femur with shaft (a) Hidden-Line (b) Gouraud Shading.

2.3.4 DISCUSSION

The method presented above to reconstruct three-dimensional surfaces of anatomical information from contours provides a good balance between automation, real-time display, and accuracy. The triangulation problem resides in the fact that contour lines do not contain sufficient information regarding the system of gradients associated with the surface they describe. Therefore using a human operator as an anatomical geometry expert system, allows



(a)

(b)

Figure 2.12: Triangulated Pelvis (a) Hidden-Line (b) Gouraud Shading.

for speedy identification and reconstruction of complex surfaces, including bifurcations. One can safely predict that the computational speed drawbacks associated with other techniques such as volume rendering will be solved in the future. However, today the speed factors associated with three-dimensional reconstruction and database real-time manipulation have a direct influence on the overall cost factors. If one wants to keep costs down, increasing the number of patients per day becomes a critical issue. In this context, the flexibility and speed offered by surface rendering techniques clearly outweighs their automation shortcomings with current hardware limitations.

Another important aspect of the surface mesh representation is the possibility to create actual models in three dimensions from the surface data using for example a stereolithographic process. It has been found that such "plastic" models play an important role as visual aids for clinicians. The production of accurate bone models from preoperative CT or MRI scans provides a means of better surgical planning of not only osteotomies but also of complex total joint replacements. The ability to visualize bony structures by producing solid models helps surgeons understand the corrections needed and create individualized templates to ensure precise bone cuts. An algorithm, presented in Appendix C, was implemented to automatically create stereolithography object files that can then be re-sliced along any axis to allow for the construction, layer by layer, of a model. Example reconstructions of a femur and pelvis data set are presented in Figure 2.13.

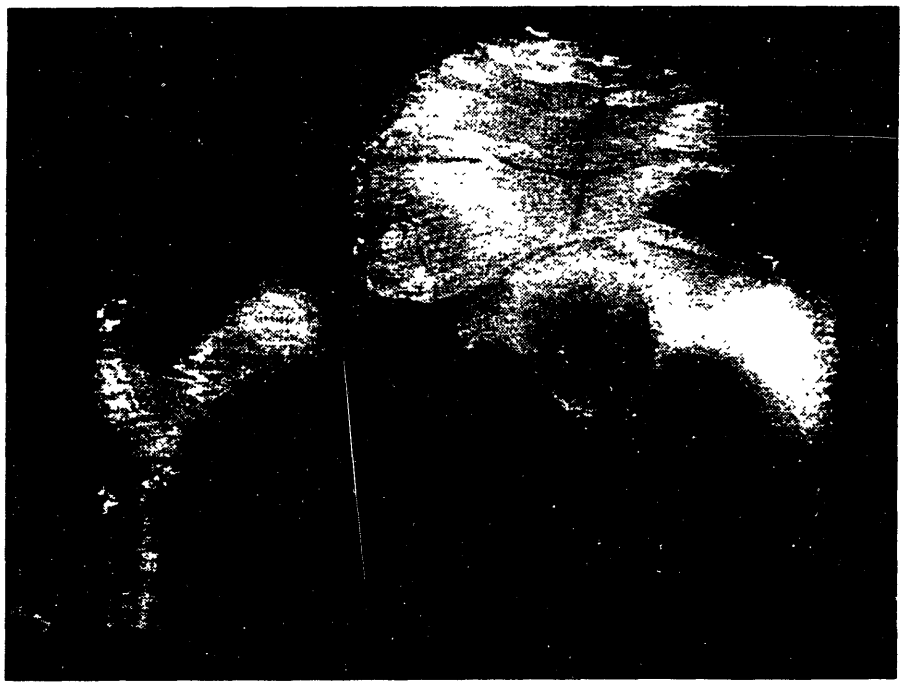


Figure 2.13: Stereolithographic Reconstructions of Femur and Hemi-Pelvis.

Stereolithography is a state-of-the-art process for making accurate prototypes directly from CAD databases without any tooling or machining. It offers a welcome shortcut in the design process by allowing fast prototyping of new components and therefore rapid feedback. Using standard CAD digital data, the laser-based process "grows" prototypes from vats of ultraviolet-sensitive liquid polymer in a matter of hours. Stereolithography performs the mathematical equivalent of slicing a three-dimensional part in horizontal cross-sections between 0.005 and 0.020 inch thick (highly detailed portions of an object require thinner layers to preserve a good surface resolution). To produce the part, a laser traces these cross-sections in succession onto the surface of a photosensitive liquid polymer. The polymer solidifies when struck by the laser's intense beam of ultraviolet light. After a cross-section is "printed", an elevator lowers the hardened portion below the surface of the liquid. The laser then prints the next cross-section directly on top of the previous one. This process repeats until the entire three-dimensional part is formed. The prototype is then pulled from the vat and exposed to intense UV light for a few minutes to complete the curing process and fuse the layers. Stereolithography can form complex shapes and geometries and is therefore very well suited to recreate anatomical structures. The software implemented as part of this project translates surface mesh reconstructions of anatomical data into stereolithography digital input which can then be sent to a service bureau for manufacturing[1].

2.4 CONCLUSIONS

The anatomical modeling technique presented in this chapter tries to address several issues that are critical to the acquisition of three-dimensional patient specific anatomy. A system has been devised which is capable with limited user intervention of analyzing CT or MRI images, of extracting the anatomical features of interest and of reconstructing the information in three-dimensions. This permits the creation of crucial anatomical databases that can be used to display structures in real-time on a computer screen or to directly manufacture polymeric models of the skeletal information. In addition, the system was also implemented to be a general purpose three-dimensional anatomical feature reconstructor and its functionality can be extended to the segmentation of tissues other than bone.

Beyond the qualitative display of the data, the protocol implemented by this project to acquire patient anatomy provides sufficient resolution and accuracy to promote further

[1]The stereolithography models created for this research were realized courtesy of Dr. Stephen J. Bresina on a custom built stereolithography system at the AO/ASIF Research Institute in Davos, Switzerland.

quantitative analyses of bone geometry and joint congruency. This is an absolute requirement for any implementation of computerized surgery simulation systems and we believe the approach presented here provides an excellent solution optimizing both accuracy and efficiency.

Most observers expect advances in memory and processing technology to lower hardware costs to the point where interactive editing of "true" three-dimensional models is possible. While such capabilities are now being investigated, the actual practice will require significant advances in software and hardware tools. The breakthrough would enable surgeons to rehearse surgical procedures on a computer model, effectively cutting through the tissue layers with an electronic scalpel in a manner similar to actual surgery. Unfortunately, the overwhelming size of anatomical data sets does not currently permit the implementation of this vision. However, and in the meantime, falling hardware costs, increases in computing power, improved system functionality, and innovative surface mesh reconstruction such as the one presented here, have combined to render real-time three-dimensional medical imaging systems a reality and a key part of modern medical analyses and thus a foundation for the initial implementation of computer surgery systems.

CHAPTER
THREE

PRINCIPAL AXES

3.1 INTRODUCTION

The importance of assigning a logical and valid coordinate system to a set of anatomical information is often overlooked. Most studies assume that the coordinate system provided by the imaging technique is adequate. Thus, little effort has been expended verifying whether such an assumption is valid, or even sufficient, to study anatomical data in a systematic manner. For most general clinical inspection of anatomical information the issue of a well-defined coordinate system is not of major importance: such investigations are usually more qualitative than quantitative. However, calibration and orientation questions definitely arise when considering anatomical data taken at different times or with different imaging systems. Simply put, the fact that it is impossible to replicate the exact position or orientation of a patient inside a scanner system from one data acquisition session to the next, makes inconceivable the collection of comparable data.

This issue becomes paramount when assembling three dimensional databases of anatomical information for comparative evaluation or for input into computer aided surgery applications. While issues of positioning and scaling can be addressed with simple algorithms to evaluate the scaling factors and translational corrections needed at the object centroid, the three degrees of freedom that describe the orientation of the object in space are much more

difficult to calculate. These rotation angles must be defined so misalignments in the data can be compensated. Since such misalignments are inherent to any medical imaging scanner system, see Figure 3.1, a coordinate system must be defined from the anatomical information. The evaluation of this *local* coordinate system can provide a standardized approach to analyzing anatomical geometry and a universal reference frame for surgical procedures.

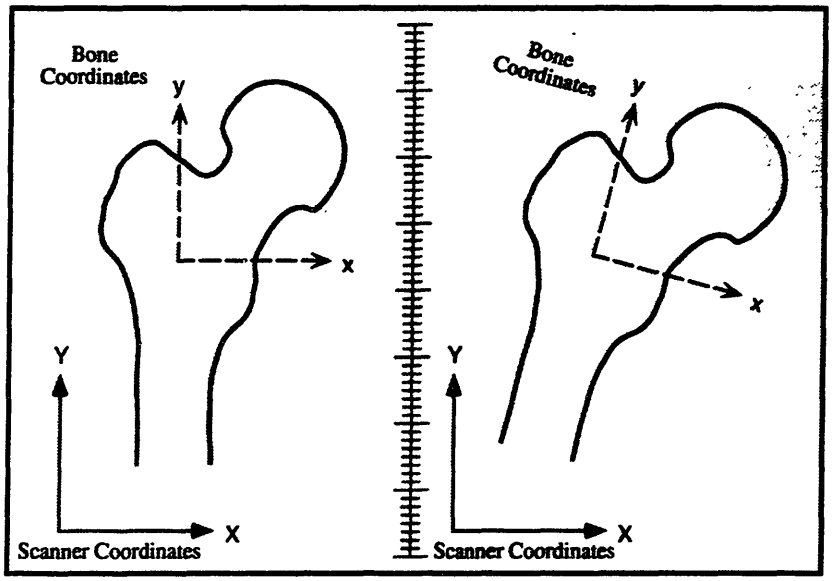


Figure 3.1: Scanner Coordinates versus Anatomical Coordinates

Since this local coordinate system is based on a subject's anatomy, we can expect it to be independent not only of the scanning method but also of the time of, or interval between data collection. Thus long-term studies analyzing the progression of disease by the acquisition of information at discrete time intervals could use such a coordinate reference frame as a baseline. In addition this approach permits the comparison of data obtained via different acquisition methods. For example, data acquired with CT and MRI could be combined to complement one another and thus provide a superset of anatomical information. Prior studies have attempted to relate the two dimensional image data provided by CT with those provided by MRI [54, 86, 87]. However, since these approaches involve only two-dimensional scaling and correlation algorithms from one image to the other, they often completely neglect the three dimensionality of the information. Moreover, since data is collected as slices at discrete locations along a translational direction, errors in matching images are unavoidable. This can sometimes be compensated by interpolating between two images, but this still results in poor evaluations of the object orientation. Such methods have been most successful at overlapping qualitative information in two dimensions to provide complimentary information on the position of malignant and surrounding tissues during radiation therapy planning.

It should also be noted that some anatomical structures, such as soft tissues, do not lead to uniform and repeatable calculation of local coordinate systems while others, such as bone, provide excellent rigid body features that can result in proper estimations. One approach could be the selection of landmark points from the anatomical images to arbitrarily define a coordinate system associated with a bone [105, 107]. For example, one could choose the greater and lesser trochanters with the femoral head *Fovea* (femoral head ligament *Teres* depression) to assign a specific coordinate system to a specific bone. However, this would require some operator interaction and thus does not preclude systematic errors related to the visual limitations presented by the imaging technique.

By focusing on the inherent three-dimensions of the information we can assign a coordinate system based on the spatial distribution of the anatomical information describing a particular tissue structure. Such an approach will have to consider the influence of the data format used as an input. More specifically, CT and MRI do not generate anatomical information with a uniform spatial distribution: the resolution in the translational axis of the scanner is much poorer than those in the transversal axes. As illustrated in Figure 3.2, the orientation of the subject in the scanner system defines the geometry of the two-dimensional slices collected by the system. Data in subfigures 3.2 (b) and (d) greatly differ though each still describes the same anatomical features.

3.2 METHOD

An approach was implemented in this research which proposes to define the orientations of rigid anatomical structures, such as bones, in scanner space. By considering all the three-dimensional points that define the geometry of a bone, we can treat this information as a swarm of points which possesses a specific set of principal axes. These, in turn, can be used to calculate the orientation of the object. While this approach can be easily generalized to apply to any anatomical feature with a rigid shape, it cannot assign a unique set of principal axes to geometric shapes which do not exhibit distinct principal directions. For example, a sphere has an infinite number of principal axes. If the sphere is rotated about its center, it is impossible to calculate from the geometric shape along a set of principal axes that could represent this reorientation. The reader can verify how this issue also applies to other mathematical shapes such as cubes and cylinders and how easily singularities can arise. Fortunately, anatomical information is rarely described by simple mathematical formulae. We propose here to consider the femur as the source of the data used to evaluate the principal axes and thus the local coordinate system. Femurs, as long bones, have a unique shape, with

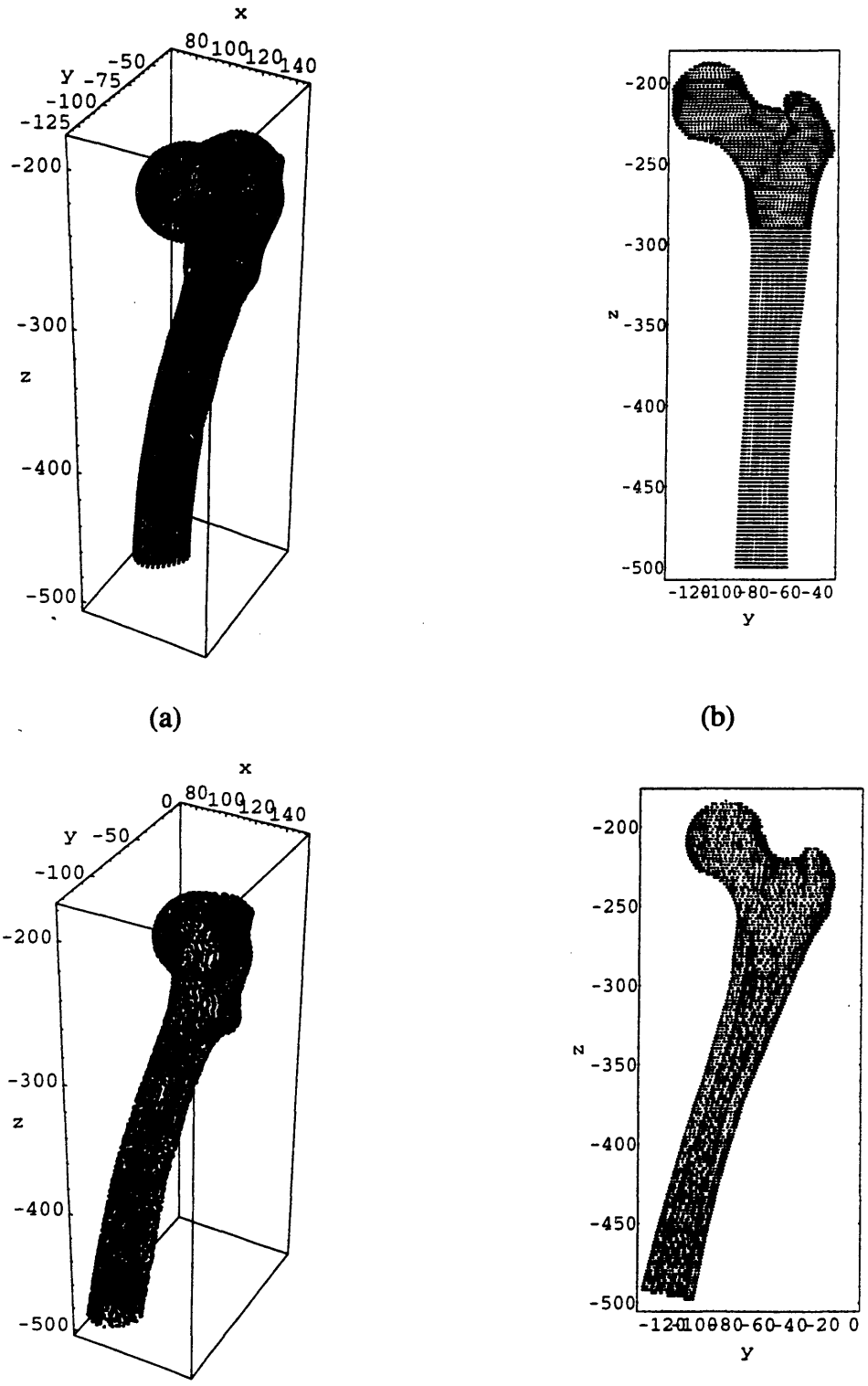


Figure 3.2: Bone Orientation and Scanner "Slicing"

distinct dimensions and spatial distributions in all three dimensions, generating a unique set of principal axes, dependent on object geometry, and whose orientation duplicates the orientation of the object.

3.2.1 THEORY

The contour points for a given structure represent a swarm of points. The principal axes and moments of inertia can be determined for this swarm of points in space. The principal axes can then serve as a local structure coordinate system. Therefore by defining points on the object, we can then calculate the orientation of the object in space relative to the local object coordinate system. A similar derivation was described by Brown [18] to determine the best fit plane to a swarm of points. A new application is presented below.

If the object points are centered about their center of mass, we can define a symmetric scatter matrix M as:

$$M = \sum_{i=1}^{N} X_i X_i^T = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i z_i \\ \sum y_i x_i & \sum y_i^2 & \sum y_i z_i \\ \sum z_i x_i & \sum z_i y_i & \sum z_i^2 \end{bmatrix} \qquad \text{(Eq. 3.1)}$$

where

$$X_i = \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix} \qquad \text{(Eq. 3.2)}$$

It can then be shown that the eigenvalues of the matrix M are the principal values of the sums of the squared errors, where the minimum eigenvalue with its corresponding eigenvector define the "best fit plane". This plane is defined by its normal unit vector, the eigenvector, and the centroid of the points. To help follow this argument, the perpendicular distance e_i between any point and the plane can be defined as the dot product of the point coordinate vector x_i and the unit normal vector n of the plane:

$$e_i = X_i^T n \qquad \text{(Eq. 3.3)}$$

The sum of the squared errors is then defined as:

$$E^2 = \sum_{i=1}^{N} \left(X_i^T n\right)^2 \qquad \text{(Eq. 3.4)}$$

$$= \sum_{i=1}^{N} \left(n^T X_i\right)\left(X_i^T n\right) \quad \text{(Eq. 3.5)}$$

$$= n^T \left(\sum_{i=1}^{N} X_i X_i^T \right) n \quad \text{(Eq. 3.6)}$$

$$= n^T M n \quad \text{(Eq. 3.7)}$$

Then, left multiplying both sides by the product of the transpose of the unit normal vector n with the unit normal vector n itself yields:

$$n^T n E^2 = n^T E^2 n = \left(n^T n\right) n^T M n = \left(n n^T n\right)^T M n = n^T M n \quad \text{(Eq. 3.8)}$$

which, since E^2 is a scalar and can be commuted with n and $n^T n=1$ because n is a unit vector, can be written as:

$$Mn = E^2 n \quad \text{(Eq. 3.9)}$$

which is equivalent to the standard form of an eigenvalue problem:

$$Mn = \lambda n \quad \text{(Eq. 3.10)}$$

where the eigenvalues λ_i are the principal values of the scatter matrix or the principal sums of squared errors E^2. To solve for the eigenvalues, Equation 3.10 can be rewritten as:

$$n^T M n - n^T \lambda n = 0 \quad \text{(Eq. 3.11)}$$

$$n^T (M - \lambda I) n = 0 \quad \text{(Eq. 3.12)}$$

The trivial solution n=0 is not possible since n is a unit vector. Thus, the eigenvalues must satisfy:

$$\det(M - \lambda I) = 0 \quad \text{(Eq. 3.13)}$$

The eigenvalues are determined by calculating the roots of this characteristic equation, and the associated eigenvectors are then evaluated by solving:

$$(M - \lambda I) n = 0 \quad \text{(Eq. 3.14)}$$

Theoretically, the scatter matrix M is sufficient to compute the principal axes. However, numerically, a matrix that has stronger diagonal elements than off-diagonal elements simplifies the computations for the eigenvalues and minimizes the sorting necessary to order the eigenvalues and their associated eigenvectors. Note that the diagonal terms of the scatter

matrix M are on the same order of magnitude as its off-diagonal terms. To this effect, we relate the matrix M to the inertia tensor matrix T of a swarm of points with:

$$M = \begin{bmatrix} \sum(x_i^2+y_i^2+z_i^2) & 0 & 0 \\ 0 & \sum(x_i^2+y_i^2+z_i^2) & 0 \\ 0 & 0 & \sum(x_i^2+y_i^2+z_i^2) \end{bmatrix} - \begin{bmatrix} \sum(y_i^2+z_i^2) & -\sum x_i y_i & -\sum x_i z_i \\ -\sum y_i x_i & \sum(x_i^2+z_i^2) & -\sum y_i z_i \\ -\sum z_i x_i & -\sum z_i y_i & \sum(x_i^2+y_i^2) \end{bmatrix} \quad \text{(Eq. 3.15)}$$

$$= \begin{bmatrix} \sum d_i^2 & 0 & 0 \\ 0 & \sum d_i^2 & 0 \\ 0 & 0 & \sum d_i^2 \end{bmatrix} - \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{xx} & I_{zz} \end{bmatrix} = \left(\sum d_i^2\right)\mathrm{I} - \mathrm{T} \quad \text{(Eq. 3.16)}$$

Where d_i is the distance of each point to the origin. Consequently the scatter matrix and inertia tensor are closely related to each other. The eigenvalues of the scatter matrix are simply shifted eigenvalues of the inertia tensor

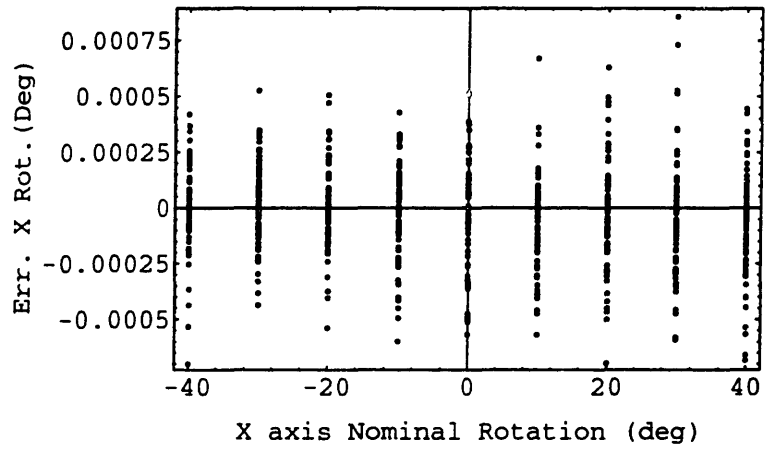
$$\lambda_{scatter,j} = \left(\sum d_i^2\right) - \lambda_{inertia,4-j} \quad \text{(Eq. 3.17)}$$

where eigenvalues subscripts j=1-3 are the minimum to maximum eigenvalues and eigenvectors of the dual eigenvalue pairs. However, the tensor matrix T provides a matrix with stronger diagonal elements and thus is a better matrix to solve the eigenvalue problem numerically.

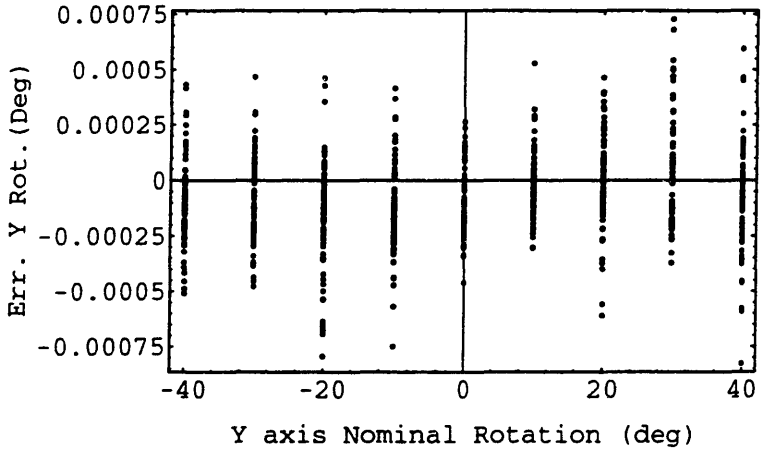
A program was implemented, which given points that define an object, will compute the "orientation" in space of that object from its eigenvectors. The software, which is listed in Appendix D uses a Jacobi method to find the eigenvalues and associated eigenvectors [100]. The software also ensures that the resulting eigenvectors define a "right handed" orthonormal coordinate system. Finally, the program solves the nonlinear equation:

$$\begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} = Eigvectors^T \quad \text{(Eq. 3.18)}$$

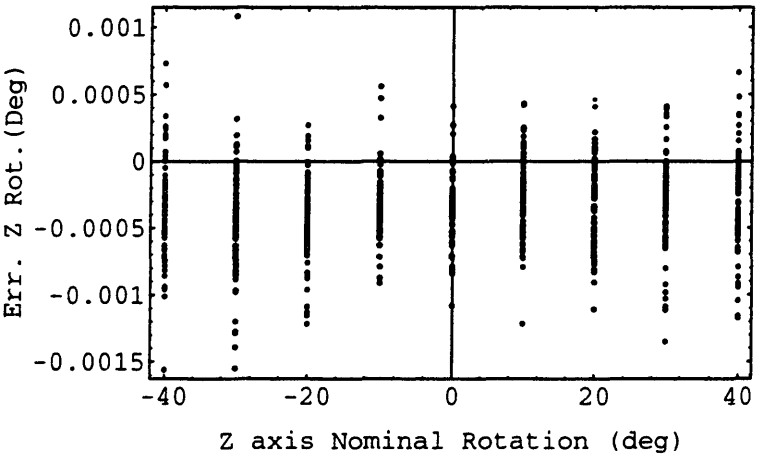
where α, β, γ are the angles of rotation along the X, Y, Z axes in that specific order and define the orientation of the object in the scanner coordinate system.



(a)



(b)



(c)

Figure 3.3: Test Simulation of Principal Axis Theory

To verify the theory and check our software, an object in the shape of a box centered at the origin and with dimensions 50.0, 100.0 and 200.0 mm was created so as to guarantee the existence of a unique set of principal axes. The initial orientation of the box was aligned with the global coordinate system. The 8 nodes that define the corner of the box were then rotated along all three axes of the global coordinate system from -40° to 40° in 10° increments. The errors between the nominal orientation and the computed principal axis orientation of the box are presented in Figure 3.3. It can be noticed that the absolute errors are less than 0.0015 degree and thus insignificant. These errors are most likely due to the discretization of the data as well as the numerical precision of the computer. In addition, no particular pattern in the error spread of the data can be detected which thus confirms the validity of the principal axis method to define the spatial orientation of an object based on its geometric shape.

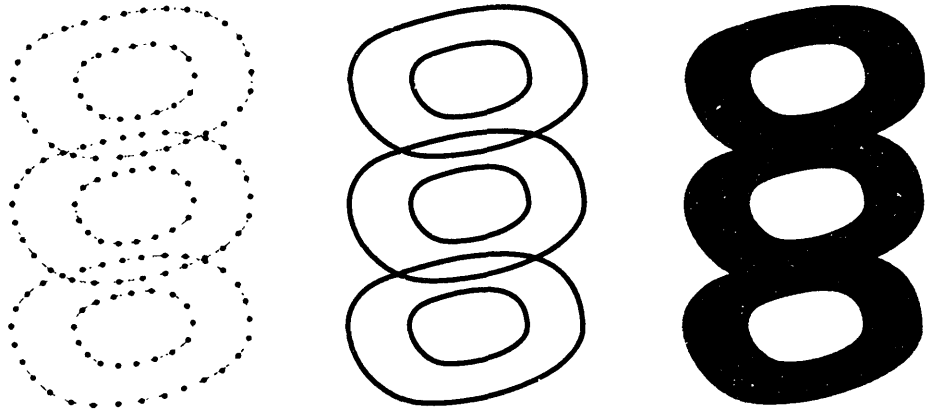
3.2.2 SIMULATIONS

While the test performed above confirms the theory, a rectangular parallelepiped is not a good simulation of the actual data acquired via CT or MRI scanner systems. These systems acquire information in discrete slices as illustrated by Figure 3.2. A different orientation provides a new set of cross-sections transverse to the scanner translational axis and not a set of rotated, with otherwise unchanged, slices. For example, a cylinder cut along its long axis will produce circular cross-sections. However, if the long axis of the cylinder is not aligned with the scanner axis, the resulting slices will be ellipsoidal.

Using the capability developed in Chapter 2 to segment anatomical data to isolate pixels belonging to a particular tissue, we can create a three dimensional data set of object points. For example, once an image has been thresholded, any pixel highlighting a cross-section of a femur bone can be converted into its three-dimensional equivalent. This results in data similar to the example illustrated in Figure 3.4(c), where full cross-section slices defining, for example, a femoral shaft have been stacked. However, the amount of data that has to be handled rapidly becomes daunting. Using contour descriptions or sampled nodes on the contour paths as in Figure 3.4(b) and 3.4(a) respectively, greatly reduces the amount of data selected to specify the geometry of the bone. While such representations clearly reduce memory storage and computational power requirements needed to calculate principal axes, it is unclear whether they provide quantitatively equivalent geometric descriptions to accurately evaluate spatial orientations.

The effects of the different possible data inputs were investigated. Using the same object as

defined above, simulations were implemented to test the principal axis theorem with sample nodes, full contours and full cross-sections as inputs. Program listings are provided in Appendix D. The process can be described in three steps. First the box is virtually reoriented with three nominal rotations. Then, a scanner simulator slices the object along the translational axis at discrete indices and creates the cross sectional representation of the object akin to a scanner image. Finally, the input points are processed by the principal axis algorithm which computes the rotation angles. These angles are then compared with the nominal values and errors are calculated. For each different anatomical data input type, the effect of axis orientation was studied individually in 1° increments as well as combined in 10° increments so as to determine the existence of coupling behavior. The angle range was limited to ±40° which should encompass all possible clinical orientations



Sampled Nodes a | Full Contours b | Full Cross-Sections c

Figure 3.4: Three Different Anatomical Data Input Types

Sampled Nodes

Node coordinates were computed at every slice at the intersection of the slice plane and the vertices of the box. Slices were created 2.0 mm apart to best simulate the scanning process and resulted in objects composed of 800 to 1,700 data points depending on the orientation. Figures 3.5 to 3.7 show simulation results for the individual X, Y and Z rotation changes respectively. The results show a strong correlation between the error magnitude of a computed rotation angle when the object is rotated about that particular axis. Errors of up to ±8.0°, ±2.5° and ±1.5° exists over the entire range in the X, Y and Z rotations. In addition, rotations about the X axis result in errors in the computed rotations about the Y and Z axes of ±0.15°. Similarly, rotations about the Y direction produced errors of ±0.4° in the computed X and ±0.15° in the computed Z angles. No coupling effects can be observed with rotations about the Z axis.

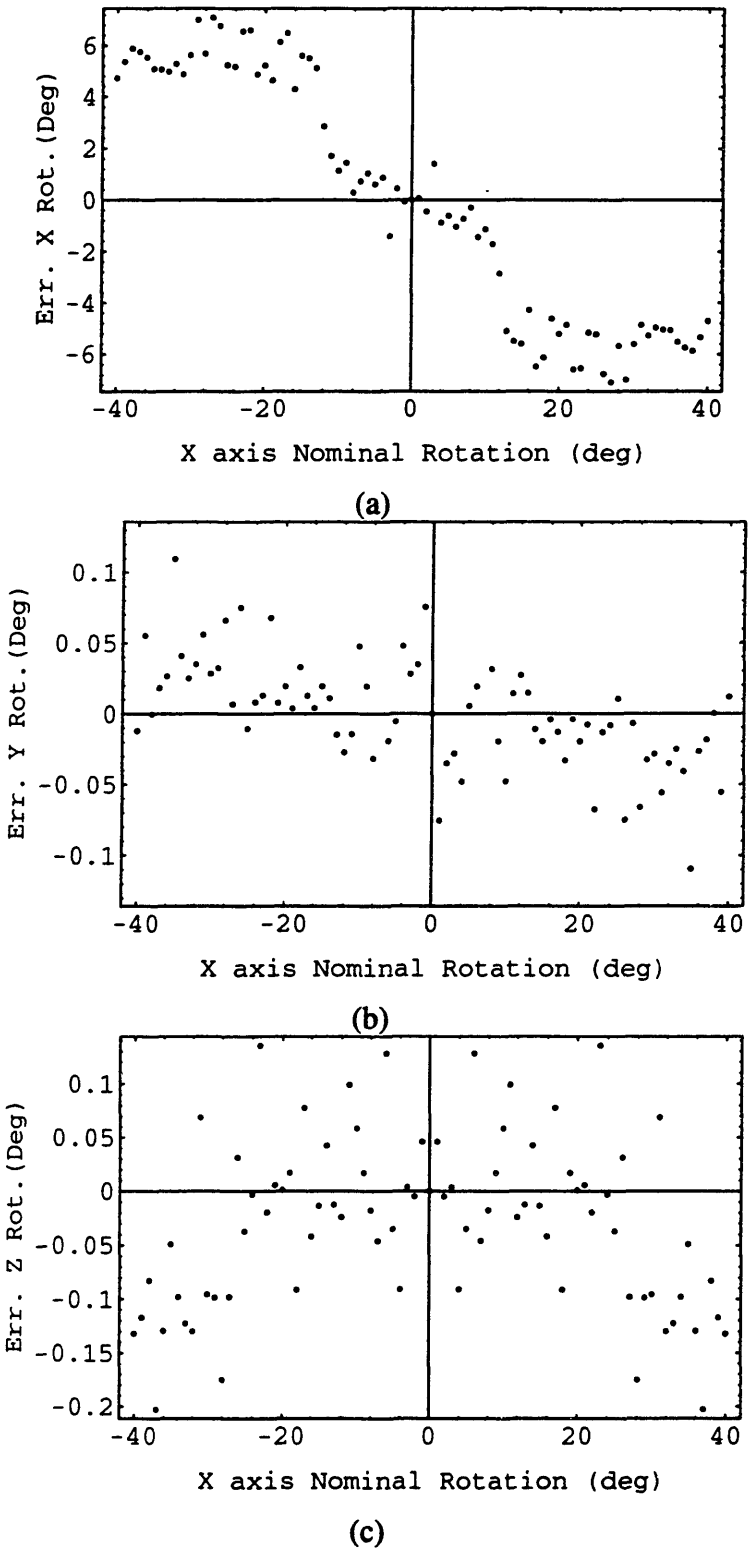
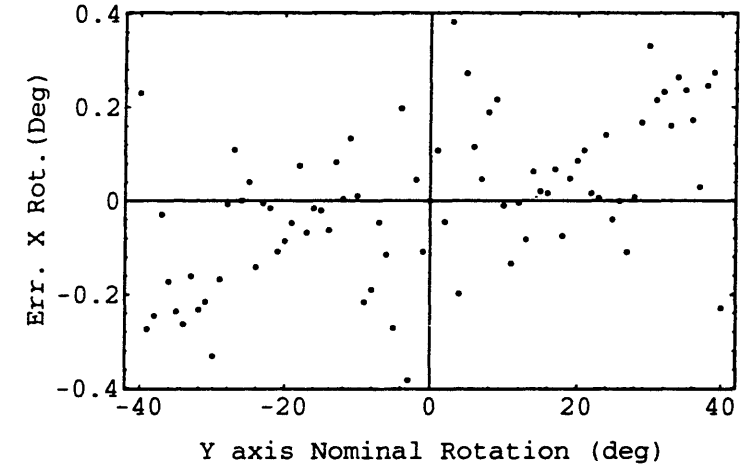
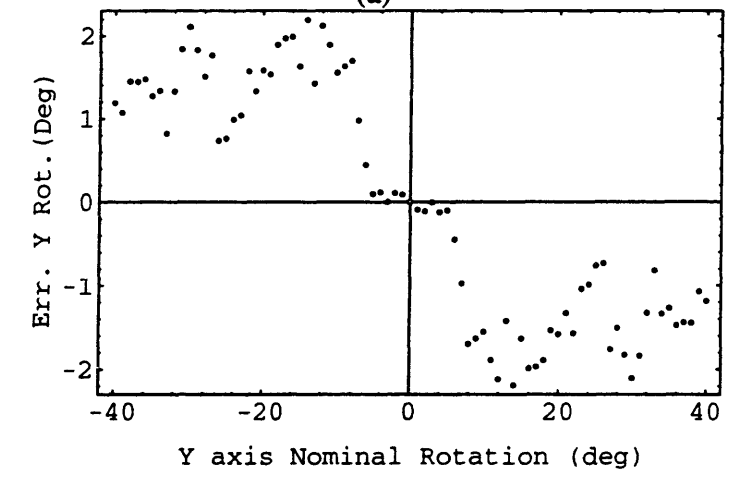


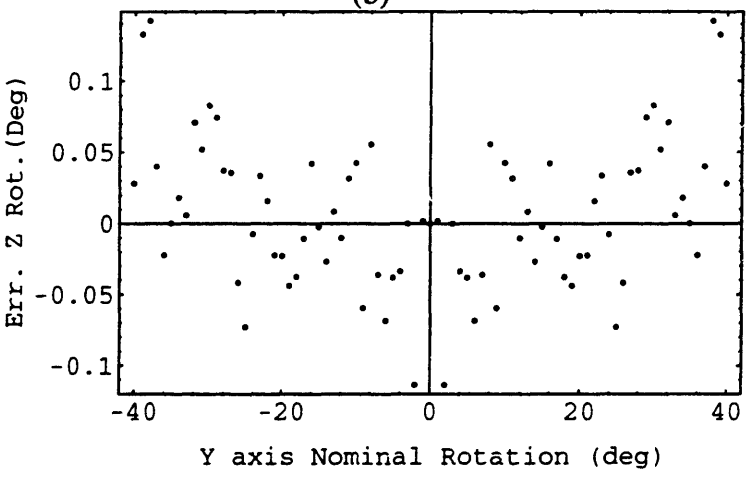
Figure 3.5: Test Simulation Sensitivity to X Rotation for Node Case



(a)

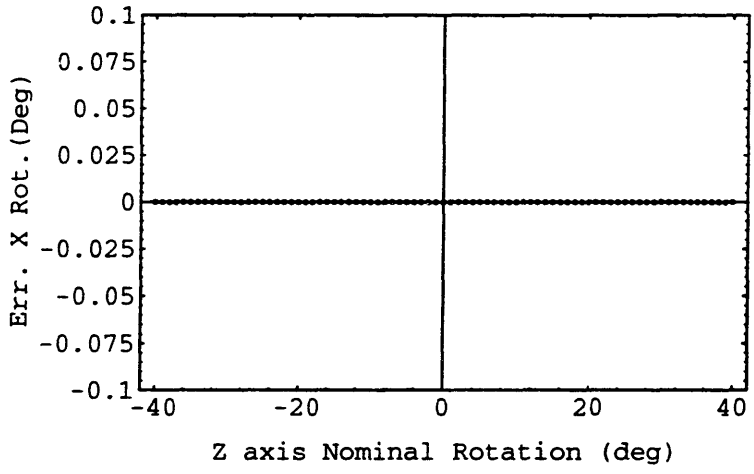


(b)

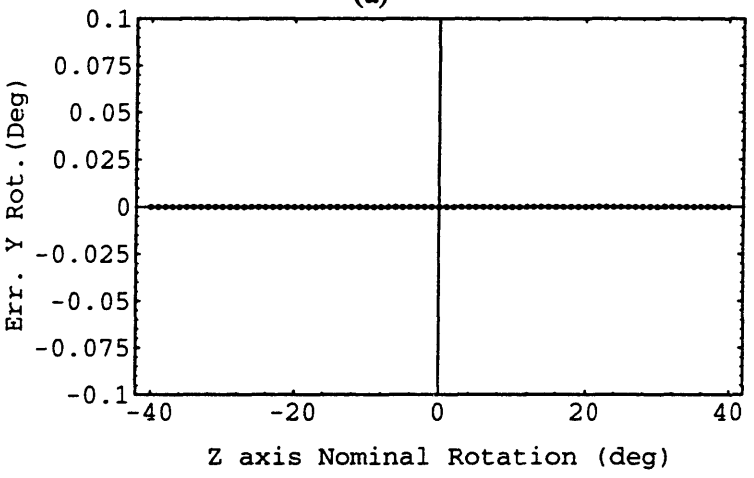


(c)

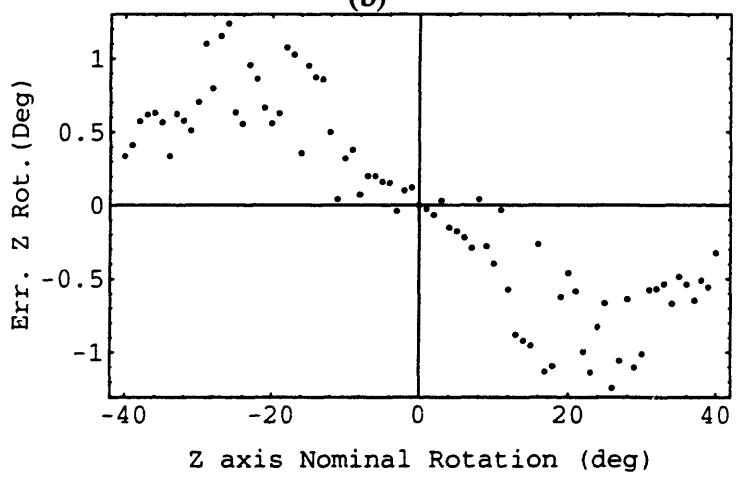
Figure 3.6: Test Simulation Sensitivity to Y Rotation for Node Case



(a)



(b)



(c)

Figure 3.7: Test Simulation Sensitivity to Z Rotation for Node Case

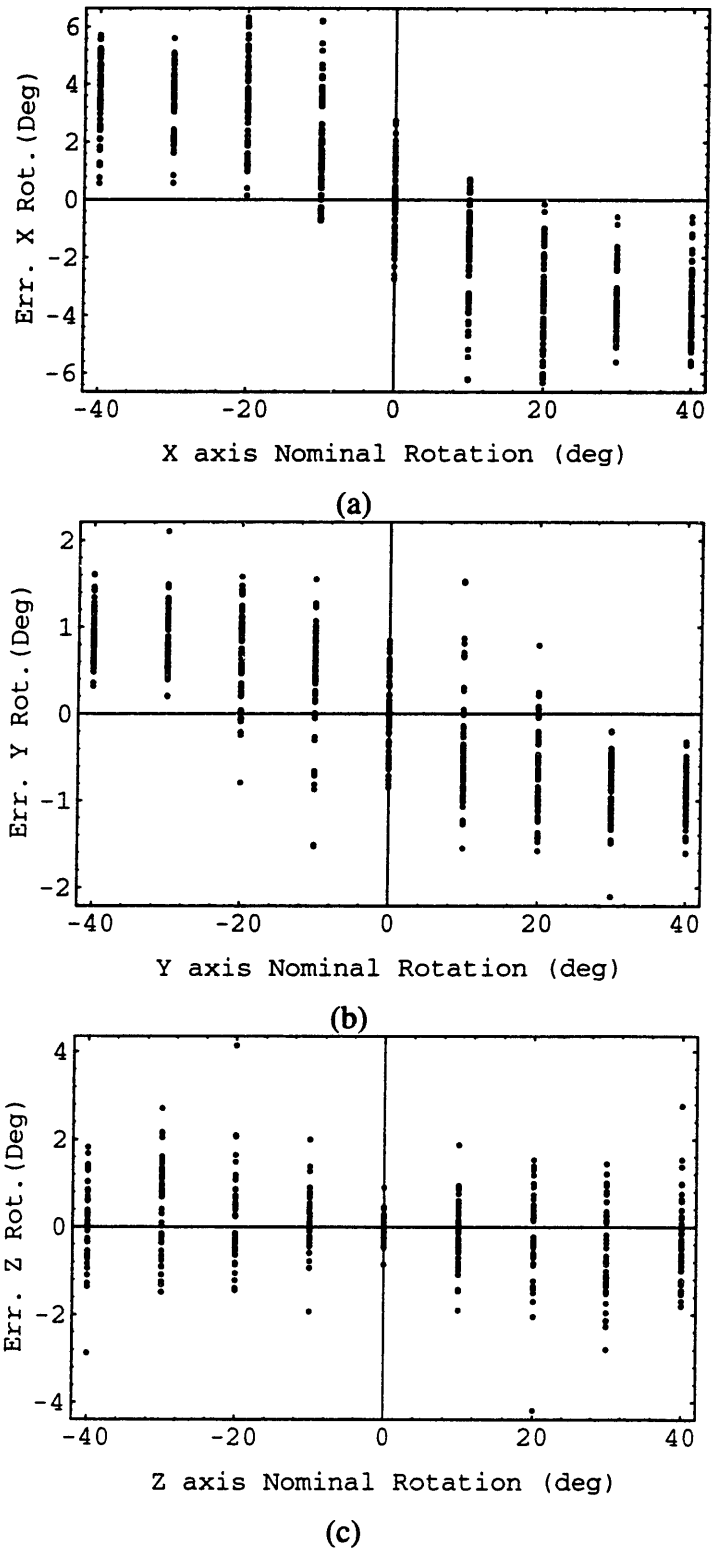


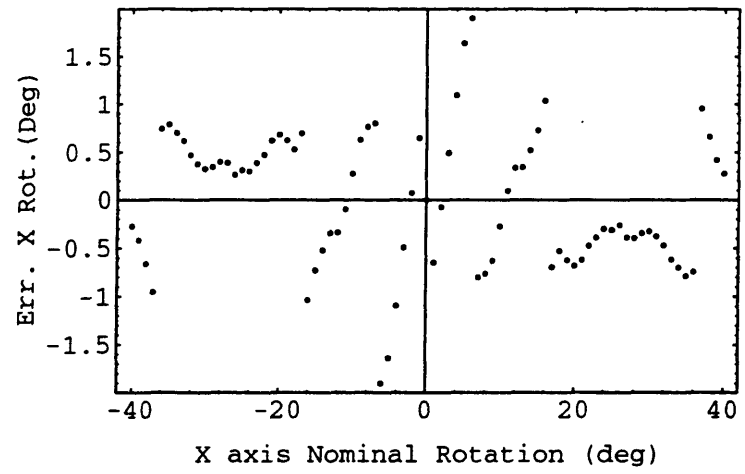
Figure 3.8: Test Simulation Sensitivity to Cumulative X, Y & Z Rotations for Node Case

Figure 3.8 presents the errors along all three axes when the object is subjected to combined reorientations . As seen previously, the magnitude of the error is primarily dependent on the magnitude of the corresponding axis reorientation. However, coupling error effects are prevalent and evident in the spread of the error. The shape and magnitude of the error results were not affected by the slice interval. For example, doubling the number of slices results in insignificant changes in the error calculations. However, changing the geometry of the object (i.e.: different dimensions or shape) affects the shape of the error curves but not their magnitude. Finally it is important to note that the computer time to completion of this simulation is about 34 minutes.

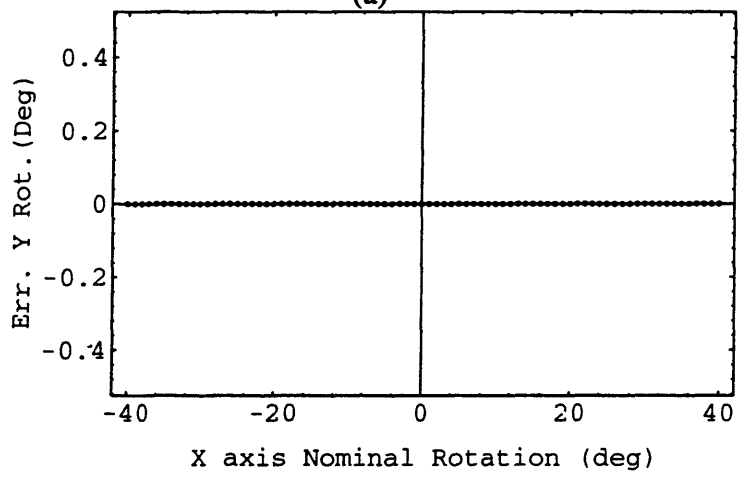
Full Contours

Full contours were formed at every slice at the intersection of the slice plane with the surface of the box. Bit map images were defined in 512 x 512 matrices with a pixel size resolution of 300μm to simulate MRI and CT images. Contours were then extracted. Slices were specified 10.0 mm apart to speedup the simulation time and resulted in objects composed of 32,000 to 68,000 data points depending on the orientation. Figures 3.9 to 3.11 show simulation results for the individual X, Y and Z rotation changes respectively. Here again, the results show a definite correlation between the error magnitude of a computed rotation angle when the object is rotated about that particular axis. However, there is no qualitative evidence of an increase in error level with increased nominal rotations as in the previous case. The shapes of the curves are much more random and the dependency can only be detected by the appearance of a larger overall error magnitude spread. Errors of up to ±1.5°, ±0.6° and ±0.4° are observed over the entire range in the X, Y and Z rotations . However, in this case, rotations along the X and Y axes result in no errors in the computed rotations along the Y, Z and X, Z rotations axes respectively while rotations along the Z direction exhibit strong coupling effects and produce very significant errors of ±1.5° in the computed X and ≈1.1° in the computed Y angles. This behavior is almost the inverse of the one described for the sampled node case. The overall magnitude of the errors is reduced by about an order of magnitude when compared with the previous case.

