MEASUREMENT OF THREE-DIMENSIONAL SCAPULAR KINEMATICS

by

ANTHONY MIN TE CHOO

B.A.Sc, The University of Toronto, 1998

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
DEPARTMENT OF MECHANICAL ENGINEERING

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

April 2001

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Mechanical Engineering

The University of British Columbia
Vancouver, Canada

Date April 18, 2001

The University of British Columbia

Abstract

Measurement of Three-Dimensional Scapular Kinematics
by

Anthony Min Te Choo

Chairperson of the Supervisory Committee: Dr. Thomas R. Oxland
Department of Mechanical Engineering

It has been well recognized that the scapula and the humerus work in concert to enable the shoulder joint to achieve its high mobility in three-dimensions. Pathologies such as impingement and instability are focused at the glenohumeral junction between the two bones. Other authors have attempted to measure scapulothoracic and scapulohumeral rhythms as a means of better understanding the shoulder joint. However, at present, the inability to practically measure scapular kinematics in a clinical setting constitutes a significant problem in the rehabilitation of shoulder pathologies.

This thesis addresses the issue of measuring three-dimensional scapular kinematics non-invasively. A novel method using a grid of skin surface markers was developed. The method was found to show promising results in a pilot study and a pre-clinical test on a cadaver. An accuracy of better than 5^{o} was achieved for all three cardan angles used to describe changes in scapular attitudes. The method quantified the regional variations encountered with surface markers and identified the optimal regions for measurement. Additionally, a prototype method utilizing two-dimensional Fourier analysis was investigated to measure scapular upward rotation through image analysis of surface features. The results were not conclusive but merit further investigation.

The gold standard used to assess accuracy of the non-invasive method was roentgen stereophotogrammetric analysis (RSA). The accuracy of the RSA system was investigated with computer simulations and benchtop analyses. System accuracy ranged from 0.21-mm to 0.67-mm for reconstruction of marker coordinates, from 0.23-mm to 0.86-mm for translation, and from 0.19^{o} to 1.05^{o} for rotations. The stereophotogrammetric calibration method was an important factor in the resultant accuracy. The accuracy was also affected by the reconstruction algorithm used. Direct Linear Transformation (DLT) was found more favorable than traditional RSA methods described by Selvik. Finally, low x-ray scanning resolution (150-ppi) was a principal hardware limitation contributing to lower accuracy.

This thesis has demonstrated a scapular measurement technique and presented some novel ideas that should contribute to the long-term development of a clinically practical, non-invasive, accurate method for measuring human scapular kinematics.

TABLE OF CONTENTS

Abstract ii

Table of Contents iii

List of Tables vii

List of Figures viii

Acknowledgements xi

Dedication xii

Chapter 1 Introduction 1

1.1 The Human Shoulder 1

1.1.1 Skeletal & Muscular Anatomy 1

1.2 Glenohumeral Joint Pathologies 5

1.2.1 Labral Damage and Instability 5

1.2.2 Cuff Damage and Impingement 8

1.2.3 Diagnosis and Rehabilitation – The Role of Scapular Measurement 10

1.2.4 Literature on Scapular Motion 14

1.3 Motivation, Objective & Scope 18

1.4 Overall Study Design 19

1.4.1 Requirements 19

1.4.2 Questions 19

1.4.3 Measurement Methods - RSA & Skin Markers 20

Chapter 2 Pilot Study 21

2.1 Objective 21

2.2 Methods 21

2.3 Results 23

2.4 Discussion 28

2.5 Conclusion 30

Chapter 3 Stereophotogrammetry 32

3.1 Stereophotogrammetric Preliminaries 32

3.2 Overview of Stereophotogrammetric Analysis Methodologies 33

3.3 Direct Linear Transformation 36

3.4 Selvik's Method for RSA 41

3.5 Selvik's Methodology 42

3.6 Discussion of DLT and Selvik's Method 44

Chapter 4 RSA Simulations 45

4.1 Part I – Calibration Cage and Analysis Algorithm 46
4.1.1 Part I – Objective 46
4.1.2 Part I – Methods 47
4.1.2.1 RSA Graphic User Interface (RSAGUI) 47
4.1.2.2 Simulated Calibration Cages and Object 49
4.1.2.3 Simulation Trials 52
4.1.3 Part I – Results 53
4.1.4 Part I – Discussion 56
4.1.5 Part I – Conclusions 63