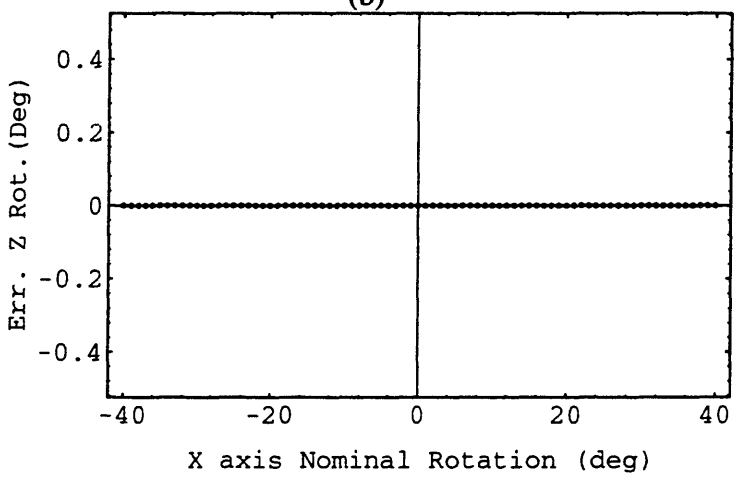
Figure 3.12 presents the errors along all three axes when the box is subjected to combined reorientations . In this case, a direct relationship between the error curve shapes and the magnitude of the nominal rotations is less evident than in the previous case. However, the coupling error effects due to the Z axis nominal rotations are still prevalent and explain the spread of the error. There is also some evidence that the error coupling due to the



(a)

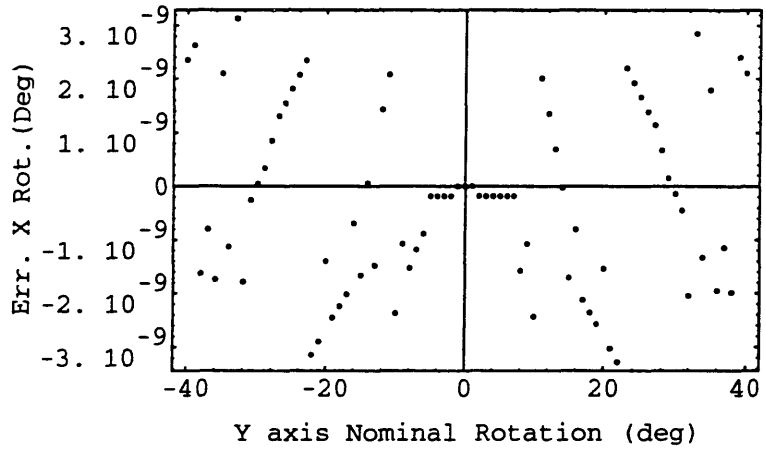


(b)

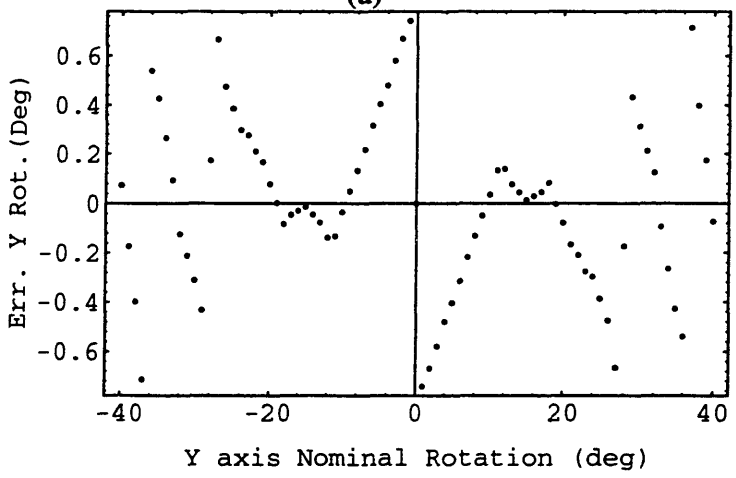


(c)

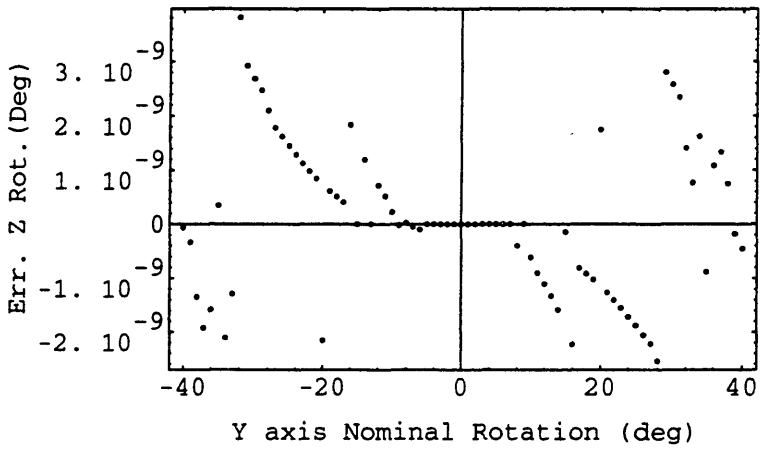
Figure 3.9: Test Simulation Sensitivity to X Rotation for Contour Case



(a)

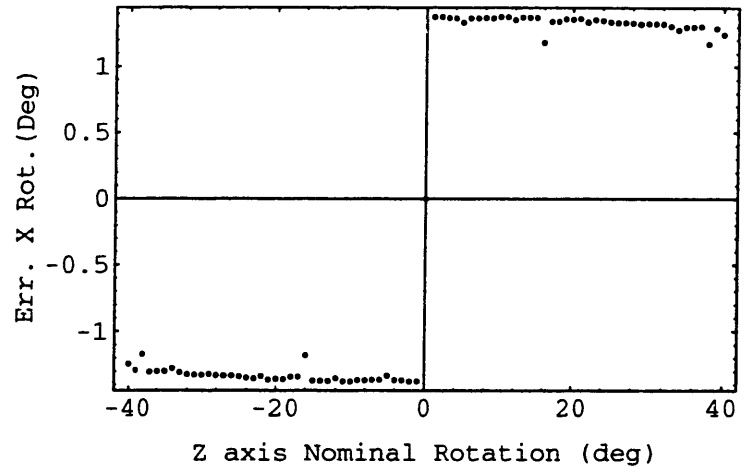


(b)

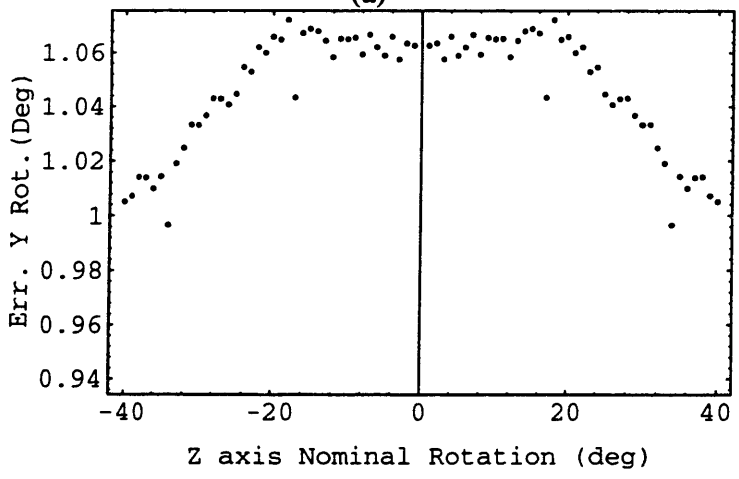


(c)

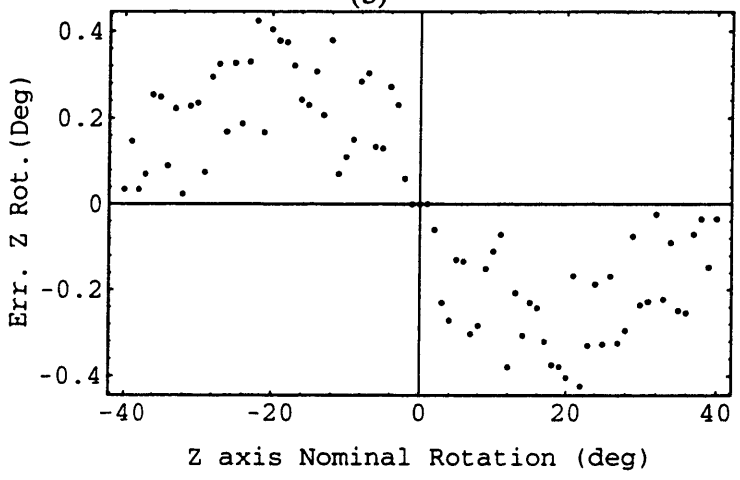
Figure 3.10: Test Simulation Sensitivity to Y Rotation for Contour Case



(a)

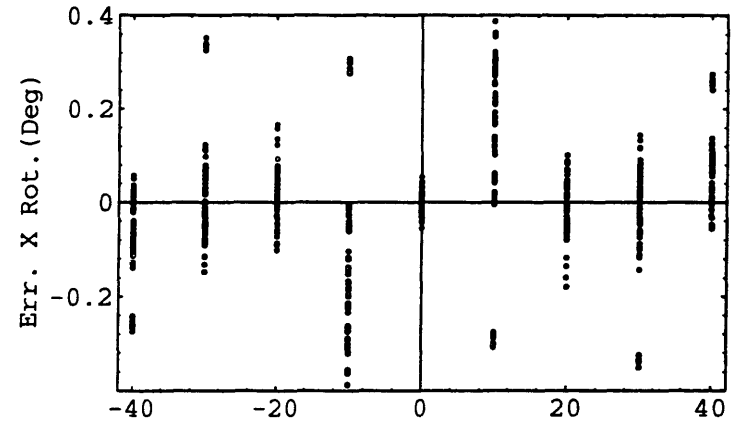


(b)

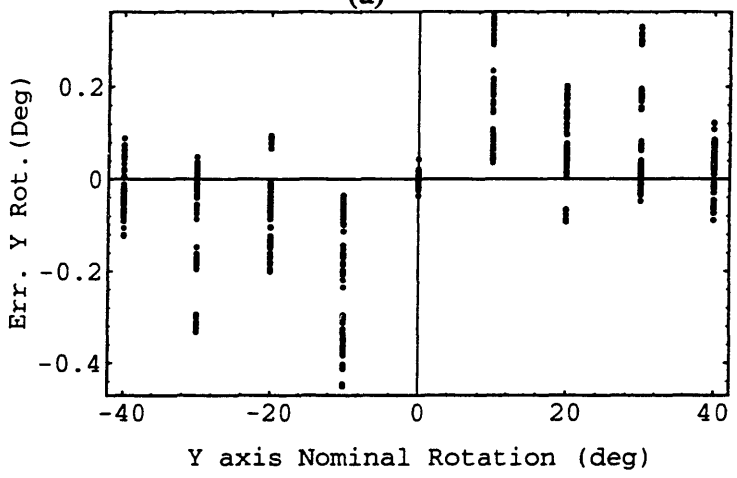


(c)

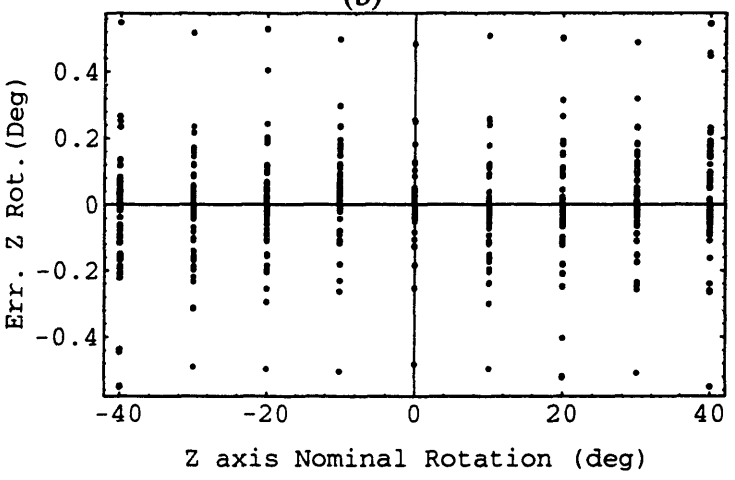
Figure 3.11: Test Simulation Sensitivity to Z Rotation for Contour Case



(a)



(b)



(c)

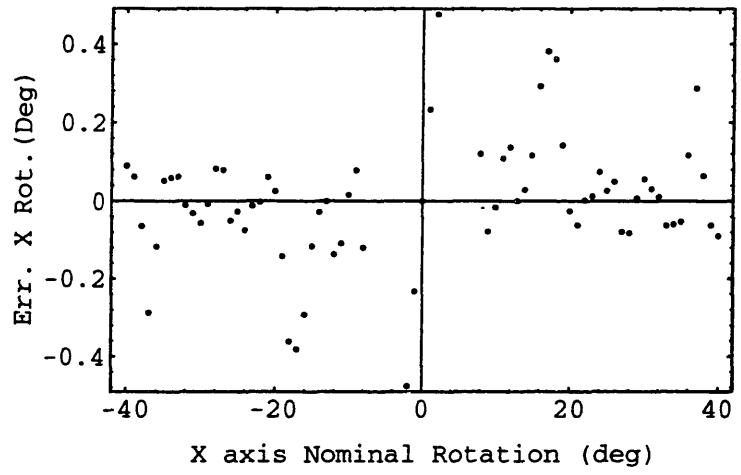
Figure 3.12: Test Simulation Sensitivity to Cumulative X, Y & Z Rotations for Contour Case

Z rotations compensates for the systematic errors in the X and Y rotations. The shape and magnitude of the error curves were found to be sensitive to the slice interval. For example, doubling the number of slices results in significant changes in the data: for the most part, the error spread is reduced. Therefore, it can be assumed that a simulation performed with a slicing interval of 2.0 mm would yield even better results. Changing the geometry of the object did not seem to affect the shape and magnitude of the error curves. This is a major benefit in using full contours as opposed to sampled nodes. However, simulation computation time increased an order of magnitude to over 5.5 hours.

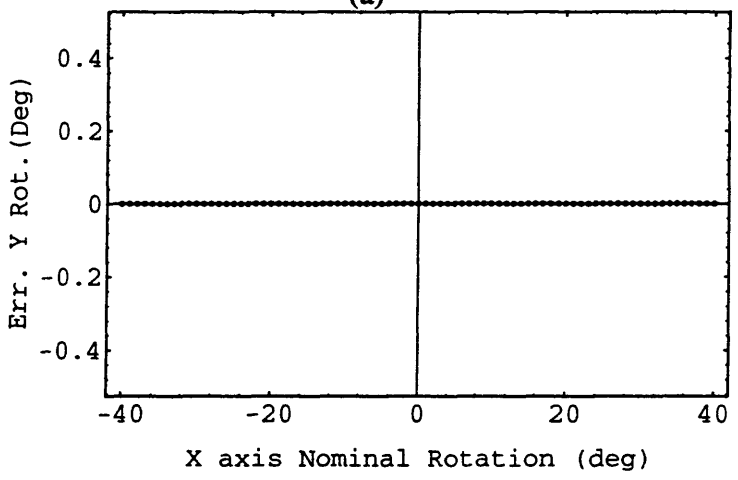
Full Cross-Sections

Full cross-sections were defined at every slice at the intersection of the slice plane with the volume of the box. As previously, bitmap images were formed in 512 x 512 matrices with a pixel size resolution of 300μm to simulate MRI and CT images. Slices were specified 10.0 mm apart to speedup the simulation time and resulted in objects made of 900,000 to 2,000,000 data points depending on the orientation. Figures 3.13 to 3.15 show simulation results for the individual X, Y and Z rotation changes respectively. At first glance the error curve patterns appear very similar to the ones computed for the full contour case. Graphically, the results are astonishingly similar with only different magnitude scales. Here again, the results show some correlation between the error magnitude of a computed rotation angle when the object is rotated about that particular axis. Also, there is no qualitative evidence of an increase in error level with increased nominal rotations. The shape of the systematic error curves is even more random than in the previous case and the dependency can only be detected by the appearance of a some detectable error magnitude spread. Errors of up to ±0.4°, ±0.2° and ±0.1° are observed over the entire range in the X, Y and Z rotations. Again, in this case, rotations about the X and Y axes result in no significant errors in the computed rotations about the Y, Z and X, Z rotations axes respectively while rotations about the Z direction exhibit some detectable errors of ±0.03° in the computed X and ±0.02° in the computed Y angles. The use of full cross-sections permits an additional reduction in the orientation angle error by a factor of 4.

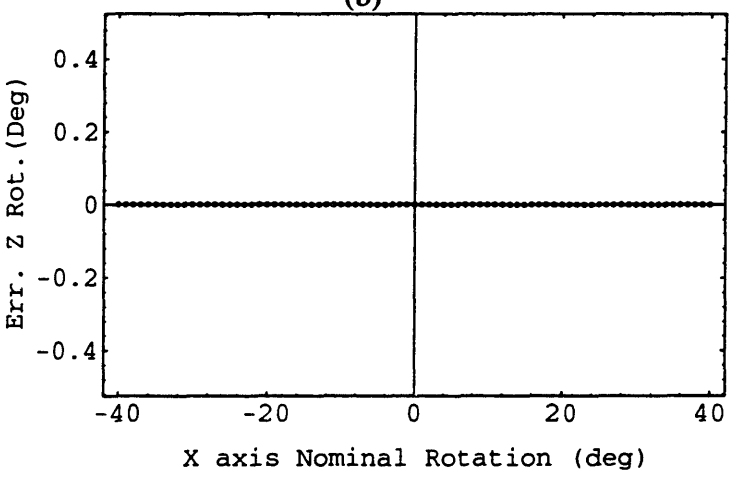
Figure 3.16 presents the errors about all three axes when the box is subjected to combined reorientations. In this case, while the existence of a direct relationship between the error curve shapes and the magnitude of the nominal rotations can still be noticed as in the previous case, the coupling error effects due to the Z axis nominal rotations are now barely detectable. The shape and magnitude of the error curves were found to be sensitive to the



(a)

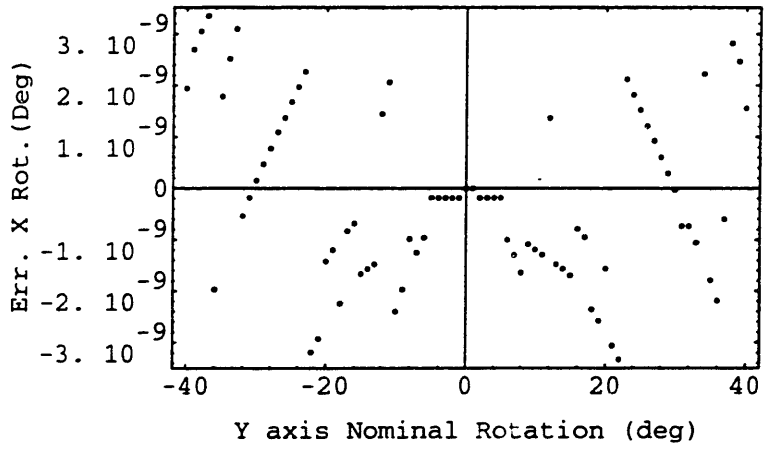


(b)

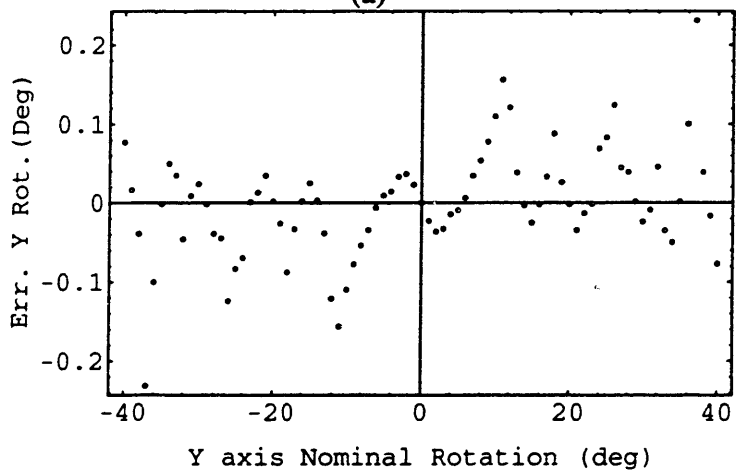


(c)

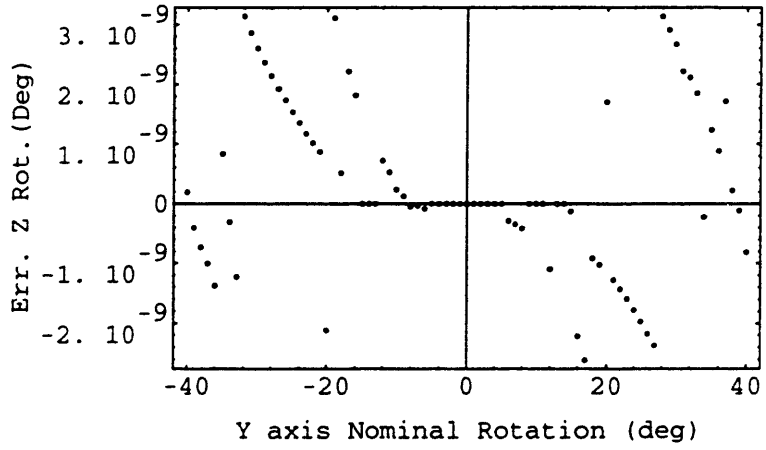
Figure 3.13: Test Simulation Sensitivity to X Rotation for Slice Case



(a)

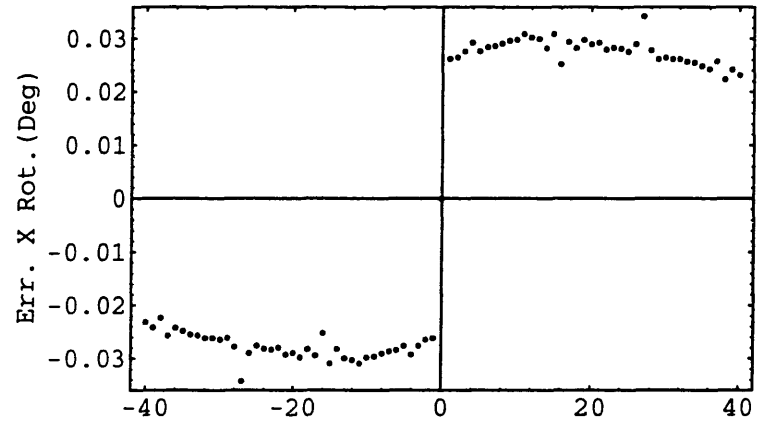


(b)

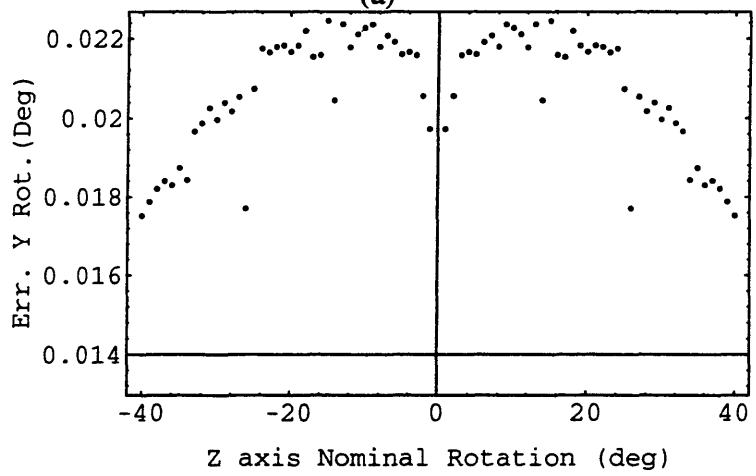


(c)

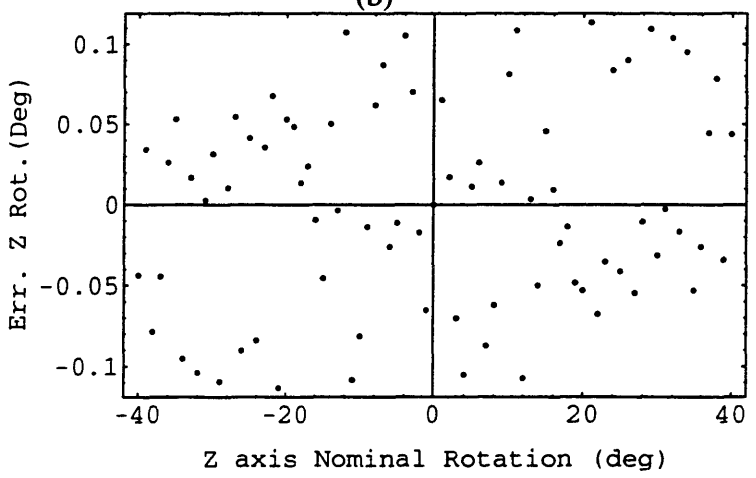
Figure 3.14: Test Simulation Sensitivity to Y Rotation for Slice Case



(a)

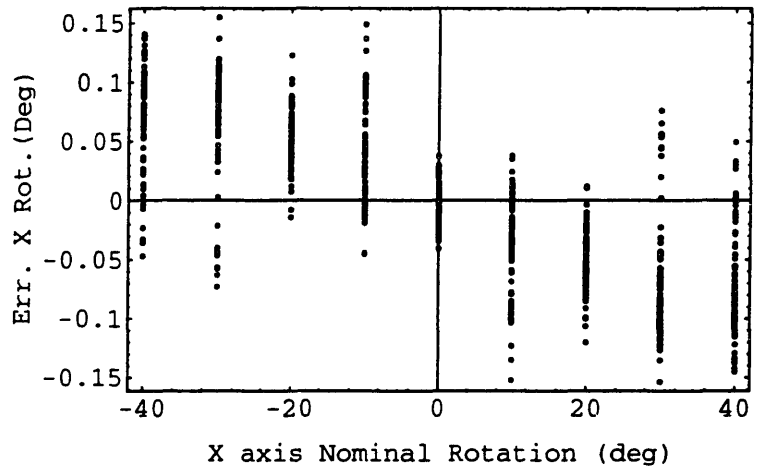


(b)

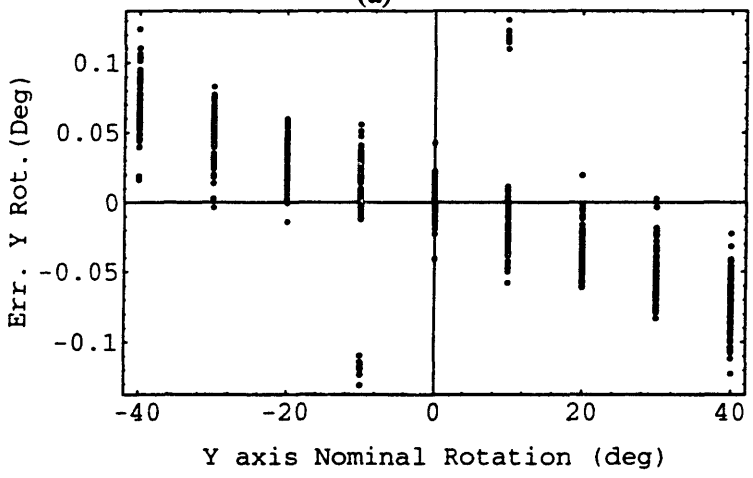


(c)

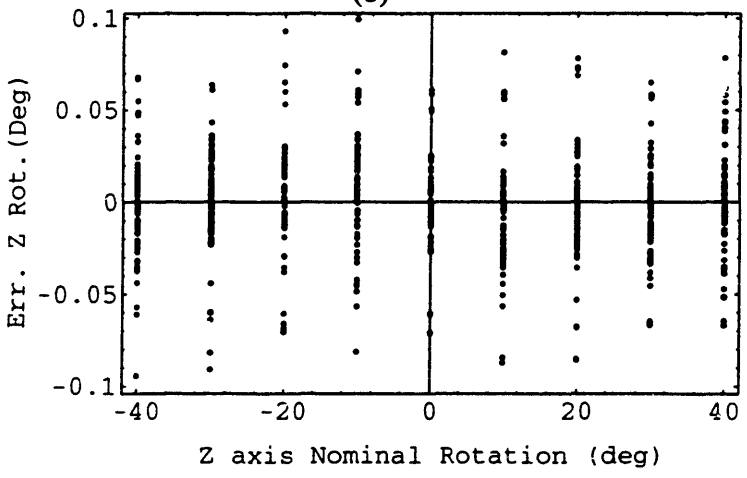
Figure 3.15: Test Simulation Sensitivity to Z Rotation for Slice Case



(a)



(b)



(c)

Figure 3.16: Test Simulation Sensitivity to Cumulative X, Y & Z Rotations for Slice Case

slice interval. For example, doubling the number of slices results in a reduction of the error spread. Therefore, it can be assumed that such a simulation performed with a slicing interval of 2.0 mm would yield even better results. Similar simulation results were obtained with different object geometries using full cross-sections. While this approach provides the best accuracy, it is the most computationally intensive and required 18.3 hours to complete.

Finally, it should be noted that the symmetry that appears in all the simulation results is expected because of the geometrical symmetry of the box about its centroid.

3.3 EXPERIMENTAL RESULTS

Simulation results clearly indicate a distinct sensitivity of the principal axis method to the data input used, especially in cases of gross misalignments between the object local coordinates and the data acquisition global coordinate system. The use of full cross section areas to obtain data points that describe the object seems to be quite adequate to minimize errors in orientation evaluations of a three dimensional box.

Table 3.1: Femur Orientations using Principal Axes and 3 Different Approaches (degrees)

	Rotation Angle	Cross-Sections	Contours	Error	Nodes	Error
CT	X	2.35	1.78	-0.57	0.83	-1.52
	Y	-8.72	-8.75	-0.03	-8.82	-0.10
	Z	9.27	9.54	0.27	9.86	0.59
MRI	X	5.83	6.04	-0.21	4.65	-1.18
	Y	12.76	12.47	-0.29	13.66	0.90
	Z	-6.33	-6.42	0.09	-6.94	0.61

Using anatomical data acquired with CT and MRI imaging systems, femur orientations were computed with the three different approaches previously described. The results are presented in Table 3.1. Inter-slice spacing was dependent on the data set and the value chosen during the scanning process. Refer to Chapter Two to review the data acquisition protocol designed for this project. Node sampling was performed on the contoured information to produce nodes 5.0 mm apart. Since the cross-sectional computed orientations were found to be the most accurate with simulations, they were used as a baseline to perform the error calculations for the other two methods. As can be seen, the overall error magnitude between the methods

is relatively small. Maximum discrepancies of about 0.6° and 1.5° are observed for the contours and sampled nodes approaches respectively. These small errors can be explained in part by the fact that the femur was oriented so as to be essentially aligned with the coordinates of the scanner system; the box simulations of all three different methods showed small errors for small misalignments between the local and global coordinate systems. Also notice that the largest error magnitude appears with the X rotations, as expected from the simulations, due mainly to variations in input information along the bone long axis, the axis where rotation presents the largest variations in point displacement.

While one might have expected larger errors for MRI than for CT data because of the lower resolution and accuracy of the former imaging technique, this is not apparent in the results. The use of a large number of data points in the orientation algorithm compensates for errors in the data, assuming that accuracy issues are the consequence of white noise: each data point behaves like another measurement of the same parameter, overlaid with a certain amount of noise.

3.4 DISCUSSION

Although not as evident in the box simulations, with actual anatomical data the full cross-section approach offers more accurate orientation calculations than the other two methods. Misalignments, with respect to common scanner systems, of a local femur coordinate system by more than 20.0° in all three rotations is not to be expected in most clinical applications. This fact explains the small differences obtained in the computed results using anatomical information. The computation benefits associated with the sampled node method then become more attractive. Albeit a small speed tradeoff, the sampled node approach provides results which are almost as accurate as the contour information. The full cross-section method takes about 4 times longer. However, these computation requirement issues will most likely disappear with future advances in the semiconductor industry.

When used with sampled nodes as input, the principal axis algorithm exhibits some sensitivity to object geometry as can be seen with the simulations as well as with the actual anatomical data. By resampling the contours to obtain nodes, some information about the shape of the object is lost. In two-dimensions one can think of the resampling as a smoothing operation on the contour of the object cross-section. Extended to three-dimensions, this results in a data set of points that do not accurately represent the object and, in effect, produces a sort of aliasing on the data. The principal axis orientation calculations

are then affected. In addition, one would expect this problem to be more prevalent along the axes that result in the largest displacement of data points. This is demonstrated in both the simulations and the actual anatomical data computations, where the rotations about the X axis are the most prone to errors. This was expected since femur, data as well as the box simulation data, have their long axis along the Z direction of the scanning system. Any misalignment in this direction results in large displacement of the data points at the extremities of the object and makes the computation of the object orientation much more sensitive to the shape of the information captured by the scanner slicing. If node sampling does not provide a sufficient description of the contour, especially for small contours, some information critical to the principal axis calculation is simply omitted. Note that as the number of nodes increases (i.e.: sampling that includes all the data points in a contour), the sampled node approach converges to the full contour approach and provides the same results.

3.5 CONCLUSION

The method presented in this chapter provides an approach to systematically computing the orientation of structures from data acquired via CT or MRI. The ability to assign a coordinate system to a particular object, such as a femur which is independent of the imaging modality offers multiple benefits. First, it becomes possible to acquire data from both imaging systems and overlay the information of one onto the other without attention to the position and orientation of the subject during the scanning processes. Similarly, comparative longitudinal geometric analyses can now be performed from different data sets collected at different times. Finally, this method permits a standardized calculation of coordinate axes for the investigations of anatomical information from different subjects: for example, the impact of certain congenital as well as degenerative conditions can be studied on a large number of subjects without introducing bias from the imaging process (Certain conditions will have gross deformities that will not allow comparison of axes).

Different inputs can be the source for computing the orientation of objects being studied. Simulations as well as calculations based on actual anatomical data sets have demonstrated that the use of volumetric information, or voxels, (i.e.: all the data on a slice collected about the object) provides the most accurate results. However, with this method a penalty is incurred in terms of computational time and memory requirements This penalty can be reduced at the cost of a small trade-off in accuracy, especially with anatomical data, by selecting surface outline of the object, or even more simply, sampled points on the surface area of the object.

This method also has some limitations. For example, the orientations of certain geometries cannot be studied using the principal axis method. Fortunately, anatomical structures rarely present the geometric singularities that would dismiss such an approach.

For the purpose of this project, the principal axis method offers for the femur a coordinate system that can be used as a reference frame for the compilation of patient specific anatomical databases and thus provide the basis for computer-aided surgery computations. While some orientation errors are unavoidable, simulations have demonstrated that they can be considerably less than what would be considered significant for clinical applications and definitely far below the ability of surgeons to discriminate. While other approaches using either complex three-dimensional correlation of images [121, 122] or plane-fitted calculations of slice area vector shifts between two contiguous images [91] have been proposed, none provide for orientation computations the complete automation, and the conceptual simplicity, as well as the full three-dimensional perspective, present in the principal axis method.

CHAPTER
FOUR

CARTILAGE THICKNESS

4.1 INTRODUCTION

With patient specific anatomical information collected and a coordinate system defined to describe the skeletal structures, we can focus our attention on evaluating the joint congruency and the cartilage quality. Both CT and MRI images provide good quantitative information about skeletal structures as demonstrated in Chapter 2. However, cartilage tissue cannot easily be imaged with CT [50] and is poorly visualized with MRI when using the "Body Coil" mandatory at the hip joint. Studies have shown that MRI can image cartilage at the knee joint with the use of a local amplified coil often referred to as the "Knee Coil" [3, 8, 67, 125]. Such methods are unfortunately not available today at the hip level, and the invasive use of image enhancing dyes such as gadolinium for MR arthrograms presents some health risks. However, to best plan a corrective osteotomy, a visualization of the cartilage thickness condition at the joint is necessary. Although MRI technology will improve and perhaps achieve the resolution and accuracy to resolve this issue, to address this imaging and visualization problem now, this thesis presents a technique to model joint geometry, its kinematic congruency and the quality, or thickness, of the cartilage present on the articular surfaces.

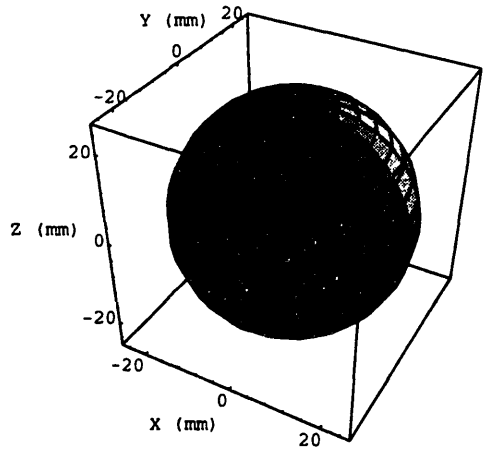
4.2 JOINT GEOMETRY

First, a method to analyze the joint geometry was devised and implemented. Using this modeling technique, the condition of the joint geometry at the bone level can be evaluated. Thus, gross deformities of the joint which would likely preclude a successful outcome with an osteotomy can easily be identified.

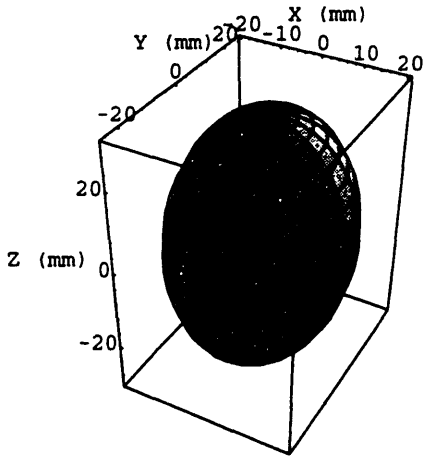
4.2.1. THEORY

In order to study the joint geometry, an algorithm was developed to find the best fit of known mathematical geometries to the large number of data points that define the bone surface of the femoral head and the acetabulum. The uneven data sampling resulting from CT or MR imaging does not permit the use of spherical harmonics to describe these anatomical geometries as had been previously adopted by other studies using ultrasound imaging [117]. To address this issue, an optimization can be performed to evaluate the optimal fit of a known geometry to a set of points in space. The global geometry of the human hip joint and aspects of the relative geometry of the articular surfaces have been the subject of several studies. Previous work has focused on two dimensional analyses from traditional X-rays [23, 51]. In three-dimensions, using these findings, three possible mathematical geometric representations were considered: a sphere, an ellipsoid and a rotated ellipsoid. These are illustrated in Figure 4.1 and are extrapolations of the best fits found with two dimensional analyses. The rotated ellipsoid, presented in Figure 4.1(c) addresses misalignment issues between the principal axes of the ellipsoid and the global coordinate system.

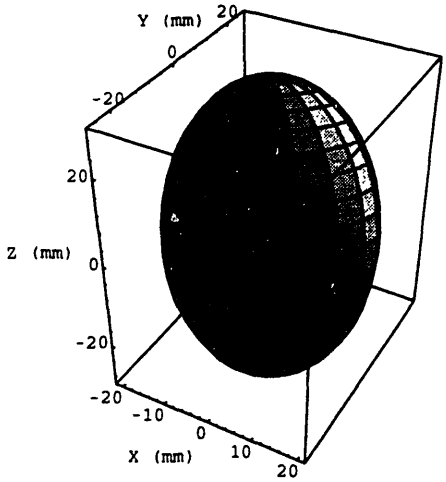
The optimization algorithm implemented and presented in Appendix E, is based on Direction Set Methods in Multidimensions which are also known as Powell's optimization methods. In 1964, Powell discovered a quadratically convergent method which produces N mutually conjugate directions. Thus with N variables, there are N directions to investigate, and with N(N+1) line minimizations one can exactly minimize a quadratic form. Brent [16] gives proofs of these statements in accessible form and the reader can further explore the theories behind Powell's optimization techniques through further reading from Acton and Jacobs [1, 58]. To improve the ability of Powell's method to find the optimal result, the algorithm adopted in this research uses a modified method which discards the direction of largest decrease at every iteration. While this may seem paradoxical, since that direction was the best of the previous iteration, it provides the best way to avoid a buildup of linear dependence and thus insures finding the global optimum. A more detailed explanation of this modified method is presented by Press *et al.* [100].



(a) Sphere



(b) Ellipsoid



(c) Rotated Ellipsoid

Figure 4.1: Hip Geometry Fit. Mathematical Models

The geometric shapes presented in Figure 4.1 can also be described mathematically. Minimizing error functions can then be inferred for each of the geometries and parameters identified for the optimization. Thus the equation of a sphere is given by:

$$\frac{(x-x_o)^2+(y-y_o)^2+(z-z_o)^2}{Radius^2}=1 \qquad \text{(Eq. 4.1)}$$

Where x_O, y_O and z_O are the coordinates of the center of the sphere. A best fit sphere to a three dimensional set of points can then be found by searching for the optimal sphere center and radius that minimize the following function, which will be referred to as the error cost function:

$$\frac{1}{n}\sum_n\left|\frac{(x-x_o)^2+(y-y_o)^2+(z-z_o)^2}{Radius^2}-1\right| \qquad \text{(Eq. 4.2)}$$

where n is the number of data points. The function is normalized by n to eliminate any dependency of the error cost function on the number of data points used as an input. One will note that in this case the optimization has 4 degrees of freedom. Similarly, the equation of an ellipsoid is:

$$\frac{(x-x_o)^2}{a^2}+\frac{(y-y_o)^2}{b^2}+\frac{(z-z_o)^2}{c^2}=1 \qquad \text{(Eq. 4.3)}$$

where a, b and c are the minor, median and major axes of the ellipsoid and x_O, y_O and z_O are the coordinates of the center of the ellipsoid. An error cost function can then be derived as follows:

$$\frac{1}{n}\sum_n\left|\frac{(x-x_o)^2}{a^2}+\frac{(y-y_o)^2}{b^2}+\frac{(z-z_o)^2}{c^2}-1\right| \qquad \text{(Eq. 4.4)}$$

For the case of the ellipsoid, note that the optimization has 6 degrees of freedom. Finally, the rotated ellipsoid can be defined using the same ellipsoid equations as above (4.3 and 4.4) and subjecting them to a pitch angle θ and a tilt angle φ (because of ellipsoidal symmetry a third rotation is not necessary to describe the rotated ellipsoid in space). The resulting rotation matrix is defined as follows:

$$\begin{bmatrix}\cos\varphi & 0 & -\sin\varphi\\ 0 & 1 & 0\\ \sin\varphi & 0 & \cos\varphi\end{bmatrix}\times\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & \sin\theta\\ 0 & -\sin\theta & \cos\theta\end{bmatrix}=\begin{bmatrix}\cos\varphi & \sin\varphi\sin\theta & -\sin\varphi\cos\theta\\ 0 & \cos\theta & \sin\theta\\ \sin\varphi & -\cos\varphi\sin\theta & \cos\varphi\cos\theta\end{bmatrix} \qquad \text{(Eq. 4.5)}$$

In this case, note that the optimization has 8 degrees of freedom and a non-unique solution: the same optimal rotated ellipsoid solution can be described by several equivalent sets of ellipsoid origins, axes and orientations.

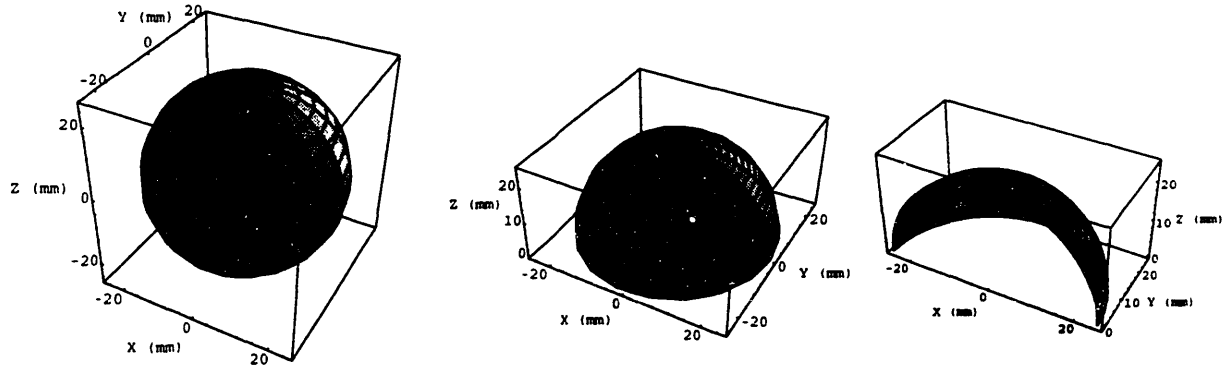
4.2.2 SIMULATIONS

The algorithm was first evaluated with simulated data. As illustrated in Figure 4.2, to best represent actual data on a femoral head and acetabulum, points were generated on the entire, one half and a quarter of the surface of a sphere (a), ellipsoid (b) and rotated ellipsoid (c). To further duplicate the effects of noise inherent in any CT or MRI acquisition system a noise level of 10% of the sphere radius and ellipsoid axes was introduced (this noise level is considered high when compared with the noise levels experienced with CT and MRI during anatomical data collection). Points were uniformly distributed over the surface area considered.

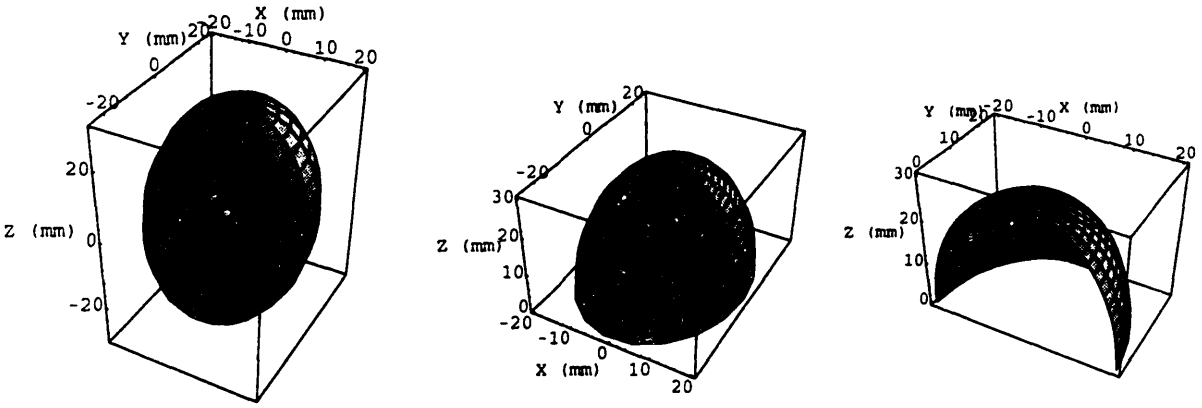
The optimization algorithms were then tested with data sets ranging from 5 to 5000 points in order to analyze the dependency of the optimization results to the density of the data. The results for the sphere case are presented in Figure 4.3. As can be seen for the complete sphere tests (a), the errors in radius rapidly drop off to less than 0.05%. In addition, the distance error between the actual center of the sphere and the optimized center was found to drop to less than 20 microns. Similar patterns can be found in the results for the half sphere (b) and the quarter sphere (c) tests. A degradation in the accuracy of the optimized calculations can be observed with data points representing half or a quarter of the sphere. However, even in the worst possible case of the quarter sphere, the errors converge to less than 0.1% in radius and less than 50 microns in center positioning.

Results for the ellipsoid optimization simulations are presented in Figure 4.4. In the case of the complete ellipsoid (a), errors in ellipsoidal axes were found to drop to less than 0.1% while the ellipsoid center positioning error converged to less than 20 microns. Again, optimization results lose accuracy for the half ellipsoid (b) and the quarter ellipsoid (c) but remained accurate to within 0.5% in axes estimation and 400 microns in ellipsoid center position evaluation. Optimization times doubled when compared to the sphere simulation tests.

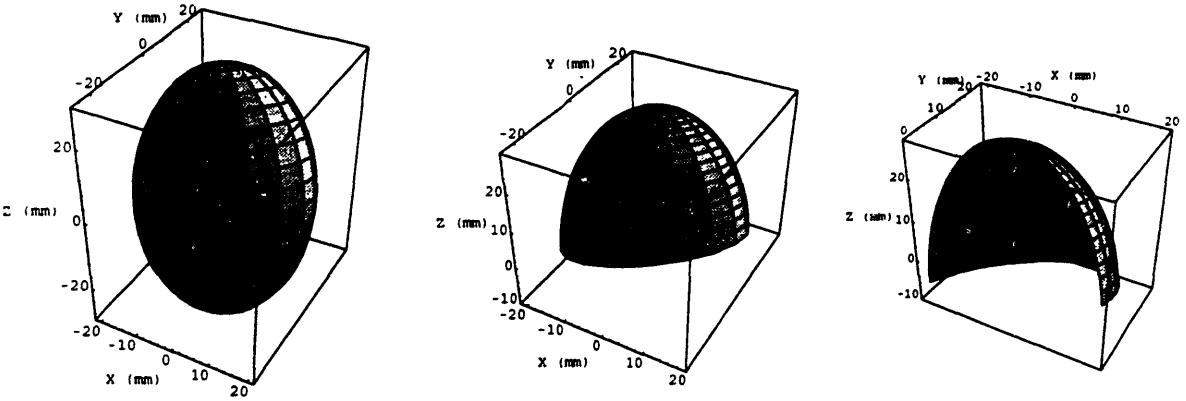
The rotated ellipsoid simulation results are presented in Figure 4.5 and 4.6. The errors found in axes dimensioning and ellipsoid center positioning are very similar to those found in the previous case. For the worst case, the quarter rotated ellipsoid (c), the ellipsoid principal axes were estimated to within 1.5% and its center to within 500 microns. In addition,



(a) Spherical Test: Radius = 25.0 mm.

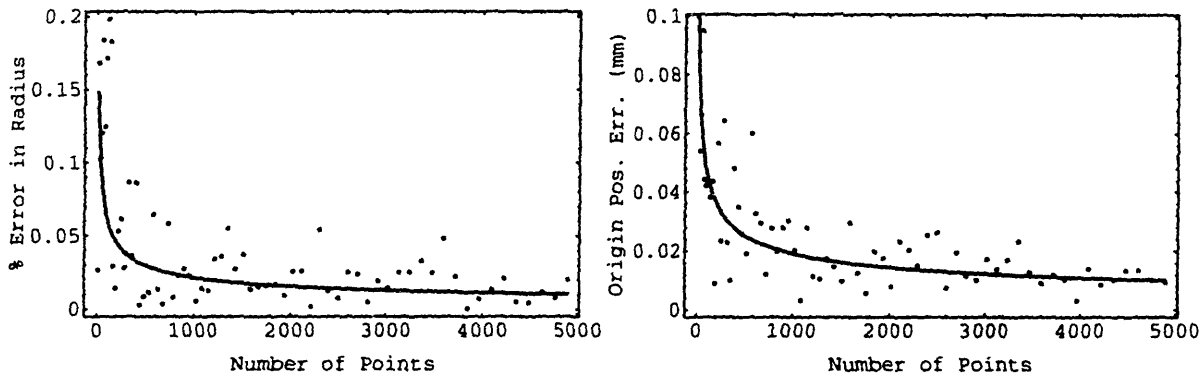


(b) Ellipsoidal Test: A = 20.0 mm, B = 25.0 mm, C = 30.0 mm.

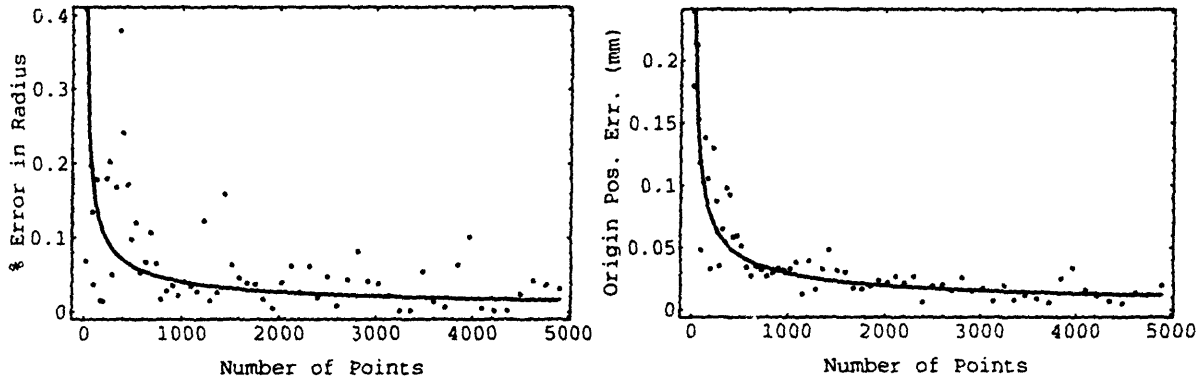


(c) Rotated Ellipsoidal Test: A = 20.0 mm, B = 25.0 mm, C = 30.0 mm, $\theta = 20°$, $\varphi = 15°$.

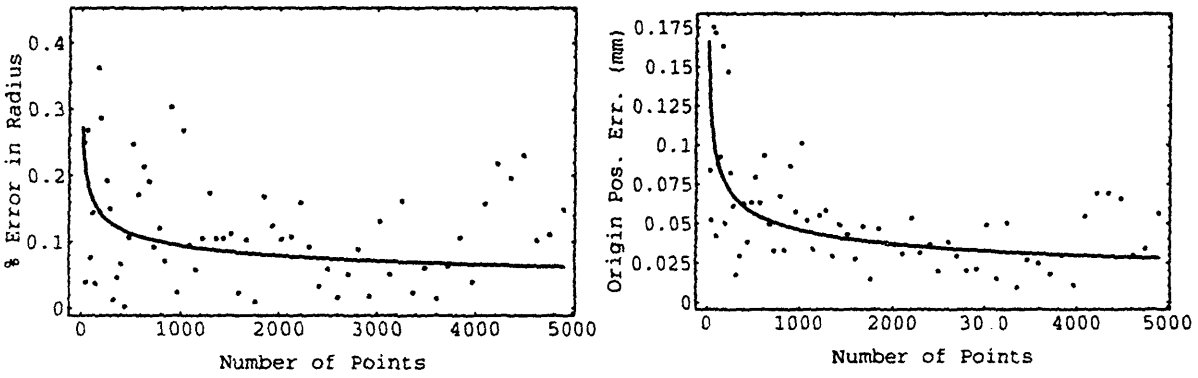
Figure 4.2: Hip Joint Geometry Simulation Test Cases.



(a) Complete Sphere

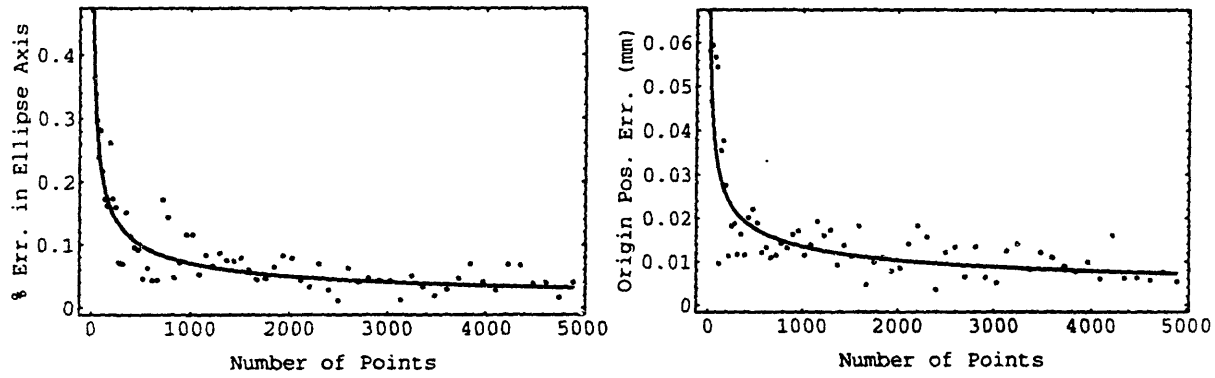


(b) Half Sphere

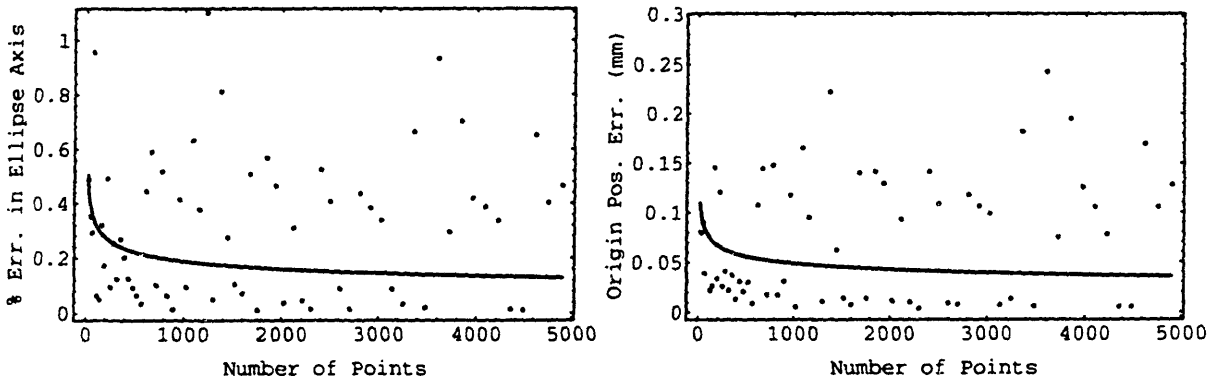


(c) Quarter Sphere

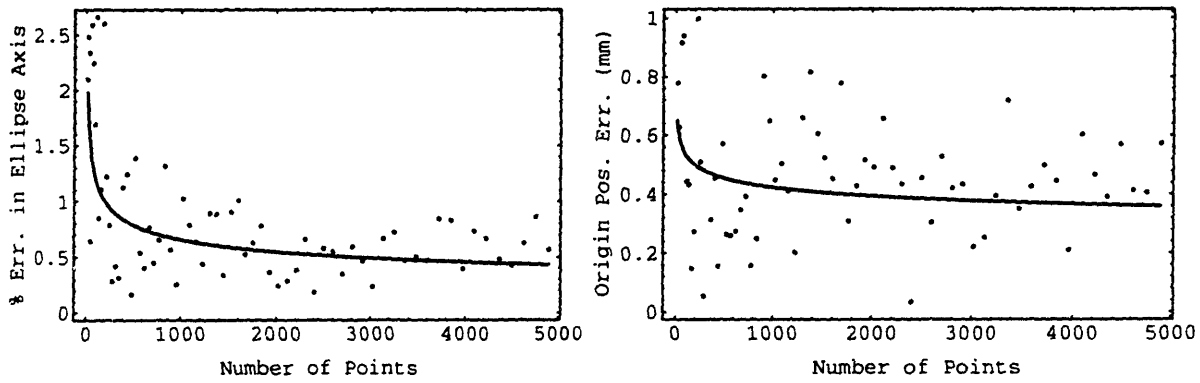
Figure 4.3: Sphere Fit Simulations



(a) Complete Ellipsoid

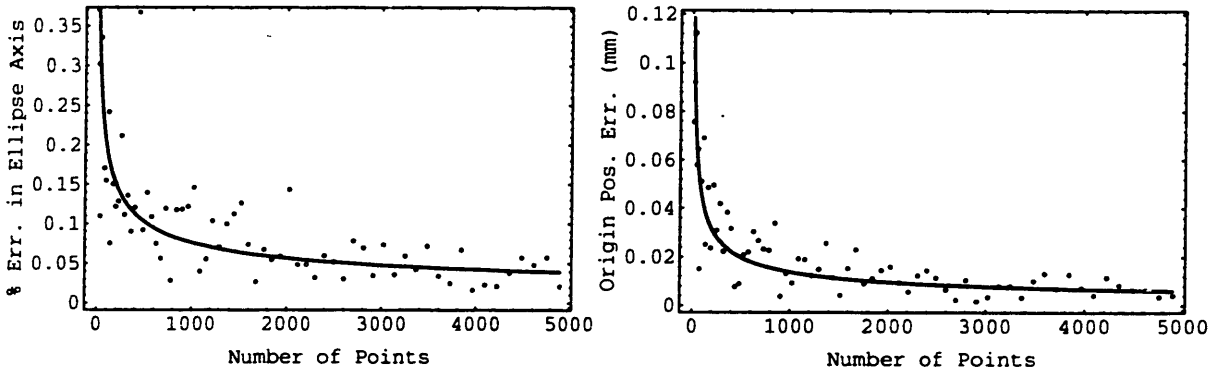


(b) Half Ellipsoid

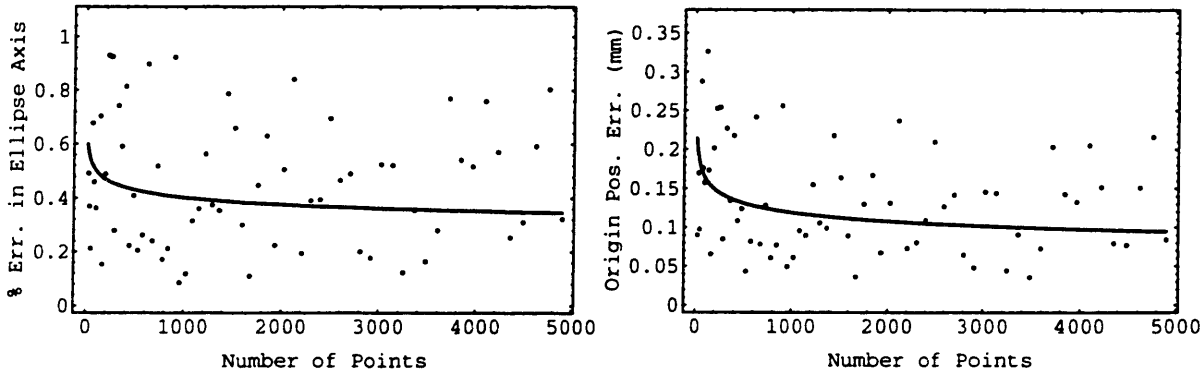


(c) Quarter Ellipsoid

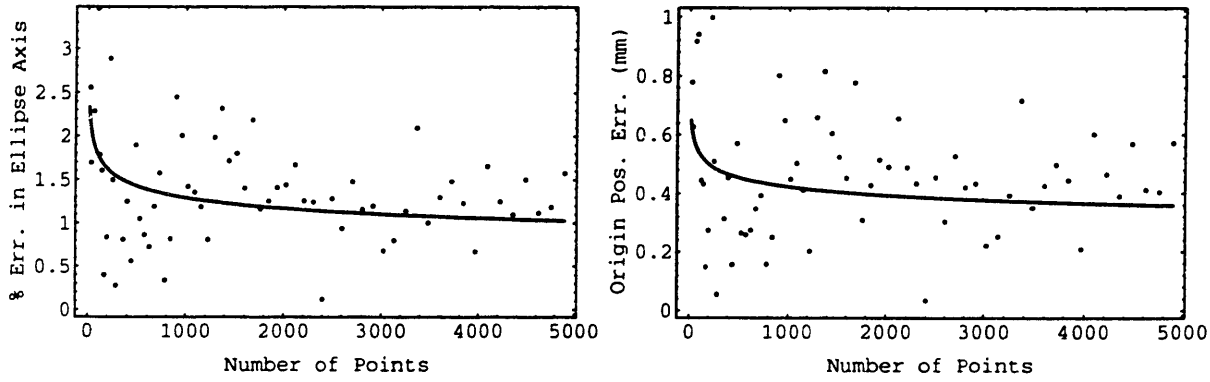
Figure 4.4: Ellipsoid Fit Simulations



(a) Rotated Complete Ellipsoid

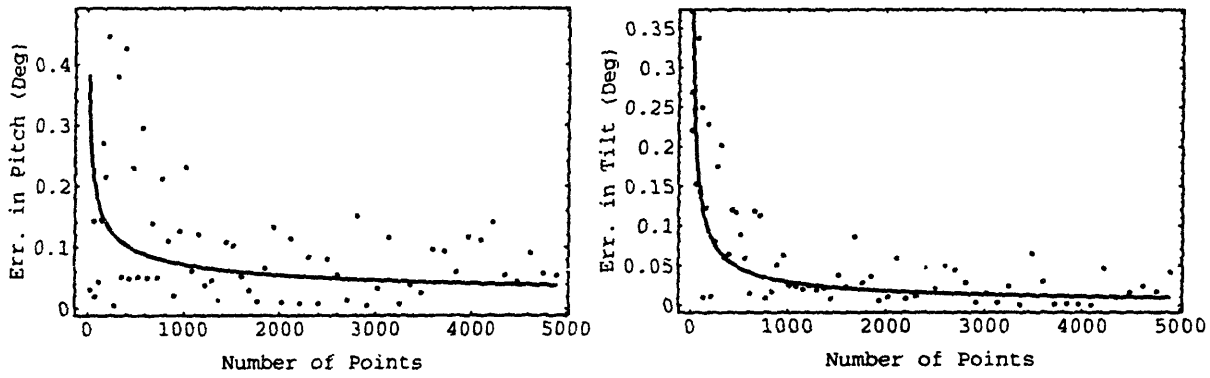


(b) Rotated Half Ellipsoid

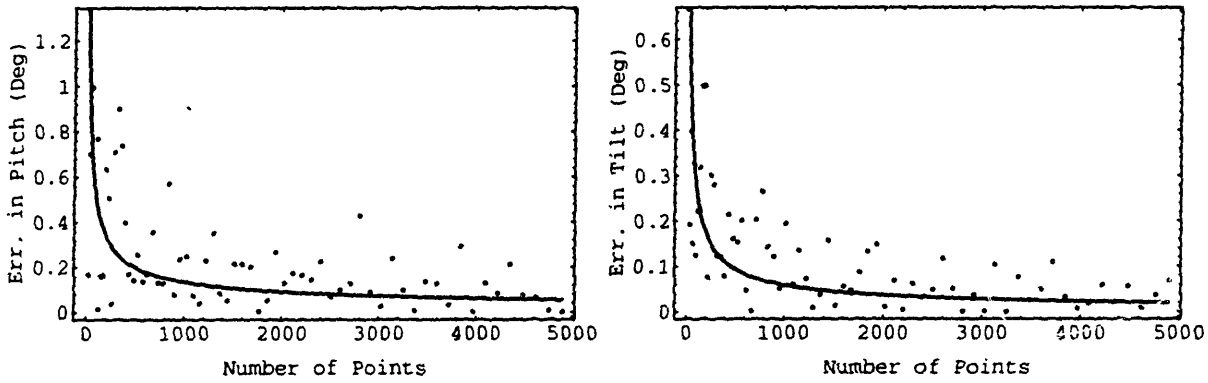


(c) Rotated Quarter Ellipsoid

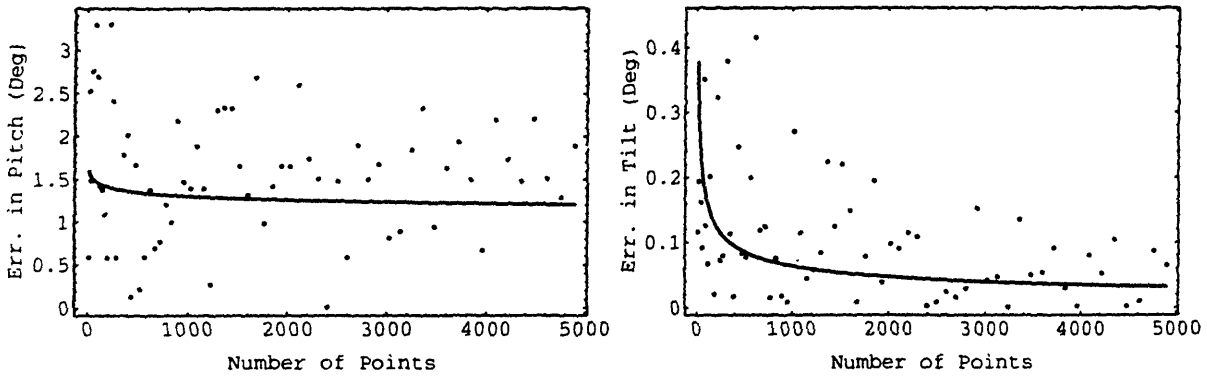
Figure 4.5: Rotated Ellipsoid Fit Simulations (Part I)



(a) Rotated Complete Ellipsoid



(b) Rotated Half Ellipsoid



(c) Rotated Quarter Ellipsoid

Figure 4.6: Rotated Ellipsoid Fit Simulations (Part II)

absolute errors in angular estimations of the pitch and tilt of the ellipsoid were found to be less than 0.1° for the rotated complete ellipsoid (a), less than 0.2° for the rotated half ellipsoid (b) and less than 1.5° for the rotated quarter ellipsoid (c). It should be noted that the increase in error magnitude observed in the orientation calculations correlates directly with the increase in the ellipsoid axes dimensions and center. Optimization times quadrupled when compared to the sphere simulation tests.

Overall, these simulations indicate the robustness of these optimization algorithms in extracting the underlying geometry from a noisy surface data. Other simulations were performed with no noise in the data and showed that the optimization algorithms were accurate to the computer floating point representation (6 decimal places or 0.001 microns). In the case of the ellipsoid and rotated ellipsoid, the algorithms also demonstrated the same level of computational accuracy as well as little sensitivity to the magnitude difference between the nominal minor and major axes.

4.3 CARTILAGE THICKNESS ESTIMATION

Using the method presented above it is possible to model the joint at the bone level by assigning mathematical models to the geometry of the femoral head and the acetabulum. This information permits analyzing the joint in terms of congruency and geometric shape but does not infer any information about the quality of the joint cartilage. Since cartilage data is difficult to obtain with CT or MRI, this thesis presents below a technique to estimate the inter-articular cartilage thickness, based on the joint geometry and kinematics.

4.3.1. THEORY

In the absence of quantitative information about the cartilage layers, the best estimate of the location of the inter-articular surfaces, or the cartilage sliding contact surfaces, is that set of points midway from the femoral head and acetabulum bone structures that form a sphere in order to satisfy kinematic integrity. We know from previous anatomical studies, as well as from kinematic analyses, that the hip joint can be mapped accurately as a ball and socket joint. From this, it can be inferred that the surfaces of the cartilage on the femoral head and that inside the acetabulum both map to a sphere. Figure 4.7 presents a diagram where the bony femoral head as well as the acetabulum and cartilage sliding contact surfaces have been simplified to circles in two dimensions. In this case, the sliding contact surfaces are defined as that circle with center mid-point from the femoral and acetabulum bone circle centers and

with radius the average of the femoral and acetabular radii. The non-congruent case illustrated in Figure 4.7 suggests how an area of the femoral head could be covered with only a thin layer of cartilage: this is of course the case in the natural joint especially when affected by osteoarthritis. Similarly, one can extend the concept to the case of two perfect spheres representing the femoral head and acetabulum geometries where the sliding contact surface is the sphere centered in between the other two sphere centers with a radius that is the average of the other two sphere radii.

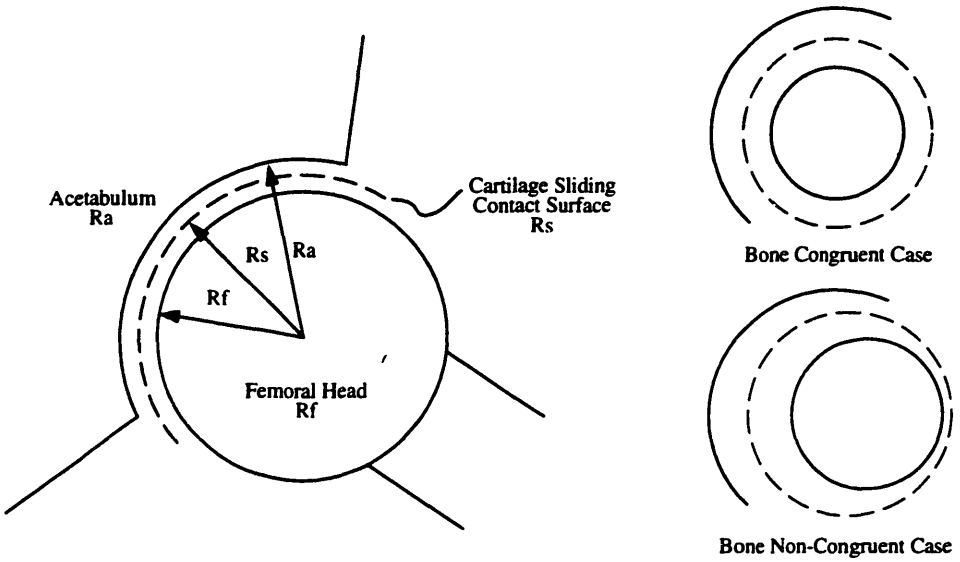


Figure 4.7: Two-Dimensional Definition of Cartilage Sliding Contact Surfaces

The reader can extend this concept to the bone of a femoral head and corresponding acetabulum mapped by two rotated ellipsoids, off-centered and with different orientations. The surface resulting from these points that are half-way between the two ellipsoids will not necessarily define a known mathematical shape. The quality of the sphere fit to the data will validate the assumption made with respect to the geometry of the cartilage sliding contact surface. The algorithm, listed in Appendix E, implements an optimization based on Powell's method similar to the one described previously. The cartilage thickness can then be calculated from the distances between the bone and the optimized articulating surface. This approach provides an estimate of the cartilage thickness, both on the femoral head and the acetabulum, based on the bone geometry and joint kinematics.

4.3.2 SIMULATIONS

Simulations for known test cases were performed to check the behavior of the algorithm. For example concentric and off-centered spheres were considered. The behavior of the algorithm was understood and is verified by the simulation results presented in Table 4.1. As expected

in this case, the resulting sliding contact surface is a sphere with center at the mid-point from the other two sphere centers and with a radius that is the average of the other two sphere radii. Verifying the algorithm for cases involving ellipsoids is not as easily done. The complexity of the geometry in three-dimensions, especially when the ellipsoids are rotated with respect to one another, makes visualization of the problem difficult. Table 4.1 also presents some results for such test cases. It is left to the reader to verify and convince himself/herself of the validity of such results.

Table 4.1: Simulation Results for the Sliding Contact Surface Estimation Algorithm.

Concentric Spheres		**Calculated Sliding Contact Sphere**
x = 0, y = 0, z = 0 and Radius = 25 mm x = 0, y = 0, z = 0 and Radius = 28 mm	⇒	x = 0, y = 0, z = 0, Radius = 26.5 mm
Off Centered Spheres		**Calculated Sliding Contact Sphere**
x = 1, y = 0, z = 0 and Radius = 25 mm x = 0, y = 0, z = 0 and Radius = 28 mm	⇒	x = 0.5, y = 0, z = 0, Radius = 26.5 mm
Concentric Ellipsoids (Pitch = Tilt = 0°)		**Calculated Sliding Contact Sphere**
x = 0, y = 0, z = 0, a = 24, b = 25 and c = 26 mm x = 0, y = 0, z = 0, a = 27, b = 28 and c = 29 mm	⇒	x = 0.003,y = 0.002, z = 0.031, Radius = 26.845 mm
Off Centered Ellipsoids (Pitch = Tilt = 0°)		**Calculated Sliding Contact Sphere**
x = 1, y = 0, z = 0, a = 24, b = 25 and c = 26 mm x = 0, y = 0, z = 0, a = 27, b = 28 and c = 29 mm	⇒	x = 0.451,y = 0.000, z = -0.019, Radius = 26.835 mm
Concentric Ellipsoids (Pitch = 20°, Tilt = 15°)		**Calculated Sliding Contact Sphere**
x = 0, y = 0, z = 0,a = 24, b = 25 and c = 26 mm x = 0, y = 0, z = 0, a = 27, b = 28 and c = 29 mm	⇒	x = -0.012,y = -0.027, z = -0.016, Radius = 26.805 mm

Overall, the simulation results presented in Table 4.1, show that the sliding contact surface optimization technique, based on the modeling of the joint bone geometry provides a unique approach to estimate the cartilage thickness distribution on the femoral head and inside the acetabulum.

4.4 ANATOMICAL RESULTS

The algorithms were then tested with anatomical information. First, such information is processed to only include the data points that define the femoral head and acetabulum bone geometry. Figure 4.8 illustrates how the anatomical data is segmented by highlighting that bone surface which supports the joint load. These load bearing surfaces which define the

femoral head and acetabulum geometries are used as input to evaluate the best fit using either a sphere, ellipsoid or rotated ellipsoid. Although counter-intuitive, the results were found to be insensitive to the selection process of the load bearing surfaces and how much of the bone information is selected to define these surfaces. Large variations in the amount of anatomical information considered did not affect the final results by much more than 10 μm, a hundredth of the overall resolution of the anatomical data collected. This permits considerable freedom in the specification of the information, which incorporates not only quantitative but also qualitative since the intervention and judgment of an operator to visually identify the load bearing surfaces is required. The large number of data points available even permits "super resolution", similar to subpixeling, since each point can be considered as one additional measurement of the same three dimensional structure. Noise effects can be reduced and virtually eliminated, while optimized geometries display accuracies greater than the original anatomical measurements.

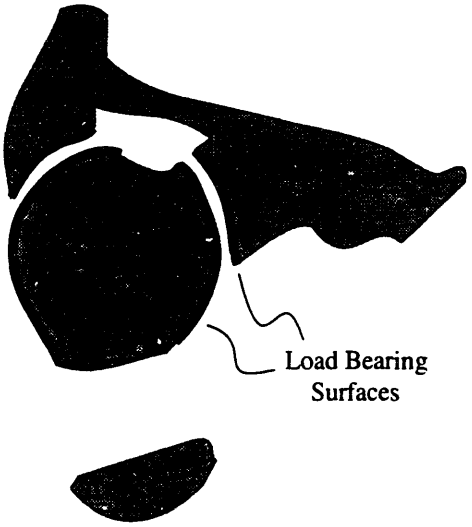
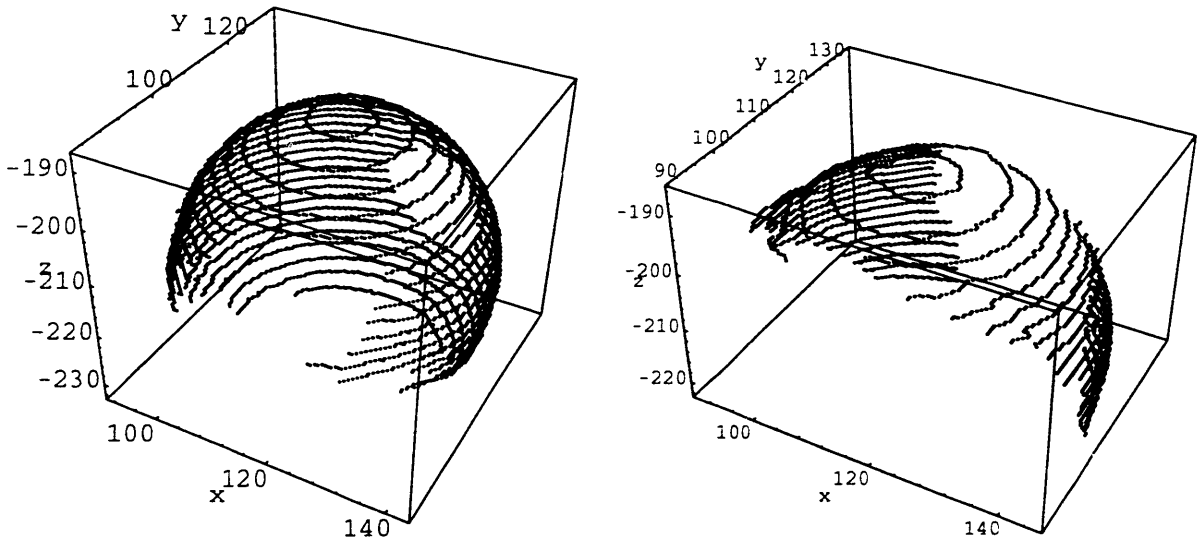


Figure 4.8: Highlighted Load Bearing Surfaces

4.4.1 CT CADAVER DATA

The geometry optimization and cartilage estimation algorithms were tested with anatomical data collected via CT imaging on a 61 year old 220 lb. male. The bone geometry selected for the load bearing surfaces of the femoral head and acetabulum are presented in Figure 4.9. For this example the femoral data is composed of 4098 points and the acetabular data of 2131 points. The results from the different optimizations are presented in Table 4.2. First notice that the fit does not dramatically improve with an increase in geometric model complexity as evidenced by the error cost functions. The centers of the sphere, ellipsoid and rotated ellipsoid fits show the same coordinates. Similarly, the magnitude of the minimum cost function did not improve much, suggesting that little can be gained by using the rotated



(a) Femoral Head (b) Acetabulum

Figure 4.9: Selected CT Femoral and Acetabulum Load Bearing Surface Data

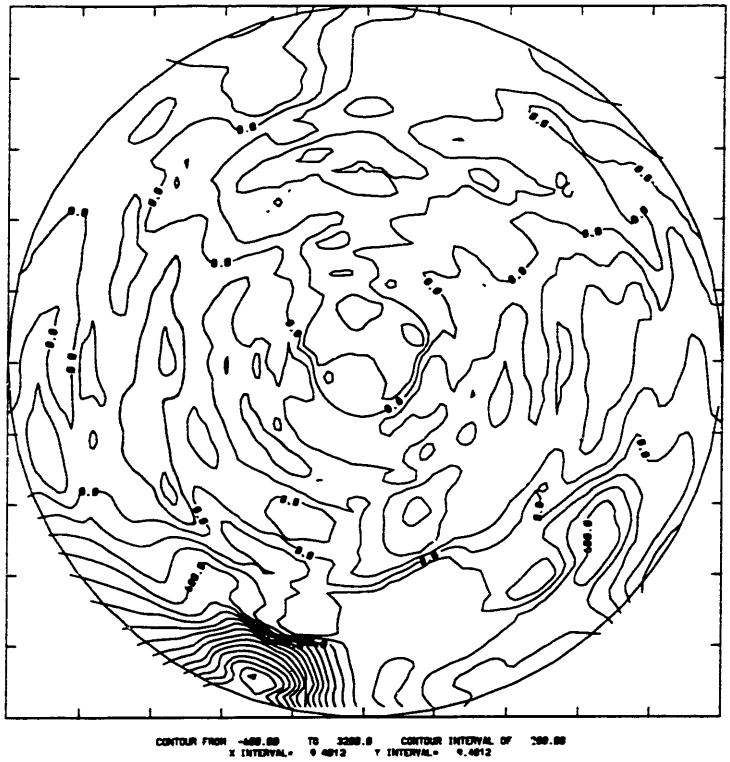
Table 4.2: CT Femoral and Acetabulum Geometric Fit Results

Femoral Data (4098 points)	**Acetabulum Fit (2131 points)**
Sphere x = 117.442, y = 108.065, z = -213.068 mm Radius = 25.368 mm Minimum function value = 0.018839	**Sphere** x = 117.699, y = 108.784, z = -212.680 mm Radius = 27.348 mm Minimum function value = 0.021233
Ellipsoid x = 117.442, y = 108.065, z = -213.068 mm a = 25.325, b = 25.585, c = 25.252 mm Minimum function value = 0.018326	**Ellipsoid** x = 117.699, y = 108.784, z = -212.680 mm a = 27.246, b = 27.543, c = 27.423 mm Minimum function value = 0.020724
Rotated Ellipsoid x = 117.442, y = 108.065, z = -213.068 mm a = 25.408, b = 25.664, c = 25.108 mm with Pitch = 22.48° and Tilt = 41.33° Minimum function value = 0.016392	**Rotated Ellipsoid** x = 117.699, y = 108.785, z = -212.680 mm a = 27.154, b = 27.869, c = 27.313 mm with Pitch = -40.11° and Tilt = -31.1° Minimum function value = 0.019201
Joint Congruency **Distance Between Ellipsoidal Centers = 857 μm**	
Estimated Cartilage Sphere **x = 117.573, y = 108.418, z = -212.864, Radius = 26.351 mm** **Minimum function value = 0.009246**	

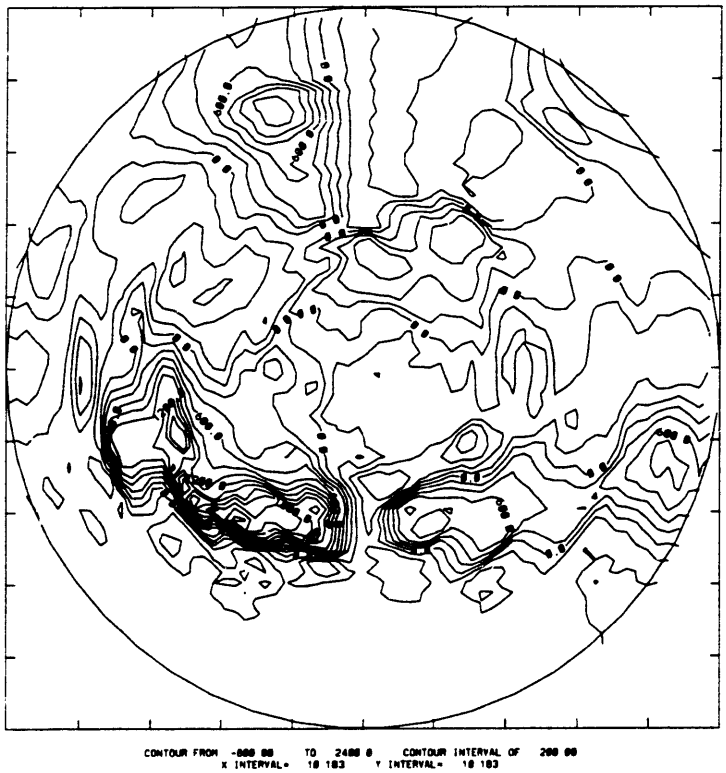
ellipsoid fit over the simpler spherical fit in this particular case. While more variations are observed for the acetabulum than for the femoral head, the differences between the minor and the major axes of the rotated ellipsoid fits are small for both, 556 μm for the femoral head and 715 μm for the acetabulum. The joint congruency, the distance between the centers of the bony femoral and acetabular surfaces, was found to be very good at 857 μm. Finally, the spherical fit to the data points forming the sliding contact surface is very accurate as demonstrated by the optimized cost function minimum value of 0.009246 translates into a mean squared error of less than 94 μm. This result confirms our original assumption that kinematic integrity at the joint requires that the sliding contact surface be a sphere. Optimization times were found to vary greatly with data sets (femoral or acetabular) but also with the number of data points in each set. Execution times ranged from as little as 35 minutes for spherical fits to as long as 2.5 hours for rotated ellipsoid fits.

The sphericity results can also be presented in a more graphical fashion. Figure 4.10 presents sphericity contour maps using a simple 3D to 2D conformal transformation. While these maps present the information in a very quantitative way, they are difficult to use to qualitatively visualize the areas on the femur and inside the acetabulum that are out of sphericity. To remedy this limitation, a three dimensional display software, FAST, was adopted and the results are presented in Figure 4.11. FAST (Flow Analysis Software Toolkit) is a software environment for visualizing data. Within FAST, a collection of separate programs (modules) run simultaneously and allow a scientist to examine the results of numerical and experimental simulations. Although NASA Ames Research Center Workstations Applications Office implemented FAST with computational fluid dynamics as its primary focus, the software can be used for other types of visualization. Implementing a few simple filters to convert the anatomical data and cartilage maps into FAST data formats is all that is required to empower users with the ability to manipulate the display in real-time and, as their viewing point changes, to monitor the sphericity patterns overlaid on the femoral head and inside the acetabulum. Providing such a display with a reference frame to the anatomical information has proven to be an invaluable visual aid.

Using the calculated sliding contact surface sphere, it is then possible to calculate the thickness distribution of cartilage on the acetabulum by computing the local distances from the bone surface to the sliding surface. Results can be presented as cartilage contour maps such as in Figure 4.12 but also as three dimensional displays such as in Figure 4.13 to enhance the qualitative understanding of where areas of thin cartilage are located. For example , Figure 4.13(a) clearly indicates in black an area above the femoral *fovea* as a region of thin or no cartilage.

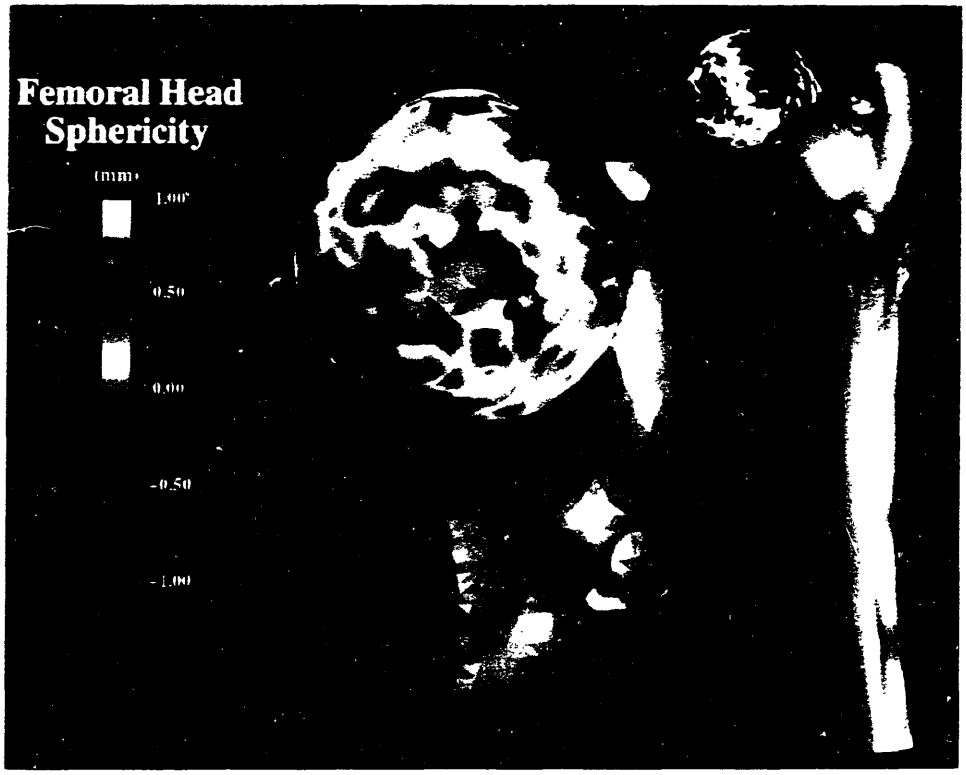


(a) Femoral Head

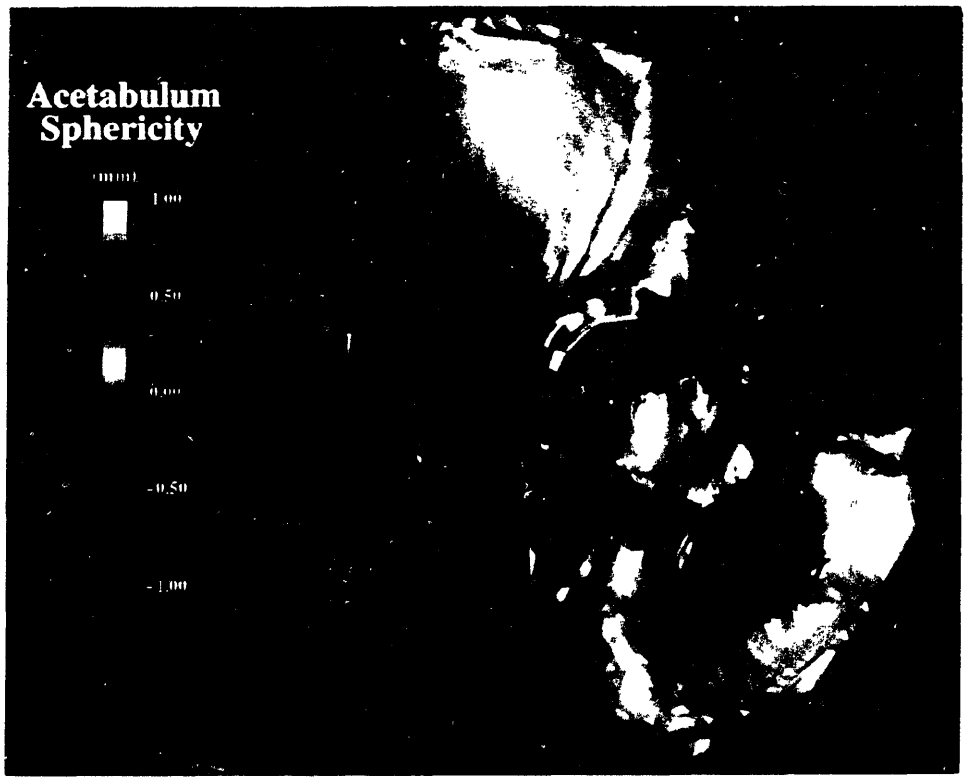


(b) Acetabulum

Figure 4.10: Femoral and Acetabular Sphericity Contour Maps

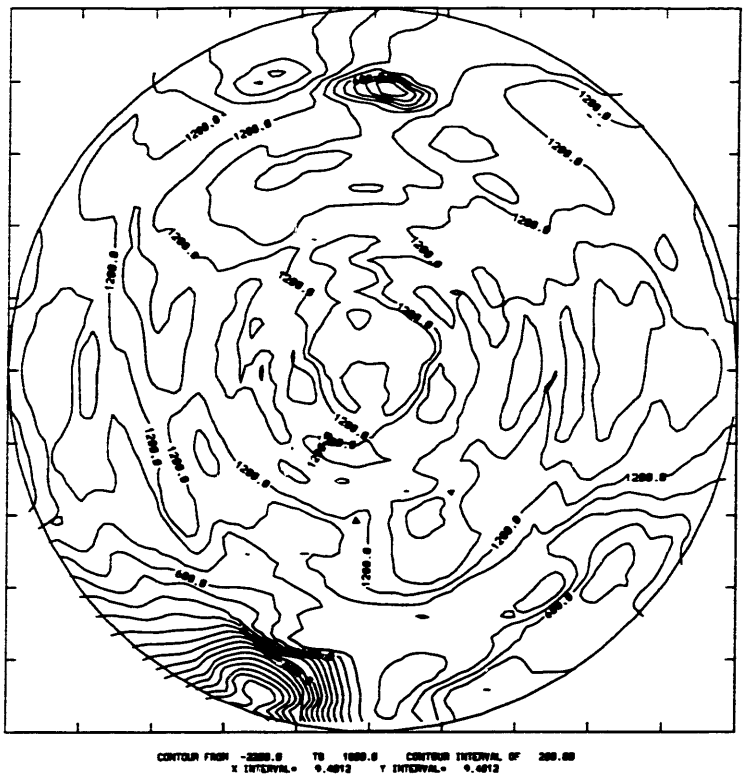


(a) Femoral Head

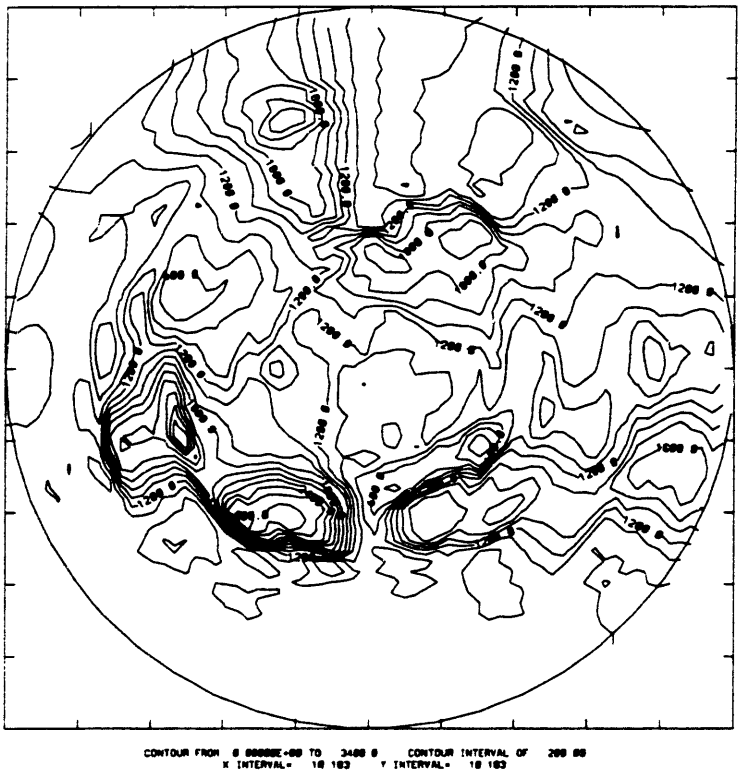


(b) Acetabulum

Figure 3.11 Femoral and Acetabular Sphericity Three-Dimensional Maps

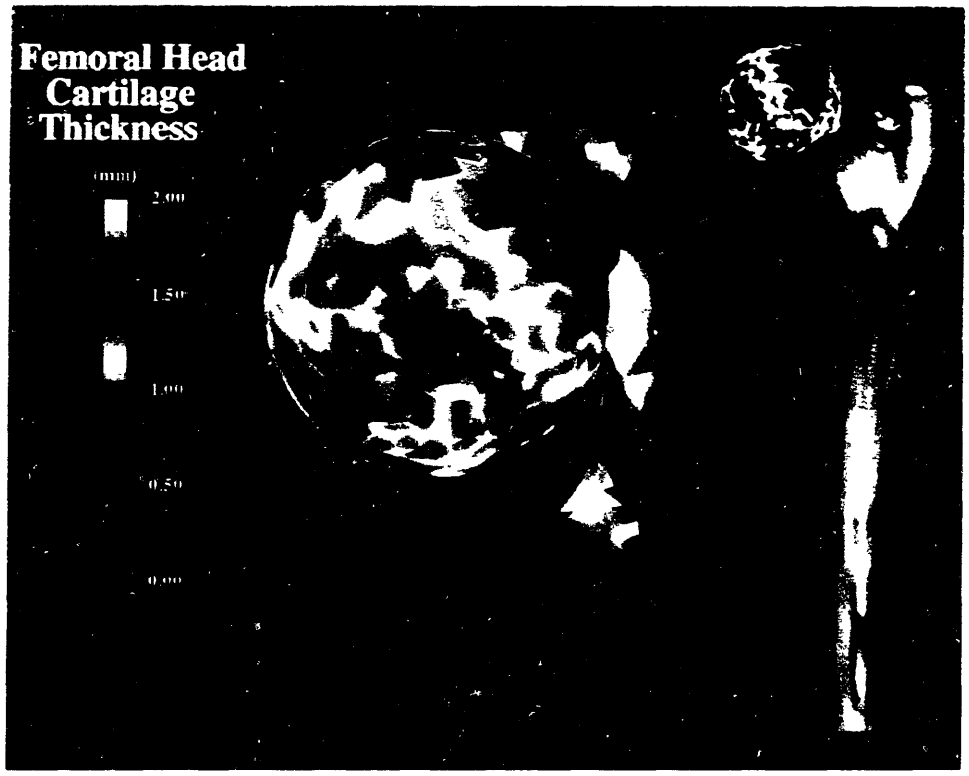


(a) Femoral Head

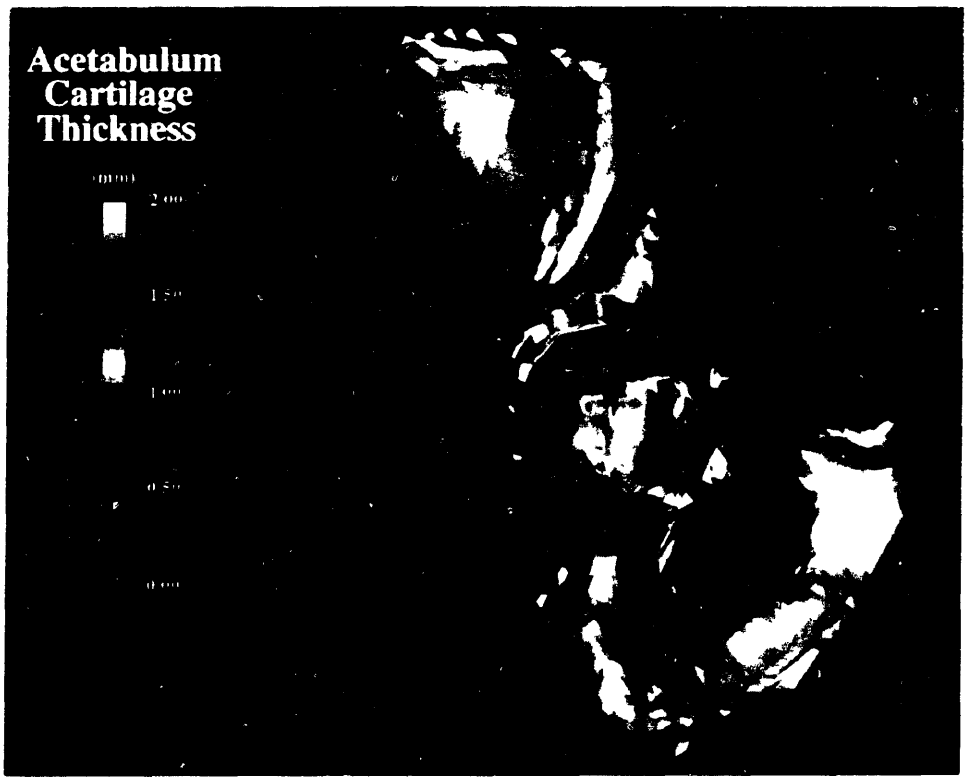


(b) Acetabulum

Figure 4.12: Femoral and Acetabular Cartilage Thickness Contour Maps



(a) Femoral Head



(b) Acetabulum

Figure 4-15: Femoral and Acetabular Cartilage Thickness Three-Dimensional Maps

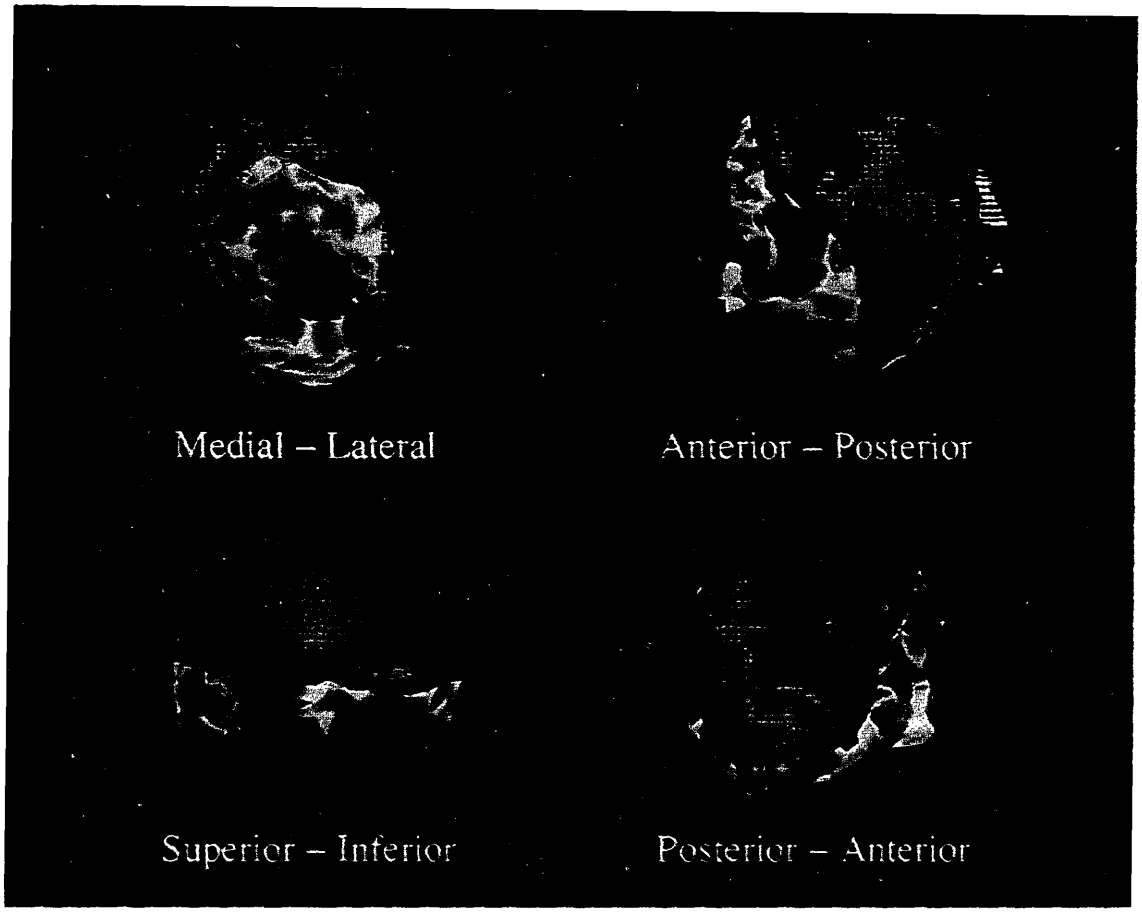


Figure 4.14: Femoral - Acetabular Cartilage Map Overlap

In addition to displaying the cartilage thickness within the joint it is important to visualize the respective areas of the femoral head and the acetabulum which contact each other as the joint moves, defined here as overlap [103, 122]. Figure 4.14 illustrates this overlap by displaying the femoral head with its cartilage thickness map and the acetabular as a translucent object riding on the femoral surface. This display can be animated throughout the entire joint kinematic range and permits a better understanding of the mechanics that bring defective or absent femoral cartilage areas opposed to the acetabular surface, resulting in contact which produces pain. Such an approach allows users to better plan the corrective osteotomy by emphasizing three dimensional aspects of the correction at the joint cartilage level.