4.2 Part II – Influence of Parameters 63
4.2.1 Part II – Objective 63
4.2.2 Part II – Methods 64
4.2.3 Part II – Results 65
4.2.4 Part II – Discussion 68
4.2.5 Part II – Conclusions 69

4.3 RSA Simulation Conclusions 69

Chapter 5 RSA Accuracy Studies 70

5.1 Measurement Accuracy 70
5.1.1 Objective 71
5.1.2 Methods 71
5.1.3 Results 72
5.1.4 Discussion 75
5.1.5 Conclusion 76

5.2 RSA Reconstruction Accuracy 77
5.2.1 Objective 77
5.2.2 Methods 77
5.2.3 Results 80
5.2.4 Discussion 82
5.2.5 Conclusion 83

5.3 RSA Translation Accuracy 84
5.3.1 Objective 84
5.3.2 Methods 84
5.3.3 Results 89
5.3.4 Discussion 92
5.3.5 Conclusion 93

5.4 RSA Rotational Accuracy 93
5.4.1 Objective 93
5.4.2 Methods 93
5.4.3 Results 95
5.4.4 Discussion 96
5.4.5 Conclusion 97

5.5 Summary Conclusions for RSA Accuracy Studies 98

Chapter 6 Scapular Kinematics in a Cadaver 99

6.1 Objective 99

6.2 Methods 99
6.3 Results 107
6.4 Discussion 117
6.5 Conclusions 121
Chapter 7 Slip Estimation using Fourier Analysis *122*
7.1 Background 122
7.2 Objective 131
7.3 Methods 131
7.4 Results 133
7.5 Discussion 136
7.6 Conclusions 137
Chapter 8 Summary, Future Work & Recommendations *138*
8.1 Summary 138
8.2 Future Work & Recommendations 140
Appendix A X-Ray Radiography *142*
A.1 General X-Ray Setup 142
A.2 Control of Image Quality 143
A.2.1 Density and Contrast 143
A.2.2 kV 143
A.2.3 mAs 144
A.3 Control of Dose 144
A.3.1 Half-Value Layer (HVL) 144
A.3.2 Filtration 144
A.3.3 Entrance Skin Exposure 145
A.3.4 Effective Dose 145
A.3.5 Methods of Determining Effective Dose 145
Appendix B RSA GUI - Matlab Code *148*
B.1 RSAGUI_CB 148
B.2 Sim3DDLT 182
B.3 SimSelvik 183
B.4 Script3DDLT 185
B.5 ScriptSelvik 186
B.6 Utilities 189
B.6.1 RSAC 189
B.6.2 Az_cb 191
B.6.3 El_cb 191
B.6.4 AddPts2D 192

B.6.5 AddPts3D 193
B.6.6 AddLns3D 193
B.6.7 AddPatch3D 194
B.6.8 Delimit 195
B.6.9 ParseDelimited 195
B.6.10 MultHomogeneous 196
B.6.11 Xpose 196
B.6.12 PlaneIntercept 197
B.6.13 LinesIntercept 198

Appendix C RSA Accuracy - Matlab Code *200*

C.1 Digital Measurement Accuracy **200**
C.1.1 AccuracyTst 200
C.1.2 MkXray 201
C.1.3 GenCentre 203
C.1.4 Psphere 205

C.2 Benchtop Accuracy Test **206**
C.2.1 DOERCage10 206
C.2.2 Tlocal 206
C.2.3 csv2xry 208

Appendix D Cadaver Kinematics - Supplementary Results *209*

D.1 Humeral Elevation Results **210**
D.1.1 Rz – Scapular Upward Rotation 210
D.1.2 Ry – Scapular Internal/External Rotation 217
D.1.3 Rx – Scapular Tipping 221

D.2 Retraction Results **225**
D.2.1 Rz – Scapular Upward Rotation 225
D.2.2 Ry – Scapular internal/external rotation 230
D.2.3 Rx – Scapular Tipping 235