4.4.2 MRI IN-VIVO DATA

The cartilage thickness estimation method was then tested with anatomical data collected via MRI imaging on a 25 year old 135 lb. male. The bone surfaces selected for the load bearing areas of the femoral head and acetabulum are presented in Figure 4.15. For this example the femoral data is composed of 1051 points and the acetabulum data of 1085 points.

Table 4.3: MRI Femoral and Acetabulum Geometric Fit Results.

Femoral Data (1051 points)	**Acetabulum Fit (1085 points)**
Sphere x = 142.886, y = 103.555, z = -62.657 mm Radius = 21.360 mm Minimum function value = 0.041950	**Sphere** x = 141.718, y = 104.330, z = -64.083 mm Radius = 28.436 mm Minimum function value = 0.076803
Ellipsoid x = 142.886, y = 103.555, z = -62.657 mm a = 21.053, b = 21.318, c = 21.885 mm Minimum function value = 0.039770	**Ellipsoid** x = 141.718, y = 104.330, z = -64.083 mm a = 29.702, b = 26.790, c = 28.714 mm Minimum function value = 0.067756
Rotated Ellipsoid x = 142.886, y = 103.555, z = -62.656 mm a = 20.791, b = 21.319, c = 21.961 mm with Pitch = 6.39° and Tilt = -16.35° Minimum function value = 0.037556	**Rotated Ellipsoid** x = 141.719, y = 104.330, z = -64.083 mm a = 30.136, b = 26.721, c = 28.690 mm with Pitch = 1.93° and Tilt = 23.97° Minimum function value = 0.066891
Joint Congruency **Distance Between Ellipsoidal Centers = 2.000 mm**	
Estimated Cartilage Sphere **x = 142.326, y = 104.000, z = -63.389, Radius = 25.113 mm** **Minimum function value = 0.032433**	

4.4.3 METHOD VALIDATION WITH MRI INFORMATION

To validate the method presented above, a comparison between optimized results and actual measurements of the cartilage surface is necessary. The joint interarticular space is filled with a thin layer of synovial fluid. This layer creates a small disturbance in the proton density excited by the magnetic field which can be detected with MRI. Though scarce, data can be collected on the location of this interarticular space. While imaging of this thin gap can be enhanced by the use of magneto-opaque dyes (Gadolinium), this procedure was not adopted because of invasiveness as well as potential health risks to the patient. Using imaging filters to enhance the MRI images of the subject's hip, it is possible to highlight information about the cartilage interface. Although extremely tedious, each data point can be manually digitized which permits the segmenting and contouring of the joint surface. Figure 4.16 shows the 1829 data points that map the interarticular cartilage sliding surface for the MRI data set presented above. The sphericity of this data is qualitatively confirmed with a simple visual inspection. To confirm it quantitatively, an optimal sphere was computed to fit the data. Several test runs were performed to test algorithm sensitivity to operator judgment. No significant changes in the optimization results were observed.

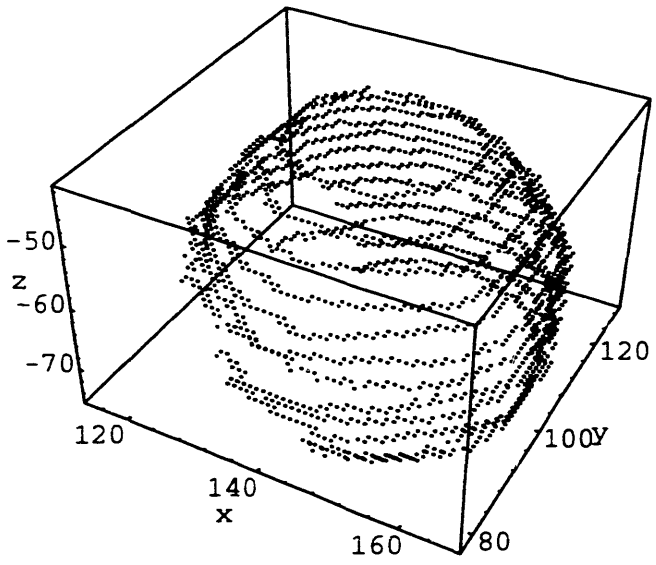


Figure 4.16: Measured MRI Joint Interarticular Space Data

Optimization results are presented in Table 4.4. As demonstrated by the minimum value of the cost function, a sphere is a good fit to model the measured interarticular gap. When the measured and the estimated cartilage surface sphere are compared, a difference of only 350 μm in center position and 337 μm in radius dimension was recorded; an astonishing result, indeed. These results clearly validate the assumptions inherent in the cartilage thickness estimation method presented in this thesis. Compared to the resolution of MRI of 930 μm, the accuracy of the cartilage estimation technique is about three times better, a consequence of the vast number of data points we use as a measure of the mathematical sphere surface. The optimization proposed in this research is similar to an optimal estimator, that is to say, a computational algorithm that processes measurements to deduce a minimum error estimate of the state of the system by utilizing 1) knowledge of the geometry of the system and of the measurements, and 2) statistics of system noises and measurement errors, both combined with initial condition information on the kinematic restrictions at the joint.

Table 4.4: Measured Cartilage Results

Measured Cartilage Sphere Fit (1829 points)
x = 142.616, y = 103.804, z = -63.400 mm Radius = 24.776 mm Minimum function value = 0.035272
Congruency of Measured Cartilage Fit to Estimated Cartilage Fit
Distance Between Spherical Centers = 350 μm Difference in Radii = 337 μm

With this approach the noise is assumed to be "white" or Gaussian and therefore every measurement is weighted the same in the final optimal solution. Among the presumed advantages of this type of data processor are that it minimizes the estimation error in a well defined statistical sense and that it utilizes all measurement data combined with pre-knowledge of the system. The corresponding potential disadvantages are sensitivity to erroneous *a priori* models and statistics, and the inherent computational burden. The theory of optimal estimation has already had application to a broad range of problem areas in medical fields, including tracer studies in nuclear medicine, statistical image enhancement, and classification of vectorcardiograms.

4.5 DISCUSSION

The method proposed here applies optimization both on redundant measurements of the bone surface geometry and then on best estimates of the cartilage surface position. Using assumptions on what best fits the geometries of the femoral head and acetabulum, sampled data points defining the subchondral bone determine the parameters of such geometries. Results from simulations, as well as from anatomical data, demonstrate that in three dimensions, rotated ellipsoids provide excellent fits to the actual geometry of the femoral head and acetabulum bone. These results extend to three-dimensions, the two-dimensional results of prior studies. For example, these results herein agree with previous work by Blowers [11] which demonstrated that younger patients have more ellipsoidal femoral heads and acetabuli than older ones. Over time, due to loading, bone growth and cartilage calcification, as well as degeneration, the bone geometries tend to become more spherical [45, 46]. Also, most of the sphericity at the bone level was found in the flexion extension trajectory of the joint kinematics, both on the femoral head and inside the acetabulum [106]. Other studies [21, 110] have indicated that congruency of the femoral head and acetabulum bone improves with age in a normal joint (as opposed to a pathological one). This process is believed to be related to the evolution of the cartilage thickness over time, especially for elderly sedentary patients who spend considerable time sitting in a chair and thus impose a constant load on the joint with no motion.

The functional basis of congruency has also been linked to degenerative disease processes in the hip joint [48, 51]. The results from this thesis, although restricted by the limited number of anatomical test cases, clearly agree with such findings. Results indicate that femoral head cartilage varies between 1 and 2.5 mm with a maximum cartilage thickness at the superior-anterior-medial area which corresponds to a window of maximum acetabulum coverage. In

addition, the algorithm results also indicate, that on average, the cartilage thickness on the femoral head is greater than inside the acetabulum. This confirms *in vivo*, work done with ultrasound investigations of *in vitro* hip joint specimens [68, 117]. While no definitive theory exists to explain this phenomenon, it is widely accepted that the cause lies with the difference in cartilage growth on concave versus convex surfaces at the cellular level [5, 118].

4.6 CONCLUSIONS

Chapter 4 presents an original method and implements appropriate algorithms to study *in vivo* three-dimensional bone geometry of the femoral head and of the acetabulum in the hip joint and thereby estimate the cartilage thicknesses on both the femoral head and inside the acetabulum, in the absence of any CT or MRI cartilage information. Simulation results and test cases using actual anatomical data validate the theory behind the methodology. While the limited number of anatomical test cases warrants caution in the interpretation of the results, it is noted that the estimation algorithms were found very accurate, beyond the resolution of the imaging system itself. In addition, optimization results confirmed and extended to three-dimensions prior two-dimensional studies contributing to increased confidence in the method.

The implementation and use of three-dimensional graphics to visualize both joint sphericity and cartilage thicknesses on the femoral head and acetabulum add new capabilities to locate areas of cartilage damage and to better understand what corrective steps are necessary to compensate for these lesions. Computer graphic techniques using transparency modeling allow users to visualize the joint overlap over the range of the joint kinematics.

In summary, the optimization method developed in this chapter allows users to:

I: Monitor changes of the joint geometry and cartilage thicknesses and study in three-dimensions the evolution over time of articular diseases such as osteoarthritis for a specific patient, without surgical intervention. Such studies may lead to a better understanding of the mechanics and biology of cartilage degenerative diseases, help identify the causes, and ultimately find a cure. To the author's knowledge, studies of three-dimensional joint congruency and cartilage thickness evolution have been restricted to date to *in vitro* examination of disarticulated specimens.

II: Use the cartilage map information to prescribe and plan surgery. The graphic displays developed in this thesis provide new visual aid tools to help clinicians evaluate the potential for, prescribe, and preoperatively plan intertrochanteric osteotomies. It is the author's hope that such displays can address most of the concerns associated with current osteotomy planning procedures.

III: Obtain the data necessary to fully automate computer implementation of an intertrochanteric osteotomy surgery planning system. With the joint geometry modeled and the cartilage thicknesses on the femoral head and inside the acetabulum estimated, it becomes possible to optimize the reorientation of the joint to maximize the overlap of cartilage thicknesses and thus identify an optimal osteotomy wedge. Such an approach is presented in Chapter 5.

CHAPTER
FIVE

OSTEOTOMY SIMULATION

5.1 INTRODUCTION

Modeling hip bone geometries and estimating cartilage thicknesses as presented in Chapter 4 provide the bases for computerized intertrochanteric osteotomy planning and surgery simulation. This chapter describes the approach developed as part of this thesis, implemented as a two step process. The first step is a numerical optimization that seeks the best reorientation of the femoral head inside the acetabulum, while the second step computes the corresponding intertrochanteric osteotomy wedge.

5.2 OSTEOTOMY PLANNING

The goal of osteotomy optimization is to maximize the amount of "good" cartilage at the joint interface. This relies on the definition of what "good" cartilage actually is. For the purpose of this thesis, "good" cartilage was defined as equivalent to thick cartilage since the osteotomy reorientation attempts to move out of joint kinematic overlap, those thinning areas of cartilage where lesions are present or most likely to develop. Using the acetabular and femoral bone data as well as the computed cartilage maps, and taking into account the kinematic range at the joint, the method developed here can find the best reorientation of the

femoral head inside the acetabulum. These correction angles form the basis of the optimal osteotomy. Taking into account joint kinematics is absolutely necessary to insure patients with maximum mobility. For example, a reorientation that only moves a bad area of cartilage away from the femoral - acetabular overlap, based on cartilage maps at the neutral kinematic position, cannot guarantee this damaged area will not overlap during different kinematic activities such as gait, rising from a chair or climbing stairs. The contemporary planning process, based as it is on X-rays and tracing the bones, is fundamentally static and does not accommodate mobility kinematics.

5.2.1. THEORY

First, the acquisition of kinematic data at the hip needs to be addressed. To capture human kinematic information, research at the Newman Laboratory for Biomechanics and Human Rehabiliitation at MIT has led to the development of a fully computerized motion analysis system. This system is used for various research projects in the laboratory and at the Massachusetts General Hospital Biomotion Laboratory where investigations are performed on normal and pathological human movement. Data is acquired by a set of infra-red-sensitive, opto-electronic Selspot cameras which measure the positions of LED markers. TRACK, the Telemetered Real-time Acquisition and Computation of Kinematics software developed at MIT, automatically acquires and processes 3-D movement data [4, 61, 73, 74, 79, 88, 89]. A piezoelectric Kistler platform built in the laboratory floor concurrently records the force magnitude and directions at the floor foot interface.

Overall gait data acquisition with TRACK is based on the following concept: rigid arrays mounting specific geometries of LED's and attached to the patients lower extremity segments allow the positions and orientations of each segment to be determined. Careful attention to the location of each array on the respective body segment and appropriate fixation causes the array to follow the anatomical/skeletal motion. The camera signals these arrays provide, once processed, become 6 degree of freedom kinematic information for each body segment, measured within a translation accuracy of about 1.0 millimeters and a rotation accuracy of 0.6 degrees (10.5 milliradians). Using this kinematic information, clinical angles at the joint can be calculated [49]. Typical results are presented in Figure 5.1 at the hip for a simple normal gait test. As can be seen from this data during a full gait cycle, the hip joint experiences about 10° of extension to 40° of flexion, 10° of adduction to 5° of abduction and 0° to 12° of external rotation. Capturing patient-specific kinematic information is critical to address differences in gait patterns among different subjects. Collecting information on the maximum kinematic range of the joint is also important, especially for the pathological joint

before surgery to identify the angles where pain occurs. The cause of pain can then be visualized and confirmed at the cartilage level using the displays presented in Chapter 4. In addition, such data can help define a normal range of hip joint kinematics that could be used as a substitute in the absence of patient specific data.

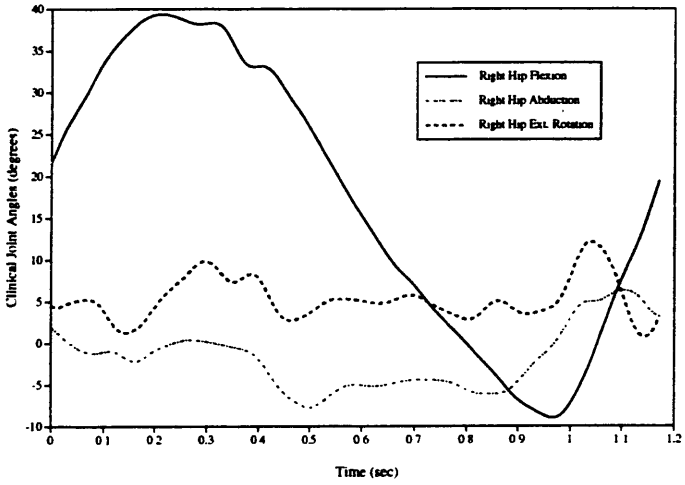


Figure 5.1: Clinical Joint Angles from Sample Hip Kinematic Data Set

The reorientation algorithm implemented for this thesis is listed in Appendix F and uses Powell's optimization method. The algorithm optimizes three reorientation angles, α, β and γ to reposition the femoral head inside the acetabulum and maximize the cartilage thickness inside the joint. These correction angles are taken about the axes of the anatomical coordinate system defined using the approach in Chapter 3, but with the origin translated to the center of the kinematic sliding contact sphere computed during the cartilage thickness estimation. The optimal reorientation is thus defined by the following rotation matrix:

$$\begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \qquad \text{(Eq. 5.1)}$$

At this optimal reorientation, the femoral head has been moved to a new position which maximizes the integral of the cartilage thickness at the overlap:

$$\iint_{Surface} (Cartilage\ Thickness)\partial A \qquad \text{(Eq. 5.2)}$$

The cartilage thickness can be defined as the minimum distance from the acetabular bony surface to the femoral head bony surface. The kinematic information can then be weighted

into the optimization by modifying equation 5.2 and evaluating it over the entire joint kinematic range which results in:

$$\int_{\theta} W(\theta) \times \left(\int_{\varphi} W(\varphi) \times \left(\int_{\phi} W(\phi) \times \left(\iint_{Surface} (Cartilage\ Thickness) \partial A \right) \right) \right) \qquad \text{(Eq. 5.3)}$$

where θ, φ and ϕ represent the clinical joint angles and the W() functions their respective weight in the optimization process. Equation 5.3 can be discretized and presented in the following form:

$$\sum_{k=1}^{p} W(\theta_k) \times \left(\sum_{l=1}^{q} W(\varphi_l) \times \left(\sum_{m=1}^{r} W(\phi_m) \times \left(\sum_{i=1}^{n} [Cartilage\ Thickness_i] \right) \right) \right) \qquad \text{(Eq. 5.4)}$$

where n is the total number of points sampled on the acetabulum, and p, q, and r are the number of discrete evaluation steps chosen for the weighting functions. Kinematic weighting functions as well as their evaluation steps can then be chosen considering the cost of computational time. Figure 5.2 illustrates example weighting functions. These variable functions emphasize the median kinematic point and reduce the effects at the extremes of the kinematic range. This approach allows clinicians to decide which part of the kinematic range should most influence the osteotomy optimization in order to promote the best possible outcome for a specific patient. However, it should be noted that such an approach can rapidly increase computation time. For example, the functions presented in Figure 5.2 have 11 evaluation steps over the clinical joint angle range. If similar weight functions are to be used for the flexion-extension, abduction-adduction, and internal-external rotation clinical angles at once, the optimization times will increase by $11 \times 11 \times 11 = 1331$.

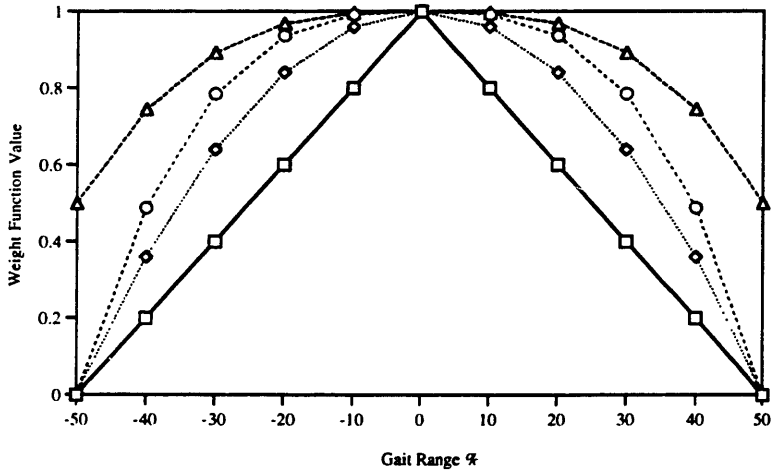


Figure 5.2: Kinematic Weighting Function Examples

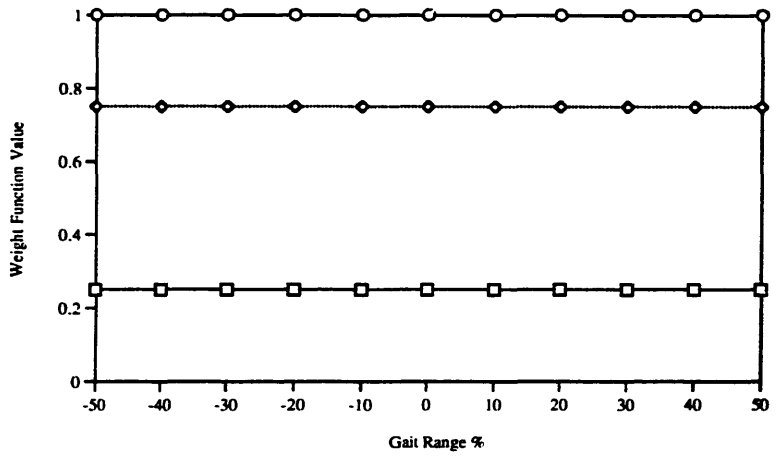


Figure 5.3: Equivalent Neutral Kinematic Weighting Functions

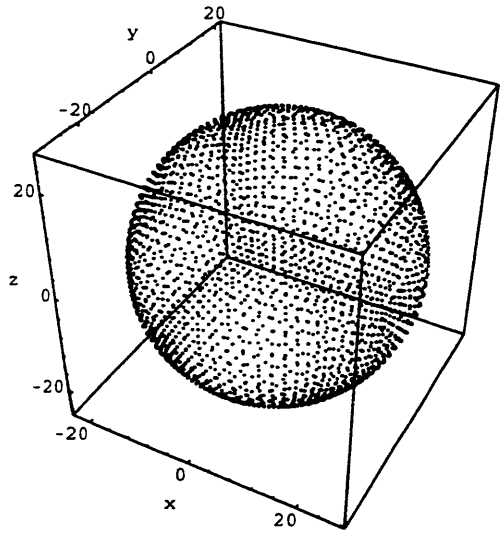
To reduce the computational time, one can either select a smaller number of time steps or use a different approach. Figure 5.3 illustrates alternative weight functions that are constant over the entire kinematic range. Since all the kinematics on both sides of the kinematic mid-point are valued equally, optimizing at the kinematic mid-point will yield the same osteotomy reorientation which would result if all the kinematics had been included. This can decidedly speed up computations. The choice of kinematic mid-points is left to the clinician's judgment depending on the functional range he/she wishes to optimize for a specific patient. Finally, issues of boundary conditions are addressed by implementing limits on the magnitude of the optimized reorientation. Because of the anatomy, it is not possible to perform surgically any reorientation with an intertrochanteric osteotomy. Osteotomy wedges are often small and rarely exhibit correction angles greater than 30° or 40° [96]. Here again, the reorientation limits can be controlled by the user and adapted to specific situations.

The software implemented for this process, while capable of using variable kinematic weights, was never fully tested using this approach. Experiments demonstrated that some optimizations would have required over 100 days of computation time to solve, and thus were not a viable solution within the research setting of this project. It should further be noted that clinical environments would not tolerate even smaller computational times because of associated delays and costs. It is also the author's belief that the constant kinematic weighting technique provides an osteotomy reorientation which can compensate for most of the joint cartilage deficiency. While the variable weighting technique is more *correct*, the additional accuracy it will contribute to the optimized reorientation will result in small correction improvements which will most likely be less than the accuracy that can be achieved in the operating room by the surgeon. Thus, the algorithms were tested against simulated data as well as anatomical data from CT and MRI using the constant kinematic

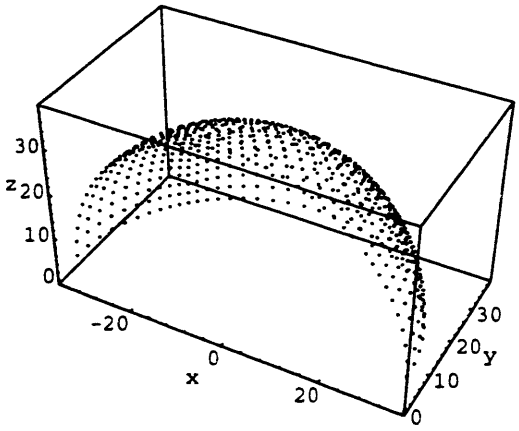
weight approach with kinematic midpoints defined at 25° of flexion, 3° of abduction and 6° of external rotation at the hip and gathered from normal gait information. The results from these simulations and tests are presented below.

5.2.2 SIMULATIONS

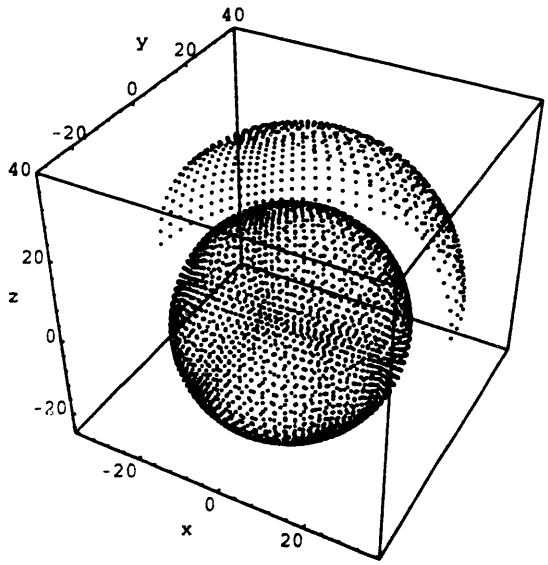
In order to test the algorithm, it was necessary to generate simulated data to represent the acetabular and femoral head bone surfaces. The femoral head was modeled as a sphere while the acetabulum was best represented by a quarter of a sphere as depicted in Figure 5.4.



(b) Femoral Head



(a) Acetabulum



(c) Simulated Joint

Figure 5.4: Simulated Acetabulum and Femoral Head

Data points were uniformly distributed over the *bone* surface of both models with 4000 and 1000 points mapping the femoral head and the acetabulum respectively. For simulation purposes, the acetabulum was given a radius of 35 mm and was centered at {0, 3.5, 3.5}. The femoral head was given a smaller radius of 25 mm to make it easier to qualitatively visualize the improvements in cartilage thickness after reorientation. The cartilage sliding surface was centered at the origin of the coordinate system. Thus, the femoral head sphere position that maximizes the amount of cartilage can be calculated to be located at {0, -3.5, -3.5}. To test the algorithm, known cases were created by back calculating and initializing the position of the femoral head to locations that would require a given correction along both the x axis (pitch) and the y axis (roll). Table 5.1 lists 11 test cases ranging from ±45° in both pitch and roll corrections. The limited number of cases is related to the amount of time required to complete the simulation. Because of the symmetry presented by the models, the third correction axis, z (yaw) could not be tested and was assumed to remain at 0°.

Table 5.1: Test Cases for Algorithm Simulations

Pitch	Roll	Original X	Original Y	Original Z
45.0	-45.0	-2.474	-4.224	0.724
40.0	-40.0	-2.249	-4.404	0.195
30.0	-30.0	-1.750	-4.546	-0.875
20.0	-20.0	-1.197	-4.413	-1.893
10.0	-10.0	-0.607	-4.045	-2.786
0.0	0.0	0.000	-3.500	-3.500
-10.0	10.0	0.607	-2.848	-4.002
-20.0	20.0	1.197	-2.164	-4.287
-30.0	30.0	1.750	-1.515	-4.375
-40.0	40.0	2.249	-0.957	-4.303
-45.0	45.0	2.474	-0.724	-4.224

Execution times took between 1.5 and 3.5 hours depending on the simulation test performed. Results are presented in Figure 5.5 where errors between the optimized correction angles and their nominal values are plotted. The pitch and roll errors were found to be small and within ±1° of their nominal value. The optimization algorithm demonstrated stability and resistance to drifting with the yaw angle computations, where any value would have satisfied the initial conditions: the errors however remained small, again mostly within ±1°. Additional experiments indicated that increasing the number of data points mapping both bone surfaces, improves the accuracy of the optimized calculations. The quarter sphere chosen to model the acetabulum as well as its alignment with the coordinate system make the roll angle computations very sensitive to any error in the pitch angle calculations. For example, using a hemi sphere instead of a quarter sphere to represent the acetabulum reduces the variability.

The crescent shape of the anatomic acetabulum covers more surface than a quarter of a sphere. Thus, it is expected that with actual anatomical data the variation will be smaller than experienced with simulations. Overall, it should be noted that the simulation error magnitudes are small and below the resolution ability of any surgeon to control in the operating room.

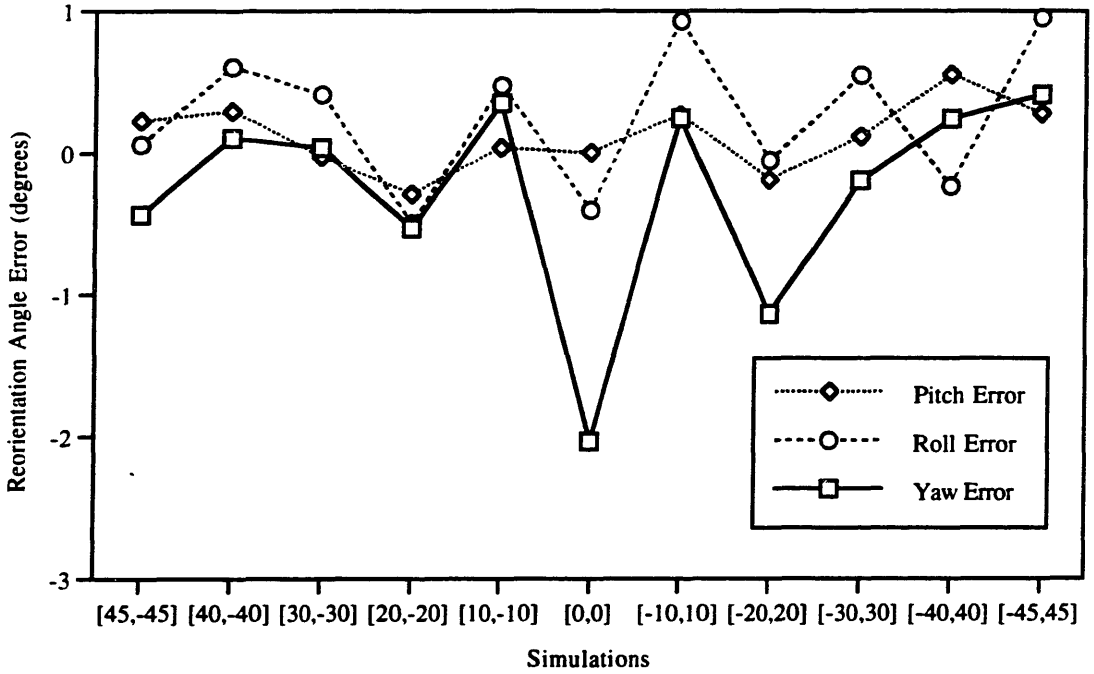


Figure 5.5: Test Case Simulation Results

5.2.3 ANATOMICAL RESULTS

The algorithm was then tested with anatomical data collected with CT and MRI. Optimizations were performed with no, ±20° and ±30° boundary conditions. The increase in cartilage thickness was then calculated to quantify the improvements achieved by the osteotomy reorientation. The osteotomy optimization results are presented in Table 5.2 for the CT case. All angles are shown in order about the x, y, and z axes of the femoral coordinate system. With no boundary conditions, the optimization resulted in reorientations greater than 90° which are not surgically feasible. The resulting increases in cartilage thickness were meaningless since they corresponded to an overlap by the acetabulum of areas of the femoral head that are not load bearing surfaces. With boundary conditions at ±20°, the algorithm converges for this case to a reorientation at -12°, 18° and 6°. The average improvement in cartilage thickness is about 13% which is significant especially with a minimum thickness improvement of 7% and a maximum thickness improvement of 16%. With boundary conditions set at ±30°, the optimization reaches the maximum allowable

reorientations, and leads to hard to believe maximum cartilage thickness improvements of over 100%, indicating that the acetabulum has started riding over non-load bearing areas of the femoral head. However, in this particular case, there is some evidence that a justifiable osteotomy could be performed with resulting cartilage improvements significant enough to recommend this hip joint as a good candidate for an intertrochanteric osteotomy. Considering that the subject is a 61 year old 220 lb. male, the indication that an osteotomy may be appropriate should not be a surprise.

Table 5.2: CT Cadaver Study Reorientation Optimization Results (2.5 hours)

Unbounded
Optimum Reorientation at: 146.0°, -14.6°, -84.0°
Cartilage Improvements: Min -4.155 %, Max 417.093 %, Ave 240.234 %.
Bounded ±20°
Optimum Reorientation at: -12.1°, 17.6°, 5.8°
Cartilage Improvements: Min 7.077 %, Max 16.115 %, Ave 12.912 %.
Bounded ±30°
Optimum Reorientation at: 26.0°, -30.0°, 30.0°
Cartilage Improvements: Min -32.223 %, Max 103.537 %, Ave 15.156 %.

Table 5.3 presents the results from the runs done with MRI data. Again, the unbounded case yields meaningless cartilage thickness improvements and reorientation solutions. For both the ±20° and ±30° bounded cases, the optimized reorientations reach their limits and yield little to no improvement in average cartilage thickness, at about 5%, and even result in a drop of the minimum cartilage thickness of almost 7%. In this case, the results clearly suggest that no significant gain in cartilage thickness can be achieved and that areas of even thinner cartilage have moved under the acetabulum overlap. Thus it can be safely assumed that this particular joint is not well suited to an intertrochanteric osteotomy, confirmed by the fact that the data is from a healthy 25 year old 135 lb. subject who should not yet be experiencing the consequences of any cartilage degenerative disease.

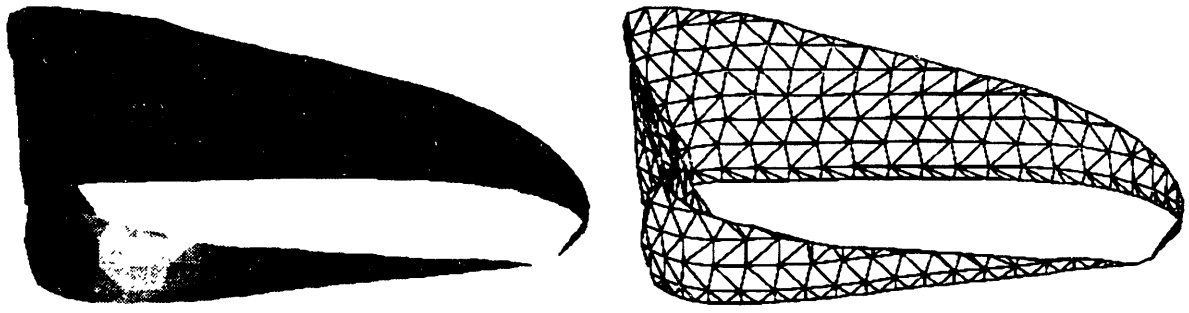
Table 5.3: MRI *In-Vivo* Study Reorientation Optimization Results (45 minutes)

Unbounded
Optimum Reorientation at: 180.0°, 98.9°, -18.7°
Cartilage Improvements: Min 58.830 %, Max 146.285 %, Ave 112.986 %.
Bounded ±20°
Optimum Reorientation at: 20.0°, 20.0°, -20.0°
Cartilage Improvements: Min -0.408 %, Max 0.398 %, Ave 4.451 %.
Bounded ±30°
Optimum Reorientation at: 29.9°, 13.7°, -30.0°
Cartilage Improvements: Min -6.836 %, Max 3.861 %, Ave 5.766 %.

5.3 SURGERY SIMULATION

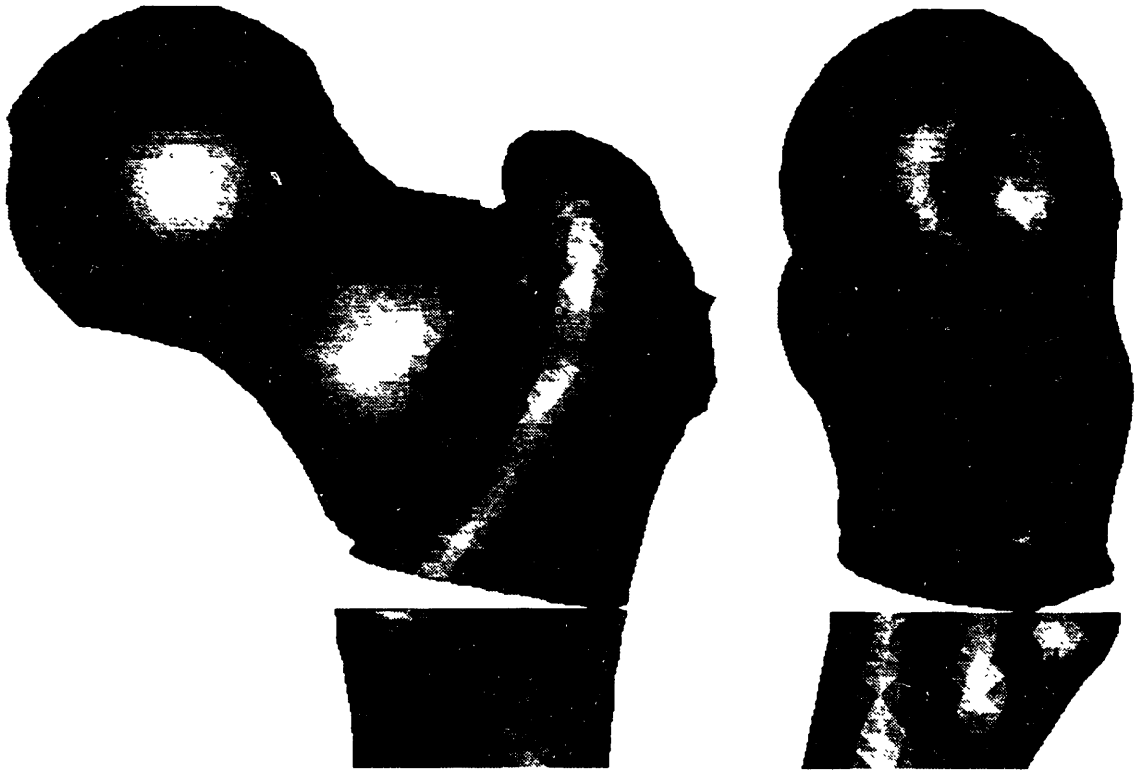
A program was implemented to calculate and display the osteotomic wedge that corresponds to the optimal osteotomy reorientation of the femoral head. It is listed in Appendix F. This program translates the femoral head reorientation to the site of the osteotomy, which is input by the user independently as the location of the transverse cut between the greater and the lesser trochanter. The program then calculates the wedge that minimizes bone stock loss from the skeletal information stored in the database. To this effect, the program identifies the location on the transverse cut that will be least affected by the reorientation. This point is the rotation vertex and corresponds to the point where the contour of the transverse cut intersects with the contour of the oblique cut. The database is then updated with surface patches sliced and sorted to reflect the surgery results. An interactive computer graphics display permits the user to manipulate the bone before and after the surgery while automatic stereolithography output is used to manufacture models describing bones before and after the simulated surgery. These models provide a realistic three-dimensional representation of what the surgeon will see in the operating room. Polymeric models of the osteotomy wedge can be used as templates to help perform the actual surgery. In addition, this program allows users to specify any reorientation other than the optimal one found by the computer and provides instant feedback on the "quality" of their proposed osteotomy by calculating the projected improvements in cartilage thickness. This is a critical aspect of this project. Providing surgeons with the ability to test several different options before committing to the specifics of the surgery, allows them the opportunity to experience the refining process so familiar to engineers that use Computer Aided Design (CAD) systems.

Figure 5.6 illustrates an intertrochanteric wedge example. Note how the surface patches have been sliced and the database reorganized to accommodate not only the planar cut but also the slanted plane that defines the wedge. Figure 5.7 displays two orthogonal views of the

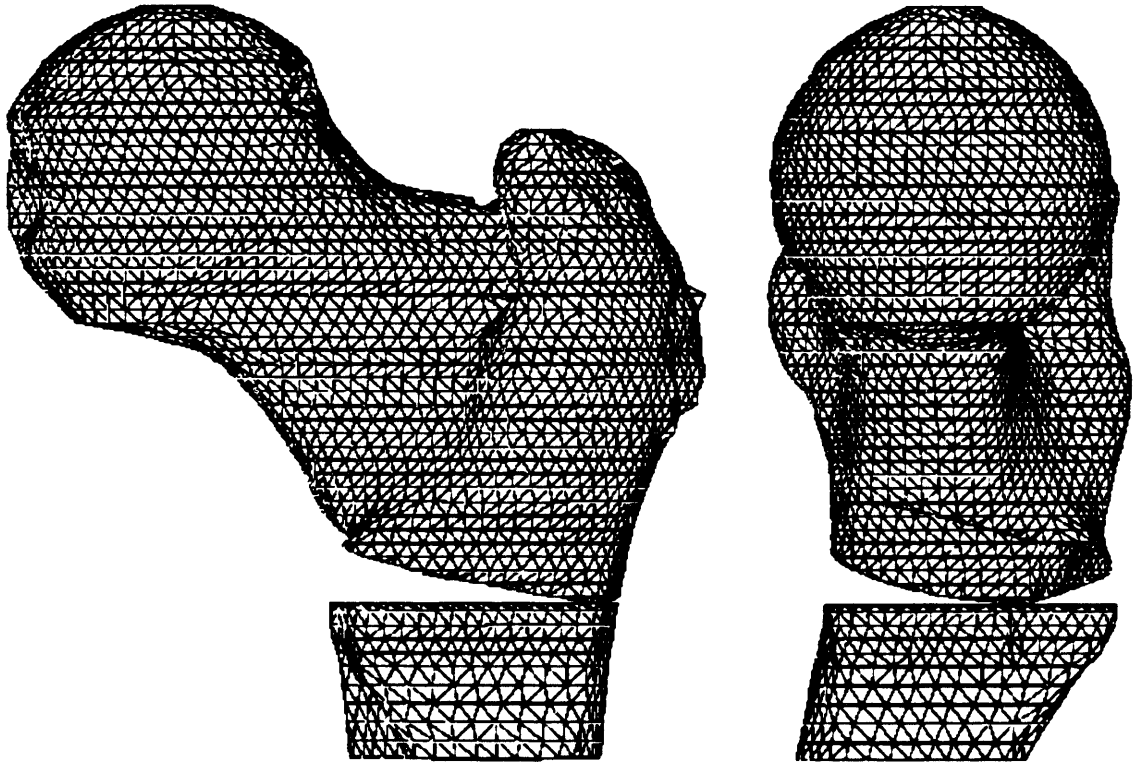


(a) Shaded (b) Hidden-Line

Figure 5.6: Three-Dimensional Display of Osteotomic Wedge

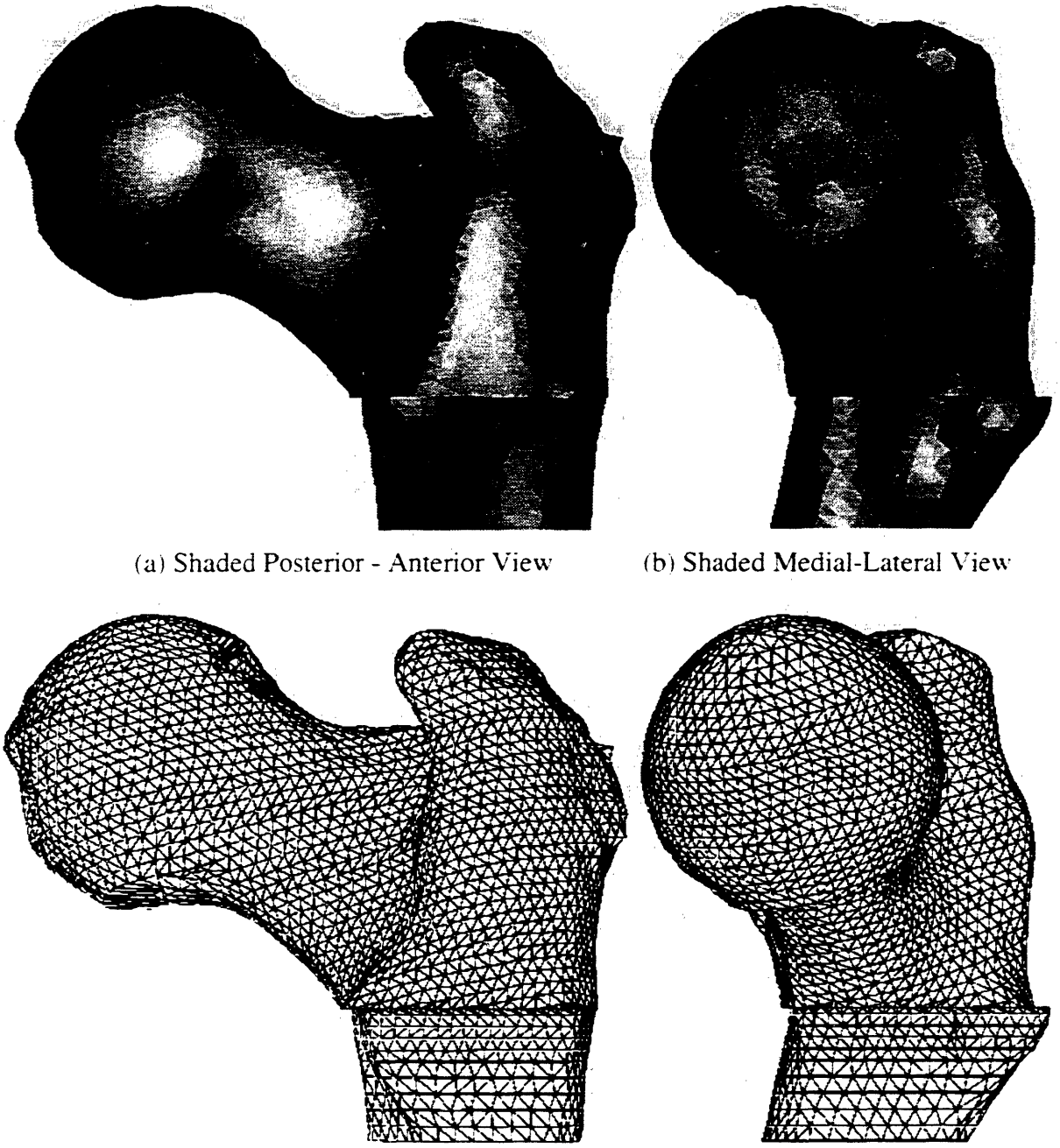


(a) Shaded Posterior - Anterior View (b) Shaded Medial-Lateral View



(c) Hidden-Line Posterior - Anterior View (d) Hidden Line Medial-Lateral View

Figure 5.7: Simulated Pre-Operative Femur With Osteotomic Wedge Removed



(a) Shaded Posterior - Anterior View (b) Shaded Medial-Lateral View

(c) Hidden-Line Posterior - Anterior View Line Medial-Lateral View

Figure 5.8: Simulated Post-Operative Femur Reorientation

proximal end of the femur with the osteotomy wedge removed. Finally, the two sections of bone are put back together on top of one another and displayed in Figure 5.8 to simulate the post-operative femoral reorientation. At their discretion, clinicians can control the relative lateral position of the proximal and distal segments of the femur. This post-operative femur bone can then be used in kinematic displays such as the one presented in Figure 4.14 to verify if the areas of damaged cartilage have been moved away from the acetabular coverage.

5.4 CONCLUSIONS

An approach was described in this chapter to fully automate the planning of intertrochanteric osteotomies. Using the joint modeling and the cartilage thickness estimation techniques developed in previous chapters, an algorithm was implemented to compute the optimal femoral head reorientation in order to maximize "good cartilage in contact with good cartilage" within osteotomy boundary conditions. The algorithm is based on Powell's method and evaluates all three correction angles in varus-valgus, flexion-extension, and internal-external rotations, unlike traditional two-dimensional planning techniques which are limited to only one or two correction angles. Simulation results and anatomical testing have validated the theory and demonstrated that the method can offer calculation accuracies superior to most surgeons' surgical accuracy.

The reorientation of the femoral head can then be transformed into the required osteotomy wedge, with computer displays helping visualize the simulated surgery results. The software automatically evaluates the projected quality of any number of reorientations beyond the "computer optimal solution", through interaction with the clinician and incorporating and testing his/her expertise. Such a system also enhances teaching the principles of intertrochanteric osteotomy planning to orthopaedic surgery residents and increases the confidence of more experienced surgeons, so that they will be less reluctant to prescribe intertrochanteric osteotomies over the traditional total hip replacement.

CHAPTER
SIX

CONCLUSIONS

6.1 REVIEW

This thesis has covered many topics and it is worth reviewing how these topics relate to the thesis objectives and conclusions. The main objective of this thesis was to implement a fully computerized system to help plan intertrochanteric osteotomies and simulate the surgery. In this endeavor, one of the goals was to improve upon the traditional intertrochanteric osteotomy planning techniques for the treatment of osteoarthritis, in order to revive osteotomies as an attractive alternative to the current widespread use of total hip replacements.

First, this thesis argues that the only viable approach is to consider patient specific information and avoid computerized methods that attempt to scale generic anatomical reconstructions to fit a particular patient. Anatomical variability and related complex non-linear scaling properties do not lend themselves to such simple modeling. Chapter 2 proposes a method to solve the issues associated with the acquisition, reconstruction and display of patient specific anatomical data. Then, this thesis focuses on the issue of assigning to anatomical information a consistent coordinate system which is independent of the imaging acquisition techniques, to thereby eliminate bias and patient-scanner misalignment. Chapter 3 describes a technique which uses patient specific three-dimensional geometry of a

structure to compute its principal axes and assign a consistent set of coordinates. The next step is to analyze the joint bone geometry and congruency, and calculate the thickness of the cartilage within the intervening space. Such information permits an evaluation of the joint condition and potential candidacy for an intertrochanteric osteotomy. Chapter 4 recommends optimization techniques to map out the geometry of the bone structures and estimate the cartilage condition. Finally, using this geometric and cartilage map at the joint, combined with kinematic knowledge, Chapter 5 unveils a technique to optimize thickness in cartilage overlap and calculate the necessary intertrochanteric osteotomy wedge.

Together, all the chapters in this thesis combine like puzzle pieces to create all the tools necessary to analyze, plan and simulate intertrochanteric osteotomies. The result is the implementation of the first computer evaluation and quantification system for intertrochanteric osteotomies, based purely on quantitative patient specific joint geometry, kinematics and cartilage maps.

6.2 CONCLUSIONS

Attaining the overall objective set out for this thesis should not overshadow the significant contributions achieved as well as the conclusions reached in each chapter. The following sections attempt to summarize the importance and significance of each chapter.

6.2.1 ANATOMICAL MODELING

Several issues critical to the acquisition of three-dimensional patient specific anatomy were addressed in Chapter 2 with the development of a novel anatomical modeling technique. With this approach, a user can, with limited intervention, analyze MRI and CT data to extract the surface of relevant anatomical features and reconstruct them in three-dimensions. These reconstructions can then be assembled into anatomical databases that can be used to display structures in real-time on a computer screen or to directly manufacture polymeric models of the skeletal information using stereolithography prototyping. The flexibility and functionality of the implemented system make it suitable not only to the extraction of bone and soft tissue information but also to any application requiring the segmentation and contouring of details from a set of two-dimensional images.

It was also demonstrated in Chapter 2 that the protocol developed for this project to acquire patient anatomy provides sufficient resolution and accuracy not only for interactive three-dimensional computer displays but also for further analyses of bone geometry and joint

congruency. This is an absolute requirement for any implementation of computerized surgery simulation systems and the approach presented in this thesis provides an excellent solution optimizing both accuracy and efficiency using current computer technology.

Advances in memory and processing technology will improve computer performance and lower hardware costs to the stage where interactive editing of "true" three-dimensional models becomes possible. This breakthrough would enable surgeons to rehearse entire surgical procedures on a computer model, effectively cutting through the tissue layers with an electronic scalpel in a manner similar to actual surgery. While the implementation of this vision is today impractical because it will require very significant advances in software and hardware tools to deal with the overwhelming size of anatomical information, a first step is taken in this thesis. This initial implementation of computer surgery has already greatly benefited from advances in real-time, three-dimensional medical imaging technologies, falling hardware costs, increases in computing power, improved system functionality, and innovative anatomical segmentation and surface mesh reconstruction methods.

6.2.2 PRINCIPAL AXIS THEORY

A method to assign a coordinate system to a particular three-dimensional CT or MRI object based on principal axis theory is presented in Chapter 3 and offers multiple benefits. First, data sets collected with CT and MRI systems on a subject without specifying or recording position and orientation during the scanning process can be overlaid and thus complement each other to form a more complete anatomical database. Similarly, misalignments inevitable between multiple data sets collected on the same subject at different times can be corrected to perform comparative geometric analyses. In addition, the standardized calculation of coordinate axes offered by this method provides users with a reference frame to investigate and compare anatomical information from different subjects: for example, the impact of certain congenital as well as degenerative conditions can be studied on a large number of subjects without introducing bias from the imaging process.

Because of the nature of CT and MRI data, different inputs can be used as the source for computing the orientation of objects being studied. It was shown with simulations, as well as calculations based on actual anatomical data, that the most accurate calculations are achieved with the use of full cross-sectional information (i.e.: all the data about an object on a slice image). With this approach, computational time and memory requirements can rapidly become constraints. However, these limitations can be offset for a small trade-off in

accuracy by using surface outlines of the object or even an adequate number of sampled points on the surface area of the object.

The orientations of certain geometries cannot be studied using the principal axes and therefore limit the functionality of this method. Fortunately, anatomical structures rarely present the geometric singularities that would dismiss such an approach. Thus, patient specific anatomical databases and computer-aided surgery simulations can be referenced to a coordinate system which is easily defined at the femur by the principal axis method. Simulations have demonstrated that orientation errors, though inevitable, can be minimized to a magnitude less than that of the inaccuracies introduced by manual surgical techniques, and thus will not affect most clinical applications. While conceptually simple, the application of the principal axis theory to the computation of anatomical structure orientations affords levels of automation and three-dimensional consideration not found in other approaches.

6.2.3 CARTILAGE THICKNESS ESTIMATION

A major contribution of this thesis is presented in Chapter 4 and is the implementation of a unique method to study *in vivo* the three-dimensional geometry of the joint as well as the cartilage coverage inside the articular space. Even in the absence of any CT or MRI cartilage surface information, the common situation using current technology, this method permits the estimation of cartilage thickness on both the femoral head and inside the acetabulum based on the geometry of the subchondral bone surfaces in the hip joint. This approach was validated with simulation results as well as test cases on actual anatomical data collected with CT and MRI. Estimation algorithms were found to be very accurate and confirmed results from previous *in vitro* studies in the literature, which increases confidence in their validity. Finally, to help visualize not only the joint sphericity but also femoral head and acetabulum cartilage quality, three-dimensional computer graphic displays were implemented. With the use of transparency algorithms, visual aids can accommodate the three-dimensional study of the femoral head - acetabulum joint kinematic overlap and further aid the planning of the necessary corrective procedures.

Beyond visually helping clinicians plan and prescribe intertrochanteric osteotomies by eliminating the two-dimensional limitations of traditional planning methods, this technique provides the necessary quantitative data for a fully computerized intertrochanteric osteotomy surgery planning system. Moreover, joint geometry and cartilage thickness changes over time can now be investigated *in vivo* using the algorithm developed for this thesis and may

lead to a better understanding of the etiology of articular diseases such as osteoarthritis. Finally, this joint geometry evaluation and cartilage thickness estimation method can be used for other applications. For example, hip joint kinematics are often studied with motion analysis systems using surface mounted markers to capture pelvic and femoral segment positions. The inevitable kinematic errors induced by soft tissue motion could be compensated for with joint geometric and kinematic constraints, thus avoiding the use of invasive bone pin markers to obtain true hip joint kinematics.

6.2.4 OSTEOTOMY PLANNING

Using the femoral and cartilage maps calculated previously, an algorithm was implemented to fully automate the planning of intertrochanteric osteotomies. Based on a numerical optimization, the algorithm attempts to maximize the overlap of good acetabulum cartilage in contact with good femoral cartilage and tries to displace away from the overlap the bad or thin areas of femoral cartilage. As opposed to traditional two-dimensional planning techniques the algorithm optimizes for all three correction angles along the varus-valgus, flexion-extension, and internal-external rotation. In addition, kinematic information at the joint is weighted into the optimization to insure the best possible reorientation for a specific patient over the range of expected future mobility. To address the issues of physical limitations surgically restricting the range of possible correction angles, the optimization algorithm can be assigned boundaries along the different correction axes. This method was tested with simulated and actual anatomical data and exhibited the ability to provide correction with accuracies in excess of the actual execution ability which can be expected from most orthopaedic surgeons.

The osteotomy wedge can then be inferred from the reorientation corrections at the femoral head. Computer graphic displays were implemented to help visualize in real-time the simulated surgery results. Evaluations of the quality of the simulated osteotomy are performed based on the cartilage thickness improvements achieved as a result of the surgery. Multiple presurgical trials can be interactively tested and evaluated in addition to the computer optimized osteotomy solution. Thus, clinicians can perform many iterations with patient specific information to familiarize themselves with the situation before the actual surgery. This system provides a safe environment for experimentation and is thus an ideal teaching aid. This method can also help surgeons decide whether or not an intertrochanteric osteotomy is appropriate for a particular patient: in some cases, the joint damage will have progressed so far that the only solution is a total hip replacement. However, it is hoped that

the progress in planning technique achieved by this method will promote the use of intertrochanteric osteotomies as viable alternatives to THRs.

6.3 FUTURE WORK

The main focus of this thesis was to develop and implement the methods and techniques for a computer aided intertrochanteric osteotomy planning and surgery simulation system. In this endeavor, several hypotheses for the modeling of the hip joint and the estimation of the cartilage thickness were presented. In order to validate them, extensive simulations as well as actual anatomical tests were performed. The results, while extremely encouraging, should however be cautiously interpreted due to the limited number of *in vivo* experiments conducted. While not the primary concern of this thesis, a clinical evaluation is an absolute prerequisite prior to the acceptance of the approach within the medical community. Clinical testing should be the next step as it can not only affirm the credibility of this work, but also challenge some of the assumptions made during the development of this computer-aided surgery system, thus resulting in modeling refinements as well as increased confidence in the software.

CT and MR images are today qualitatively sufficient to help visually diagnose gross medical problems and were shown in this thesis to be quantitatively adequate, in particular for CT, to model the hip joint. Surgery simulation systems can only be as accurate as the anatomical information acquisition and reconstruction. To be successfully simulated, some surgery situations will require higher accuracies than those needed for this project. Therefore, improvements in medical imaging technologies would greatly benefit surgery simulation systems.

If this computer-aided surgery system is to be used within a clinical setting, the three-dimensional segmentation and reconstruction of patient specific anatomical information needs further automation. The solution proposed in this thesis requires a certain level of user intervention which may not be acceptable in the clinical setting. Initially minimizing the amount of user intervention during the reconstruction process will certainly help, however complete automation is the key to its success. This is still a difficult problem that has not yet been fully answered. In addition, in a typical medical environment the computation time requirements will have to be reduced due to their prohibitive cost. This can be achieved by optimizing the algorithms for speed. This work in no way claims to have implemented the fastest algorithms to solve the optimization issues, and thus performance improvements should be achievable. Moreover, computational power never ceases to increase with the

introduction of new hardware. This should partially alleviate the computational burden experienced throughout the development and testing of the algorithms in this thesis.

Surgery simulation systems will also have to face the limitations associated with a surgeon's ability to perform the recommended procedure. No matter how accurate any surgery simulation will be, it will be limited by the surgical precision that can actually be performed in the operating room. Some projects have already started addressing this issue with the introduction of computer controlled robotics devices in the operating room [60, 64, 94, 114, 115]. Their use so far has been very limited.

It is my hope that the insight gained from this thesis will help promote the use of intertrochanteric osteotomies and reverse the trend that has made total hip replacements the method of choice in most, if not all, cases of osteoarthritis. Ultimately, this will improve the quality of life of patients suffering from osteoarthritis by providing them with a desirable solution especially if they are young and active. This thesis addresses patient specific anatomy modeling as well as quantitative aspects of surgery simulations in an attempt to emphasize the importance of these issues. It is my aim that future developments of computer surgery simulation systems incorporate these crucial perspectives.

BIBLIOGRAPHY

[1] Acton, F. S., *Numerical Methods That Work.* 1970, New York, N.Y.: Harper and Row.

[2] Adams, L., *et al.*, *Computer Assisted Surgery.* IEEE Computational Graphics Applications, 1990. **10**(3): pp. 43.

[3] Adams, M. E. and Li, D. K. B., *Magnetic Resonance Imaging of Joint Lesions*, Workshop Conference Hoechst-Werk Albert: Articular Cartilage Biochemistry, 1985, (Edited by Kuettner, K. E., Schleyerbach, R., and Hascall, V. C.), Wiesbaden, Germany, pp. 331-345.

[4] Antonsson, E. K., *A Three-Dimensional Kinematic Data Acquisition and Intersegmental Dynamic Analysis System for Human Motion*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Ph.D. Thesis, 1982.

[5] Armstrong, C. G. and Gardner, D. L., *Thickness and Distribution of Human Femoral Head Articular Cartilage.* Annals of the Rheumatic Diseases, 1977. **36**: pp. 407.

[6] Artzy, E., Frieder, G., and Herman, G. T., *The Theory, Design, Implementation and Evaluation of a Three Dimensional Surface Detection Algorithm.* Computational Graphics and Image Processing, 1981. **15**(1): pp. 1.

[7] Bailey, T. E. and Hall, J. E., *Chiari Medial Displacement Osteotomy.* Journal Pediatr. Orthop., 1985. **5**(1): pp. 635-641.

[8] Baker, D. G., Schumacher Jr., H. R., and Wolf, G. L., *Nuclear Magnetic Resonance Evaluation of Synovial Fluid and Articular Tissues.* The Journal of Rheumatology, 1985. **12**(6): pp. 1062-1065.

[9] Bloch, F., *Nuclear Induction.* Physics Review, 1946. **70**(7 and 8): pp. 460-474.

[10] Blount, W. P., *Proximal Osteotomies of the Femur.* Instructional Course Lectures, ed. Pease, C. N. and Edwards, J. W. 1952, Ann Arbor, Michigan: The American Academy of Orthopaedic Surgeons.

[11] Blowers, D. H., Elson, R., and Korley, E., *An Investigation of the Sphericity of the Femoral Head.* Medical and Biological Engineering, 1972. **10**(1): pp. 762.

[12] Bombelli, R., *Osteoarthritis of the Hip.* 1976, New York: Springer-Verlag.

[13] Bottomley, P. A., *NMR in Medecine.* Computerized Radiology, 1984. **8**(2): pp. 57-77.

[14] Bottomley, P. A. and Edelstein, W. A., *Power Deposition in Whole-Body NMR Imaging.* Medical Physics, 1981. **8**(1): pp. 510-512.

[15] Brand, R. A. and Pedersen, D. R., *Computer Modeling of Surgery and a Consideration of the Mechanical Effects of Proximal Femoral Osteotomies.* The Hip Society, 1984. **Otto E. Aufranc Award.**

[16] Brent, R. P., *Algorithms for Minimization without Derivatives.* Vol. Chapter 7. 1973, Englewood Cliffs, N.J: Prentice-Hall.

[17] Brown, G. A., *Determination of Body Segment Parameters using Computerized Tomography and Magnetic Resonance Imaging,* Dept. of Mechanical Engineering, Massachusetts Institute of Technology, Master's Thesis, 1987.

[18] Brown, G. A., *Load-Bearing Role of The Human Knee Meniscus,* Health Sciences and Technology Division, Massachusetts Institute of Technology, Ph.D. Thesis, 1990.

[19] Budinger, T. F., *Thresholds for Physiological Effects Due to RF and Magnetic Fields Used in NMR Imaging.* IEEE Transaction on Nuclear Science, 1979. **NS-26**(1): pp. 2821-2825.

[20] Budinger, T. F., *Nuclear Magnetic Resonance (NMR) In Vivo Studies: Known Threshold for Health Effects.* Journal of Computer Assisted Tomography, 1981. **5**(6): pp. 800-811.

[21] Bullough, P. G., Goodfellow, J., Greenwald, A. S., and O'Connor, J. J., *Incongruent Surfaces in the Human Hip Joint.* Nature, 1968. **217**: pp. 1280.

[22] Byers, P. D., *The Effect of High Femoral Osteotomy on Osteoarthritis of the Hip: an Anatomical Study of Six Hip Joints.* Journal of Bone and Joint Surgery, 1974. **56B**(279).

[23] Cathcart, R. F., *The Shape of the Femoral Head and Results from Clinical Use of More Normally Shaped Non-Spherical Hip-Replacement.* Journal of Bone and Joint Surgery, 1972. **43**(A): pp. 1559.

[24] Chang, G. H.-P., *Computer-Aided Surgery: an Interactive Simulation System for Intertrochanteric Osteotomy*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Master's Thesis, 1987.

[25] Christiansen, H. N. and Sederberg, T. W., *Conversion of Complex Contour Line Definitions into Polygonal Element Mosaics.* Computer Graphics, 1978. **13**(2): pp. 187-192.

[26] Cormack, A. M., *Representation of a Function by its Line Integrals, with Some Radiological Applications.* The Journal of Applied Physics, 1963. **34**(1): pp. 2722-2727.

[27] Damadian, R., *Tumor Detection by Nuclear Magnetic Resonance.* Science, 1971. **171**(3976): pp. 1151-1153.

[28] Delp, S. L., *Surgery Simulation: A Computer Graphics System to Analyze and Design Musculoskeletal Reconstruction of the Lower Limb*, Stanford University, Ph.D. Thesis, 1990.

[29] Delp, S. L., Bleck, E. E., Zajac, F. E., and Bollinin, G., *Biomechanical Analysis of the Chiari Pelvic Osteotomy.* Clinical Orthopaedics, 1990. **254**: pp. 189-198.

[30] Delp, S. L., *et al.*, *An Iteractive Graphics-Based Model of the Lower Extremity to Study Orthopaedic Surgical Procedures.* IEEE Transactions on Biomedical Engineering, 1990. **37**(8): pp. 757-767.

[31] Deutsch, E. S. and Fram, J. R., *A Quantitative Study of the Orientation Bias of Some Edge Detector Schemes.* IEEE Transactions on Computations, 1978. **C-27**(3): pp. 205-213.

[32] Dixon, R. L. and Ekstrand, K. E., *The Physics of Proton NMR.* Medical Physics, 1982. **9**(6): pp. 807-818.

[33] Dwyer III, S. J., *et al.*, *Medical Processing in Diagnnostic Radiology.* Institute of Electronics and Electrical Engineers Transactions on Nuclear Science, 1980. **27**(1): pp. 1047.

[34] Eck, M., Hoscheck, J., and Weber, U., *Three-Dimensional Determination of an Oblique Osteotomy in the Hip by Mathematical Optimization Fulfilling Some Anatomical Demands.* J. Biomechanics, 1990. **23**(10): pp. 1061-1067.

[35] Fellingham, L. L., Vogel, J., Lau, C., and Dev, P., *Interactive Graphics and 3D Modeling for Surgical Planning and Prosthesis and Implant Design*, NCGA Conference and Exposition, 1986, Anaheim, California.

[36] Ferguson Jr., A. B., *High Intertrochanteric Osteotomy for Osteo-Arthritis of the Hip: a Procedure to Streamline the Defective Joint.* Journal of Bone and Joint Surgery, 1964. **46A**(1159).

[37] Ferguson Jr., A. B., *The Pathological Changes in Degenerative Arthritis of the Hip and Treatment by Rotational Osteotomy.* Journal of Bone and Joint Surgery, 1964. **46A**(1337).

[38] Fisherman, E. K. and et al., *Volumetric Rendering Techniques: Applications for Three-Dimensional Imaging of the Hip.* Radiology, 1987. **163**(1): pp. 737-738.

[39] Frain, P. and Merle d'Aubigné, R., *Action Mécanique de l'Antéversion Fémorale sur la Hanche: Degré de Validité de la Théorie de Pauwels.* Revue de Chirurgie Orthopédique, 1981. **67**(1): pp. 1-9.

[40] Fuchs, H., Kedem, Z. M., and Uselton, S. P., *Optimal Surface Reconstruction from Planar Contours.* Communications of the ACM, 1977. **20**(10): pp. 693-702.

[41] Galland, W. I., *The Bifurcation Operation: Indications, Technique, and Results.* Surgery, Gynecology and Obstetrics, 1930. **50**(90).

[42] Galloway Jr., R. L., Maciunas, R. J., and Edwards II, C. A., *Interactive Image-Guided Neurosurgery.* IEEE Transactions on Biomedical Engineering, 1992. **39**(12): pp. 1226-1231.

[43] Ganapathy, S. and Dennehy, T. G., *A New General Triangulation Method for Planar Contours.* ACM: Computer Graphics, 1982. **16**(3): pp. 69-75.

[44] Giordano, T. A., *Pegasus Takes Flight: Engineering 3-D Medical Imaging Systems.* Soma, 1987. : pp. 44-50.

[45] Goodfellow, J. W. and Bullough, P. G., *Studies on Age Changes in the Human Hip Joint.* Journal of Bone and Joint Surgery, 1968. **50**(B): pp. 222.

[46] Goodfellow, J. W. and Mitsou, A., *Joint Surface Incongruity and its Maintenance.* Journal of Bone and Joint Surgery, 1977. **59**(B): pp. 4.

[47] Granholm, J. W., Robertson, D. D., Walker, P. S., and Nelson, P. C., *Computer Design of Custom Femoral Stem Prostheses.* IEEE CG&A, 1987. **272**(1716): pp. 26-35.

[48] Greenwald, A. S. and O'Connor, J. J., *The Transmission of Load through the Human Hip Joint.* Journal of Biomechanics, 1971. **4**: pp. 507.

[49] Grood, E. S. and Suntay, W. J., *A Joint Coordinate System for the Clinical Description of Three-Dimensional Motions: Applications to the Knee.* Journal of Biomechanical Engineering, 1983. **105**(Transaction of the ASME).

[50] Hall, F. M. and Wyshak, G., *Thickness of Articular Cartilage in The Normal Knee.* The Journal of Bone and Joint Surgery, 1980. **62-A**(3): pp. 408-413.

[51] Hammond, B. T. and Charnley, J., *The Sphericity of the Femoral Head.* Medical and Biological Engineering, 1967. **5**(1): pp. 445.

[52] Harris, W. H. and Sledge, C. B., *Total Hip and Total Knee Replacement.* The New England Journal of Medicine, 1990. **323**(11 & 12): pp. 725-731 & 801-807.

[53] Hemmy, D. C., David, D. J., and Herman, G. T., *Three-Dimensional Reconstruction of Craniofacial Deformity Using Computed Tomography.* Neurosurgery, 1983. **13**(5): pp. 534-541.

[54] Herman, G. T. and Liu, H. K., *Three-Dimensional Display of Human Organs from Computed Tomograms.* Computer Graphic Image, 1979. **9**(1): pp. 1-21.

[55] Herman, G. T. and Udupa, J. K., *Display of 3-D Digital Images: Computational Foundations and Medical Applications.* IEEE: Computer Graphics and Applications, 1983. **1**(1): pp. 39-45.

[56] Hey Groves, W. E., *Surgical Treatment of Osteoarthritis of the Hip.* British Medical Journal, 1933. **1**(3).

[57] Holmes, W. S., *Design of a Photointerpretation Automaton*, Proceedings of the Fall Joint Computer Conference, 1962, , pp. 27-35.

[58] Jacobs, D. A. H., *The State of the Art in Numerical Analysis.* 1977, London, England: Academic Press.

[59] Johnston, R. C., Brand, R. A., and Crowninshield, R. D., *Reconstruction of the Hip - a Mathematical Approach to Determine Optimum Geometric Relationships.* Journal of Bone and Joint Surgery, 1979. **61A**(639).

[60] Joskowicz, L. and Taylor, R. H., *Hip Implant Insertability Analysis: a Medical Instance of the Peg-In-Hole Problem*, Proceedings of the IEEE International Conference on Robotics and Automation, 1993, Atlanta, Georgia, pp. 901-908.

[61] Karlsson, J. O. M., *Using Axodes to Compare Biokinematic Data Measured with Bone- and Skin-Mounted Markers*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Master's Thesis, 1990.

[62] Keppel, E., *Approximating Complex Surfaces by Triangulation of Contour Lines.* IBM Journal of Research Development, 1975. **19**: pp. 2-11.

[63] Kernighan, B. W. and Ritchie, D. M., *The C Programming Language.* 1988, Englewood Cliffs, New Jersey: Prentice Hall.

[64] Kienzle, T. C., Stulberg, S. D., Peshkin, M., Quaid, A., and Wu, C.-h., *An Integrated CAD-Robotics System for Total Knee Replacement Surgery*, Proceedings of the IEEE International Conference on Robotics and Automation, 1993, Atlanta, Georgia, pp. 889-900.

[65] Klaue, K., Wallin, A., and Ganz, R., *CT Evaluation of Coverage and Congruency of the Hip Prior to Osteotomy.* Clin. Orthop. Rel. Res., 1988. **232**: pp. 15-25.

[66] Knodt, H., *Osteo-Arthritis of the Hip Joint: Etiology and Treatment by Osteotomy.* Journal of Bone and Joint Surgery, 1964. **46A**(1326).

[67] König, H., Sauter, R., Deimling, M., and Vogt, M., *Cartilage Disorders: Comparison of Spin-Echo, CHESS, and FLASH Sequence MR Images.* Musculoskeletal Radiology, 1987. **164**(3): pp. 753-758.

[68] Kurrat, H. and Oberlander, W., *The Thickness of the Cartilage in the Hip Joint.* Journal of Anatomy, 1978. **126**: pp. 145.

[69] Lauterbur, P. C., *Image Formation by Induced Local Interactions: Examples Employing Nuclear Magnetic Resonance.* Nature, 1973. **242**(5394): pp. 190-191.

[70] Ledley, R. S., *Introduction to Computerized Tomography.* Computers in Biology and Medecine, 1976. **6**(1): pp. 239-246.

[71] Lévesque, S., *Automatic Three-Dimensional Mesh Generation of Skeletal Structures*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Master's Thesis, 1989.

[72] Lloyd-Roberts, C. G., *The Role of Capsular Changes in Osteoarthritis of the Hip Joint.* Journal of Bone and Joint Surgery, 1953. **35B**(627).

[73] Lord, P. J., *3D_Gait: A Three Dimensional Computer Graphic Display for Human Motion Analysis from TRACK Gait Data*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Bachelor Thesis, 1987.

[74] Lord, P. J., *Real-Time Analysis and Display of Human Movement*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Master's Thesis, 1989.

[75] Lowendorf, C. S., *The Bifurcation Operation: A Study of Late Results.* Journal of Bone and Joint Surgery, 1933. **15**(463).

[76] MacKenzie, J. F., *Osteo-Arthritis of Hip and Knee: Description of a Surgical Treatment.* British Medical Journal, 1936. **1**(306).

[77] Mann, R. W., *Biennial Report Department of Mechanical Engineering.* 1983-84 and 1984-85, Massachusetts Institute of Technology:

[78] Mann, R. W., *Computer Aided Surgery*, Rehabilitation Engineering Society of North America (RESNA), 1985, Memphis, Tennessee, Vol. 8th Annual Conference, pp. 160-162.

[79] Mansfield, P. K., *A Large Volume Close-Range Photogrammetric System*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Ph.D. Thesis, 1990.

[80] Manuceau, J., *Modèle à Trois Dimensions de l'Articulation de la Hanche.* Revue de Chirurgie Orthopédique, 1991. **77**(1): pp. 293-300.

[81] Maquet, P. and Radin, E. L., *Osteotomy as an Alternative to Total Hip Replacement in Young Adults.* Clinical Orthopaedics, 1977. **123**(138).

[82] Marsh, J. L. and Vannier, M. W., *The 'Third' Dimension in Craniofacial Surgery.* Plastic and Reconstructive Surgery, 1983. **76**(6): pp. 759-767.

[83] McMurray, T. P., *Osteo-Arthritis of the Hip Joint.* British Journal of Surgery, 1935. **22**(716).

[84] McMurray, T. P., *Osteo-Arthritis of the Hip Joint.* Journal of Bone and Joint Surgery, 1939. **21**(1).

[85] Millis, M. B. and Murphy, S. B., *Use of Computer Tomographic Reconstruction in Planning Osteotomies of the Hip*, 19th Open Scientific Meeting of the Hip Society, 1991, Anaheim, California, pp. 154-159.

[86] Mitchell, D. G., *et al., Avascular Necrosis of the Femoral Head: Morphologic Assessment by MR Imaging imaging with CT Correlation.* Radiology, 1986. **161**(1): pp. 739.

[87] Mitchell, D. G., *et al., Femoral Head Avascular Necrosis: Correlation with MR Imaging, Radiographic Staging, Radionuclide Imaging, and Clinical Findings.* Radiology, 1987. **162**(1): pp. 709.

[88] Murphy, M. C., *A Comparison of Skeletal Kinematic Measured in Vivo Using Different Methods for Attaching Markers*, East Coast Clinical Gait Laboratory, 1987, Bethesda, Maryland.

[89] Murphy, M. C., *Geometry and the Kinematics of the Normal Human Knee*, Dept. of Mech. Eng., Massachusetts Institute of Technology, Ph.D. Thesis, 1990.

[90] Murphy, S. B., *et al., The Planning of Orthopaedic Reconstructive Surgery Using Computer-Aided Simulation and Design.* Computational Medical Imaging and Graphics, 1988. **12**(1): pp. 33-45.

[91] Murphy, S. B., Kijewski, P. K., and Simon, S. R., *Normal Acetabular Anteversion.* 33rd Annual Meeting, Othopaedic Research Society, 1987. : p. 206.

[92] Murphy, S. B., *et al., Computer-Aided Simulation, Analysis, and Design in Orthopaedic Surgery.* Orthopedic Clinics of North America, 1986. **17**(4): pp. 637-649.

[93] Osborne, G. V. and Fahrni, W. H., *Oblique Displacement Osteotomy for Osteoarthritis of the Hip Joint.* Journal of Bone and Joint Surgery, 1950. **32B**(148).

[94] Paul, H. and et al., *A Surgical Robot for Total Hip Replacement Surgery*, Proceedings of IEEE International Conference on Robotics and Automation, 1992, Nice, France.

[95] Pauwels, F., *Biomechanics of the Normal and Diseased Hip.* 1976, New York: Springer-Verlag.

[96] Pauwels, F., *Biomechanical Principles of Varus/Valgus Intertrochanteric Osteotomy (Pauwels I and II) in the Treatment of Osteoarthritis of the Hip,* in *The Intertrochanteric Osteotomy,* Scatzker, J., Editor. 1984, Springer-Verlag: Berlin - Heildelberg. pp. 3-24.

[97] Pedotti, A., Krishnan, V. V., and Stark, L., *Optimization of Muscle-Force Sequencing in Human Locomotion.* Mathematical Bioscience, 1978. **38**(57).

[98] Pieper, S. D., *CAPS: Computer-Aided Plastic Surgery*, Media Arts and Sciences Section, School of Architecture and Planning, Massachusetts Institute of Technology, Ph.D. Thesis, 1991.

[99] Poss, R., *Valgus-Extensions Osteotomy for Primary Protrusio Acetabuli and Coxa Vara.* Strategies in Orthopaedic Surgery, 1986. **5**(2).

[100] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes in C: The art of Scientific Computing.* 1989, New York: Cambridge University Press.

[101] Purcell, E. M., Torrey, H. C., and Pound, R. V., *Resonance Absorption by Nuclear Magnetic Moments in a Solid.* Physics Review, 1946. **69**(1): pp. 37-38.

[102] Pykett, I. L., *et al.*, *Principles of Nuclear Magnetic Resonance Imaging.* Radiology, 1982. **143**(1): pp. 157-168.

[103] Rab, G. T., *Mapping of Surface Area 'Containment' of the Femoral Head During Walking.* Journal of Biomechanics, 1986. **19**(2): pp. 159-163.

[104] Rhodes, M. L., Kuo, Y.-M., and Rothman, S. L. G., *An Application of Computer Graphics and Networks to Anatomic Model and Prosthesis Manufacturing.* IEEE CG&A, 1987. **272**(1716): pp. 12-25.

[105] Rugg, S. G., Gregor, R. J., Mandelbaum, B. R., and Chiu, L., *In Vivo Moment Arm Calculations at the Ankle Using Magnetic Resonance Imaging (MRI).* Journal of Biomechanics, 1990. **23**(5): pp. 495-501.

[106] Rushfelt, P. D., *Human Hip Joint Geometry and the Resulting Pressure Distribution*, Mechanical Engineering, Massachusetts Institute of Technology, Sc.D. Thesis Thesis, 1978.

[107] Sangeorzan, B. P., Judd, R. P., and Sangeorzan, B. J., *Mathematical Analysis of Single-Cut Osteotomy for Complex Long Bone Deformity.* Journal of Biomechanics, 1989. **22**(11/12): pp. 1271-1278.

[108] Santore, R. F. and Bombelli, R., *Long-Term Follow-Up of the Bombelli Experience with Osteotomy for Osteoarthritis: Results at 11 Years in The Hip*, 1983, St. Louis.

[109] Simon, J. C. and Haralick, R. M., *Digital Image Processing*, Proceedings of the NATO Advanced Study Institute, 1980, Bonas, France, June 23 - July 4, pp. 105-148.

[110] Simon, W. H., Friedenberg, S., and Richardson, S., *Joint Congruence. A Correlation of Joint Congruence and Thickness of Articular Cartilage in Dogs.* Journal of Bone and Joint Surgery, 1973. **55**(A): pp. 1614.

[125] Wojtys, E., Wilson, M., Buckwalter, K., Braunstein, E., and Martel, W., *Magnetic Resonance Imaging of Knee Hyaline Cartilage - Intra-Articular Pathology*, 33rd Annual Meeting, Orthopaedic Research Society, 1987, San Francisco, California.

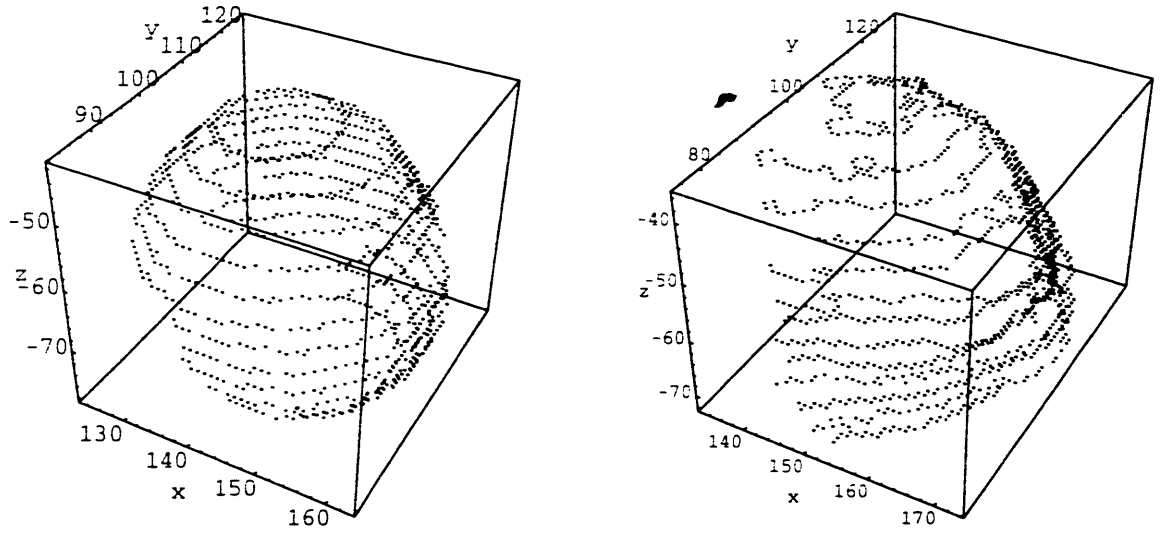
[126] Wolff, S., Crooks, L. E., Brown, P., Howard, R., and Painter, R. B., *Tests for DNA and Chromosomal Damage Induced by Nuclear Magnetic Resonance Imaging.* Radiology, 1980. **136**(1): pp. 707-710.

[127] Woolson, S. T., Dev, P., Fellingham, L. L., and Vassiliadis, A., *Three-Dimensional Imaging of Bone from Computerized Tomography.* Clinical Orthopaedics, 1986. **202**(a): pp. 239-248.

[128] Yoshioka, Y. and Cooke, T. D. V., *Femoral Anteversion: Assessment Based on Function Axes.* Journal of Orthopaedic Research, 1987. **5**(1): pp. 86-91.

[129] Young, S. W., White, D. N., Kaplan, E. N., Dev, P., and Wood, S. L., *A Contour Finding Algorithm for Automated Tumor and Organ Volume Calculation and Three-Dimensional CRT, CT scan display.*, Proceedings of the Association of University Radiologists, 1983, Mobile, Alabama.

[130] Young, S. W., *et al.*, *An NMR and CT Contour Finding Algorithm for Automated Volume Calculation, Three-Dimensional SRT Scan Display, and Prosthesis Milling*, Proceedings of the Radiological Society of North America, 1983, Chicago, Illinois.



(a) Femoral Head (b) Acetabulum

Figure 4.15: Selected MRI Femoral and Acetabular Load Bearing Surface Data

The results from the different optimizations are presented in Table 4.3. In this case the improvement in the fit from the sphere to the rotated ellipsoid is much more noticeable than for the CT data. The magnitude of the minimum cost function significantly decreases and thus indicates that a rotated ellipsoid is a much better fit than a sphere for this data. While the center coordinates of the sphere, ellipsoid and rotated ellipsoid remained the same, the differences between the minor and the major axes of the rotated ellipsoid fits are as large as 1.170 mm for the femoral head and 3.415 mm for the acetabulum. The centers of the best fit geometries for both the femoral head and acetabulum were found to be 2.0 mm apart, indicating reduced joint congruency at the bone level when compared with the CT test case.

However, the algorithm found a good fit to the set of data points, a sphere forming the sliding contact surface midway between the femoral head and the acetabulum optimal-fit-rotated-ellipsoid-geometries. The minimum cost function of 0.032433 can be translated into a mean squared error that is less than 350 μm. This result confirms our original assumption that the sliding contact surface must be a sphere to satisfy kinematic integrity at the joint even if, at the bone level, the femoral head and the acetabulum are not spherical and congruent. Optimization computation times were found to vary greatly with data sets (femoral or acetabulum) but also with the number of data points in each set. Execution times ranged from as little as 45 minutes for spherical fits to as long as 3.5 hours for rotated ellipsoid fits. The increase in processing time over the CT test case was caused by a slower convergence of the optimization algorithm due to the lower resolution and accuracy as well as the higher noise level inherent to MRI imaging.

[111] Smith, D. K., An, K. N., Robb, R. A., Berquist, T. H., and Chao, E. Y., *Accuracy of CT and MRI Based 3-D Reconstructions of Knee Anatomy*, 34th Annual Meeting of the Orthopaedic Research Society, 1988, Atlanta, Georgia, February 1-4.

[112] Sumner, D. R., Mockbee, B., Morse, K., Cram, T., and Pitt, M., *Computed Tomography and Automated Image Analysis of Prehistoric Femora.* American Journal of Physiological Anthropology, 1985. **68**(1): pp. 225-232.

[113] Sumner, D. R., Olson, C. L., Freeman, P. M., Lobick, J. J., and Andriacchi, T. P., *Computed Tomographic Measurement of Cortical Bone Geometry.* Journal of Biomechanics, 1989. **22**(6/7): pp. 649-653.

[114] Taylor, R. H., *et al.*, *Robotic Total Hip Replacement Surgery in Dogs.* IEEE Engineering in Medecine & Biology Society: Medical Applications of Robotics, 1989. **1**(1): pp. 1-3.

[115] Taylor, R. H. e. a., *An Image-based Robotic System for Hip Replacement Surgery.* Journal of the Robotics Society of Japan, 1990. **1**: pp. 111-116.

[116] Tello, R. J., *Design and Implementation of an Intelligent Biomedical Image Processing Environment*, Department of Mechanical Engineering, Massachusetts Institute of Technology, Master's Thesis, 1983.

[117] Tepic, S., *Congruency of the Human Hip Joint*, Mechanical Engineering, Massachusetts Institute of Technolgy, Master's Thesis, 1980.

[118] Tepic, S., *Dynamics of and Entropy Production in the Cartilage layers of the Synovial Joints*, Mechanical Engineering Department, Massachusetts Institute of Technology, Sc.D. Thesis, 1982.

[119] Udupa, J. K., *Display of 3-D Information in Discrete 3-D scenes Produced by Computerized Tomography.* Proceedings of the IEEE, 1983. **71**(3): pp. 420-431.

[120] Vannier, M. W., Marsh, J. L., and Warren, J. O., *Three Dimensional Computer Graphics for Craniofacial Surgical Planning and Evaluation.* Computer Graphics, 1983. **17**(3).

[121] Wallin, A. and Klaue, K., *Three-Dimensional in vivo Modelling and Evaluation of Hip Coverage*, Biostereometrics '88, 5th International Meeting, 1988, (Edited by Baumann, J. U. and Herron, R. E.), Basel, Switzerland, pp. 163-170.

[122] Wallin, A., Klaue, K., Ganz, R., and Perren, S. M., *Three-Dimensional Evaluation of Pathological Joints and Simulation of Corrective Surgical Procedures*, National Computer Graphics Association (NCGA) '88, 9th Annual Conference and Exposition, 1988, Anaheim, California, Vol. III, pp. 198-207.

[123] Wardle, E. N., *Displacement Osteotomy of the Upper End of the Femur.* Journal of Bone and Joint Surgery, 1955. **37B**(568).

[124] Wiberg, G., *Studies on Dysplastic Acetabula and Congenital Subluxation of the Hip joint with Special Reference to the Complication of Osteoarthritis.* Acta Chir. Scand., 1939. **83 (Suppl.)**(58).

APPENDIX
A

ANATOMICAL DATA

A.1 CT CADAVER DATA

CT: Massachusetts General Hospital
Department of Radiology
32 Fruit Street
Boston, MA 02114

Sex: Male
Weight: 220 lbs
Age: 61
Date: 04/09/91

Total Images: 104
Raw Size: 538624 bytes (512*512*2 bytes + Header Size)
Header Size: 14336bytes
Striped Size: 524288 bytes (512*512*2 bytes)

For positioning: A = anterior, P = posterior, R = right, L = left, I = inferior, S = superior and all numbers in mm.

Saggital slices of pelvis (scouts): 3 images total, file001 through file003
Info from X-ray films / File header information not decoded

Matrix	Dia	Thick	Skip	Pro	Contrast
512x512	51.2 cm	200.0 mm	0.0 mm	400	None

Positioning: centered I/S @ 1010 mm.

file001:	R0	file002:	R0	file003:	A525

Axial slices of ilium: 24 images total, file004 through file028
Info from X-ray films / File header information not decoded

Matrix	Dia	Thick	Skip	Pro	Contrast
512x512	240 mm	5.0 mm	5.0 mm	221	None

Positioning: centered A/P & R/L @ 200 mm

file004:	I60	file013:	I105	file022:	I150
file005:	I65	file014:	I110	file023:	I155
file006:	I70	file015:	I115	file024:	I160
file007:	I75	file016:	I120	file025:	I165
file008:	I80	file017:	I125	file026:	I170
file009:	I85	file018:	I130	file027:	I175
file010:	I90	file019:	I135	file028:	I180
file011:	I95	file020:	I140		
file012:	I100	file021:	I145		

Axial slices of hip: 54 images total, file029 through file082
Info from X-ray films / File header information not decoded

Matrix	Dia	Thick	Skip	Pro	Contrast
512x512	240 mm	2.0 mm	2.0 mm	202	None

Positioning: centered A/P & R/L @ 200 mm

file029:	I182	file047:	I218	file065:	I254
file030:	I184	file048:	I220	file066:	I256
file031:	I186	file049:	I222	file067:	I258
file032:	I188	file050:	I224	file068:	I260
file033:	I190	file051:	I226	file069:	I262
file034:	I192	file052:	I228	file070:	I264
file035:	I194	file053:	I230	file071:	I266
file036:	I196	file054:	I232	file072:	I268
file037:	I198	file055:	I234	file073:	I270
file038:	I200	file056:	I236	file074:	I272
file039:	I202	file057:	I238	file075:	I274
file040:	I204	file058:	I240	file076:	I276
file041:	I206	file059:	I242	file077:	I278
file042:	I208	file060:	I244	file078:	I280
file043:	I210	file061:	I246	file079:	I282
file044:	I212	file062:	I248	file080:	I284
file045:	I214	file063:	I250	file081:	I286
file046:	I216	file064:	I252	file082:	I288

Axial slices of proximal femur: 22 images total, file083 through file104
No X-ray films / File header information not decoded

Matrix	Dia	Thick	Skip	Pro	Contrast
512x512	240 mm	10.0 mm	10.0 mm	250	None

Positioning: centered A/P & R/L @ 200 mm

file083:	I290	file091:	I370	file099:	I450
file084:	I300	file092:	I380	file100:	I460
file085:	I310	file093:	I390	file101:	I470
file086:	I320	file094:	I400	file102:	I480
file087:	I330	file095:	I410	file103:	I490
file088:	I340	file096:	I420	file104:	I500
file089:	I350	file097:	I430		
file090:	I360	file098:	I440		