Appendix E Cadaver Kinematics - Matlab Code *240*

E.1 matchStereoPts **240**

E.2 assignPts **241**

E.3 solveKin **242**

E.4 mkT **244**

E.5 defMarkers **245**

Appendix F Slip Estimation with Fourier Analysis - Matlab Code *247*

F.1 solveFFT **247**

F.2 convertSP **252**

F.3 defSP **253**

Reference List *254*

LIST OF TABLES

Number	*Page*
Table 1-1: Manual Tests	*11*
Table 1-2: UCLA Rating Scale	*12*
Table 1-3: American Shoulder and Elbow Surgeons Scale	*13*
Table 2-1: Comparison of 3-D rotations	*28*
Table 3-1: Stereophotogrammetric Parameters	*34*
Table 4-1: Comparison of Methods	*56*
Table 5-1: Measurement Accuracy Results	*73*
Table 5-2: Reconstruction Accuracy	*81*
Table 5-3: Self-calibration Pairs	*89*
Table 5-4: Initial Calibration Pairs	*89*
Table 5-5: Absolute Rotations	*95*
Table 5-6: Relative Rotations	*95*
Table 5-7: RSA Accuracy Summary	*98*
Table 6-1: Example Digitized Bone Markers	*109*
Table 6-2: Example Point Pairing	*109*
Table 6-3: Reference Distance between Markers	*110*
Table 6-4: Marker Distances at 45° Elevation	*110*
Table 7-1: Scapular Feature Upward Rotation	*136*

LIST OF FIGURES

Number *Page*

Figure 1-1: Shoulder – Skeletal Components 2
Figure 1-2: Shoulder Muscular Attachments 3
Figure 1-3: The Rotator Cuff 4
Figure 1-4: The Glenoid 5
Figure 1-5: SLAP Lesions 6
Figure 1-6: Hospital for Special Surgery Instability Grading 7
Figure 1-7: Acromion Morphology 9
Figure 1-8: Reduced Subacromial Space 9
Figure 1-9: State Transition Diagram of Glenohumeral Pathology 10
Figure 2-1: Pilot Patches 22
Figure 2-2: Optoelectronic skin markers 22
Figure 2-3: Anatomical coordinate systems 22
Figure 2-4: Scapular Tipping 24
Figure 2-5: Scapular Internal/External Rotation 24
Figure 2-6: Scapular Elevation 25
Figure 2-7: Composite of Literature Data 26
Figure 2-8: Inferior Patches & Literature 26
Figure 2-9: Peripheral Patches & Literature 27
Figure 2-10: Ideal Patches & Literature 27
Figure 2-11: Skin Patch Summary 30
Figure 3-1: Stereophotogrammetric Setup 33
Figure 3-2: Image and Lab Coordinate Systems 34
Figure 3-3: The Node Point 37
Figure 3-4: Vector A 37
Figure 3-5: Vector B 38
Figure 3-6: Central Projection 42
Figure 4-1: Calibration Cages 45
Figure 4-2: RSAGUI Window 48
Figure 4-3: RSAGUI Rendering 48
Figure 4-4: Virtual projective cage 49
Figure 4-5: Projective Cage Exposure 50
Figure 4-6: Virtual Biplanar Cage 51
Figure 4-7: Biplanar Cage Exposure 51
Figure 4-8: Virtual object 51
Figure 4-9: Simulation Sets 52
Figure 4-10: Zero Simulation 54
Figure 4-11: Projective Cage Simulation 54
Figure 4-12: Biplanar Cage Simulation 55
Figure 4-13: Comparison of overall RMSE 58
Figure 4-14: Optimization - Single versus Multiple Steps 60
Figure 4-15: Control Point Extrapolation 61
Figure 4-16: Manufacturing Accuracy 66
Figure 4-17: Source Distance 66
Figure 4-18: Asymmetric Object Position 67
Figure 4-19: Projective Cage Interpolation Accuracy 68
Figure 5-1: Measurement Accuracy Image 72
Figure 5-2: Sphere Radius Effect 74

Figure 5-3: X-ray Transmissibility Effect 74
Figure 5-4: Radius and Transmissibility 75
Figure 5-5: Simulated and Actual 0.8-mm Exposures 76
Figure 5-6: Simulated and Actual 3-mm Exposures 76
Figure 5-7: Alignment Markers 78
Figure 5-8: Biplanar Cage Dimensions 79
Figure 5-9: Coordinate Comparisons 80
Figure 5-10: Cassette Changes 81
Figure 5-11: Distance Accuracy 82
Figure 5-12: Translation Markers 85
Figure 5-13: Translation x-axis 86
Figure 5-14: Translation z-axis 86
Figure 5-15: Sample Points 87
Figure 5-16: Translation Accuracy Setup 87
Figure 5-17: Effect of Calibration Method 91
Figure 5-18: Directional Dependence 91
Figure 5-19: Calibration Method & Direction 92
Figure 5-20: Rotation Marker Carriers 94
Figure 5-21: Rotation Test Setup 94
Figure 5-22: Rotation Error 96
Figure 6-1: Implantation Instruments 101
Figure 6-2: Thoracic Marker Carrier 101
Figure 6-3: Grid on Skin Surface 103
Figure 6-4: Specimen on Side 104
Figure 6-5: Pairing and Assignment Ambiguity 105
Figure 6-6: Anatomical Coordinate Systems 106
Figure 6-7: Examples of Poor X-rays 108
Figure 6-8: Scapular upward rotation RMSE during humeral elevation 112
Figure 6-9: Scapular upward rotation - lateral scapular spine patches 113
Figure 6-10: Scapular internal/external rotation - RMSE < 5° 113
Figure 6-11: Scapular tipping - inferior scapular spine patches 114
Figure 6-12: Scapular upward rotation – patches near scapular spine 115
Figure 6-13: Scapular internal/external rotation – patches on scapular spine 116
Figure 6-14: Scapular tipping - inferior scapular spine patches 116
Figure 6-15: Effect of missing skin marker 117
Figure 6-16: Scapular tipping 119
Figure 6-17: Shoulder coordinates of the international shoulder group 120
Figure 7-1: Representation of a signal 123
Figure 7-2: Two images with relative rotation 126
Figure 7-3: 3-D Magnitude of Fourier spectra 127
Figure 7-4: 2-D Magnitude of Fourier spectra 128
Figure 7-5: Δ(k) function 129
Figure 7-6: Locus where Δ(k) = 0 129
Figure 7-7: Histogram of locus Δ(k) = 0 130
Figure 7-8: Ideal Skin Boundary 132
Figure 7-9: Actual Skin Boundary 132
Figure 7-10: Interpolated Skin Surfaces 134
Figure 7-11: Locus of Δ(k) = 0 between 45° & 90° of elevation 135
Figure 7-12: Histogram of Δ(k) = 0 between 45° & 90° of elevation 135
Figure 7-13: Lagrangian versus Eulerian descriptions 137
Figure A-1: X-ray Imaging Setup 143
Figure D-1: Scapular upward rotation RMSE during humeral elevation 210

Figure D-2: Scapular upward rotation - inferior boarder patches 211
Figure D-3: Scapular upward rotation - medial boarder patches 212
Figure D-4: Scapular upward rotation – central patches 213
Figure D-5: Scapular upward rotation - patches in region of scapular spine 214
Figure D-6: Scapular upward rotation - medial scapular spine patches 215
Figure D-7: Scapular upward rotation - lateral scapular spine patches 216
Figure D-8: Scapular intenal/external rotation RMSE during humeral elevation 217
Figure D-9: Scapular internal/external rotation – patches 1 to 26 218
Figure D-10: Scapular internal/external rotation – patches 27 to 51 219
Figure D-11: Scapular internal/external rotation patches with RMSE < 5° 220
Figure D-12: Scapular tipping RMSE during humeral elevation 221
Figure D-13: Scapular tipping - inferior boarder patches 222
Figure D-14: Scapular tipping - superior and medial boarder patches 223
Figure D-15: Scapular tipping - inferior scapular spine patches 224
Figure D-16: Scapular upward rotation RMSE during humeral retraction 225
Figure D-17: Scapular upward rotation - inferior boarder patches 226
Figure D-18: Scapular upward rotation - superior and medial boarder patches 227
Figure D-19: Scapular upward rotation - lateral and inferior scapular spine patches 228
Figure D-20: Scapular upward rotation error due to missing skin markers 229
Figure D-21: Scapular internal/external rotation RMSE during humeral retraction 230
Figure D-22: Scapular internal/external rotation – patches 1 to 24 231
Figure D-23: Scapular internal/external rotation – patches 26 to 51 232
Figure D-24: Scapular internal/external rotation – patches on scapular spine 233
Figure D-25: Scapular internal/external rotation error due to missing skin markers 234
Figure D-26: Scapular tipping RMSE during humeral retraction 235
Figure D-27: Scapular tipping - inferior boarder patches 236
Figure D-28: Scapular tipping - superior and medial boarder patches 237
Figure D-29: Scapular tipping - inferior scapular spine patches 238
Figure D-30: Scapular tipping error due to missing skin markers 239