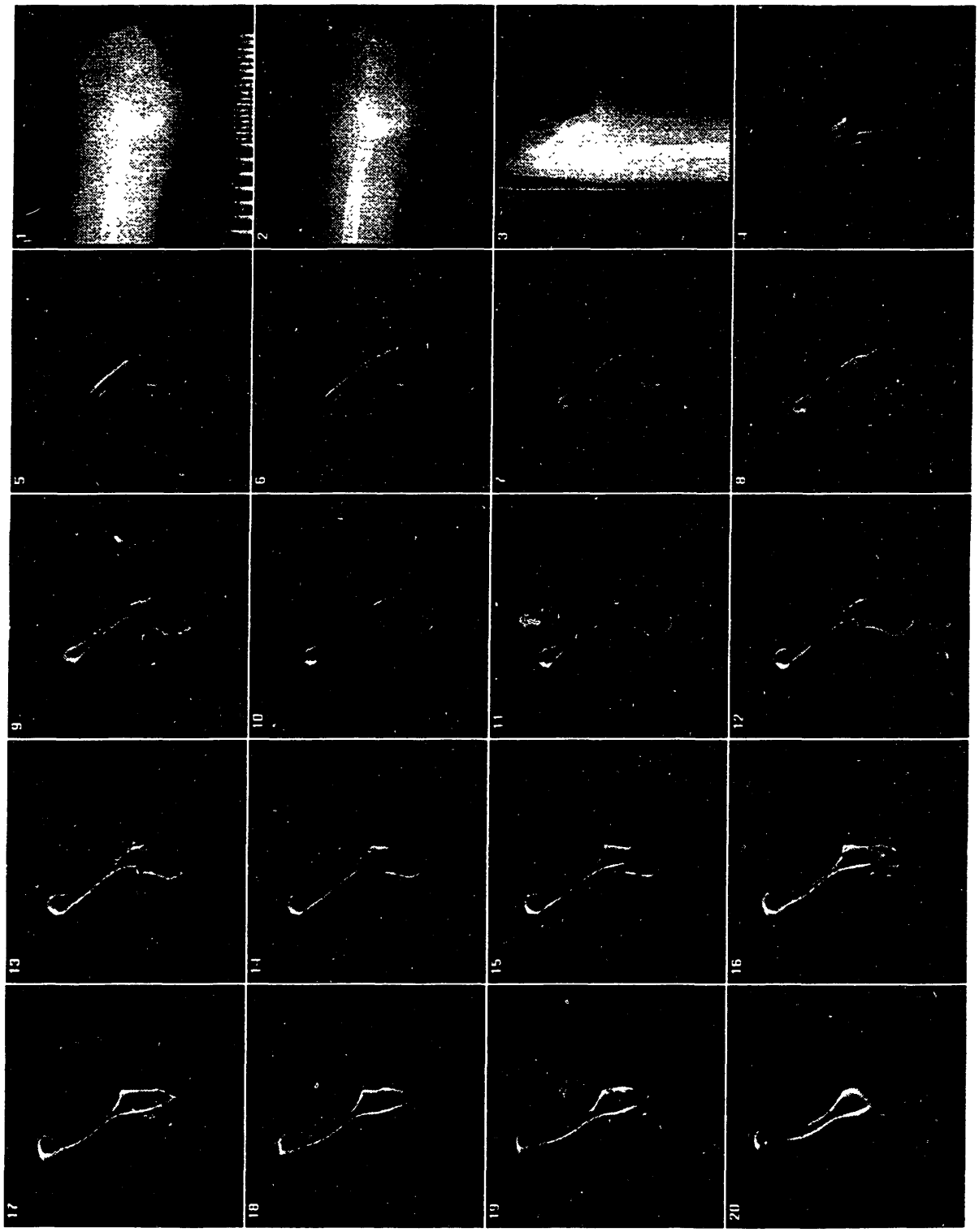


Figure A1: CT Data: Scans 1 through 20

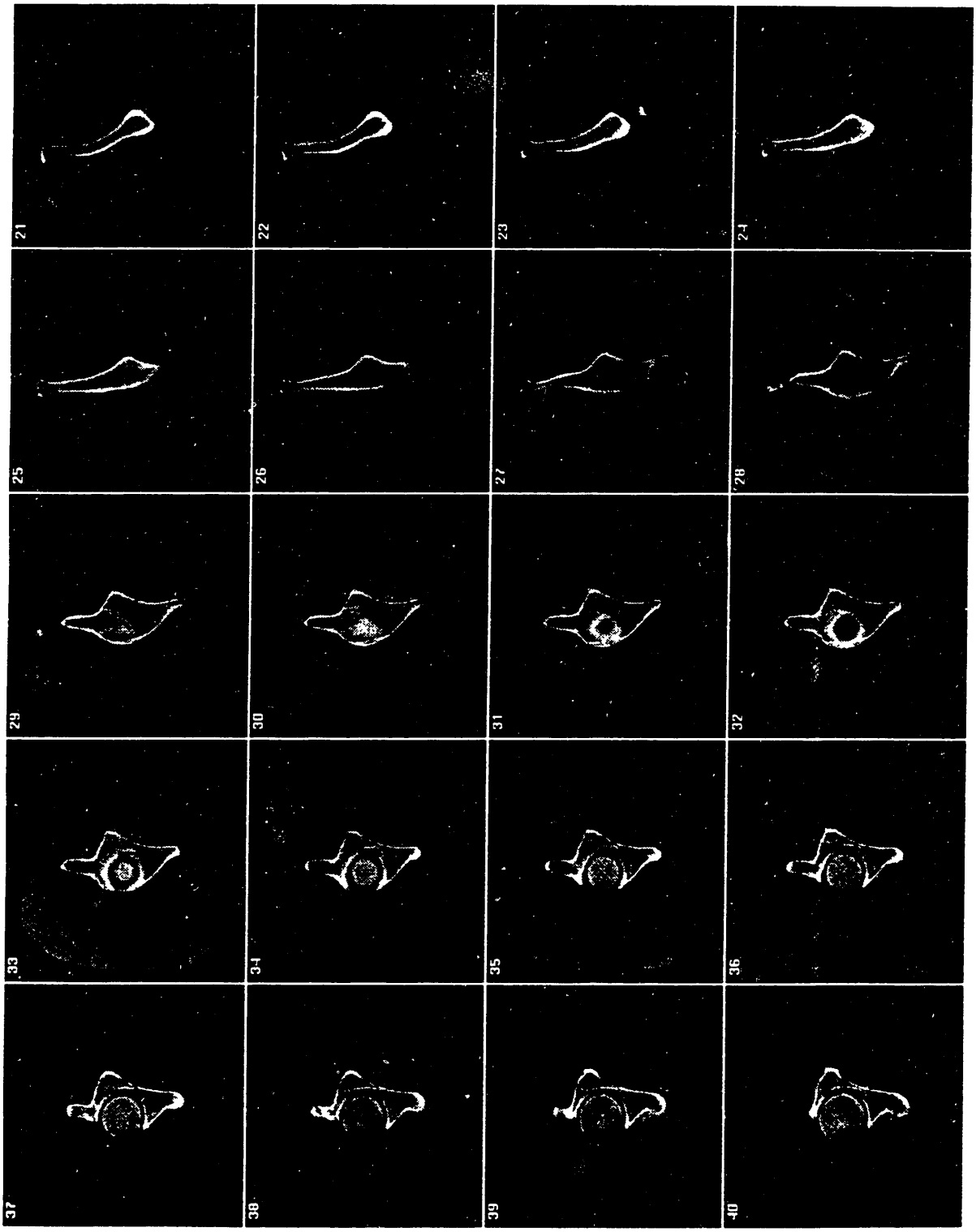


Figure A2. CT Data Scans 21 through 40

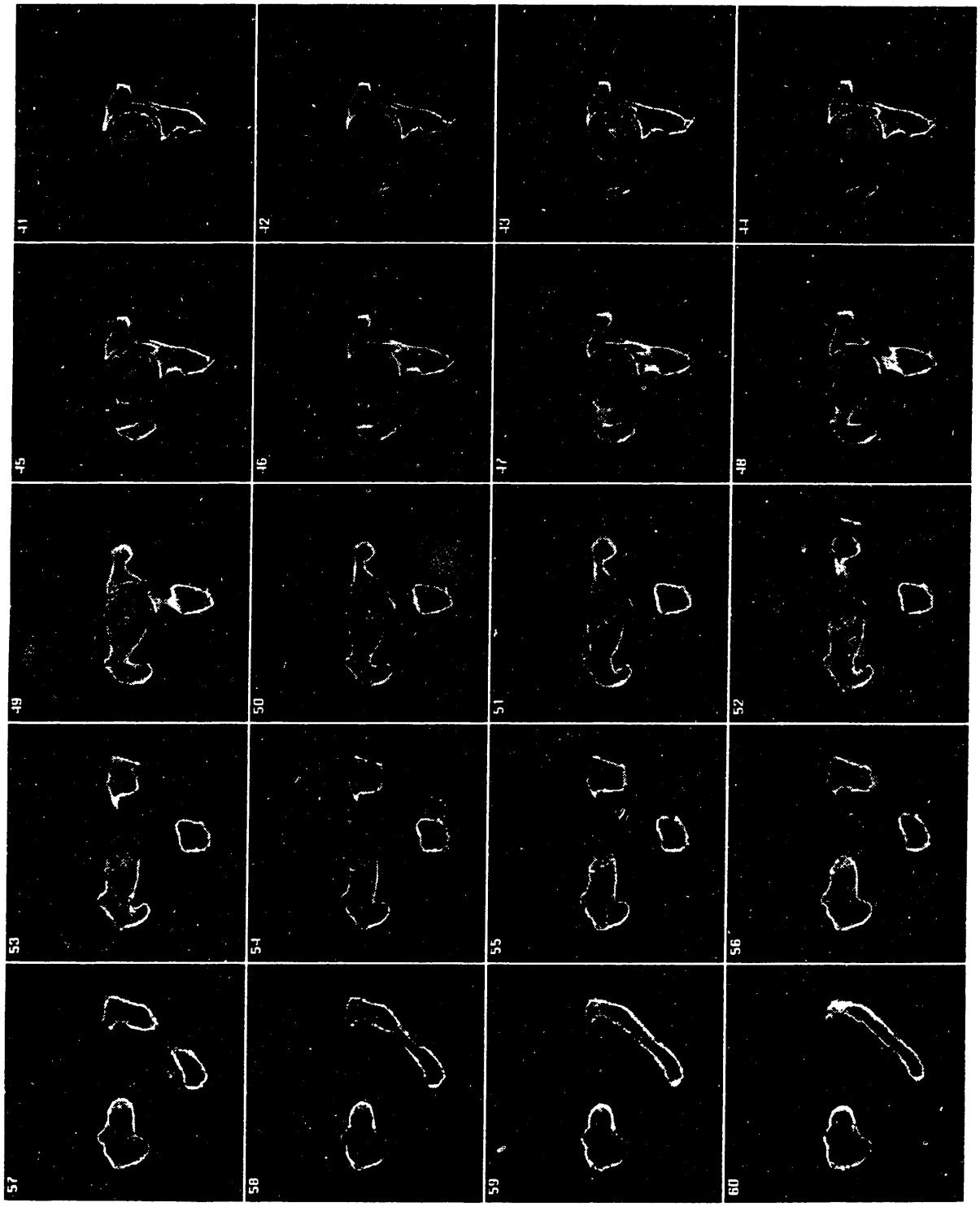


Figure A3: CT Data: Scans 41 through 60

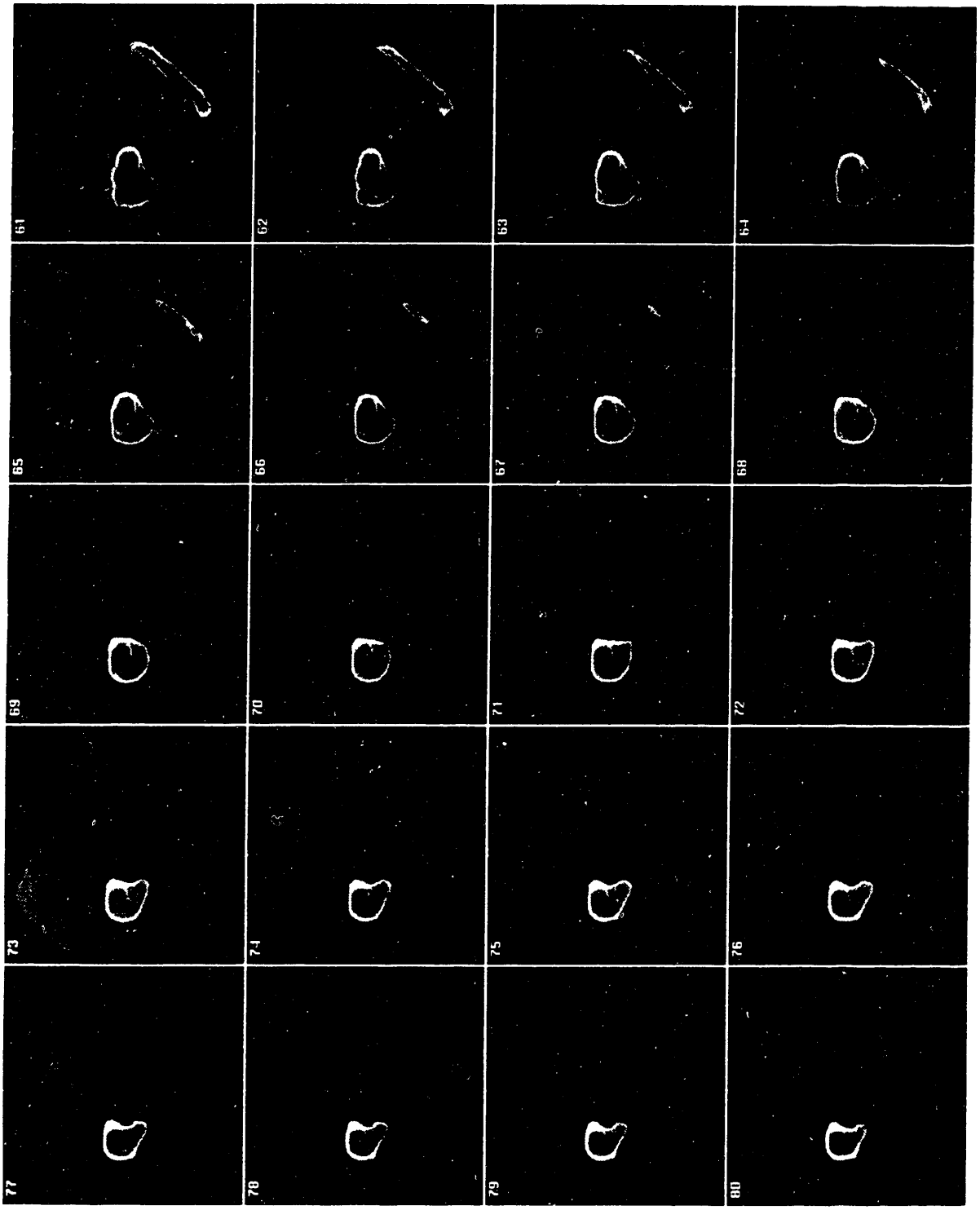


Figure A4. CT Data: Scans 61 through 80

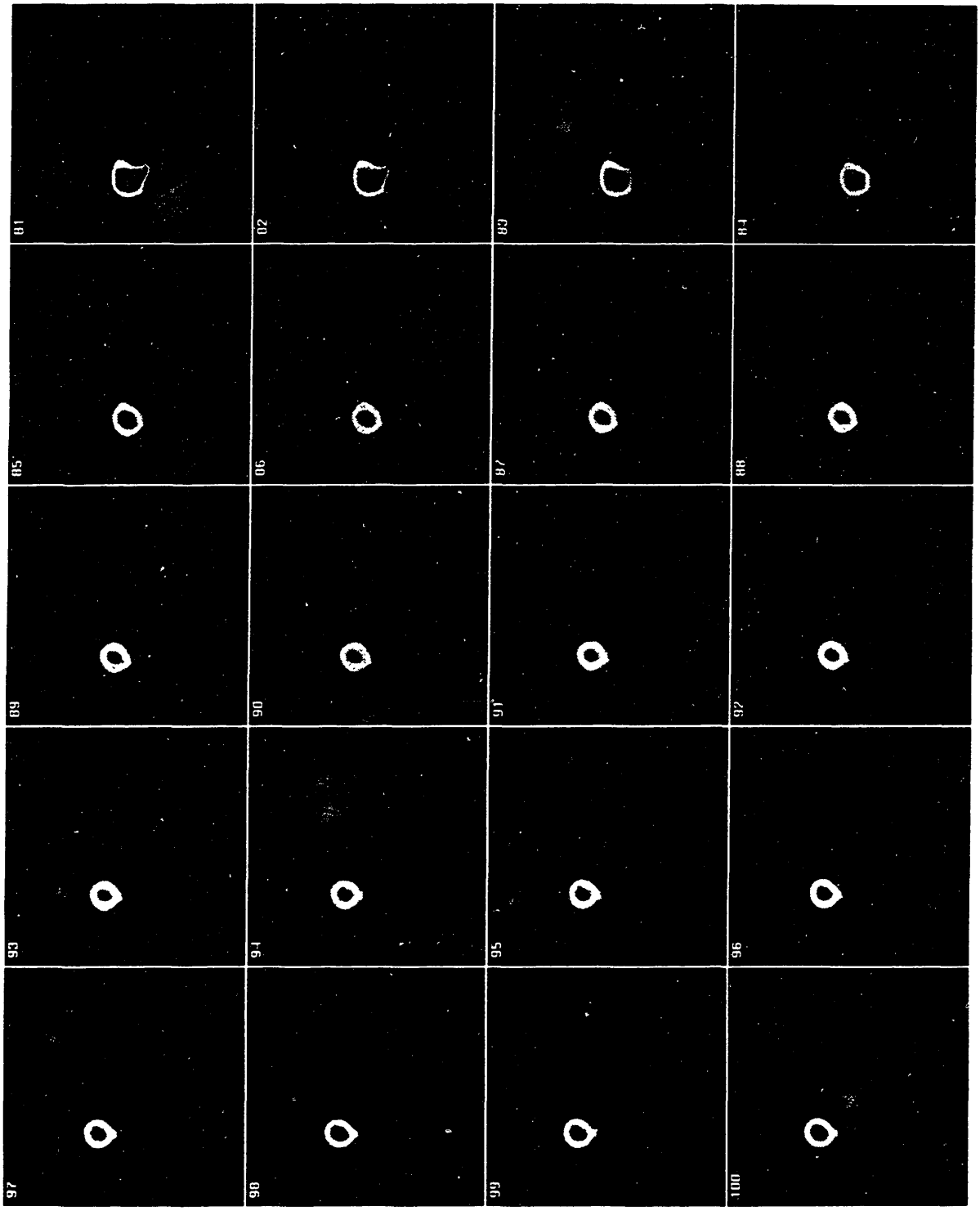


Figure A5: CT Data: Scans 81 through 100

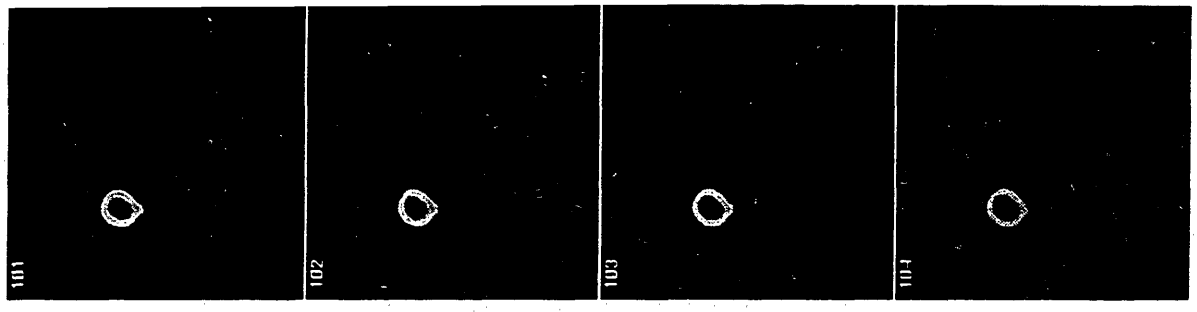


Figure A6: CT Data: Scans 101 through 104

A.2 MRI IN-VIVO DATA

MRI: MGH Imaging Center
Building 149 - 13th Street
Charlestown Navy Yard
Charlestown, MA 02129

Sex: Male
Weight: 135 lbs
Age: 24
Date: 08/03/92

Total Images: 69
Raw Size: 145408 bytes (256*256*2 bytes + Header Size)
Header Size: 14336bytes
Striped Size: 131072 bytes (256*256*2 bytes)

For positioning: A = anterior, P = posterior, R = right, L = left, I = inferior, S = superior and all numbers in mm.

Coronal slices of pelvis (scouts): 6 images total, file001 through file006
Info from X-ray films / File header information not decoded

Matrix	FOV	Thick	Skip	TR	TE
256x128, 1 NEX	48 cm	10.0 mm	14.0 mm	18	5.0Fr

Positioning

R240 / L240 ⇔ centered R0/L0 and I240 / S240 ⇔ centered I0/S0

file001:	P50	file003:	P22	file005:	A6
file002:	P36	file004:	P8	file006:	A20

Axial slices of hip: 10 images total, file007 through file016
Info from X-ray films / File header information not decoded

Matrix	FOV	Thick	Skip	TR	TE
256x256, 2 NEX	20 cm	3.0 mm	13.0 mm	600	12.0Fr 1/1

Positioning

A100 / P100 ⇔ centered A0/P0 and R194 / L6 ⇔ centered R94

file007:	I15	file011:	I67	file015:	I119
file008:	I28	file012:	I80	file016:	I132
file009:	I41	file013:	I93		
file010:	I54	file014:	I106		

Axial slices of hip: 10 images total, file017 through file026
Info from X-ray films / File header information not decoded

Matrix	FOV	Thick	Skip	TR	TE
256x256, 2 NEX	24 cm	3.0 mm	13.0 mm	600	11.0Fr 1/1

Positioning
A120 / P120 ⇔ centered A0/P0 and R214 / L26 ⇔ centered R94

file017:	I15	file021:	I67	file025:	I119
file018:	I28	file022:	I80	file026:	I132
file019:	I41	file023:	I93		
file020:	I54	file024:	I106		

Axial slices of hip: 3 images total, file027 through file029
No X-ray films / File header information not decoded

Matrix	FOV	Thick	Skip	TR	TE
256x256, 2 NEX	24 cm	3.0 mm	3.0 mm	600	23.0Fr 1/1

Positioning

A120 / P120 ⇔ centered A0/P0 and R214 / L26 ⇔ centered R94

file027:	I15	file028:	I18	file029:	I21

Axial slices of hip: 40 images total, file030 through file069
Info from X-ray films / File header information not decoded

Matrix	FOV	Thick	Skip	TR	TE
256x256, 4 NEX	24 cm	3.0 mm	3.0 mm	600	23.0Fr 1/1

Positioning

A120 / P120 ⇔ centered A0/P0 and R214 / L26 ⇔ centered R94

file030:	I15	file044:	I57	file058:	I99
file031:	I18	file045:	I60	file059:	I102
file032:	I21	file046:	I63	file060:	I105
file033:	I24	file047:	I66	file061:	I108
file034:	I27	file048:	I69	file062:	I111
file035:	I30	file049:	I72	file063:	I114
file036:	I33	file050:	I75	file064:	I117
file037:	I36	file051:	I78	file065:	I120
file038:	I39	file052:	I81	file066:	I123
file039:	I42	file053:	I84	file067:	I126
file040:	I45	file054:	I87	file068:	I129
file041:	I48	file055:	I90	file069:	I132
file042:	I51	file056:	I93		
file043:	I54	file057:	I96		

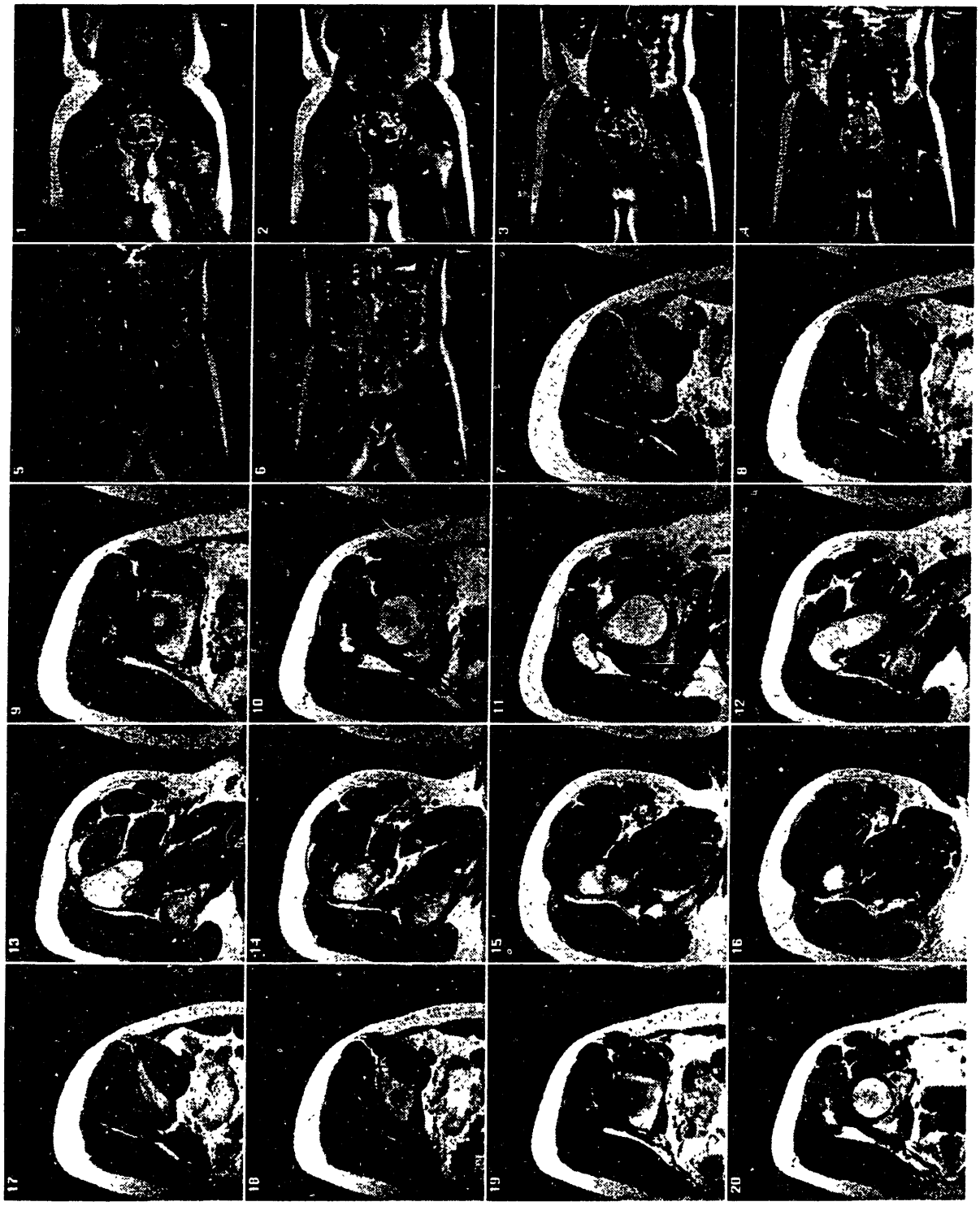


Figure A7: MRI Data: Scans 1 through 20

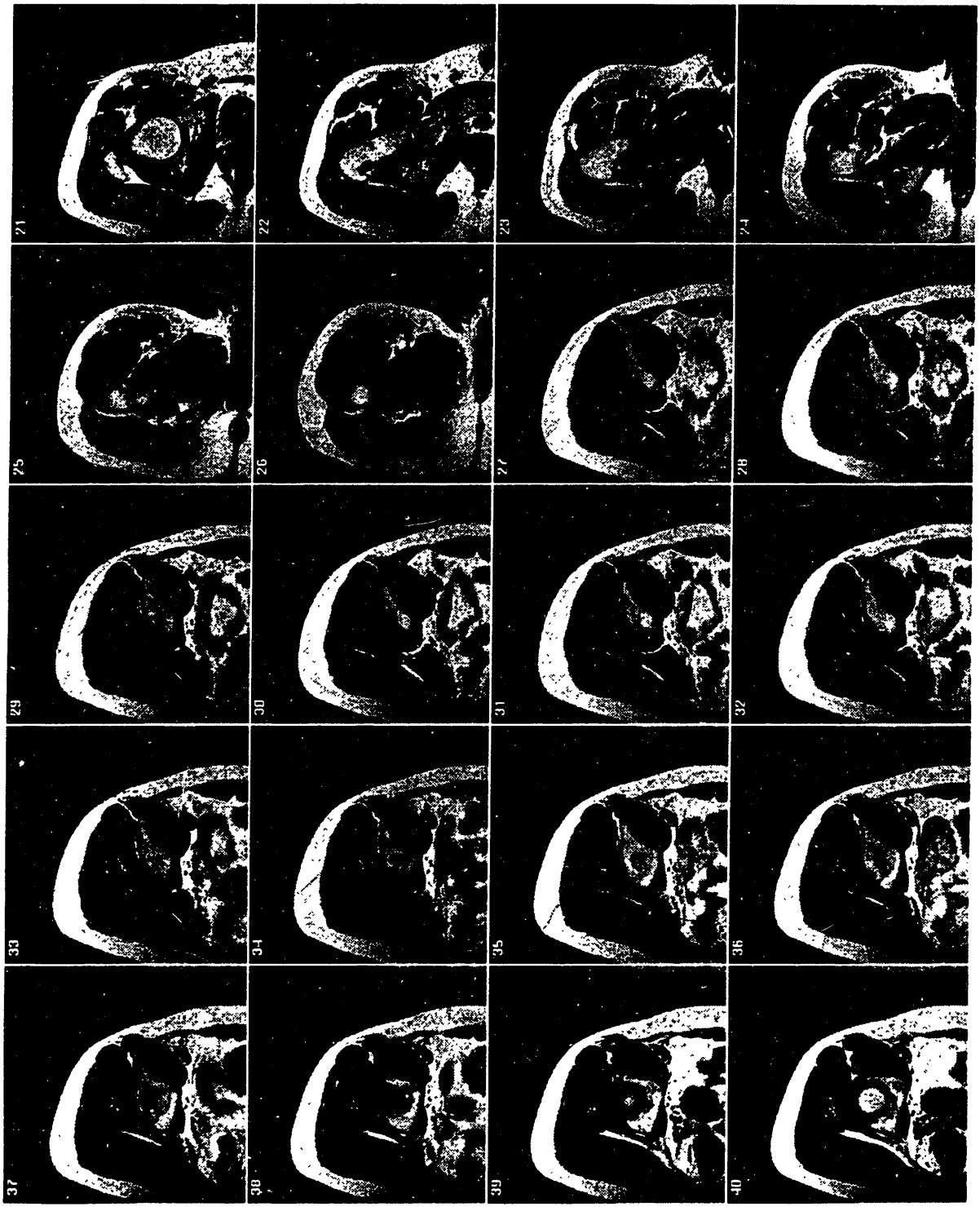


Figure A8: MRI Data: Scans 21 through 40

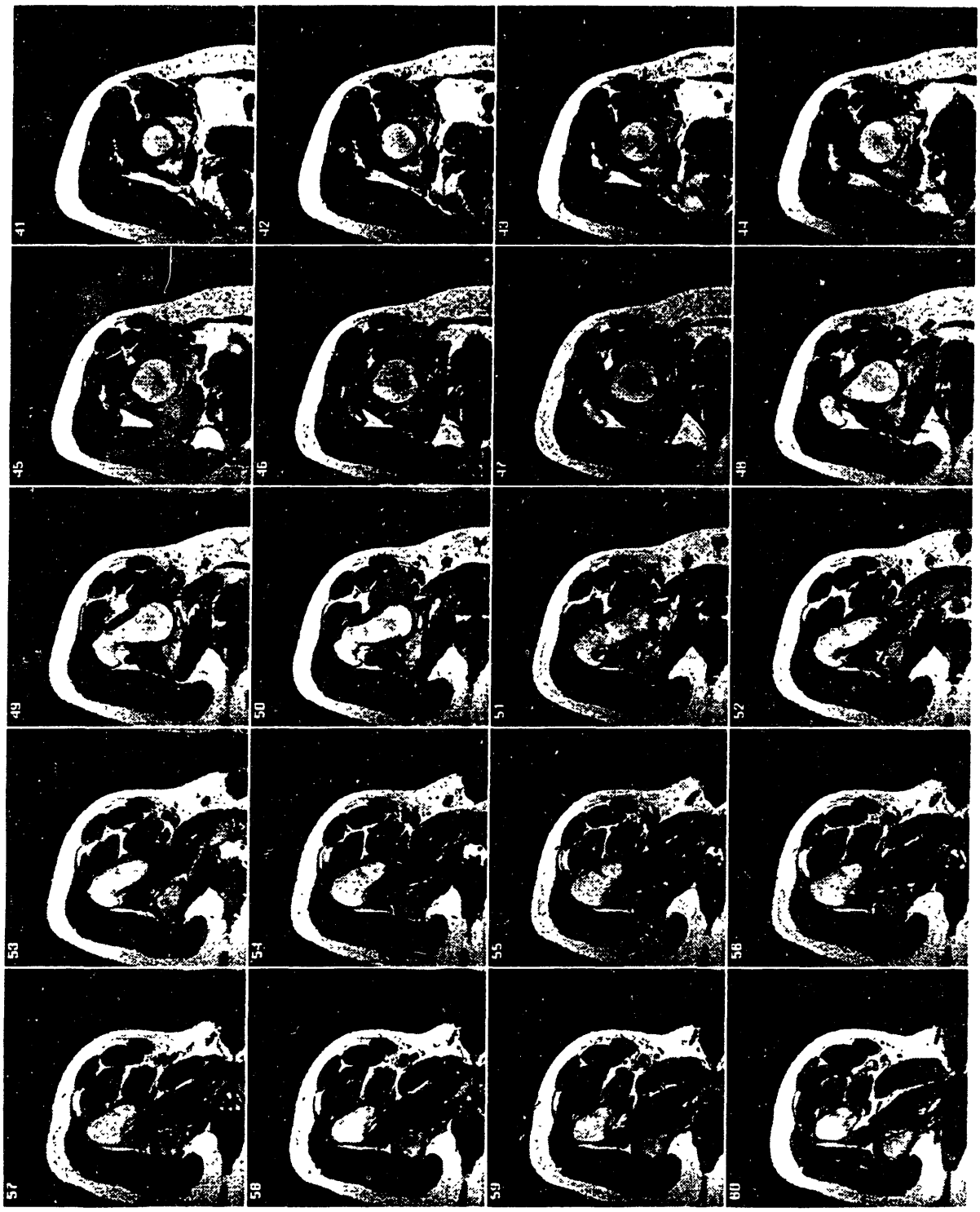


Figure A9: MRI Data: Scans 41 through 60

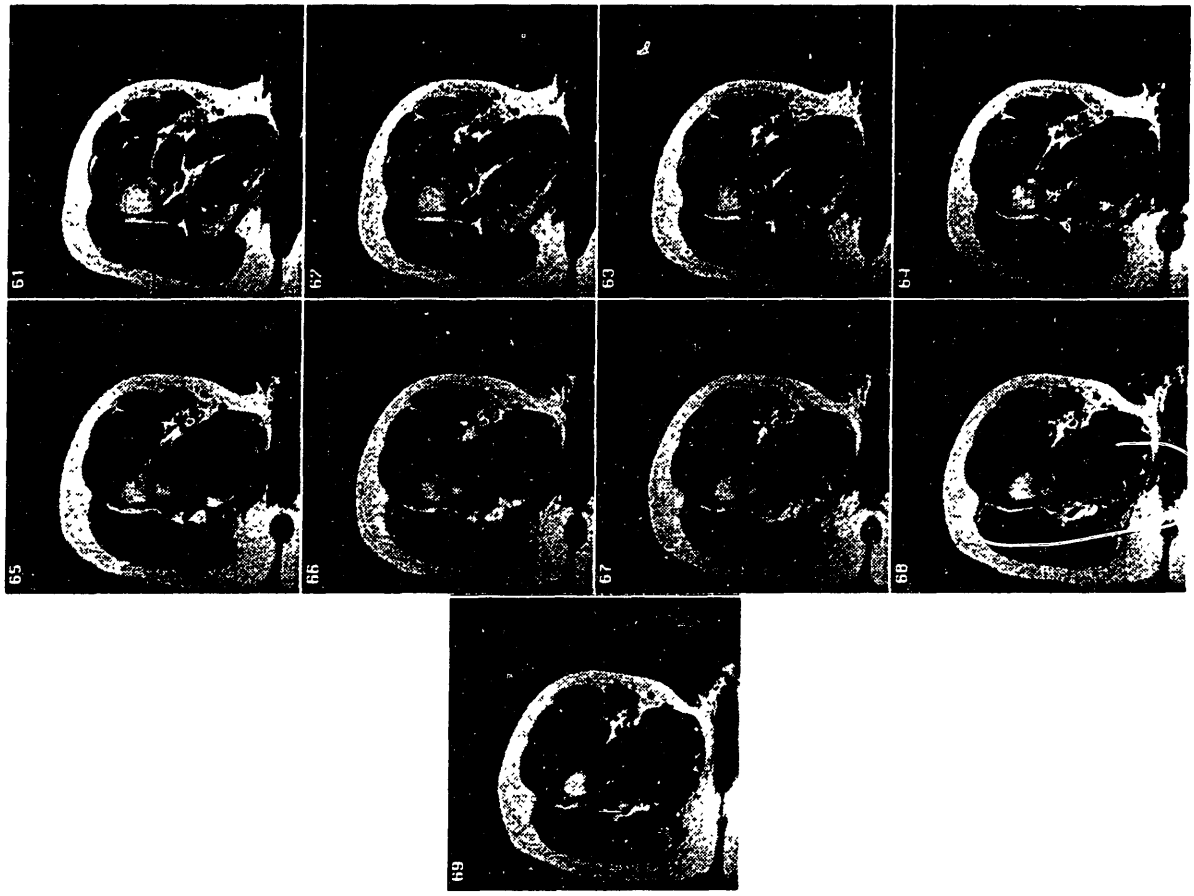


Figure A10: MRI Data: Scans 61 through 69

APPENDIX
B

SEGMENTATION SOFTWARE

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *  view_scan.c
 *
 *  Description: This program allows viewing of CT or MRI scans and
 *               contouring of anatomical information with some user
 *               input.  Any information in the image can be thresholded
 *               and thus segmented into distinct contours.
 *
 *  Created by:  Patrick J. Lord        18-May-94
 *
 *  Modified by:
 *
 *  Known Bugs:
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
 *                      Copyright (C) 1994
 *               Massachusetts Institute of Technology
 *                    Cambridge, Massachusetts
 *
 *  This software is subject to change without notice and should not
 *  be construed as a commitment by MIT.  MIT assumes no responsibility
 *  for the use or reliability of its software.
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 */
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <gl.h>
#include <device.h>
#include <math.h>

#define OFFSET      1024
#define HITHRESH      20
#define LOTHRESH       1
#define FOFFSET     8450
#define IMAGESIZE   512
#define NBFILES     104
#define MAXPTS      2000
```

```
#define MAX(A, B)    ((A) > (B) ? (A) : (B))
#define MIN(A, B)    ((A) < (B) ? (A) : (B))
int gid1;

void usage(command)
char command[];
    {
    printf("\nusage: %s [filename] [width] [height] \007\n\n", command);
    }

main(argc, argv)
int argc;
char *argv[];
    {
    char line[80], file[80];
    char loop=TRUE, normalized=FALSE, threshold=FALSE, blanked=FALSE;
    short val,moveit;
    Device dev;
    int width, height;
    short *image;
    struct stat buf;
    int i, index=0, counter=0;

    height = width = IMAGESIZE;
    image = (short *)calloc(width*height, sizeof(short));

    initialize(argv[0], width, height);
    sprintf(file,"../images/00%d.img",8450);
    read_image(file, image, width, height);
    draw_image(image, width, height);
    sprintf(line,"%s: %s %dx%d", argv[0], file, width, height);
    wintitle(line);

    while (loop)
        {
        while (qtest())
            {
            dev = qread(&val);
            if (dev == ESCKEY)
                {
                gexit();
                exit();
                }
            else if (dev == REDRAW)
                {
                reshapeviewport();
                draw_image(image, width, height);
                }
            else if (dev == BUT145)
                {
                index=MIN(index+HITHRESH, 256);
                printf("Threshold Index: %d\n", index);
                for(i=index-HITHRESH;i<index;i++) mapcolor(i+OFFSET,0,0,0);
                qreset();
                }
            else if (dev == BUT146)
                {
                index=MAX(index-HITHRESH, 0);
                printf("Threshold Index: %d\n", index);
                for(i=index;i<index+HITHRESH;i++) mapcolor(i+OFFSET,i,i,i);
                qreset();
                }
            else if (dev == BUT147)
                {
                index=MIN(index+LOTHRESH, 256);
                printf("Threshold Index: %d\n", index);
                for(i=index-LOTHRESH;i<index;i++) mapcolor(i+OFFSET,0,0,0);
                qreset();
                }
            else if (dev == BUT148)
                {
                index=MAX(index-LOTHRESH, 0);
                printf("Threshold Index: %d\n", index);
                for(i=index;i<index+LOTHRESH;i++) mapcolor(i+OFFSET,i,i,i);
                qreset();
                }
            else if (dev == BUT149)
                {
                if(!normalized) for(i=0;i<index;i++) mapcolor(i+OFFSET,i,i,i);
                else for(i=0;i<index;i++) mapcolor(i+OFFSET,0,0,0);
```

```
                normalized = !normalized;
                qreset();
                }
            else if (dev == BUT150)
                {
                if(threshold) for(i=index;i<256;i++) mapcolor(i+OFFSET,i,i,i);
                else for(i=index;i<256;i++) mapcolor(i+OFFSET,255,255,255);
                threshold = !threshold;
                qreset();
                }
            else if (dev == BUT151)
                {
                find_contour(index+OFFSET);
                qreset();
                }
            else if (dev == BUT152)
                {
                find_fudge();
                qreset();
                }
            else if (dev == BUT153)
                {
                if(blanked && threshold)
                    for(i=index;i<256;i++) mapcolor(i+OFFSET,255,255,255);
                else if(blanked && !threshold)
                    for(i=index;i<256;i++) mapcolor(i+OFFSET,i,i,i);
                else for(i=index;i<256;i++) mapcolor(i+OFFSET,0,0,0);
                blanked = !blanked;
                qreset();
                }
            else if (dev == BUT79)
                {
                counter++;
                counter = (NBFILES+counter)%NBFILES;
                sprintf(file,"../images/00%d.img",FOFFSET+counter);
                read_image(file, image, width, height);
                draw_image(image, width, height);
                sprintf(line,"%s: %s %dx%d", argv[0], file, width, height);
                wintitle(line);
                qreset();
                }
            else if (dev == BUT72)
                {
                counter--;
                counter = (NBFILES+counter)%NBFILES;
                sprintf(file,"../images/00%d.img",FOFFSET+counter);
                read_image(file, image, width, height);
                draw_image(image, width, height);
                sprintf(line,"%s: %s %dx%d", argv[0], file, width, height);
                wintitle(line);
                qreset();
                }
            }
        sleep(1);
        }
    }

initialize(command, width, height)
char command[];
int width, height;
    {
    int i;
    char line[80];

    prefsize(width, height);
    sprintf(line,"%s", command);
    gid1 = winopen(line);

    gconfig();
    for(i=256;i<4096;i++) mapcolor(i,0,0,0);
    for(i=0;i<256;i++) mapcolor(i+OFFSET,i,i,i);

    qdevice(ESCKEY);
    qdevice(BUT72);
    qdevice(BUT79);
    qdevice(BUT145);
    qdevice(BUT146);
    qdevice(BUT147);
    qdevice(BUT148);
    qdevice(BUT149);
    qdevice(BUT150);
```

```
    qdevice(BUT151);
    qdevice(BUT152);
    qdevice(BUT153);
    qdevice(LEFTMOUSE);
    qenter(REDRAW,gid1);
    }

draw_image(image, width, height)
Colorindex *image;
int width, height;
    {
    winset(gid1);
    rectwrite( 0, 0, width-1, height-1, image);
    }

read_image(file, image, width, height)
char file[];
short *image;
int width, height;
    {
    int binfil, i;
    if((binfil=open(file,O_RDONLY))==(-1))
        {
        printf("can't open %s\n",file);
        exit();
        }
    read(binfil, image, width*height*sizeof(short));
    for(i=0;i<width*height;i++)
        {
        *(image+i) /= 16;
        *(image+i) += OFFSET;
        }
    close(binfil);
    }

find_contour(index)
int index;
    {
    int j,x,y,position;
    int contour[MAXPTS][2], contour1[MAXPTS][2], nb_pts, nb_pts1;
    short thresh, *data, *datas;
    char found, looped;

    data = (short *)calloc(IMAGESIZE*IMAGESIZE, sizeof(short));
    datas = (short *)calloc(IMAGESIZE*IMAGESIZE, sizeof(short));
    rectread( 0, 0, IMAGESIZE-1, IMAGESIZE-1, (Colorindex *)data);
    for(j=0;j<IMAGESIZE*IMAGESIZE;j++)
        {
        if(*(data+j) > index) *(data+j)=255;
        else *(data+j)=0;
        *(datas+j) = *(data+j);
        }
    find_start(&contour[0][0],&contour[0][1],&thresh, data);

    x=contour[0][0];
    y=contour[0][1];
    *(data+y*IMAGESIZE+x)= 0;
    x++;
    y--;
    position=7;
    looped=FALSE;
    for(nb_pts=1; nb_pts<MAXPTS && !looped; nb_pts++)
        {
        found = FALSE;
        for(j=0;j<8 && !found;j++)
            {
            switch(position%8)
                {
                case 0:
                if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
                else y++;
                break;
                case 1:
                if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
                else x--;
                break;
                case 2:
                if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
                else x--;
                break;
                case 3:
```

```
            if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
            else y--;
            break;
            case 4:
            if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
            else y--;
            break;
            case 5:
            if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
            else x++;
            break;
            case 6:
            if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
            else x++;
            break;
            case 7:
            if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
            else y++;
            break;
            }
        if(!found) position++;
        }

    contour[nb_pts][0]= x;
    contour[nb_pts][1]= y;
    *(data+y*IMAGESIZE+x)= 0;

    if((((x-1)==contour[0][0] && (y-1)==contour[0][1]) ||
        ((x-1)==contour[0][0] && (y)  ==contour[0][1]) ||
        ((x-1)==contour[0][0] && (y+1)==contour[0][1]) ||
        ((x)  ==contour[0][0] && (y-1)==contour[0][1]) ||
        ((x)  ==contour[0][0] && (y+1)==contour[0][1]) ||
        ((x+1)==contour[0][0] && (y-1)==contour[0][1]) ||
        ((x+1)==contour[0][0] && (y)  ==contour[0][1]) ||
        ((x+1)==contour[0][0] && (y+1)==contour[0][1]))&& nb_pts>8 )
            looped=TRUE;

    position = (position+7)%8;
    switch(position)
        {
        case 0:
        x++;
        break;
        case 1:
        x++;
        y++;
        break;
        case 2:
        y++;
        break;
        case 3:
        y++;
        x--;
        break;
        case 4:
        x--;
        break;
        case 5:
        x--;
        y--;
        break;
        case 6:
        y--;
        break;
        case 7:
        x++;
        y--;
        break;
        }
    }

for(j=0;j<IMAGESIZE*IMAGESIZE;j++) *(data+j) = *(datas+j);
x=contour1[0][0]=contour[0][0];
y=contour1[0][1]=contour[0][1];
*(data+y*IMAGESIZE+x)= 0;
x--;
y++;
position=3;
looped=FALSE;
for(nb_pts1=1; nb_pts1<MAXPTS && !looped; nb_pts1++)
    {
```

```
found = FALSE;
for(j=0;j<8 && !found;j++)
    {
    switch(position%8)
        {
        case 0:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else y--;
        break;
        case 1:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else y--;
        break;
        case 2:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else x++;
        break;
        case 3:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else x++;
        break;
        case 4:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else y++;
        break;
        case 5:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else y++;
        break;
        case 6:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else x--;
        break;
        case 7:
        if(*(data+y*IMAGESIZE+x)>=thresh) found=TRUE;
        else x--;
        break;
        }
    if(!found) position = (position+7)%8;
    }

contour1[nb_pts1][0]= x;
contour1[nb_pts1][1]= y;
*(data+y*IMAGESIZE+x)= 0;

if((((x-1)==contour1[0][0] && (y-1)==contour1[0][1]) ||
    ((x-1)==contour1[0][0] && (y)  ==contour1[0][1]) ||
    ((x-1)==contour1[0][0] && (y+1)==contour1[0][1]) ||
    ((x)  ==contour1[0][0] && (y-1)==contour1[0][1]) ||
    ((x)  ==contour1[0][0] && (y+1)==contour1[0][1]) ||
    ((x+1)==contour1[0][0] && (y-1)==contour1[0][1]) ||
    ((x+1)==contour1[0][0] && (y)  ==contour1[0][1]) ||
    ((x+1)==contour1[0][0] && (y+1)==contour1[0][1]))&& nb_pts1>8 )
        looped=TRUE;

position = (position+1)%8;
switch(position)
    {
    case 0:
    x++;
    break;
    case 1:
    x++;
    y++;
    break;
    case 2:
    y++;
    break;
    case 3:
    y++;
    x--;
    break;
    case 4:
    x--;
    break;
    case 5:
    x--;
    y--;
    break;
    case 6:
    y--;
```

```
            break;
            case 7:
            x++;
            y--;
            break;
            }
        }

    printf("Number of points in Type 1 contour %d\n", nb_pts);
    printf("Number of points in Type 2 contour %d\n", nb_pts1);
    if(nb_pts1<nb_pts)
        {
        printf("Type 2 contour selected\n");
        nb_pts = nb_pts1;
        for(j=0;j<nb_pts;j++)
            {
            contour[j][0]=contour1[j][0];
            contour[j][1]=contour1[j][1];
            }
        }
    else printf("Type 1 contour selected\n");

    check_duplicate(nb_pts, contour);
    check_direction(nb_pts, contour);

    /* Draw contour */
    ortho2(0.,(float)IMAGESIZE-1.,0.,(float)IMAGESIZE-1.);
    move2i(contour[nb_pts-1][0],contour[nb_pts-1][1]);
    color(GREEN);
    for(j=0;j<nb_pts/2;j++) draw2i(contour[j][0],contour[j][1]);
    color(RED);
    for(j=nb_pts/2;j<nb_pts;j++) draw2i(contour[j][0],contour[j][1]);

    printf("%d\n", nb_pts);
    for(j=0;j<nb_pts;j++)
        {
        printf("%4d%4d",contour[j][0],contour[j][1]);
        if((j+1)%9 == 0) printf("\n");
        }
    printf("\n");
    fflush(stdout);
    }

find_start(x,y,thresh, data)
int *x,*y;
short *thresh, *data;
    {
    int xorg,yorg;
    char found = FALSE;

    getorigin((long *)&xorg,(long *)&yorg);
    printf("Ready: Select start point\n\007");
    while(!found)
        {
        printf("\r");
        fflush(stdout);
        while(!getbutton(LEFTMOUSE));
        *x = (int)getvaluator(MOUSEX) - xorg;
        *y = (int)getvaluator(MOUSEY) - yorg;
        *thresh = *(data+ *y*IMAGESIZE+ *x);
        printf("x = %d ; y =%d ; threshold = %d ",*x,*y,*thresh);
        if(*x>0 && *x<IMAGESIZE &&
            *y>0 && *y<IMAGESIZE && *thresh>0) found=TRUE;
        }
    printf("\n");
    }

find_fudge()
    {
    int xorg,yorg, x,y, index=0,j;
    char quit=FALSE;
    int contour[2000][2];

    getorigin((long *)&xorg,(long *)&yorg);
    printf("Ready: Select start point\n");
    printf("\n");
    fflush(stdout);
    while(!getbutton(LEFTMOUSE));
    x = (int)getvaluator(MOUSEX) - xorg;
    y = (int)getvaluator(MOUSEY) - yorg;
    printf("x = %d ; y =%d\n",x,y);
```

```
    contour[index][0]= x;
    contour[index][1]= y;
    index++;

    while(!quit)
        {
        while(contour[index-1][0] == x && contour[index-1][1] == y)
            {
            x = (int)getvaluator(MOUSEX) - xorg;
            y = (int)getvaluator(MOUSEY) - yorg;
            }

        contour[index][0]= x;
        contour[index][1]= y;
        if(contour[index][0]==contour[0][0] &&
            contour[index][1]==contour[0][1]) quit=TRUE;
        else index++;
        }

    printf("Number of points = %d\n", index);
    /* Draw contour */
    ortho2(0.,(float)IMAGESIZE-1.,0.,(float)IMAGESIZE-1.);
    move2i(contour[index-1][0],contour[index-1][1]);
    color(RED);
    for(j=0;j<index;j++) draw2i(contour[j][0],contour[j][1]);

    printf("%d\n", index);
    for(j=0;j<index;j++)
        {
        printf("%4d%4d",contour[j][0],contour[j][1]);
        if((j+1)%9 == 0) printf("\n");
        }
    fflush(stdout);
    }

check_duplicate(nb_pts, contour)
int nb_pts, contour[MAXPTS][2];
    {
    int i, j;

    printf("Checking for Duplicate Points in Contour\n");
    for(i=0;i<nb_pts;i++)
    for(j=i+1;j<nb_pts;j++)
        if(contour[i][0]==contour[j][0] && contour[i][1]==contour[j][1])
            printf("ERROR: Duplicate in contour @ %d and %d\n\007",i+1,j+1);
    }

check_direction(nb_pts, contour)
int nb_pts, contour[MAXPTS][2];
    {
    int i;
    float a1, b1, c1, a2, b2, c2, a3, b3, c3;
    float total_a=0., total_b=0., total_c=0.;

    printf("Checking Contour Direction\n");
    for(i=1;i<nb_pts;i++)
        {
        a1 = (float)(contour[i-1][0] - contour[i][0]);
        b1 = (float)(contour[i-1][1] - contour[i][1]);
        c1 = 0.;
        a2 = (float)(contour[(i+1)%nb_pts][0] - contour[i][0]);
        b2 = (float)(contour[(i+1)%nb_pts][1] - contour[i][1]);
        c2 = 0.;
        cross_product(a1,b1,c1,a2,b2,c2,&a3,&b3,&c3);
        total_a += a3;
        total_b += b3;
        total_c += c3;
        }
    printf("Resultant Vector is: % f % f % f\n", total_a, total_b, total_c);
    printf("Contour Direction is: ");
    if(total_c > 0.) printf("clockwise\n");
    else printf("counter clockwise\n");
    }

cross_product(a1,b1,c1,a2,b2,c2,a3,b3,c3)
float a1, b1, c1, a2, b2, c2, *a3, *b3, *c3;
    {
    *a3 = b1*c2-c1*b2;
    *b3 = c1*a2-a1*c2;
    *c3 = a1*b2-b1*a2;
    }
```

APPENDIX
C

STEREOLITHOGRAPHY SOFTWARE

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *  stlviewer.c version 1.0
 *
 *  Description:  This program will read a STL ASCII file and display
 *  the object.  One option in the mouse menu allows to slice the object
 *  along one of the coordinate axes and generate bitmaps for the DAVOS
 *  stereolithography system.
 *
 *  Usage: Self-Explanatory
 *
 *  Created by: Patrick J. Lord     18-May-94
 *
 *  Modified by:
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
 *                      Copyright (C) 1994
 *               Massachusetts Institute of Technology
 *                     Cambridge, Massachusetts
 *
 *  This software is subject to change without notice and should not
 *  be construed as a commitment by MIT.  MIT assumes no responsibility
 *  for the use or reliability of its software.
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 */
#include <stdio.h>
#include <math.h>
#include <fcntl.h>
#include <gl.h>
#include <device.h>

#define MAXTRIANG   20000
#define MAXVERTEX   30
#define OFFSET      1024
#define SCALE_FAC   1.0
#define SCALE_FAC2  1.95
#define MAX(A, B)   ((A) > (B) ? (A) : (B))
#define MIN(A, B)   ((A) < (B) ? (A) : (B))

struct hiddline_info {
```

```
    int order;
    float zvalue; };

struct point { float x, y, z;};

float polygons[MAXTRIANG][MAXVERTEX][3];
int poly_vert[MAXTRIANG];
float normals[MAXTRIANG][MAXVERTEX][3];
struct point avgxyz_triang[MAXTRIANG];
int nb_triang;
float xmin,xmax,ymin,ymax,zmin,zmax,scaf;
char hidden_line = FALSE, shading = FALSE, draw_normals=FALSE;

static float shiny_material[] = {EMISSION, 0.2,0.2,0.2,
                                 AMBIENT, 0.2,0.2,0.2,
                                 DIFFUSE, 0.2,0.2,0.2,
                                 SPECULAR, 0.45,0.45,0.45,
                                 SHININESS, 10.,
                                 ALPHA, 1.0,
                                 LMNULL};

static float lt0[] = {LCOLOR,1,1,1,
                      POSITION,0,0,1,0,
                      LMNULL};

static float lm[] = {LOCALVIEWER, 0,
                     LMNULL};
int menu, submenu;
long gid1, gid2;

main()
    {
    long dev;
    short val;
    int movexy,move_z;
    short x = 0,y = 0,z = 0, old_x = 0, old_y = 0, old_z = 0;
    char quit=FALSE, automatic_tumble = FALSE;
    long menuval;

    initialize();
    readdata();
    drawstructure(x,y,z);
    while (!quit)
        {
        while (qtest())
            {
            dev = qread(&val);
            if (dev == ESCKEY) quit = TRUE;
            else if (dev == REDRAW)
                {
                reshapeviewport();
                drawstructure(x,y,z);
                }
            else if (dev == RIGHTMOUSE)
                {
                if (val) menuval = dopup(menu);
                if(menuval==1)
                    {
                    readdata();
                    drawstructure(x,y,z);
                    }
                else if(menuval==3)
                    {
                    slice_object();
                    }
                else if(menuval==4)
                    {
                    quit = TRUE;
                    }
                else if(menuval==10)
                    {
                    hidden_line = FALSE;
                    shading=FALSE;
                    zbuffer(FALSE);
                    shademodel(FLAT);
                    drawstructure(x,y,z);
                    }
                else if(menuval==11)
                    {
                    hidden_line = TRUE;
                    shading=FALSE;
```

```
                zbuffer(FALSE);
                shademodel(FLAT);
                drawstructure(x,y,z);
                }
            else if(menuval==12)
                {
                shademodel(FLAT);
                hidden_line = FALSE;
                shading=TRUE;
                lsetdepth(getgdesc(GD_ZMIN), getgdesc(GD_ZMAX));
                zbuffer(TRUE);
                lmdef(DEFMATERIAL,1,0,shiny_material);
                lmdef(DEFLIGHT,1,0,lt0);
                lmdef(DEFLMODEL,1,0,lm);
                lmbind(MATERIAL,1);
                lmbind(LIGHT0,1);
                lmbind(LMODEL,1);
                drawstructure(x,y,z);
                }
            else if(menuval==13)
                {
                shademodel(GOURAUD);
                hidden_line = FALSE;
                shading=TRUE;
                lsetdepth(getgdesc(GD_ZMIN), getgdesc(GD_ZMAX));
                zbuffer(TRUE);
                lmdef(DEFMATERIAL,1,0,shiny_material);
                lmdef(DEFLIGHT,1,0,lt0);
                lmdef(DEFLMODEL,1,0,lm);
                lmbind(MATERIAL,1);
                lmbind(LIGHT0,1);
                lmbind(LMODEL,1);
                drawstructure(x,y,z);
                }
            else if(menuval==14)
                {
                x = y = z = 0;
                drawstructure(x,y,z);
                }
            else if(menuval==15)
                {
                automatic_tumble = !automatic_tumble;
                }
            else if(menuval==16)
                {
                draw_normals = !draw_normals;
                drawstructure(x,y,z);
                }
            }
        else if(dev == LEFTMOUSE)
            movexy = val;   /* left mouse is down */
        else if(dev == MIDDLEMOUSE)
            move_z = val;   /* middle mouse is down */
        }
    if(movexy)
        {
        x += getvaluator(MOUSEX) - old_x;
        y += getvaluator(MOUSEY) - old_y;
        drawstructure(x,y,z);
        }
    else if(move_z)
        {
        z += getvaluator(MOUSEX) - old_z;
        drawstructure(x,y,z);
        }
    else if(automatic_tumble)
        {
        x+=20;
        y+=20;
        z+=20;
        drawstructure(x,y,z);
        }
    else{
        sleep(1);
        old_x = getvaluator(MOUSEX);
        old_y = getvaluator(MOUSEY);
        old_z = getvaluator(MOUSEX);
        }
    }
textinit();
gexit();
```

```
    exit();
    }

initialize()
    {
    char *menu_str =    "Menu Options %t| \
                        Read STL data file | \
                        Display Options %m| \
                        Slice Object %l| \
                        Quit Program";

    char *submenu_str =" Wire Mesh %x10| \
                        Hidden Line %x11| \
                        Flat Shading %x12| \
                        Gouraud Shading %x13| \
                        Reset Orientation %x14| \
                        Automatic Tumble on/off %x15| \
                        Draw Normals on/off %x16";
    foreground();
#ifdef INDIGO
    prefposition(0, 511, 255, 767);
#else
    prefposition(100, 611, 412, 923);
#endif
    gid1 = winopen("Display");
    wintitle("M.I.T. Newman Laboratory: .STL Viewer");
    keepaspect(1,1);
    winconstraints();

    RGBmode();
    doublebuffer();
    shademodel(FLAT);
    drawmode(NORMALDRAW);
    gconfig();
    mmode(MVIEWING);
    color(BLACK);
    clear();
    swapbuffers();

#ifdef INDIGO
    prefposition(512, 1023, 255, 767);
#else
    prefposition(668, 1179, 412, 923);
#endif
    gid2 = winopen("Slice");
    wintitle("Davos Stereolithography Format");
    color(BLACK);
    clear();

#ifdef INDIGO
    textport(0, 1023, 10, 260);
#else
    textport(0, 1279, 100, 350);
#endif
    tpon();

    qdevice(ESCKEY);
    qdevice(REDRAW);
    qdevice(LEFTMOUSE);
    qdevice(MIDDLEMOUSE);
    qdevice(RIGHTMOUSE);
    qenter(REDRAW,gid1);

    submenu = defpup(submenu_str);
    menu = defpup(menu_str, submenu);

    printf("\n\n\n");
    printf("\n    M.I.T. STL Format to Davos Stereolithography Bitmap ");
    printf("Format Converter\n");
    printf("\n         Newman Laboratory for Biomechanics and Human Rehabilitation\n");
    printf("                  Massachusetts Institute of Technology\n");
    printf("                        77 Massachusetts Avenue\n");
    printf("                     Cambridge, MA 02139, U.S.A.\n");
    printf("\n                     Version 0.7b by Patrick J. Lord\n\n");
    printf("\n\n");

    }

readdata()
    {
    register i, j;
```

```
char filename[80];
FILE *ascfil;
int pindex=0, nindex=0, vindex=0;
char solidname[80], stringdat[80], done=FALSE;
float temp;

printf("\nEnter stereolithography file name [.stl]: \007");
fflush(stdout);
scanf("%s", filename);

if((ascfil=fopen(filename,"r")) == NULL)
    {
    printf("Sorry.... Can't open datafile!\n");
    exit();
    }

while(!done)
    {
    fscanf(ascfil,"%s", stringdat);
    if(!strcmp(stringdat, "SOLID"))
        {
        fscanf(ascfil,"%s", solidname);
        printf("Reading info for solid: %s \n", solidname);
        }
    else if(!strcmp(stringdat, "FACET"))
        {
        fscanf(ascfil,"%s", stringdat);
        if(!strcmp(stringdat, "NORMAL"))
            {
            fscanf(ascfil,"%e %e %e",
                &normals[nindex][0][0],
                &normals[nindex][0][1],
                &normals[nindex][0][2]);
            nindex++;
            }
        fscanf(ascfil,"%s", stringdat);
        if(!strcmp(stringdat, "OUTER"))
            {
            fscanf(ascfil,"%s", stringdat);
            if(!strcmp(stringdat, "LOOP"))
                {
                fscanf(ascfil,"%s", stringdat);
                while(!strcmp(stringdat, "VERTEX"))
                    {
                    if((vindex+1)>MAXVERTEX)
                        {
                        printf("Error: Facet has at least %d vertices &",
                            vindex+1);
                        printf(" the maximum allowed is %d \n", MAXVERTEX);
                        exit(0);
                        }
                    fscanf(ascfil,"%e %e %e",
                        &polygons[pindex][vindex][0],
                        &polygons[pindex][vindex][1],
                        &polygons[pindex][vindex][2]);
                    vindex++;
                    fscanf(ascfil,"%s", stringdat);
                    }
                if(!strcmp(stringdat, "ENDLOOP"))
                    {
                    poly_vert[pindex]=vindex;
                    vindex=0;
                    for(j=1;j<poly_vert[pindex];j++)
                        {
                        normals[pindex][j][0] = normals[pindex][0][0];
                        normals[pindex][j][1] = normals[pindex][0][1];
                        normals[pindex][j][2] = normals[pindex][0][2];
                        }
                    fscanf(ascfil,"%s", stringdat);
                    }
                }
            }
        if(!strcmp(stringdat, "ENDFACET")) pindex++;
        }
    else if(!strcmp(stringdat, "END"))
        {
        fscanf(ascfil,"%s", stringdat);
        if(!strcmp(stringdat, "SOLID"))
            {
            done = TRUE;
            }
```

```
            }
        }
    fclose(ascfil);
    nb_triang = pindex;
    printf("Number of Facets is %d, ", nb_triang);
    printf("Number of Normals is %d \n", nindex);
    printf("Done reading info for solid: %s \n", solidname);

    xmin = polygons[0][0][0];
    xmax = xmin;
    ymin = polygons[0][0][1];
    ymax = ymin;
    zmin = polygons[0][0][2];
    zmax = zmin;
    for(i=0;i<nb_triang;i++)
        for(j=0;j<poly_vert[i];j++)
        {
        xmin = MIN(xmin, polygons[i][j][0]);
        xmax = MAX(xmax, polygons[i][j][0]);
        ymin = MIN(ymin, polygons[i][j][1]);
        ymax = MAX(ymax, polygons[i][j][1]);
        zmin = MIN(zmin, polygons[i][j][2]);
        zmax = MAX(zmax, polygons[i][j][2]);
        }

    for(i=0;i<nb_triang;i++)
        {
        for(j=0;j<poly_vert[i];j++)
            {
            polygons[i][j][0] -= (xmax+xmin)/2.;
            polygons[i][j][1] -= (ymax+ymin)/2.;
            polygons[i][j][2] -= (zmax+zmin)/2.;
            avgxyz_triang[i].x += polygons[i][j][0];
            avgxyz_triang[i].y += polygons[i][j][1];
            avgxyz_triang[i].z += polygons[i][j][2];
            }
        avgxyz_triang[i].x /= (float)poly_vert[i];
        avgxyz_triang[i].y /= (float)poly_vert[i];
        avgxyz_triang[i].z /= (float)poly_vert[i];
        }

    temp = (xmax+xmin)/2.;
    xmin -= temp;
    xmax -= temp;
    temp = (ymax+ymin)/2.;
    ymin -= temp;
    ymax -= temp;
    temp = (zmax+zmin)/2.;
    zmin -= temp;
    zmax -= temp;

    printf("\txmin: %f, xmax: %f, xdist: %f \n", xmin, xmax, xmax-xmin);
    printf("\tymin: %f, ymax: %f, ydist: %f \n", ymin, ymax, ymax-ymin);
    printf("\tzmin: %f, zmax: %f, zdist: %f \n", zmin, zmax, zmax-zmin);

    winset(gid1);
    scaf = (xmax-xmin)/SCALE_FAC;
    scaf = MAX(scaf, (ymax-ymin)/SCALE_FAC);
    scaf = MAX(scaf, (zmax-zmin)/SCALE_FAC);
    ortho(-scaf,scaf,-scaf,scaf,-scaf,scaf);
    }

structure()
    {
    register i,j;
    float vector[3];
    struct hiddline_info sorted_triangs[MAXTRIANG];

    if(shading)
        {
        for(i=0;i<nb_triang;i++)
            {
            concave(TRUE);
            bgnpolygon();
            for(j=0;j<poly_vert[i];j++)
                {
                n3f((float *)&normals[i][j][0]);
                v3f((float *)&polygons[i][j][0]);
                }
            endpolygon();
            }
```

```
        if(draw_normals)
            {
            RGBcolor(255,0,0);
            for(i=0;i<nb_triang;i++)
                {
                bgnline();
                v3f((float *)&avgxyz_triang[i]);
                vector[0] = avgxyz_triang[i].x + normals[i][0][0]*scaf/10.;
                vector[1] = avgxyz_triang[i].y + normals[i][0][1]*scaf/10.;
                vector[2] = avgxyz_triang[i].z + normals[i][0][2]*scaf/10.;
                v3f(vector);
                endline();
                }
            }
        }

    else if(hidden_line)
        {
        sort_triangs(sorted_triangs, avgxyz_triang, nb_triang);
        for(i=0;i<nb_triang;i++)
            {
            RGBcolor(0,0,0);
            concave(TRUE);
            bgnpolygon();
            for(j=0;j<poly_vert[sorted_triangs[i].order];j++)
                v3f((float *)&polygons[sorted_triangs[i].order][j][0]);
            endpolygon();

            RGBcolor(255,255,255);
            bgnclosedline();
            for(j=0;j<poly_vert[sorted_triangs[i].order];j++)
                v3f((float *)&polygons[sorted_triangs[i].order][j][0]);
            endclosedline();

            if(draw_normals)
                {
                RGBcolor(255,0,0);
                bgnline();
                v3f((float *)&avgxyz_triang[sorted_triangs[i].order]);
                vector[0] = avgxyz_triang[sorted_triangs[i].order].x +
                            normals[sorted_triangs[i].order][0][0]*scaf/10.;
                vector[1] = avgxyz_triang[sorted_triangs[i].order].y +
                            normals[sorted_triangs[i].order][0][1]*scaf/10.;
                vector[2] = avgxyz_triang[sorted_triangs[i].order].z +
                            normals[sorted_triangs[i].order][0][2]*scaf/10.;
                v3f(vector);
                endline();
                }
            }
        }
    else
        {
        RGBcolor(255,255,255);
        for(i=0;i<nb_triang;i++)
            {
            bgnclosedline();
            for(j=0;j<poly_vert[i];j++) v3f((float *)&polygons[i][j][0]);
            endclosedline();
            }

        if(draw_normals)
            {
            RGBcolor(255,0,0);
            for(i=0;i<nb_triang;i++)
                {
                bgnline();
                v3f((float *)&avgxyz_triang[i]);
                vector[0] = avgxyz_triang[i].x + normals[i][0][0]*scaf/10.;
                vector[1] = avgxyz_triang[i].y + normals[i][0][1]*scaf/10.;
                vector[2] = avgxyz_triang[i].z + normals[i][0][2]*scaf/10.;
                v3f(vector);
                endline();
                }
            }
        }
    }

drawstructure(x,y,z)
short x,y,z;
    {
```

```
    winset(gid1);
    pushmatrix();
    /*
    rotate(x+900, 'x');
    rotate(y, 'y');
    rotate(z, 'z');
    */
    polarview(0.,x,y-900,z);
    czclear(0x000000, getgdesc(GD_ZMAX));
    structure();
    axes();
    popmatrix();
    swapbuffers();
    }

axes()
    {
    float origin[3], vectorx[3], vectory[3], vectorz[3];

    origin[0] = xmin;
    origin[1] = ymin;
    origin[2] = zmin;
    vectorx[0] = xmin+scaf;
    vectorx[1] = ymin;
    vectorx[2] = zmin;
    vectory[0] = xmin;
    vectory[1] = ymin+scaf;
    vectory[2] = zmin;
    vectorz[0] = xmin;
    vectorz[1] = ymin;
    vectorz[2] = zmin+scaf;

    RGBcolor(255,0,0);
    bgnline();
    v3f(origin);
    v3f(vectorx);
    endline();
    RGBcolor(255,255,255);
    cmov(vectorx[0], vectorx[1], vectorx[2]);
    charstr("x");
    RGBcolor(0,255,0);
    bgnline();
    v3f(origin);
    v3f(vectory);
    endline();
    RGBcolor(255,255,255);
    cmov(vectory[0], vectory[1], vectory[2]);
    charstr("y");
    RGBcolor(0,0,255);
    bgnline();
    v3f(origin);
    v3f(vectorz);
    endline();
    RGBcolor(255,255,255);
    cmov(vectorz[0], vectorz[1], vectorz[2]);
    charstr("z");
    }

sort_triangs(zvalue_order,xyzvalues,n)
    struct hiddline_info zvalue_order[];
    struct point xyzvalues[];
    int  n;
    {
    int i;
    Matrix T;
    int comp_function();

    getmatrix(T);
    for(i=0;i<n;i++)
        {
        zvalue_order[i].order = i;
        zvalue_order[i].zvalue = xyzvalues[i].x*T[0][2] +
                                 xyzvalues[i].y*T[1][2] +
                                 xyzvalues[i].z*T[2][2] +
                                 T[3][2];
        }
    qsort((char *)zvalue_order,n,sizeof(zvalue_order[0]),comp_function);
    }

comp_function(ep,ep2)
struct hiddline_info *ep,*ep2;
```

```
    {
    if(ep->zvalue > ep2->zvalue) return(1);
    else if(ep->zvalue == ep2->zvalue) return(0);
    else return(-1);
    }

slice_object()
    {
    char inplane, filename[80], nullchr = '\000';
    int i, j, k, l, index=0, slice_axis, slice_number, binfil;
    float a, x, y, x1, y1, z1, x2, y2, z2, plane;
    float xc[MAXVERTEX], yc[MAXVERTEX], scaf2, vector[2], tmp;
    float start_plane, end_plane, delta_plane, pixel_size;

    winset(gid2);
    winpop();

    printf("\nChoose slice axis [1 = X, 2 = Y, 3 = Z]: \007");
    scanf("%d", &slice_axis);

    if(slice_axis == 1)
        {
        start_plane = xmin;
        end_plane = xmax;
        printf("\n\txmin: %f, xmax: %f, xdist: %f\n\n", xmin, xmax, xmax-xmin);
        scaf2 = (ymax-ymin)/SCALE_FAC2;
        scaf2 = MAX(scaf2, (zmax-zmin)/SCALE_FAC2);
        }
    else if (slice_axis == 2)
        {
        start_plane = ymin;
        end_plane = ymax;
        printf("\n\tymin: %f, ymax: %f, ydist: %f\n\n", ymin, ymax, ymax-ymin);
        scaf2 = (xmax-xmin)/SCALE_FAC2;
        scaf2 = MAX(scaf2, (zmax-zmin)/SCALE_FAC2);
        }
    else
        {
        start_plane = zmin;
        end_plane = zmax;
        printf("\n\tzmin: %f, zmax: %f, zdist: %f\n\n", zmin, zmax, zmax-zmin);
        scaf2 = (xmax-xmin)/SCALE_FAC2;
        scaf2 = MAX(scaf2, (ymax-ymin)/SCALE_FAC2);
        }

    subpixel(TRUE);
    ortho2(-scaf2,scaf2,-scaf2,scaf2);

    printf("Enter position of start slice [e.g.: %f]: ", start_plane);
    scanf("%f", &start_plane);

    printf("Enter position of end slice [e.g.: %f]: ", end_plane);
    scanf("%f", &end_plane);

    printf("Enter number of slices [e.g.: 100]: ");
    scanf("%d", &slice_number);

    printf("Enter bitmap data filename [e.g.: foo.raw]: ");
    scanf("%s", filename);

    if((binfil=open(filename,O_WRONLY|O_CREAT, 0644))==(-1))
        {
        printf("Sorry.... Can't open file %s!\n", filename);
        exit();
        }

    printf("\n\tFile: %s\n", filename);
    delta_plane = (end_plane - start_plane) / (slice_number -1);
    printf("\tInter slice spacing is %f units (mm,cm,inches, ...)\n",
            delta_plane);
    pixel_size = 2.*scaf2/512.;
    printf("\tPixel Size is %f units (mm,cm,inches, ...) \n\n", pixel_size);

    /* Write header info */
    write(binfil, &slice_number, sizeof(int));
    write(binfil, &delta_plane, sizeof(float));
    write(binfil, &pixel_size, sizeof(float));
    for(i=0;i<500;i++) write(binfil, &nullchr, sizeof(char));

    for(l=0;l<slice_number;l++)
        {
```

```
plane = start_plane + ((float)l*delta_plane);
color(BLACK);
clear();
color(WHITE);
printf("\tSlice %3d at position % f \n", l+1, plane);
fflush(stdout);
for(i=0; i<nb_triang; i++)
    {
    /* Check if polygon is flat with cut plane */
    inplane = TRUE;
    for(j=0; j<poly_vert[i]; j++)
        {
        if(slice_axis == 1) z1 = polygons[i][j][0];
        else if (slice_axis == 2) z1 = polygons[i][j][1];
        else z1 = polygons[i][j][2];
        if (z1 != plane) inplane = FALSE;
        }
    if(inplane)
        {
        color(GREEN);
        concave(TRUE);
        bgnpolygon();
        for(j=0; j<poly_vert[i]; j++)
            {
            if(slice_axis == 1)
                {
                vector[0] = polygons[i][j][1];
                vector[1] = polygons[i][j][2];
                }
            else if (slice_axis == 2)
                {
                vector[0] = polygons[i][j][2];
                vector[1] = polygons[i][j][0];
                }
            else
                {
                vector[0] = polygons[i][j][0];
                vector[1] = polygons[i][j][1];
                }
            v2f(vector);
            }
        endpolygon();
        color(WHITE);
        }
    else for(j=0; j<poly_vert[i]; j++)
        {
        if(slice_axis == 1)
            {
            z1 = polygons[i][j][0];
            x1 = polygons[i][j][1];
            y1 = polygons[i][j][2];
            z2 = polygons[i][(j+1)%poly_vert[i]][0];
            x2 = polygons[i][(j+1)%poly_vert[i]][1];
            y2 = polygons[i][(j+1)%poly_vert[i]][2];
            }
        else if (slice_axis == 2)
            {
            y1 = polygons[i][j][0];
            z1 = polygons[i][j][1];
            x1 = polygons[i][j][2];
            y2 = polygons[i][(j+1)%poly_vert[i]][0];
            z2 = polygons[i][(j+1)%poly_vert[i]][1];
            x2 = polygons[i][(j+1)%poly_vert[i]][2];
            }
        else
            {
            x1 = polygons[i][j][0];
            y1 = polygons[i][j][1];
            z1 = polygons[i][j][2];
            x2 = polygons[i][(j+1)%poly_vert[i]][0];
            y2 = polygons[i][(j+1)%poly_vert[i]][1];
            z2 = polygons[i][(j+1)%poly_vert[i]][2];
            }

        if((z1!=z2) &&
        ((plane<=z1 && plane>=z2) || (plane>=z1 && plane<=z2)))
            {
            a = (plane - z1) / (z2 - z1);
            x = a*(x2 - x1) + x1;
            y = a*(y2 - y1) + y1;
```

```
                    xc[index] = x;
                    yc[index] = y;
                    index++;
                    }
                }
            if (index > 0)
                {
                if(index%2 == 0) /* normal case */
                    {
                    for(j=0; j<index; j++)
                        for(k=j+1;k<index;k++)
                            if(xc[k] < xc[j])
                                {
                                tmp = xc[j];
                                xc[j] = xc[k];
                                xc[k] = tmp;
                                tmp = yc[j];
                                yc[j] = yc[k];
                                yc[k] = tmp;
                                }
                    for(j=0; j<index; j++)
                        for(k=j+1;k<index;k++)
                            if(yc[k] < yc[j])
                                {
                                tmp = yc[j];
                                yc[j] = yc[k];
                                yc[k] = tmp;
                                tmp = xc[j];
                                xc[j] = xc[k];
                                xc[k] = tmp;
                                }
                    for(j=0; j<index; j+=2)
                        {
                        bgnline();
                        vector[0] = xc[j];
                        vector[1] = yc[j];
                        v2f(vector);
                        vector[0] = xc[j+1];
                        vector[1] = yc[j+1];
                        v2f(vector);
                        endline();
                        }
                    }
                else
                    {
                    /* Unidentified Case */
                    printf("Unidentified Case \007\n");
                    printf("Index = %d \n", index);
                    for(j=0; j<poly_vert[i]; j++)
                        {
                        for(k=0;k<3;k++) printf("%f ", polygons[i][j][k]);
                        printf("\n");
                        }
                    for(k=0;k<index;k++) printf("%f %f\n", xc[k], yc[k]);
                    }
                }
            index = 0;
            }
        fillin();
        write_bitmap(binfil);
        }
    close(binfil);
    }

write_bitmap(binfil)
int binfil;
    {
    int i, j, k, index=0;
    char bitmap[512][64];
    unsigned long image[512][512];

    lrectread(0, 0, 511, 511, (unsigned long *)&image[0][0]);
    for(i=0;i<512;i++)
        {
        index=0;
        for(j=0;j<64;j++)
            {
            bitmap[i][j] = 0;
            for(k=0;k<8;k++)
                {
                if(image[i][index] != 0)
```

```
                    {
                    bitmap[i][j] += (char)pow(2., (float)(7-k));
                    }
                index++;}}}
    write(binfil, bitmap, 512*64*sizeof(char));
    }

fillin()
    {
    char solid=TRUE, found=FALSE;
    int i, j;
    unsigned long image[512][512];

    lrectread(0, 0, 511, 511, (unsigned long *)&image[0][0]);
    FloodFill4(1,1, BLACK, BLUE, &image[0][0]);
    /* little fix at the corners */
    image[0][0] = BLUE;
    image[511][0] = BLUE;
    image[0][511] = BLUE;
    image[511][511] = BLUE;
    for(i=256;i<512;i++)
        for(j=0;j<512;j++)
            {
            if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
            else if(image[i][j] == GREEN) solid = FALSE;
            else if(image[i][j] == BLACK)
                {
                if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
                else FloodFill4(i,j, BLACK, RED, &image[0][0]);
                }
            }
    for(i=255;i>=0;i--)
        for(j=0;j<512;j++)
            {
            if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
            else if(image[i][j] == GREEN) solid = FALSE;
            else if(image[i][j] == BLACK)
                {
                if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
                else FloodFill4(i,j, BLACK, RED, &image[0][0]);
                }
            }
    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            if(image[i][j] == RED)
                {
                image[i][j] = BLACK;
                found = TRUE;
                }
    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            {
            if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
            else if(image[i][j] == GREEN) solid = FALSE;
            else if(image[i][j] == BLACK)
                {
                if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
                else FloodFill4(i,j, BLACK, RED, &image[0][0]);
                }
            }
    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            if(image[i][j] == RED)
                {
                image[i][j] = BLACK;
                found = TRUE;
                }
    if(found)
        {
    for(i=0;i<512;i++)
        for(j=511;j>=0;j--)
            {
            if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
            else if(image[i][j] == GREEN) solid = FALSE;
            else if(image[i][j] == BLACK)
                {
                if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
                else FloodFill4(i,j, BLACK, RED, &image[0][0]);
                }
            }
        found = FALSE;
```

```
    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            if(image[i][j] == RED)
                {
                image[i][j] = BLACK;
                found = TRUE;
                }
    }
if(found)
    {
for(i=511;i>=0;i--)
    for(j=0;j<512;j++)
        {
        if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
        else if(image[i][j] == GREEN) solid = FALSE;
        else if(image[i][j] == BLACK)
            {
            if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
            else FloodFill4(i,j, BLACK, RED, &image[0][0]);
            }
        }
    found = FALSE;
    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            if(image[i][j] == RED)
                {
                image[i][j] = BLACK;
                found = TRUE;
                }
    }
if(found)
    {
for(i=511;i>=0;i--)
    for(j=511;j>=0;j--)
        {
        if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
        else if(image[i][j] == GREEN) solid = FALSE;
        else if(image[i][j] == BLACK)
            {
            if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
            else FloodFill4(i,j, BLACK, RED, &image[0][0]);
            }
        }
    found = FALSE;
    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            if(image[i][j] == RED)
                {
                image[i][j] = BLACK;
                found = TRUE;
                }
    }
if(found)
    {
for(j=0;j<512;j++)
    for(i=0;i<512;i++)
        {
        if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
        else if(image[i][j] == GREEN) solid = FALSE;
        else if(image[i][j] == BLACK)
            {
            if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
            else FloodFill4(i,j, BLACK, RED, &image[0][0]);
            }
        }
    found = FALSE;
    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            if(image[i][j] == RED)
                {
                image[i][j] = BLACK;
                found = TRUE;
                }
    }
if(found)
    {
for(j=511;j>=0;j--)
    for(i=0;i<512;i++)
        {
        if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
        else if(image[i][j] == GREEN) solid = FALSE;
```

```
            else if(image[i][j] == BLACK)
                {
                if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
                else FloodFill4(i,j, BLACK, RED, &image[0][0]);
                }
            }
        found = FALSE;
        for(i=0;i<512;i++)
            for(j=0;j<512;j++)
                if(image[i][j] == RED)
                    {
                    image[i][j] = BLACK;
                    found = TRUE;
                    }
        }
    if(found)
        {
    for(j=0;j<512;j++)
        for(i=511;i>=0;i--)
            {
            if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
            else if(image[i][j] == GREEN) solid = FALSE;
            else if(image[i][j] == BLACK)
                {
                if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
                else FloodFill4(i,j, BLACK, RED, &image[0][0]);
                }
            }
        found = FALSE;
        for(i=0;i<512;i++)
            for(j=0;j<512;j++)
                if(image[i][j] == RED)
                    {
                    image[i][j] = BLACK;
                    found = TRUE;
                    }
        }
    if(found)
    for(j=511;j>=0;j--)
        for(i=511;i>=0;i--)
            {
            if(image[i][j] == RED || image[i][j] == BLUE) solid = TRUE;
            else if(image[i][j] == GREEN) solid = FALSE;
            else if(image[i][j] == BLACK)
                {
                if(solid) FloodFill4(i,j, BLACK, GREEN, &image[0][0]);
                else FloodFill4(i,j, BLACK, RED, &image[0][0]);
                }
            }

    for(i=0;i<512;i++)
        for(j=0;j<512;j++)
            {
            if(image[i][j] == RED) image[i][j] = BLACK;
            else if(image[i][j] == BLUE) image[i][j] = BLACK;
            else if(image[i][j] == GREEN) image[i][j] = WHITE;
            }
    lrectwrite(0, 0, 511, 511, (unsigned long *)&image[0][0]);
    }

FloodFill4(x, y, oldvalue, newvalue, image)
int x, y;
long oldvalue, newvalue;
unsigned long image[512][512];
    {
    if(image[x][y]==oldvalue)
        {
        image[x][y] = newvalue;
        if((x>0 && x<511) && (y>0 && y<511))
            {
            FloodFill4(x, y-1, oldvalue, newvalue, image);
            FloodFill4(x, y+1, oldvalue, newvalue, image);
            FloodFill4(x-1, y, oldvalue, newvalue, image);
            FloodFill4(x+1, y, oldvalue, newvalue, image);
            }
        }
    }
```

APPENDIX
D

PRINCIPAL AXIS SOFTWARE

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *  principal_axes.c
 *
 *  Description: This program computes principal axes from a set of data
 *               points representing an object in space.  The
 *               orientation of the principal axes is also calulated with
 *               respect to the global reference frame.  This program
 *               takes advantage of functions from Numerical Recipes.
 *
 *  Created by:  Patrick J. Lord        18-May-94
 *
 *  Modified by:
 *
 *  Known Bugs:
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
 *                        Copyright (C) 1994
 *                Massachusetts Institute of Technology
 *                      Cambridge, Massachusetts
 *
 *  This software is subject to change without notice and should not
 *  be construed as a commitment by MIT.  MIT assumes no responsibility
 *  for the use or reliability of its software.
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 */
#include <stdio.h>
#include <math.h>
#include <nr.h>
#include <nrutil.h>

#define MAXPTS      100000

#define MAX(A, B)   ((A) > (B) ? (A) : (B))
#define MIN(A, B)   ((A) < (B) ? (A) : (B))

#define RAD2DEG 180./M_PI
#define SQR(a)  ((a)*(a))
#define ORIENT  3
#define FTOL    1.0e-6
```

```
struct defpoint {
    float x,y,z;
    };

struct defpoint points[MAXPTS];
float result_mat[3][3];
int nb_total;

float orient(x)
float x[];
    {
    int i, j;
    float interm_mat[3][3], orient_mat[3][3], error=0.;
    float rotx_mat[3][3], roty_mat[3][3], rotz_mat[3][3];

    /* Setup X rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotx_mat[1][1] =  cos(x[1]);
    rotx_mat[1][2] =  sin(x[1]);
    rotx_mat[2][1] = -sin(x[1]);
    rotx_mat[2][2] =  cos(x[1]);

    /* Setup Y rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
    roty_mat[0][0] =  cos(x[2]);
    roty_mat[0][2] = -sin(x[2]);
    roty_mat[2][0] =  sin(x[2]);
    roty_mat[2][2] =  cos(x[2]);

    /* Setup Z rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotz_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotz_mat[0][0] =  cos(x[3]);
    rotz_mat[0][1] =  sin(x[3]);
    rotz_mat[1][0] = -sin(x[3]);
    rotz_mat[1][1] =  cos(x[3]);

    /* Compute Full Rotation Matrix */
    mulmatr(rotx_mat, roty_mat, interm_mat, 3, 3, 3);
    mulmatr(interm_mat, rotz_mat, orient_mat, 3, 3, 3);

    /* Compute Principal Axis Orientation error function */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) error += SQR(orient_mat[i][j] - result_mat[i][j]);

    return error;
    }

void mulmatr(a,b,r,n,m,l)
float *a, *b ,*r;
int n, m, l;
    {
    int i, j, k;
    for(i=0;i<n;i++)
        for(j=0;j<l;j++)
            {
            *(r+(i*l+j)) = 0.;
            for(k=0;k<m;k++)
                *(r+(i*l+j)) += *(a+(i*m+k)) * *(b+(k*l+j));
            }
    }

main (argc, argv)
int argc;
char *argv[];
    {
    int i,j, n=3, nrot=0, iter;
    struct defpoint centroid;
    float sum_d, sum_xx, sum_yy, sum_zz, sum_xy, sum_xz, sum_yz;
    float mean_x, mean_y, mean_z, inertia_matrix[3][3], tensor_matrix[3][3];
    float *di, **vi, **ai, *dt, **vt, **at, fret, **xi, p[ORIENT+1];

    if(argc<2)
        {
        printf("\nERROR => Usage: %s inputfilename\n\007\n", argv[0]);
        exit(0);
        }
```

```
readdata(argv[1], points, &nb_total);

/* Find Centroid of Data Points */
centroid.x = centroid.y = centroid.z = 0.;
for(i=0;i<nb_total;i++)
    {
    centroid.x += points[i].x;
    centroid.y += points[i].y;
    centroid.z += points[i].z;
    }
centroid.x /= (float)nb_total;
centroid.y /= (float)nb_total;
centroid.z /= (float)nb_total;
printf("Data Centroid is at % f % f % f\n",
    centroid.x, centroid.y, centroid.z);

/* Center Data Points to Centroid */
for(i=0;i<nb_total;i++)
    {
    points[i].x -= centroid.x;
    points[i].y -= centroid.y;
    points[i].z -= centroid.z;
    }


/* Create the Scatter Matrix */
sum_d=sum_xx=sum_yy=sum_zz=sum_xy=sum_xz=sum_yz=0.0;
for(i=0;i<nb_total;i++)
    {
    sum_xx += SQR(points[i].x);
    sum_yy += SQR(points[i].y);
    sum_zz += SQR(points[i].z);
    sum_xy += points[i].x * points[i].y;
    sum_xz += points[i].x * points[i].z;
    sum_yz += points[i].y * points[i].z;
    sum_d  += SQR(points[i].x) + SQR(points[i].y) + SQR(points[i].z);
    }

inertia_matrix[0][0] = sum_xx;
inertia_matrix[1][1] = sum_yy;
inertia_matrix[2][2] = sum_zz;
inertia_matrix[0][1] = inertia_matrix[1][0] = sum_xy;
inertia_matrix[0][2] = inertia_matrix[2][0] = sum_xz;
inertia_matrix[1][2] = inertia_matrix[2][1] = sum_yz;
printf("\nThe scatter matrix is:\n");
for(i=0;i<3;i++)
    {
    for(j=0;j<3;j++) printf("% f ", inertia_matrix[i][j]);
    printf("\n");
    }

di = vector(1,3);
vi = matrix(1,3,1,3);
ai = convert_matrix(&inertia_matrix[0][0],1,3,1,3);
jacobi(ai,n,di,vi,&nrot);
eigsrt(di,vi,n);
printf("\nNumber of Jacobi rotations: %d\n", nrot);
printf("The Eigenvalues are: % f % f % f\n", di[1], di[2], di[3]);
printf("The Eigenvectors are:\n");
for(i=1;i<=3;i++)
    {
    for(j=1;j<=3;j++) printf("% f ", vi[i][j]);
    printf("\n");
    }

tensor_matrix[0][0] = sum_yy + sum_zz;
tensor_matrix[1][1] = sum_xx + sum_zz;
tensor_matrix[2][2] = sum_yy + sum_xx;
tensor_matrix[0][1] = tensor_matrix[1][0] = -sum_xy;
tensor_matrix[0][2] = tensor_matrix[2][0] = -sum_xz;
tensor_matrix[1][2] = tensor_matrix[2][1] = -sum_yz;
printf("\nThe Tensor matrix is:\n");
for(i=0;i<3;i++)
    {
    for(j=0;j<3;j++) printf("% f ", tensor_matrix[i][j]);
    printf("\n");
    }

nrot = 0;
dt = vector(1,3);
vt = matrix(1,3,1,3);
```

```
    at = convert_matrix(&tensor_matrix[0][0],1,3,1,3);
    jaccbi(at,n,dt,vt,&nrot);
    eigsrt(dt,vt,n);
    printf("\nNumber of Jacobi rotations: %d\n", nrot);
    printf("The Eigenvalues are: % f % f % f\n", dt[1], dt[2], dt[3]);
    printf("The Eigenvectors are:\n");
    for(i=1;i<=3;i++)
        {
        for(j=1;j<=3;j++) printf("% f ", vt[i][j]);
        printf("\n");
        }

    printf("\nVerification: d^2 = % f\n", sum_d);
    for(i=1;i<=3;i++)
        {
        printf("% f = d^2 - % f = % f =  % f\n", di[i], dt[4-i], sum_d-dt[4-i], di[i] -
(sum_d-dt[4-i]));
        }

    /* Start Optimization */
    for(i=1;i<=3;i++)
        if(vt[i][i] < 0.)
            for(j=1;j<=3;j++) vt[j][i] = -vt[j][i];
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) result_mat[i][j] = vt[j+1][i+1];
    for(i=0;i<=ORIENT;i++) p[i] = 0.;
    xi=matrix(1,ORIENT,1,ORIENT);
    for(i=1;i<=ORIENT;i++)
        for(j=1;j<=ORIENT;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
    powell(p,xi,ORIENT,FTOL,&iter,&fret, orient);
    free_matrix(xi, 1, ORIENT, 1, ORIENT);

    /* Print Out Results */
    printf("\nRESULTS\n");
    printf("Optimization Completed in %d Iterations\n",iter);
    printf("Optimum at: ");
    for(i=1;i<=ORIENT;i++) printf("%f deg ", p[i] * RAD2DEG);
    printf("\nMinimum function:  % f\n",fret);

    printf("\nThe Object is rotated at its centroid: % f % f % f\n",
    centroid.x, centroid.y, centroid.z);
    printf("FIRST    by % f degrees along the X axis\n", p[1] * RAD2DEG);
    printf("THEN     by % f degrees along the Y axis\n", p[2] * RAD2DEG);
    printf("and LAST by % f degrees along the Z axis\n\n", p[3] * RAD2DEG);

    free_convert_matrix(ai,1,3,1,3);
    free_vector(di,1,3);
    free_matrix(vi,1,3,1,3);
    free_convert_matrix(at,1,3,1,3);
    free_vector(dt,1,3);
    free_matrix(vt,1,3,1,3);

    exit();
    }

readdata(filename, data, npts)
char filename[];
struct defpoint data[];
int *npts;
    {
    FILE *ascfil;

    if((ascfil=fopen(filename,"r")) == NULL)
        {
        printf("Sorry.... Can't open datafile %s!\n", filename);
        exit();
        }
    *npts = 0;
    while((fscanf(ascfil,"%e %e %e", &data[*npts].x,
                                     &data[*npts].y,
                                     &data[*npts].z)) != EOF &&
                                     *npts<MAXPTS) (*npts)++;
    fclose(ascfil);
    if(*npts==MAXPTS)
        {
        printf("\\007n\nWARNING: Max num of points (MAXPTS) exceeded\n\n\007");
        exit(0);
        }
    }
```

APPENDIX
E

CARTILAGE ESTIMATION SOFTWARE

E.1 JOINT GEOMETRY OPTIMIZATION

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *  best_fit.c
 *
 *  Description: This program calculates the best fit sphere, ellipsoid,
 *               and rotated ellipsoid for a given set of data points
 *               defining the surface of the femoral head or acetabulum.
 *
 *  Created by:  Patrick J. Lord        18-May-94
 *
 *  Modified by:
 *
 *  Known Bugs:
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
 *                      Copyright (C) 1994
 *               Massachusetts Institute of Technology
 *                    Cambridge, Massachusetts
 *
 *  This software is subject to change without notice and should not
 *  be construed as a commitment by MIT.  MIT assumes no responsibility
 *  for the use or reliability of its software.
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <nr.h>
#include <nrutil.h>

#define MAXPTS      20000

#define MAX(A, B)   ((A) > (B) ? (A) : (B))
```

```
#define MIN(A, B)   ((A) < (B) ? (A) : (B))

#define SPHERE_NDIM 4
#define ELLIPS_NDIM 6
#define O_ELLIPS_NDIM   8
#define FTOL    1.0e-6
#define SQR(a)  ((a)*(a))

struct defpoint {
    float x,y,z;
    };

struct defpoint point[MAXPTS];
int nb_total;

float func_sphere(x)
float x[];
    {
    int i;
    float error=0.;

    for(i=0;i<nb_total;i++)
        error += fabs((SQR(point[i].x-x[1]) + SQR(point[i].y-x[2]) +
                  SQR(point[i].z-x[3]))/SQR(x[4]) - 1.0);
    return error;
    }

float func_ellipsoid(x)
float x[];
    {
    int i;
    float error=0.;

    for(i=0;i<nb_total;i++)
        error += fabs(SQR(point[i].x-x[1])/SQR(x[4]) +
                      SQR(point[i].y-x[2])/SQR(x[5]) +
                      SQR(point[i].z-x[3])/SQR(x[6]) - 1.0);
    return error;
    }

float func_oriented_ellipsoid(x)
float x[];
    {
    int i,j;
    float error=0., data_pin[3], data_pout[3];
    float rotx_mat[3][3], roty_mat[3][3];

    /* Setup X rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotx_mat[1][1] =  cos(x[7]);
    rotx_mat[1][2] =  sin(x[7]);
    rotx_mat[2][1] = -sin(x[7]);
    rotx_mat[2][2] =  cos(x[7]);

    /* Setup Y rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
    roty_mat[0][0] =  cos(x[8]);
    roty_mat[0][2] = -sin(x[8]);
    roty_mat[2][0] =  sin(x[8]);
    roty_mat[2][2] =  cos(x[8]);

    for(i=0;i<nb_total;i++)
        {
        /* Translate to center */
        data_pin[0] = point[i].x - x[1];
        data_pin[1] = point[i].y - x[2];
        data_pin[2] = point[i].z - x[3];

        /* Rotate data along Y */
        mulmatr(data_pin, roty_mat, data_pout, 1, 3, 3);

        /* Rotate data along X */
        mulmatr(data_pout, rotx_mat, data_pin, 1, 3, 3);

        /* Translate back out off center */
        data_pin[0] += x[1];
        data_pin[1] += x[2];
        data_pin[2] += x[3];
```

```
        error += fabs(SQR(data_pin[0]-x[1])/SQR(x[4]) +
                      SQR(data_pin[1]-x[2])/SQR(x[5]) +
                      SQR(data_pin[2]-x[3])/SQR(x[6]) - 1.0);
        }
    return error;
    }

void mulmatr(a,b,r,n,m,l)
float *a, *b ,*r;
int n, m, l;
    {
    int i, j, k;
    for(i=0;i<n;i++)
        for(j=0;j<l;j++)
            {
            *(r+(i*l+j)) = 0.;
            for(k=0;k<m;k++)
                *(r+(i*l+j)) += *(a+(i*m+k)) * *(b+(k*l+j));
            }
    }

main ()
    {
    int i, iter, j;
    float fret, **xi;
    float p[O_ELLIPS_NDIM+1];
    time_t t1, t2;

    readdata();

    for(i=0;i<=O_ELLIPS_NDIM;i++) p[i] = 1.0;

    /* Optimize for Sphere */
    xi=matrix(1,SPHERE_NDIM,1,SPHERE_NDIM);
    for(i=1;i<=SPHERE_NDIM;i++)
        for(j=1;j<=SPHERE_NDIM;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
    t1 = time(NULL);
    powell(p,xi,SPHERE_NDIM,FTOL,&iter,&fret, func_sphere);
    t2 = time(NULL);
    printf("\nSphere Iterations: %d in %d seconds\n", iter, t2-t1);
    printf("Minimum at: ");
    for(i=1;i<=SPHERE_NDIM;i++) printf("%f ", p[i]);
    printf("\nMinimum function value = %f \n", fret);
    free_matrix(xi, 1, SPHERE_NDIM, 1, SPHERE_NDIM);

    /* Kludge to fix converging to infinity */
    for(i=0;i<nb_total;i++)
        {
        point[i+nb_total].x = 2.0*p[1] - point[i].x;
        point[i+nb_total].y = 2.0*p[2] - point[i].y;
        point[i+nb_total].z = 2.0*p[3] - point[i].z;
        }
    nb_total *= 2;

    /* Optimize for Ellipsoid */
    p[5] = p[6] = p[4];
    xi=matrix(1,ELLIPS_NDIM,1,ELLIPS_NDIM);
    for(i=1;i<=ELLIPS_NDIM;i++)
        for(j=1;j<=ELLIPS_NDIM;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
    t1 = time(NULL);
    powell(p,xi,ELLIPS_NDIM,FTOL,&iter,&fret, func_ellipsoid);
    t2 = time(NULL);
    printf("\nEllipsoid Iterations: %d in %d seconds\n", iter, t2-t1);
    printf("Minimum at: ");
    for(i=1;i<=ELLIPS_NDIM;i++) printf("%f ", p[i]);
    printf("\nMinimum function value = %f \n", fret);
    free_matrix(xi, 1, ELLIPS_NDIM, 1, ELLIPS_NDIM);

    /* Optimize for Oriented Ellipsoid */
    p[7] = p[8] = p[9] = 0.0;
    xi=matrix(1,O_ELLIPS_NDIM,1,O_ELLIPS_NDIM);
    for(i=1;i<=O_ELLIPS_NDIM;i++)
        for(j=1;j<=O_ELLIPS_NDIM;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
    t1 = time(NULL);
    powell(p,xi,O_ELLIPS_NDIM,FTOL,&iter,&fret, func_oriented_ellipsoid);
    t2 = time(NULL);
    printf("\nOriented Ellipsoid Iterations: %d in %d seconds\n", iter, t2-t1);
    printf("Minimum at: ");
    for(i=1;i<=ELLIPS_NDIM;i++) printf("%f ", p[i]);
    printf("\nwith rotations: ");
    for(i=ELLIPS_NDIM+1;i<=O_ELLIPS_NDIM;i++) printf("%f ", -p[i]*180./M_PI);
```

```
    printf("\nMinimum function value = %f \007\n\n", fret);
    free_matrix(xi, 1, O_ELLIPS_NDIM, 1, O_ELLIPS_NDIM);

    exit();
    }

readdata()
    {
    FILE *ascfil;
    char filename[80];
    int index =0;

    printf("Enter filename: ");
    scanf("%s", filename);
    if((ascfil=fopen(filename,"r")) == NULL)
        {
        printf("Sorry.... Can't open datafile!\n");
        exit();
        }
    while((fscanf(ascfil,"%f %f %f", &point[index].x,
                                     &point[index].y,
                                     &point[index].z)) != EOF) index++;
    fclose(ascfil);
    nb_total = index;
    }
```

E.2 CARTILAGE THICKNESS OPTIMIZATION

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *  estimate_cartilage.c
 *
 *  Description: This program computes the best fit sphere for the contact
 *               sliding surface from data points defining the bone
 *               surfaces of the femoral head and acetabulum.  From this,
 *               cartilage distribution maps are calculated for both
 *               surfaces of the joint.
 *
 *  Created by:  Patrick J. Lord        18-May-94
 *
 *  Modified by:
 *
 *  Known Bugs:
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
 *                      Copyright (C) 1994
 *               Massachusetts Institute of Technology
 *                     Cambridge, Massachusetts
 *
 *  This software is subject to change without notice and should not
 *  be construed as a commitment by MIT.  MIT assumes no responsibility
 *  for the use or reliability of its software.
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <nr.h>
#include <nrutil.h>


#define MAXPTS      10000

#define MAX(A, B)   ((A) > (B) ? (A) : (B))
#define MIN(A, B)   ((A) < (B) ? (A) : (B))

#define SPHERE_NDIM 4
#define ELLIPS_NDIM 6
#define O_ELLIPS_NDIM   8
#define FTOL    1.0e-6
#define SQR(a)  ((a)*(a))
```

```
struct defpoint {
    float x,y,z;
    };

struct defpoint point[MAXPTS];
int nb_total;

float func_sphere(x)
float x[];
    {
    int i;
    float error=0.;

    for(i=0;i<nb_total;i++)
        error += fabs((SQR(point[i].x-x[1]) + SQR(point[i].y-x[2]) +
                    SQR(point[i].z-x[3]))/SQR(x[4]) - 1.0);
    return(error/(float)nb_total);
    }

float func_ellipsoid(x)
float x[];
    {
    int i;
    float error=0.;

    for(i=0;i<nb_total;i++)
        error += fabs(SQR(point[i].x-x[1])/SQR(x[4]) +
                    SQR(point[i].y-x[2])/SQR(x[5]) +
                    SQR(point[i].z-x[3])/SQR(x[6]) - 1.0);
    return(error/(float)nb_total);
    }

float func_oriented_ellipsoid(x)
float x[];
    {
    int i,j;
    float error=0., data_pin[3], data_pout[3];
    float rotx_mat[3][3], roty_mat[3][3];

    /* Setup X rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotx_mat[1][1] =  cos(x[7]);
    rotx_mat[1][2] =  sin(x[7]);
    rotx_mat[2][1] = -sin(x[7]);
    rotx_mat[2][2] =  cos(x[7]);

    /* Setup Y rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
    roty_mat[0][0] =  cos(x[8]);
    roty_mat[0][2] = -sin(x[8]);
    roty_mat[2][0] =  sin(x[8]);
    roty_mat[2][2] =  cos(x[8]);

    for(i=0;i<nb_total;i++)
        {
        /* Translate to center */
        data_pin[0] = point[i].x - x[1];
        data_pin[1] = point[i].y - x[2];
        data_pin[2] = point[i].z - x[3];

        /* Rotate data along Y */
        mulmatr(data_pin, roty_mat, data_pout, 1, 3, 3);

        /* Rotate data along X */
        mulmatr(data_pout, rotx_mat, data_pin, 1, 3, 3);

        /* Translate back out off center */
        data_pin[0] += x[1];
        data_pin[1] += x[2];
        data_pin[2] += x[3];

        error += fabs(SQR(data_pin[0]-x[1])/SQR(x[4]) +
                    SQR(data_pin[1]-x[2])/SQR(x[5]) +
                    SQR(data_pin[2]-x[3])/SQR(x[6]) - 1.0);
        }
    return(error/(float)nb_total);
    }

void mulmatr(a,b,r,n,m,l)
```

```
float *a, *b ,*r;
int n, m, l;
    {
    int i, j, k;
    for(i=0;i<n;i++)
        for(j=0;j<l;j++)
            {
            *(r+(i*l+j)) = 0.;
            for(k=0;k<m;k++)
                *(r+(i*l+j)) += *(a+(i*m+k)) * *(b+(k*l+j));
            }
    }
main ()
    {
    FILE *ascfil;
    char femoral_result[80], acetabulum_result[80];
    struct defpoint point1[MAXPTS], point2[MAXPTS];
    struct defpoint point3[MAXPTS], point4[MAXPTS];
    float p[SPHERE_NDIM], p1[O_ELLIPS_NDIM+1], p2[O_ELLIPS_NDIM+1];
    time_t t1, t2, t3, t4;
    int nb_total1, nb_total2;
    int i, j, iter, npts;
    float fret, **xi;
    float x_orig, y_orig, z_orig, a_radius, b_radius, c_radius;
    float pitch_angle, tilt_angle, rotx_mat[3][3], roty_mat[3][3];
    float distance, mindistance, data_pin[3], data_pout[3];
    int index;
    float xo, yo, zo, radius, dist;


    readdata("FEMORAL", point1, &nb_total1);
    readdata("ACETABULUM", point2, &nb_total2);
    printf("\nEnter RESULT FEMORAL cartilage data file name: ");
    scanf("%s", femoral_result);
    printf("\nEnter RESULT ACETABULUM cartilage data file name: ");
    scanf("%s", acetabulum_result);

    t1 = time(NULL);

    printf("\nOptimizing FEMORAL Geometry\n"); fflush(stdout);
    optimize_geometry(point1, nb_total1, p1);

    printf("\nOptimizing ACETABULUM Geometry\n"); fflush(stdout);
    optimize_geometry(point2, nb_total2, p2);

    printf("\nOptimizing SLIDING SURFACE Geometry\n"); fflush(stdout);

    npts = 100;

    printf("Creating Femoral Ellipsoid\n"); fflush(stdout);
    x_orig =  p1[1];
    y_orig =  p1[2];
    z_orig =  p1[3];
    a_radius = p1[4];
    b_radius = p1[5];
    c_radius = p1[6];
    pitch_angle = -p1[7];
    tilt_angle = -p1[8];

    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotx_mat[1][1] =  cos(pitch_angle);
    rotx_mat[1][2] =  sin(pitch_angle);
    rotx_mat[2][1] = -sin(pitch_angle);
    rotx_mat[2][2] =  cos(pitch_angle);

    for(i=0;i<3;i++)
        for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
    roty_mat[0][0] =  cos(tilt_angle);
    roty_mat[0][2] = -sin(tilt_angle);
    roty_mat[2][0] =  sin(tilt_angle);
    roty_mat[2][2] =  cos(tilt_angle);

    make_ellips_data(point3, x_orig, y_orig, z_orig,
                     a_radius, b_radius, c_radius, npts);

    for(i=0;i<nb_total;i++)
        {
        data_pin[0] = point3[i].x - x_orig;
        data_pin[1] = point3[i].y - y_orig;
        data_pin[2] = point3[i].z - z_orig;
```

```
    mulmatr(data_pin, rotx_mat, data_pout, 1, 3, 3);
    mulmatr(data_pout, roty_mat, data_pin, 1, 3, 3);
    point3[i].x = data_pin[0] + x_orig;
    point3[i].y = data_pin[1] + y_orig;
    point3[i].z = data_pin[2] + z_orig;
    }

printf("Creating Acetabulum Ellipsoid\n"); fflush(stdout);
x_orig =  p2[1];
y_orig =  p2[2];
z_orig =  p2[3];
a_radius = p2[4];
b_radius = p2[5];
c_radius = p2[6];
pitch_angle = -p2[7];
tilt_angle = -p2[8];

for(i=0;i<3;i++)
    for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
rotx_mat[1][1] =  cos(pitch_angle);
rotx_mat[1][2] =  sin(pitch_angle);
rotx_mat[2][1] = -sin(pitch_angle);
rotx_mat[2][2] =  cos(pitch_angle);

for(i=0;i<3;i++)
    for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
roty_mat[0][0] =  cos(tilt_angle);
roty_mat[0][2] = -sin(tilt_angle);
roty_mat[2][0] =  sin(tilt_angle);
roty_mat[2][2] =  cos(tilt_angle);

make_ellips_data(point4, x_orig, y_orig, z_orig,
                 a_radius, b_radius, c_radius, npts);

for(i=0;i<nb_total;i++)
    {
    data_pin[0] = point4[i].x - x_orig;
    data_pin[1] = point4[i].y - y_orig;
    data_pin[2] = point4[i].z - z_orig;
    mulmatr(data_pin, rotx_mat, data_pout, 1, 3, 3);
    mulmatr(data_pout, roty_mat, data_pin, 1, 3, 3);
    point4[i].x = data_pin[0] + x_orig;
    point4[i].y = data_pin[1] + y_orig;
    point4[i].z = data_pin[2] + z_orig;
    }

printf("Generating Cartilage Interface Data Points\n"); fflush(stdout);
for(i=0; i<nb_total; i++)
    {
    index=0;
    mindistance = SQR(point4[0].x - point3[i].x) +
                  SQR(point4[0].y - point3[i].y) +
                  SQR(point4[0].z - point3[i].z);
    for(j=0;j<nb_total;j++)
        {
        distance = SQR(point4[j].x - point3[i].x) +
                   SQR(point4[j].y - point3[i].y) +
                   SQR(point4[j].z - point3[i].z);
        if(distance < mindistance)
            {
            mindistance = distance;
            index = j;
            }
        }
    point[i].x = (point3[i].x + point4[index].x)/2.0;
    point[i].y = (point3[i].y + point4[index].y)/2.0;
    point[i].z = (point3[i].z + point4[index].z)/2.0;
    }

printf("Starting Cartilage Sphere Optimization\n"); fflush(stdout);
for(i=0;i<=O_ELLIPS_NDIM;i++) p[i] = 1.0;