ACKNOWLEDGEMENTS

I am most grateful to Dr. Tom Oxland for giving me the opportunity to join the great group of people at the Orthopaedic Engineering Research Lab. His clarity of thought, patience, guidance and support have been invaluable to me throughout this project. I would also like to thank Dr. Bill Regan for his enthusiasm and bringing this project to the lab. Many thanks to Dr. Donna MacIntyre for her support, particularly with ethics approval, and Dr. Tony Hodgson for the discussions and helpful emails. I am also very thankful to Darrell Goertzen for his support on many aspects of this project.

Many others have taken time to assist me and have somehow contributed to the completion of this project.

Many thanks,

J. PAMELA GRANT, HANSPETER FREI, JOHNSON GO, JEFF GORDON, CARLA HABER, JAN HOWE, ZAMEER HIRJI, SUSANNE KOMILI, CHRIS LANE, NGOC LUU, LINA MADILAO, JOHN MALONEY, MARGARET MCCREADY, WINNIE NG, DON PEAKE, DR. RAY POLACK, MARK RAMSEY, DR. BONITA SAWATZKY, DR. JOE SON-HING, DR. VLAD STANESCU, MAGGIE STUART, JUAY SENG TAN, CHARLTON WANG, BETTY WINCUP

DEDICATION

for my ever supportive parents Lip Kung and Keow Geen
&
Ngoc and my little Cianna

Chapter 1

INTRODUCTION

1.1 The Human Shoulder

1.1.1 Skeletal & Muscular Anatomy

The skeletal and muscular anatomy of the shoulder demonstrates the need to measure scapular motion.[i] The scapula, humerus and clavicle are the three primary bones forming the shoulder (Figure 1-1). The clavicle serves to transfer load from the arm to the thorax and is a pivot point at which the scapula rotates. The anatomy of the shoulder is such that humeral elevation, relative to the scapula, is restricted to less than 90^{o}. This is due to abutment of the greater tuberosity of the humerus and the coracoacriomial arch that lies superior to the glenoid cavity. Yet the arm is able to elevate well above 90^{o} in normal individuals. The reason is that the scapula is also free to move and the total elevation of the arm is the combined effect of humeral motion with respect to the scapula, and the scapular motion with respect to the thorax. This project is focused on measuring this latter motion.

The junction between the humeral head and the scapula's glenoid process is the glenohumeral joint. This, however, is not the only joint in the shoulder. The shoulder joint is more aptly named the shoulder joint complex (SJC) because it actually consists of four joints. The acromioclavicular joint refers to the articulation between the acromion process of the scapula and the clavicle. The sternoclavicular joint denotes the articulation between the clavicle and the sternum. Finally, the scapulothoracic joint

[i] The nervous and vascular systems in the shoulder are not directly related to this thesis and have thus been omitted for brevity.

refers to a virtual joint between the scapular body and the thoracic wall along which the scapula translates.