/* Optimize for Sphere */
xi=matrix(1,SPHERE_NDIM,1,SPHERE_NDIM);
for(i=1;i<=SPHERE_NDIM;i++)
    for(j=1;j<=SPHERE_NDIM;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
t3 = time(NULL);
powell(p,xi,SPHERE_NDIM,FTOL,&iter,&fret, func_sphere);
t4 = time(NULL);
printf("\nSphere Iterations: %d in %d seconds\n", iter, t4-t3);
printf("Minimum at: ");
```

```
    for(i=1;i<=SPHERE_NDIM;i++) printf("%f ", p[i]);
    printf("\nMinimum function value = %f \n", fret);
    free_matrix(xi, 1, SPHERE_NDIM, 1, SPHERE_NDIM);

    xo = p[1];
    yo = p[2];
    zo = p[3];
    radius = p[4];
    printf("\nSliding Sphere Results: \n");
    printf("Optimum radius: %f %f %f\n",radius,radius-p1[4],radius-p2[4]);
    printf("Sphere  origin: %f %f %f\n", xo, yo, zo);
    printf("Femoral variation: %f %f %f\n", xo-p1[1], yo-p1[2], zo-p1[3]);
    printf("Acetabulum variation: %f %f %f\n", xo-p2[1], yo-p2[2], zo-p2[3]);

    printf("\nComputing and Writing Femoral Cartilage Result Data ...");
    fflush(stdout);
    if((ascfil=fopen(femoral_result,"w")) == NULL)
        {
        printf("Sorry.... Can't open datafile!\n");
        exit(0);
        }
    fprintf(ascfil,"%d %f %f %f %f\n", nb_total1, xo, yo, -zo, p1[4]);
    for(j=0;j<nb_total1;j++)
        {
        dist = sqrt((point1[j].x-xo)*(point1[j].x-xo) +
                    (point1[j].y-yo)*(point1[j].y-yo) +
                    (point1[j].z-zo)*(point1[j].z-zo));
        fprintf(ascfil,"%.2f  %.2f  %.2f  %.2f\n",
            point1[j].x, point1[j].y, -point1[j].z, radius-dist);
        }
    fclose(ascfil);
    printf(" Done.\nOutput Data has %d points \n", nb_total1);

    printf("\nComputing and Writing Acetabulum Cartilage Result Data ...");
    fflush(stdout);
    if((ascfil=fopen(acetabulum_result,"w")) == NULL)
        {
        printf("Sorry.... Can't open datafile!\n");
        exit(0);
        }
    fprintf(ascfil,"%d %f %f %f %f\n", nb_total2, xo, yo, -zo, p2[4]);
    for(j=0;j<nb_total2;j++)
        {
        dist = sqrt((point2[j].x-xo)*(point2[j].x-xo) +
                    (point2[j].y-yo)*(point2[j].y-yo) +
                    (point2[j].z-zo)*(point2[j].z-zo));
        fprintf(ascfil,"%.2f  %.2f  %.2f  %.2f\n",
            point2[j].x, point2[j].y, -point2[j].z, dist-radius);
        }
    fclose(ascfil);
    printf(" Done.\nOutput Data has %d points \n", nb_total2);

    t2 = time(NULL);
    printf("\nTotal Optimization done in %d seconds\n",t2-t1);
    exit(1);
    }

readdata(text, p, nb_pts)
char text[];
struct defpoint p[];
int *nb_pts;
    {
    FILE *ascfil;
    char filename[80];
    int index =0;

    printf("\nEnter %s data file name: ", text);
    scanf("%s", filename);
    if((ascfil=fopen(filename,"r")) == NULL)
        {
        printf("Sorry.... Can't open datafile!\n");
        exit(0);
        }
    printf("Reading Data ..."); fflush(stdout);
    while((fscanf(ascfil,"%f %f %f", &p[index].x,
                                     &p[index].y,
                                     &p[index].z)) != EOF) index++;
    fclose(ascfil);

    *nb_pts = index;
    printf(" Done.\nInput Data has %d points \n", *nb_pts);
```

```
    }

optimize_geometry(pt, nb_pts, p)
struct defpoint pt[];
int nb_pts;
float p[];
    {
    int i, j, iter;
    float fret, **xi;
    time_t t1, t2;

    nb_total = nb_pts;
    for(i=0;i<nb_total;i++)
        {
        point[i].x = pt[i].x;
        point[i].y = pt[i].y;
        point[i].z = pt[i].z;
        }

    for(i=0;i<=O_ELLIPS_NDIM;i++) p[i] = 1.0;

    /* Optimize for Sphere */
    xi=matrix(1,SPHERE_NDIM,1,SPHERE_NDIM);
    for(i=1;i<=SPHERE_NDIM;i++)
        for(j=1;j<=SPHERE_NDIM;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
    t1 = time(NULL);
    powell(p,xi,SPHERE_NDIM,FTOL,&iter,&fret, func_sphere);
    t2 = time(NULL);
    printf("\nSphere Iterations: %d in %d seconds\n", iter, t2-t1);
    printf("Minimum at: ");
    for(i=1;i<=SPHERE_NDIM;i++) printf("%f ", p[i]);
    printf("\nMinimum function value = %f \n", fret);
    free_matrix(xi, 1, SPHERE_NDIM, 1, SPHERE_NDIM);

    for(i=0;i<nb_total;i++)
        {
        point[i+nb_total].x = 2.0*p[1] - point[i].x;
        point[i+nb_total].y = 2.0*p[2] - point[i].y;
        point[i+nb_total].z = 2.0*p[3] - point[i].z;
        }
    nb_total *= 2;

    /* Optimize for Ellipsoid */
    p[5] = p[6] = p[4];
    xi=matrix(1,ELLIPS_NDIM,1,ELLIPS_NDIM);
    for(i=1;i<=ELLIPS_NDIM;i++)
        for(j=1;j<=ELLIPS_NDIM;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
    t1 = time(NULL);
    powell(p,xi,ELLIPS_NDIM,FTOL,&iter,&fret, func_ellipsoid);
    t2 = time(NULL);
    printf("\nEllipsoid Iterations: %d in %d seconds\n", iter, t2-t1);
    printf("Minimum at: ");
    for(i=1;i<=ELLIPS_NDIM;i++) printf("%f ", p[i]);
    printf("\nMinimum function value = %f \n", fret);
    free_matrix(xi, 1, ELLIPS_NDIM, 1, ELLIPS_NDIM);

    /* Optimize for Oriented Ellipsoid */
    p[7] = p[8] = p[9] = 0.0;
    xi=matrix(1,O_ELLIPS_NDIM,1,O_ELLIPS_NDIM);
    for(i=1;i<=O_ELLIPS_NDIM;i++)
        for(j=1;j<=O_ELLIPS_NDIM;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
    t1 = time(NULL);
    powell(p,xi,O_ELLIPS_NDIM,FTOL,&iter,&fret, func_oriented_ellipsoid);
    t2 = time(NULL);
    printf("\nOriented Ellipsoid Iterations: %d in %d seconds\n", iter, t2-t1);
    printf("Minimum at: ");
    for(i=1;i<=ELLIPS_NDIM;i++) printf("%f ", p[i]);
    printf("\nwith rotations: ");
    for(i=ELLIPS_NDIM+1;i<=O_ELLIPS_NDIM;i++) printf("%f ", -p[i]*180./M_PI);
    printf("\nMinimum function value = %f \007\n\n", fret);
    free_matrix(xi, 1, O_ELLIPS_NDIM, 1, O_ELLIPS_NDIM);
    }

make_ellips_data(p, x_orig, y_orig, z_orig, a_radius, b_radius, c_radius, npts)
struct defpoint p[];
float x_orig, y_orig, z_orig, a_radius, b_radius, c_radius;
int npts;
    {
    int i, j;
    float theta, phi;
```

```
nb_total=0;
for(i=0;i<npts;i++)
    {
    theta = -M_PI/2.0 + ((float)i*M_PI)/(float)npts;
    for(j=0;j<npts;j++)
        {
        phi = ((float)j * 2.0 * M_PI)/(float)npts;
        p[nb_total].x = a_radius*cos(theta)*cos(phi) + x_orig;
        p[nb_total].y = b_radius*cos(theta)*sin(phi) + y_orig;
        p[nb_total].z = c_radius*sin(theta) + z_orig;
        nb_total++;
        }
    }
}
```

APPENDIX
F

OSTEOTOMY SIMULATION SOFTWARE

F.1 JOINT REORIENTATION OPTIMIZATION

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *  osteotomy.c
 *
 *  Description: This program calculates the best possible reorientation
 *               of the femoral head inside the acetabulum, based on
 *               cartilage maps, geometric constraints, mobility
 *               kinematics, and surgery angle correction limitations.
 *
 *  Created by:  Patrick J. Lord        18-May-94
 *
 *  Modified by:
 *
 *  Known Bugs:
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
 *                      Copyright (C) 1994
 *               Massachusetts Institute of Technology
 *                    Cambridge, Massachusetts
 *
 *  This software is subject to change without notice and should not
 *  be construed as a commitment by MIT.  MIT assumes no responsibility
 *  for the use or reliability of its software.
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 */
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <nr.h>
#include <nrutil.h>

#define MAXPTS      6000
```

```
#define MAX(A, B)    ((A) > (B) ? (A) : (B))
#define MIN(A, B)    ((A) < (B) ? (A) : (B))

#define BOUND    2.*M_PI/3.
#define RAD2DEG 180./M_PI
#define OSTEO    3
#define SQR(a)   ((a)*(a))

struct defpoint {
    float x,y,z;
    };

struct defpoint ace_point[MAXPTS], fem_point[MAXPTS], new_point[MAXPTS];
int ace_total, fem_total;
float ftol, flexmin, flexmax, abdumin, abdumax, rotamin, rotamax;
float famaxdist;

float osteo_reor(x)
float x[];
    {
    int i, j;
    float mindist, dist, error=0., bound=1.;

    /* Boundaries on variables */
    if (x[1]<flexmin || x[1]>flexmax ||
        x[2]<abdumin || x[2]>abdumax ||
        x[3]<rotamin || x[3]>rotamax) bound = 1000000.;

    /* Rotate femoral head to new orientation */
    femur_rotate(x[1], x[2], x[3]);

    /* Compute Cartilage Thickness error function */
    for(i=0;i<ace_total;i++)
        {
        mindist = SQR(ace_point[i].x - new_point[0].x) +
                  SQR(ace_point[i].y - new_point[0].y) +
                  SQR(ace_point[i].z - new_point[0].z);
        for(j=1;j<fem_total;j++)
            {
            dist = SQR(ace_point[i].x - new_point[j].x) +
                   SQR(ace_point[i].y - new_point[j].y) +
                   SQR(ace_point[i].z - new_point[j].z);
            if(dist<mindist) mindist = dist;
            }
        error += famaxdist - sqrt(mindist);
        }
    return error*bound;
    }

void femur_rotate(alpha, beta, gamma)
float alpha, beta, gamma;
    {
    int i, j;
    float data_pin[3], data_pout[3];
    float rotx_mat[3][3], roty_mat[3][3], rotz_mat[3][3];

    /* Setup X rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotx_mat[1][1] =  cos(alpha);
    rotx_mat[1][2] =  sin(alpha);
    rotx_mat[2][1] = -sin(alpha);
    rotx_mat[2][2] =  cos(alpha);

    /* Setup Y rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
    roty_mat[0][0] =  cos(beta);
    roty_mat[0][2] = -sin(beta);
    roty_mat[2][0] =  sin(beta);
    roty_mat[2][2] =  cos(beta);

    /* Setup Z rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotz_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotz_mat[0][0] =  cos(gamma);
    rotz_mat[0][1] =  sin(gamma);
    rotz_mat[1][0] = -sin(gamma);
    rotz_mat[1][1] =  cos(gamma);

    /* Rotate femoral head to new orientation */
```

```
    for(i=0;i<fem_total;i++)
        {
        data_pin[0] = fem_point[i].x;
        data_pin[1] = fem_point[i].y;
        data_pin[2] = fem_point[i].z;

        /* Rotate data along X */
        mulmatr(data_pin, rotx_mat, data_pout, 1, 3, 3);

        /* Rotate data along Y */
        mulmatr(data_pout, roty_mat, data_pin, 1, 3, 3);

        /* Rotate data along Z */
        mulmatr(data_pin, rotz_mat, data_pout, 1, 3, 3);

        new_point[i].x = data_pout[0];
        new_point[i].y = data_pout[1];
        new_point[i].z = data_pout[2];
        }
    }

void mulmatr(a,b,r,n,m,l)
float *a, *b ,*r;
int n, m, l;
    {
    int i, j, k;
    for(i=0;i<n;i++)
        for(j=0;j<l;j++)
            {
            *(r+(i*l+j)) = 0.;
            for(k=0;k<m;k++)
                *(r+(i*l+j)) += *(a+(i*m+k)) * *(b+(k*l+j));
            }
    }

main (argc, argv)
int argc;
char *argv[];
    {
    FILE *ascfil;
    char dummy[80], femfilename[40], acefilename[40];
    int i, iter, j;
    float fret, **xi;
    float p[OSTEO+1], amaxdist, fmaxdist, dist, mindist;
    float omint, omaxt, oavet, fmint, fmaxt, favet;
    time_t t1, t2;

    if(argc<2)
        {
        printf("\nERROR => Usage: %s inputfilename\n\007\n", argv[0]);
        exit(0);
        }

    /* Read Input Information */
    if((ascfil=fopen(argv[1],"r")) == NULL)
        {
        printf("Sorry.... Can't open INPUT datafile!\n");
        exit();
        }
    fscanf(ascfil,"%s %s", dummy, femfilename);
    fscanf(ascfil,"%s %s", dummy, acefilename);
    fscanf(ascfil,"%s %e", dummy, &ftol);
    fscanf(ascfil,"%s %f %f", dummy, &flexmin, &flexmax);
    fscanf(ascfil,"%s %f %f", dummy, &abdumin, &abdumax);
    fscanf(ascfil,"%s %f %f", dummy, &rotamin, &rotamax);
    fclose(ascfil);

    /* Read Femoral and Acetabular Data */
    readdata(femfilename, fem_point, &fem_total);
    readdata(acefilename, ace_point, &ace_total);

    printf("\n   OSTEOTOMY OPTIMIZATION ALGORITHM\n\n");
    printf("Femoral File   : %s  => %4d points.\n", femfilename, fem_total);
    printf("Acetabulum File: %s  => %4d points.\n", acefilename, ace_total);
    printf("Optimization Tolerance: %e\n", ftol);
    printf("Boundaries in Flexion  : % 5.1f to % 5.1f deg\n", flexmin, flexmax);
    printf("Boundaries in Abduction: % 5.1f to % 5.1f deg\n", abdumin, abdumax);
    printf("Boundaries in Rotation : % 5.1f to % 5.1f deg\n", rotamin, rotamax);

    /* Convert to RADIANS */
    flexmin /= RAD2DEG;
```

```
flexmax /= RAD2DEG;
abdumin /= RAD2DEG;
abdumax /= RAD2DEG;
rotamin /= RAD2DEG;
rotamax /= RAD2DEG;

printf("\nORIGINAL EVALUATION\n");
/* Compute Maximum Possible Distance Between Data Sets */
amaxdist = SQR(ace_point[0].x) + SQR(ace_point[0].y) + SQR(ace_point[0].z);
for(i=1;i<ace_total;i++)
    {
    dist = SQR(ace_point[i].x) + SQR(ace_point[i].y) + SQR(ace_point[i].z);
    if(dist>amaxdist) amaxdist = dist;
    }

fmaxdist = SQR(fem_point[0].x) + SQR(fem_point[0].y) + SQR(fem_point[0].z);
for(i=1;i<fem_total;i++)
    {
    dist = SQR(fem_point[i].x) + SQR(fem_point[i].y) + SQR(fem_point[i].z);
    if(dist<fmaxdist) fmaxdist = dist;
    }
famaxdist = 2.*(sqrt(amaxdist) - sqrt(fmaxdist));
printf("Max Interdistance is   : %.3f mm\n", famaxdist);

/* Compute Original Cartilage Thickness */
omint = SQR(ace_point[0].x - fem_point[0].x) +
        SQR(ace_point[0].y - fem_point[0].y) +
        SQR(ace_point[0].z - fem_point[0].z);
omaxt = oavet = 0.;
for(i=0;i<ace_total;i++)
    {
    mindist = SQR(ace_point[i].x - fem_point[0].x) +
              SQR(ace_point[i].y - fem_point[0].y) +
              SQR(ace_point[i].z - fem_point[0].z);
    for(j=1;j<fem_total;j++)
        {
        dist = SQR(ace_point[i].x - fem_point[j].x) +
               SQR(ace_point[i].y - fem_point[j].y) +
               SQR(ace_point[i].z - fem_point[j].z);
        if(dist<mindist) mindist = dist;
        }
    if(omint>mindist) omint = mindist;
    if(omaxt<mindist) omaxt = mindist;
    oavet += sqrt(mindist);
    }
omint = sqrt(omint);
omaxt = sqrt(omaxt);
oavet/= ace_total;
printf("Cartilage Thickness: Min %.3f mm, Max %.3f mm, Ave %.3f mm.\n",
        omint, omaxt, oavet);

fflush(stdout);
/* Start Optimization */
for(i=0;i<=OSTEO;i++) p[i] = 0.;
xi=matrix(1,OSTEO,1,OSTEO);
for(i=1;i<=OSTEO;i++)
    for(j=1;j<=OSTEO;j++) xi[i][j]=(i == j ? 1.0 : 0.0);
t1 = time(NULL);
powell(p,xi,OSTEO,ftol,&iter,&fret, osteo_reor);
t2 = time(NULL);
free_matrix(xi, 1, OSTEO, 1, OSTEO);

/* Print Out Results */
printf("\n\007RESULTS\n");
printf("Optimization Completed: %d Iterations in %d seconds\n",iter,t2-t1);
printf("Optimum at: ");
for(i=1;i<=OSTEO;i++) printf("%f deg ", p[i] * RAD2DEG);
printf("\nMinimum function:  % f (% f) \n",fret,fret/(float)ace_total);

printf("\nFINAL EVALUATION\n");
/* Rotate Femoral Head as per Optimized Osteotomy */
femur_rotate(p[1], p[2], p[3]);

/* Compute Final Cartilage Thickness */
fmint = SQR(ace_point[0].x - new_point[0].x) +
        SQR(ace_point[0].y - new_point[0].y) +
        SQR(ace_point[0].z - new_point[0].z);
fmaxt = favet = 0.;
for(i=0;i<ace_total;i++)
    {
    mindist = SQR(ace_point[i].x - new_point[0].x) +
```

```
                SQR(ace_point[i].y - new_point[0].y) +
                SQR(ace_point[i].z - new_point[0].z);
        for(j=1;j<fem_total;j++)
            {
            dist = SQR(ace_point[i].x - new_point[j].x) +
                   SQR(ace_point[i].y - new_point[j].y) +
                   SQR(ace_point[i].z - new_point[j].z);
            if(dist<mindist) mindist = dist;
            }
        if(fmint>mindist) fmint = mindist;
        if(fmaxt<mindist) fmaxt = mindist;
        favet += sqrt(mindist);
        }
    fmint = sqrt(fmint);
    fmaxt = sqrt(fmaxt);
    favet/= ace_total;
    printf("Cartilage Thickness: Min %.3f mm, Max %.3f mm, Ave %.3f mm.\n",
            fmint, fmaxt, favet);

    /* Compute Improvements */
    printf("\nIMPROVEMENTS\n");
    printf("Thickness Change: Min %.3f %%, Max %.3f %%, Ave %.3f %%.\n",
            ((fmint-omint)/omint)*100., ((fmaxt-omaxt)/omaxt)*100.,
            ((favet-oavet)/oavet)*100.);

    exit(1);
    }

readdata(filename, data, npts)
char filename[];
struct defpoint data[];
int *npts;
    {
    FILE *ascfil;

    if((ascfil=fopen(filename,"r")) == NULL)
        {
        printf("Sorry.... Can't open datafile %s!\n", filename);
        exit();
        }
    *npts = 0;
    while((fscanf(ascfil,"%e %e %e", &data[*npts].x,
                                     &data[*npts].y,
                                     &data[*npts].z)) != EOF) (*npts)++;
    fclose(ascfil);
    }
```

F.2 OSTEOTOMY WEDGE CALCULATION

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *  osteotomy_cut.c
 *
 *  Description: This program uses the information from the optimal
 *               reorientation algorithm to compute the intertrochanteric
 *               ostetomy wedge.  The wedge is optimize for minimal bone
 *               loss, and the databases are updated to reflect the
 *               the effects of the surgery simulation.
 *
 *  Created by:  Patrick J. Lord        18-May-94
 *
 *  Modified by:
 *
 *  Known Bugs:
 *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
 *                      Copyright (C) 1994
 *               Massachusetts Institute of Technology
 *                    Cambridge, Massachusetts
 *
 *  This software is subject to change without notice and should not
 *  be construed as a commitment by MIT.  MIT assumes no responsibility
 *  for the use or reliability of its software.
 *
```

```
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 */
#include <stdio.h>
#include <math.h>
#include <fcntl.h>

#define MAX(A, B)    ((A) > (B) ? (A) : (B))
#define MIN(A, B)    ((A) < (B) ? (A) : (B))

#define RAD2DEG (float)(180./M_PI)
#define TRUE    1
#define FALSE   !TRUE

main(argc, argv)
int argc;
char *argv[];
    {
    int i, j, k, index;
    float z_planar, xr, yr, zr;
    float rotx_mat[3][3], roty_mat[3][3], rotz_mat[3][3];
    float data_in[3], data_out[3], mean[3];

    float dist, maxdist;

    int nb_nodes, newnb_nodes;
    float *nodes, *rnodes, *new_nodes, offset[3];
    int *triangs, *utriangs, *ltriangs, *wtriangs, *ftriangs, *frtriangs;
    int nb_triangs, unb_triangs, lnb_triangs, wnb_triangs, fnb_triangs;

    if(argc<2)
        {
        printf("\nERROR => Usage: %s inputfilename\n\007\n", argv[0]);
        exit(0);
        }

    readdata(argv[1], &nodes, &triangs, &nb_nodes, &nb_triangs);

    printf("Enter a vertical axis for the planar cut: ");
    scanf("%f", &z_planar);

    if((utriangs = (int *)calloc(nb_triangs*3, sizeof(int))) == NULL)
        {
        printf("\nERROR Memory allocation\n\n");
        exit(0);
        }
    if((ltriangs = (int *)calloc(nb_triangs*3, sizeof(int))) == NULL)
        {
        printf("\nERROR Memory allocation\n\n");
        exit(0);
        }
    unb_triangs = lnb_triangs = 0;

    /* Perform Planar Cut */
    pcut(&nodes, &nb_nodes,
triangs,nb_triangs,utriangs,&unb_triangs,ltriangs,&lnb_triangs,z_planar);

    /* Enter Osteotomy Correction Angles */
    printf("Enter Osteotomy Correction Angles [xr, yr, zr]: ");
    scanf("%f %f %f", &xr, &yr, &zr);
    xr /= RAD2DEG;
    yr /= RAD2DEG;
    zr /= RAD2DEG;

    /* Setup X rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotx_mat[1][1] =  cos(xr);
    rotx_mat[1][2] =  sin(xr);
    rotx_mat[2][1] = -sin(xr);
    rotx_mat[2][2] =  cos(xr);

    /* Setup Y rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
    roty_mat[0][0] =  cos(yr);
    roty_mat[0][2] = -sin(yr);
    roty_mat[2][0] =  sin(yr);
    roty_mat[2][2] =  cos(yr);

    /* Setup Z rotation matrix */
```

```
for(i=0;i<3;i++)
    for(j=0;j<3;j++) rotz_mat[i][j]=(i == j ? 1.0 : 0.0);
rotz_mat[0][0] =  cos(zr);
rotz_mat[0][1] =  sin(zr);
rotz_mat[1][0] = -sin(zr);
rotz_mat[1][1] =  cos(zr);

if((wtriangs = (int *)calloc(nb_triangs*3, sizeof(int))) == NULL)
    {
    printf("\nERROR Memory allocation\n\n");
    exit(0);
    }
if((ftriangs = (int *)calloc(nb_triangs*3, sizeof(int))) == NULL)
    {
    printf("\nERROR Memory allocation\n\n");
    exit(0);
    }
if((frtriangs = (int *)calloc(nb_triangs*3, sizeof(int))) == NULL)
    {
    printf("\nERROR Memory allocation\n\n");
    exit(0);
    }
wnb_triangs = fnb_triangs = 0;

if((rnodes = (float *)calloc(nb_nodes*3, sizeof(float))) == NULL)
    {
    printf("\nERROR Memory allocation\n\n");
    exit(0);
    }
for(i=0; i<nb_nodes; i++)
    for(j=0; j<3; j++) *(rnodes+i*3+j) = *(nodes+i*3+j);

index = 0;
for(i=0; i<3; i++) mean[i] = 0.;
for(i=0; i<nb_nodes; i++)
    {
    if(*(nodes+i*3+2) == z_planar)
        {
        for(j=0; j<3; j++) mean[j] += *(nodes+i*3+j);
        index++;
        }
    }
for(i=0; i<3; i++) mean[i] /= (float)index;

/* Find on Planar Cut Min and Max */
maxdist = 0.;
for(i=0; i<nb_nodes; i++)
    {
    if(*(nodes+i*3+2) == z_planar)
        {
        dist = 0.;
        for(j=0; j<3; j++) offset[j] = *(nodes+i*3+j);
        for(k=0; k<nb_nodes; k++)
            {
            if(*(nodes+k*3+2) == z_planar)
                {
                for(j=0; j<3; j++) data_in[j] = *(nodes+k*3+j) - offset[j];
                mulmatr(data_in, rotx_mat, data_out, 1, 3, 3);
                mulmatr(data_out, roty_mat, data_in, 1, 3, 3);
                dist += data_in[2];
                }
            }
        if(dist > maxdist)
            {
            maxdist = dist;
            index = i;
            }
        }
    }

/* Translate & Rotate Nodes */
for(j=0; j<3; j++) offset[j] = *(nodes+index*3+j);
for(i=0; i<nb_nodes; i++)
    {
    for(j=0; j<3; j++) data_in[j] = *(nodes+i*3+j) - offset[j];
    mulmatr(data_in, rotx_mat, data_out, 1, 3, 3);
    mulmatr(data_out, roty_mat, data_in, 1, 3, 3);
    for(j=0; j<3; j++) *(rnodes+i*3+j) = data_in[j] + offset[j];
    }

for(j=0; j<3; j++) data_in[j] = mean[j] - offset[j];
```

```
    mulmatr(data_in, rotx_mat, data_out, 1, 3, 3);
    mulmatr(data_out, roty_mat, data_in, 1, 3, 3);
    for(j=0; j<3; j++) mean[j] = data_in[j] + offset[j];

    for(i=0; i<nb_nodes; i++)
        {
        for(j=0; j<3; j++) data_in[j] = *(rnodes+i*3+j) - mean[j];
        mulmatr(data_in, rotz_mat, data_out, 1, 3, 3);
        for(j=0; j<3; j++) *(rnodes+i*3+j) = data_out[j] + mean[j];
        }

    /* Perform Angle Cut */
    pcut(&rnodes, &nb_nodes,
ltriangs,lnb_triangs,wtriangs,&wnb_triangs,ftriangs,&fnb_triangs,z_planar);

    for(i=0; i<fnb_triangs*3; i++) *(frtriangs+i) = *(ftriangs+i);

    /* Derotate Original Nodes */
    /* Setup X rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotx_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotx_mat[1][1] =  cos(-xr);
    rotx_mat[1][2] =  sin(-xr);
    rotx_mat[2][1] = -sin(-xr);
    rotx_mat[2][2] =  cos(-xr);

    /* Setup Y rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) roty_mat[i][j]=(i == j ? 1.0 : 0.0);
    roty_mat[0][0] =  cos(-yr);
    roty_mat[0][2] = -sin(-yr);
    roty_mat[2][0] =  sin(-yr);
    roty_mat[2][2] =  cos(-yr);

    /* Setup Z rotation matrix */
    for(i=0;i<3;i++)
        for(j=0;j<3;j++) rotz_mat[i][j]=(i == j ? 1.0 : 0.0);
    rotz_mat[0][0] =  cos(-zr);
    rotz_mat[0][1] =  sin(-zr);
    rotz_mat[1][0] = -sin(-zr);
    rotz_mat[1][1] =  cos(-zr);

    nodes = (float *)realloc((void *)nodes, nb_nodes*3*sizeof(float));

    for(i=0; i<nb_nodes; i++)
        {
        for(j=0; j<3; j++) data_in[j] = *(rnodes+i*3+j) - mean[j];
        mulmatr(data_in, rotz_mat, data_out, 1, 3, 3);
        for(j=0; j<3; j++) *(nodes+i*3+j) = data_out[j] + mean[j];
        }

    for(i=0; i<nb_nodes; i++)
        {
        for(j=0; j<3; j++) data_in[j] = *(nodes+i*3+j) - offset[j];
        mulmatr(data_in, roty_mat, data_out, 1, 3, 3);
        mulmatr(data_out, rotx_mat, data_in, 1, 3, 3);
        for(j=0; j<3; j++) *(nodes+i*3+j) = data_in[j] + offset[j];
        }

    if((new_nodes = (float *)calloc(nb_nodes * 3, sizeof(float))) == NULL)
        {
        printf("\nERROR Memory allocation\n\n");
        exit(0);
        }

    printf("Sorting %s: %d patches %d nodes => ",argv[2],unb_triangs,nb_nodes);
    fflush(stdout);
    sort_nodes(unb_triangs, nb_nodes, &newnb_nodes, utriangs, nodes, new_nodes);
    write_output(argv[2], unb_triangs, newnb_nodes, utriangs, new_nodes);
    printf("%d actual nodes\n", newnb_nodes);

    printf("Sorting %s: %d patches %d nodes => ",argv[3],lnb_triangs,nb_nodes);
    fflush(stdout);
    sort_nodes(lnb_triangs, nb_nodes, &newnb_nodes, ltriangs, nodes, new_nodes);
    write_output(argv[3], lnb_triangs, newnb_nodes, ltriangs, new_nodes);
    printf("%d actual nodes\n", newnb_nodes);

    printf("Sorting %s: %d patches %d nodes => ",argv[4],wnb_triangs,nb_nodes);
    fflush(stdout);
    sort_nodes(wnb_triangs, nb_nodes, &newnb_nodes, wtriangs, nodes, new_nodes);
    write_output(argv[4], wnb_triangs, newnb_nodes, wtriangs, new_nodes);
```

```
    printf("%d actual nodes\n", newnb_nodes);

    printf("Sorting %s: %d patches %d nodes => ",argv[5],fnb_triangs,nb_nodes);
    fflush(stdout);
    sort_nodes(fnb_triangs, nb_nodes, &newnb_nodes, ftriangs, nodes, new_nodes);
    write_output(argv[5], fnb_triangs, newnb_nodes, ftriangs, new_nodes);
    printf("%d actual nodes\n", newnb_nodes);

    printf("Sorting %s: %d patches %d nodes => ",argv[6],fnb_triangs,nb_nodes);
    fflush(stdout);
    sort_nodes(fnb_triangs, nb_nodes, &newnb_nodes,frtriangs,rnodes, new_nodes);
    write_output(argv[6], fnb_triangs, newnb_nodes, frtriangs, new_nodes);
    printf("%d actual nodes\n", newnb_nodes);

    cfree(new_nodes);
    cfree(rnodes);
    cfree(frtriangs);
    cfree(ftriangs);
    cfree(wtriangs);
    cfree(ltriangs);
    cfree(utriangs);
    cfree(triangs);
    cfree(nodes);
    exit();
    }

sort_nodes(nb_triangs, nb_nodes, newnb_nodes, triangs, nodes, new_nodes)
int nb_triangs, nb_nodes, *newnb_nodes, *triangs;
float *nodes, *new_nodes;
    {
    char duplicate;
    int i, j, k;

    for(j=0; j<3; j++) *(new_nodes+j) = *(nodes+(*(triangs)-1)*3+j);
    *newnb_nodes = 1;
    for(k=1; k<nb_triangs*3; k++)
        if(*(triangs+k)== *triangs)
            *(triangs+k) = *newnb_nodes+nb_nodes;
    *triangs = *newnb_nodes;

    for(i=1; i<nb_triangs*3; i++)
        {
        duplicate = FALSE;
        j=0;
        if(*(triangs+i)< nb_nodes+1)
            {
            while(!duplicate && j<*newnb_nodes)
                {
                if (*(new_nodes+j*3+0) == *(nodes+(*(triangs+i)-1)*3+0) &&
                    *(new_nodes+j*3+1) == *(nodes+(*(triangs+i)-1)*3+1) &&
                    *(new_nodes+j*3+2) == *(nodes+(*(triangs+i)-1)*3+2))
                        {
                        duplicate = TRUE;
                        for(k=i+1; k<nb_triangs*3; k++)
                            if(*(triangs+k)== *(triangs+i))
                                *(triangs+k) = j+1+nb_nodes;
                        *(triangs+i) = j+1;
                        }
                j++;
                }
            if(!duplicate)
                {
                for(j=0; j<3; j++)
                    *(new_nodes+(*newnb_nodes)*3+j) =
                        *(nodes+(*(triangs+i)-1)*3+j);
                (*newnb_nodes)++;
                for(k=i+1; k<nb_triangs*3; k++)
                    if(*(triangs+k)== *(triangs+i))
                        *(triangs+k) = *newnb_nodes+nb_nodes;
                *(triangs+i) = *newnb_nodes;
                }
            }
        }
    for(i=1; i<nb_triangs*3; i++)
        if(*(triangs+i)>nb_nodes) *(triangs+i) -=nb_nodes;
    }

void mulmatr(a,b,r,n,m,l)
float *a, *b ,*r;
int n, m, l;
    {
```

```
    int i, j, k;
    for(i=0;i<n;i++)
        for(j=0;j<l;j++)
            {
            *(r+(i*l+j)) = 0.;
            for(k=0;k<m;k++)
                *(r+(i*l+j)) += *(a+(i*m+k)) * *(b+(k*l+j));
            }
    }

pcut(pnodes,cnb_nodes, triangs,nb_triangs,utriangs,anb_triangs,ltriangs,bnb_triangs,z_planar)
float **pnodes;
int *cnb_nodes, *triangs,nb_triangs,*utriangs,*anb_triangs,*ltriangs,*bnb_triangs;
float z_planar;
   {
    int i, j, unb_triangs, lnb_triangs, nb_nodes;
    float alpha, *nodes;

    unb_triangs = *anb_triangs;
    lnb_triangs = *bnb_triangs;
    nb_nodes = *cnb_nodes;
    nodes = *pnodes;
    fflush(stdout);

    /* Perform Planar Cut */
    for(i=0; i<nb_triangs; i++)
        {
        if((*(nodes+(*(triangs+i*3+0)-1)*3+2) >= z_planar &&
            *(nodes+(*(triangs+i*3+1)-1)*3+2) >= z_planar &&
            *(nodes+(*(triangs+i*3+2)-1)*3+2) >= z_planar) ||
            (*(nodes+(*(triangs+i*3+0)-1)*3+2) <= z_planar &&
             *(nodes+(*(triangs+i*3+1)-1)*3+2) <= z_planar &&
             *(nodes+(*(triangs+i*3+2)-1)*3+2) <= z_planar))
            {
            if(*(nodes+(*(triangs+i*3+0)-1)*3+2) >= z_planar &&
              *(nodes+(*(triangs+i*3+1)-1)*3+2) >= z_planar &&
               *(nodes+(*(triangs+i*3+2)-1)*3+2) >= z_planar)
                {
                for(j=0;j<3;j++) *(utriangs+unb_triangs*3+j) = *(triangs+i*3+j);
                unb_triangs++;
                }
            if(*(nodes+(*(triangs+i*3+0)-1)*3+2) <= z_planar &&
               *(nodes+(*(triangs+i*3+1)-1)*3+2) <= z_planar &&
               *(nodes+(*(triangs+i*3+2)-1)*3+2) <= z_planar)
                {
                for(j=0;j<3;j++) *(ltriangs+lnb_triangs*3+j) = *(triangs+i*3+j);
                lnb_triangs++;
                }
            }
        else
            {
            if((*(nodes+(*(triangs+i*3+0)-1)*3+2) - z_planar) *
               (*(nodes+(*(triangs+i*3+1)-1)*3+2) - z_planar) < 0. &&
               (*(nodes+(*(triangs+i*3+2)-1)*3+2) - z_planar) *
               (*(nodes+(*(triangs+i*3+1)-1)*3+2) - z_planar) < 0.)
                {
                nodes = (float *)realloc((void *)nodes,
                        (nb_nodes+2)*3*sizeof(float));
                alpha = (z_planar - *(nodes+(*(triangs+i*3+0)-1)*3+2)) /
                        (*(nodes+(*(triangs+i*3+1)-1)*3+2) -
                        *(nodes+(*(triangs+i*3+0)-1)*3+2));
                *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+0)-1)*3+0) +
                    alpha*(*(nodes+(*(triangs+i*3+1)-1)*3+0) -
                    *(nodes+(*(triangs+i*3+0)-1)*3+0));
                *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+0)-1)*3+1) +
                    alpha*(*(nodes+(*(triangs+i*3+1)-1)*3+1) -
                    *(nodes+(*(triangs+i*3+0)-1)*3+1));
                *(nodes+nb_nodes*3+2) = z_planar;
                nb_nodes++;
                alpha = (z_planar - *(nodes+(*(triangs+i*3+1)-1)*3+2)) /
                        (*(nodes+(*(triangs+i*3+2)-1)*3+2) -
                        *(nodes+(*(triangs+i*3+1)-1)*3+2));
                *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+1)-1)*3+0) +
                    alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+0) -
                    *(nodes+(*(triangs+i*3+1)-1)*3+0));
                *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+1)-1)*3+1) +
                    alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+1) -
                    *(nodes+(*(triangs+i*3+1)-1)*3+1));
                *(nodes+nb_nodes*3+2) = z_planar;
                nb_nodes++;
```

```
        if(*(nodes+(*(triangs+i*3+1)-1)*3+2) >= z_planar)
            {
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+0);
            *(ltriangs+lnb_triangs*3+1) = nb_nodes-1;
            *(ltriangs+lnb_triangs*3+2) = *(triangs+i*3+2);
            lnb_triangs++;
            *(ltriangs+lnb_triangs*3+0) = nb_nodes;
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+2);
            *(ltriangs+lnb_triangs*3+2) = nb_nodes-1;
            lnb_triangs++;
            *(utriangs+unb_triangs*3+0) = nb_nodes-1;
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+2) = nb_nodes;
            unb_triangs++;
            }
        else
            {
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+0);
            *(utriangs+unb_triangs*3+1) = nb_nodes-1;
            *(utriangs+unb_triangs*3+2) = *(triangs+i*3+2);
            unb_triangs++;
            *(utriangs+unb_triangs*3+0) = nb_nodes;
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+2);
            *(utriangs+unb_triangs*3+2) = nb_nodes-1;
            unb_triangs++;
            *(ltriangs+lnb_triangs*3+0) = nb_nodes-1;
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+2) = nb_nodes;
            lnb_triangs++;
            }
        }
    else if((*(nodes+(*(triangs+i*3+0)-1)*3+2) - z_planar) *
            (*(nodes+(*(triangs+i*3+2)-1)*3+2) - z_planar) < 0. &&
            (*(nodes+(*(triangs+i*3+1)-1)*3+2) - z_planar) *
            (*(nodes+(*(triangs+i*3+2)-1)*3+2) - z_planar) < 0.)
        {
        nodes = (float *)realloc((void *)nodes,
                (nb_nodes+2)*3*sizeof(float));
        alpha = (z_planar - *(nodes+(*(triangs+i*3+1)-1)*3+2)) /
                (*(nodes+(*(triangs+i*3+2)-1)*3+2) -
                *(nodes+(*(triangs+i*3+1)-1)*3+2));
        *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+1)-1)*3+0) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+0) -
            *(nodes+(*(triangs+i*3+1)-1)*3+0));
        *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+1)-1)*3+1) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+1) -
            *(nodes+(*(triangs+i*3+1)-1)*3+1));
        *(nodes+nb_nodes*3+2) = z_planar;
        nb_nodes++;
        alpha = (z_planar - *(nodes+(*(triangs+i*3+0)-1)*3+2)) /
                (*(nodes+(*(triangs+i*3+2)-1)*3+2) -
                *(nodes+(*(triangs+i*3+0)-1)*3+2));
        *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+0)-1)*3+0) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+0) -
            *(nodes+(*(triangs+i*3+0)-1)*3+0));
        *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+0)-1)*3+1) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+1) -
            *(nodes+(*(triangs+i*3+0)-1)*3+1));
        *(nodes+nb_nodes*3+2) = z_planar;
        nb_nodes++;

        if(*(nodes+(*(triangs+i*3+2)-1)*3+2) <= z_planar)
            {
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+0);
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+2) = nb_nodes;
            unb_triangs++;
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+1) = nb_nodes-1;
            *(utriangs+unb_triangs*3+2) = nb_nodes;
            unb_triangs++;
            *(ltriangs+lnb_triangs*3+0) = nb_nodes-1;
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+2);
            *(ltriangs+lnb_triangs*3+2) = nb_nodes;
            lnb_triangs++;
            }
        else {
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+0);
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+2) = nb_nodes;
            lnb_triangs++;
```

```
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+1) = nb_nodes-1;
            *(ltriangs+lnb_triangs*3+2) = nb_nodes;
            lnb_triangs++;
            *(utriangs+unb_triangs*3+0) = nb_nodes-1;
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+2);
            *(utriangs+unb_triangs*3+2) = nb_nodes;
            unb_triangs++;
            }
        }
    else if((*(nodes+(*(triangs+i*3+1)-1)*3+2) - z_planar) *
            (*(nodes+(*(triangs+i*3+0)-1)*3+2) - z_planar) < 0. &&
            (*(nodes+(*(triangs+i*3+2)-1)*3+2) - z_planar) *
            (*(nodes+(*(triangs+i*3+0)-1)*3+2) - z_planar) < 0.)
        {
        nodes = (float *)realloc((void *)nodes,
                (nb_nodes+2)*3*sizeof(float));
        alpha = (z_planar - *(nodes+(*(triangs+i*3+0)-1)*3+2)) /
                (*(nodes+(*(triangs+i*3+1)-1)*3+2) -
                *(nodes+(*(triangs+i*3+0)-1)*3+2));
        *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+0)-1)*3+0) +
            alpha*(*(nodes+(*(triangs+i*3+1)-1)*3+0) -
            *(nodes+(*(triangs+i*3+0)-1)*3+0));
        *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+0)-1)*3+1) +
            alpha*(*(nodes+(*(triangs+i*3+1)-1)*3+1) -
            *(nodes+(*(triangs+i*3+0)-1)*3+1));
        *(nodes+nb_nodes*3+2) = z_planar;
        nb_nodes++;
        alpha = (z_planar - *(nodes+(*(triangs+i*3+0)-1)*3+2)) /
                (*(nodes+(*(triangs+i*3+2)-1)*3+2) -
                *(nodes+(*(triangs+i*3+0)-1)*3+2));
        *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+0)-1)*3+0) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+0) -
            *(nodes+(*(triangs+i*3+0)-1)*3+0));
        *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+0)-1)*3+1) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+1) -
            *(nodes+(*(triangs+i*3+0)-1)*3+1));
        *(nodes+nb_nodes*3+2) = z_planar;
        nb_nodes++;

        if(*(nodes+(*(triangs+i*3+0)-1)*3+2) <= z_planar)
            {
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+0);
            *(ltriangs+lnb_triangs*3+1) = nb_nodes-1;
            *(ltriangs+lnb_triangs*3+2) = nb_nodes;
            lnb_triangs++;
            *(utriangs+unb_triangs*3+0) = nb_nodes-1;
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+2) = nb_nodes;
            unb_triangs++;
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+2);
            *(utriangs+unb_triangs*3+2) = nb_nodes;
            unb_triangs++;
            }
        else {
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+0);
            *(utriangs+unb_triangs*3+1) = nb_nodes-1;
            *(utriangs+unb_triangs*3+2) = nb_nodes;
            unb_triangs++;
            *(ltriangs+lnb_triangs*3+0) = nb_nodes-1;
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+2) = nb_nodes;
            lnb_triangs++;
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+2);
            *(ltriangs+lnb_triangs*3+2) = nb_nodes;
            lnb_triangs++;
            }
        }
    else {
        nodes = (float *)realloc((void *)nodes,
                (nb_nodes+1)*3*sizeof(float));
        if(*(nodes+(*(triangs+i*3+0)-1)*3+2) == z_planar)
            {
            alpha = (z_planar - *(nodes+(*(triangs+i*3+1)-1)*3+2)) /
                    (*(nodes+(*(triangs+i*3+2)-1)*3+2) -
                    *(nodes+(*(triangs+i*3+1)-1)*3+2));
            *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+1)-1)*3+0) +
                alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+0) -
                *(nodes+(*(triangs+i*3+1)-1)*3+0));
```

```
        *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+1)-1)*3+1) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+1) -
            *(nodes+(*(triangs+i*3+1)-1)*3+1));
        *(nodes+nb_nodes*3+2) = z_planar;
        nb_nodes++;

        if(*(nodes+(*(triangs+i*3+1)-1)*3+2) > z_planar)
            {
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+0);
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+2) =  nb_nodes;
            unb_triangs++;
            *(ltriangs+lnb_triangs*3+0) =  nb_nodes;
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+2);
            *(ltriangs+lnb_triangs*3+2) = *(triangs+i*3+0);
            lnb_triangs++;
            }
        else {
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+0);
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+2) =  nb_nodes;
            lnb_triangs++;
            *(utriangs+unb_triangs*3+0) =  nb_nodes;
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+2);
            *(utriangs+unb_triangs*3+2) = *(triangs+i*3+0);
            unb_triangs++;
            }
        }
    else if(*(nodes+(*(triangs+i*3+1)-1)*3+2) == z_planar)
        {
        alpha = (z_planar - *(nodes+(*(triangs+i*3+0)-1)*3+2)) /
                (*(nodes+(*(triangs+i*3+2)-1)*3+2) -
                *(nodes+(*(triangs+i*3+0)-1)*3+2));
        *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+0)-1)*3+0) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+0) -
            *(nodes+(*(triangs+i*3+0)-1)*3+0));
        *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+0)-1)*3+1) +
            alpha*(*(nodes+(*(triangs+i*3+2)-1)*3+1) -
            *(nodes+(*(triangs+i*3+0)-1)*3+1));
        *(nodes+nb_nodes*3+2) = z_planar;
        nb_nodes++;

        if(*(nodes+(*(triangs+i*3+0)-1)*3+2) > z_planar)
            {
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+0);
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+2) =  nb_nodes;
            unb_triangs++;
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+2);
            *(ltriangs+lnb_triangs*3+2) =  nb_nodes;
            lnb_triangs++;
            }
        else
            {
            *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+0);
            *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+1);
            *(ltriangs+lnb_triangs*3+2) =  nb_nodes;
            lnb_triangs++;
            *(utriangs+unb_triangs*3+0) = *(triangs+i*3+1);
            *(utriangs+unb_triangs*3+1) = *(triangs+i*3+2);
            *(utriangs+unb_triangs*3+2) =  nb_nodes;
            unb_triangs++;
            }
        }
    else if(*(nodes+(*(triangs+i*3+2)-1)*3+2) == z_planar)
        {
        alpha = (z_planar - *(nodes+(*(triangs+i*3+0)-1)*3+2)) /
                (*(nodes+(*(triangs+i*3+1)-1)*3+2) -
                *(nodes+(*(triangs+i*3+0)-1)*3+2));
        *(nodes+nb_nodes*3+0) = *(nodes+(*(triangs+i*3+0)-1)*3+0) +
            alpha*(*(nodes+(*(triangs+i*3+1)-1)*3+0) -
            *(nodes+(*(triangs+i*3+0)-1)*3+0));
        *(nodes+nb_nodes*3+1) = *(nodes+(*(triangs+i*3+0)-1)*3+1) +
            alpha*(*(nodes+(*(triangs+i*3+1)-1)*3+1) -
            *(nodes+(*(triangs+i*3+0)-1)*3+1));
        *(nodes+nb_nodes*3+2) = z_planar;
        nb_nodes++;

        if(*(nodes+(*(triangs+i*3+1)-1)*3+2) > z_planar)
            {
```

```
                        *(utriangs+unb_triangs*3+0) =  nb_nodes;
                        *(utriangs+unb_triangs*3+1) = *(triangs+i*3+1);
                        *(utriangs+unb_triangs*3+2) = *(triangs+i*3+2);
                        unb_triangs++;
                        *(ltriangs+lnb_triangs*3+0) = *(triangs+i*3+0);
                        *(ltriangs+lnb_triangs*3+1) =  nb_nodes;
                        *(ltriangs+lnb_triangs*3+2) = *(triangs+i*3+2);
                        lnb_triangs++;
                        }
                    else
                        {
                        *(ltriangs+lnb_triangs*3+0) =  nb_nodes;
                        *(ltriangs+lnb_triangs*3+1) = *(triangs+i*3+1);
                        *(ltriangs+lnb_triangs*3+2) = *(triangs+i*3+2);
                        lnb_triangs++;
                        *(utriangs+unb_triangs*3+0) = *(triangs+i*3+0);
                        *(utriangs+unb_triangs*3+1) =  nb_nodes;
                        *(utriangs+unb_triangs*3+2) = *(triangs+i*3+2);
                        unb_triangs++;
                        }
                    }
                }
            }
        }
    *anb_triangs = unb_triangs;
    *bnb_triangs = lnb_triangs;
    *cnb_nodes = nb_nodes;
    *pnodes = nodes;
    }

readdata(filename, nodes, triangs, nb_nodes, nb_triangs)
char filename[];
float **nodes;
int **triangs, *nb_nodes, *nb_triangs;
    {
    int binfil;

    if((binfil=open(filename,O_RDONLY))==(-1))
        {
        printf("Sorry.... Can't open file %s!\n", filename);
        exit();
        }
    printf("Reading data file %s:\n", filename);
    read(binfil, nb_triangs, sizeof(int));
    printf("Number of Triangular Patches is %d\n", *nb_triangs);
    read(binfil, nb_nodes, sizeof(int));
    printf("Number of Object Nodes is %d\n", *nb_nodes);

    if((*triangs = (int *)calloc( *nb_triangs * 3, sizeof(int))) == NULL)
        {
        printf("\nERROR Memory allocation\n\n");
        exit(0);
        }
    if((*nodes = (float *)calloc(*nb_nodes * 3, sizeof(float))) == NULL)
        {
        printf("\nERROR Memory allocation\n\n");
        exit(0);
        }
    read(binfil, *triangs, *nb_triangs*3*sizeof(int));
    read(binfil, *nodes, *nb_nodes*3*sizeof(float));
    close(binfil);
    }

write_output(filename, nb_triangs, nb_nodes, triangs, nodes)
char filename[];
int nb_triangs, nb_nodes, *triangs;
float *nodes;
    {
    int binfil;

    if((binfil=open(filename,O_WRONLY|O_CREAT, 0644))==(-1))
        {
        printf("Sorry.... Can't open output file %s!\n", filename);
        exit();
        }
    write(binfil,&nb_triangs,sizeof(int));
    write(binfil,&nb_nodes,sizeof(int));
    write(binfil,triangs,nb_triangs*3*sizeof(int));
    write(binfil,nodes,nb_nodes*3*sizeof(float));
    close(binfil);
    }
```