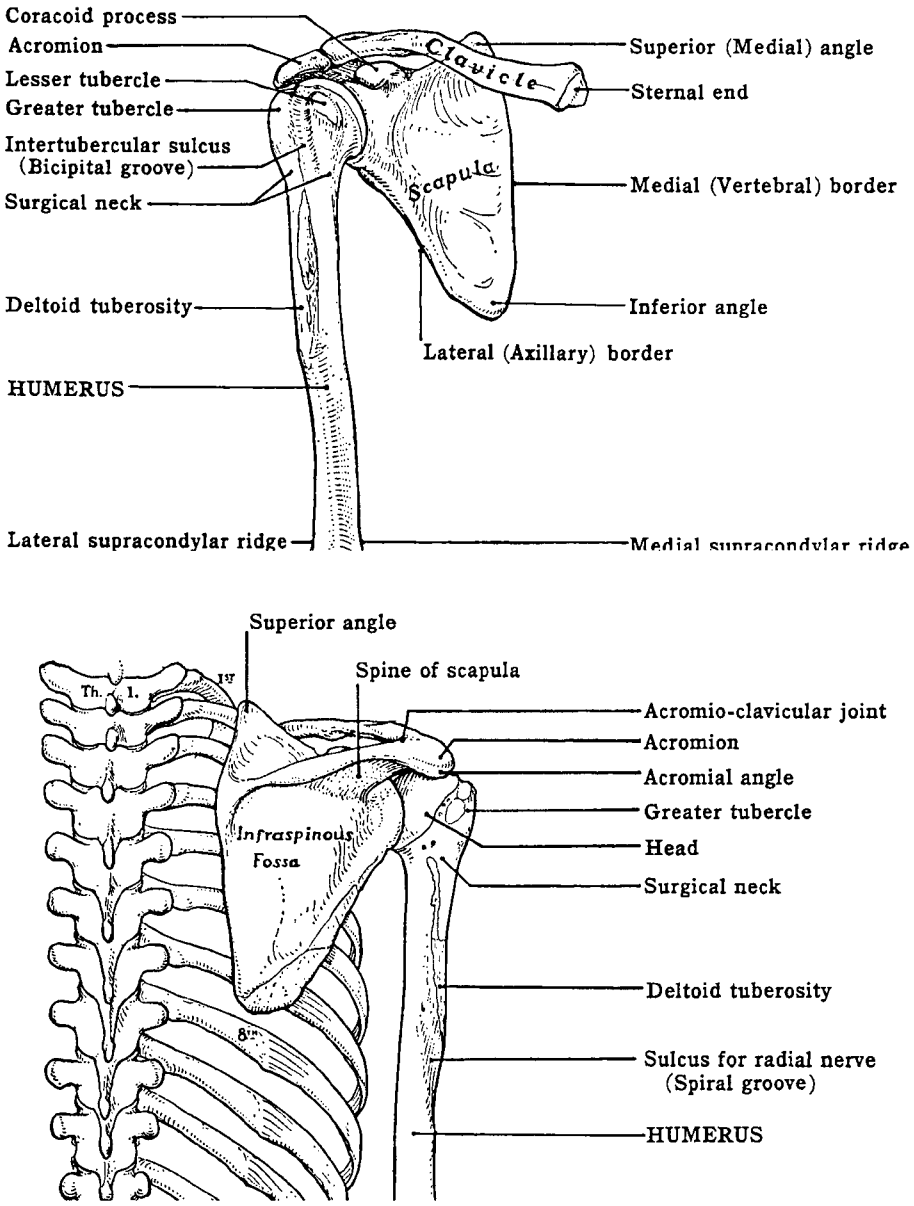

Figure 1-1: Shoulder – Skeletal Components
Top: Anterior view, Bottom: Posterior view. Diagrams are from Anderson.[5]

Following the skeletal anatomy, the muscular structures in the shoulder link the scapula and the humerus to mobilize the shoulder through a huge range of motion. The scapula is free to move in three dimensions and is entirely stabilized by muscular structures. Figure 1-2 shows the attachments of these structures. They include the rhomboids, levator scapulae, trapezius, and serratus anterior. The humerus is linked to the scapula at the glenohumeral joint via the rotator cuff muscles. The cuff consists of subscapularis, supraspinatus, infraspinatus and teres minor (Figure 1-3).

The rotator cuff functions to mobilize the humerus and to stabilize the glenohumeral joint. The cuff muscles pull the humeral head into the socket formed by the labrum (discussed next) and glenoid fossa. The joint remains stable throughout motion as long as the humeral head is firmly centred and cradled within the deep socket.

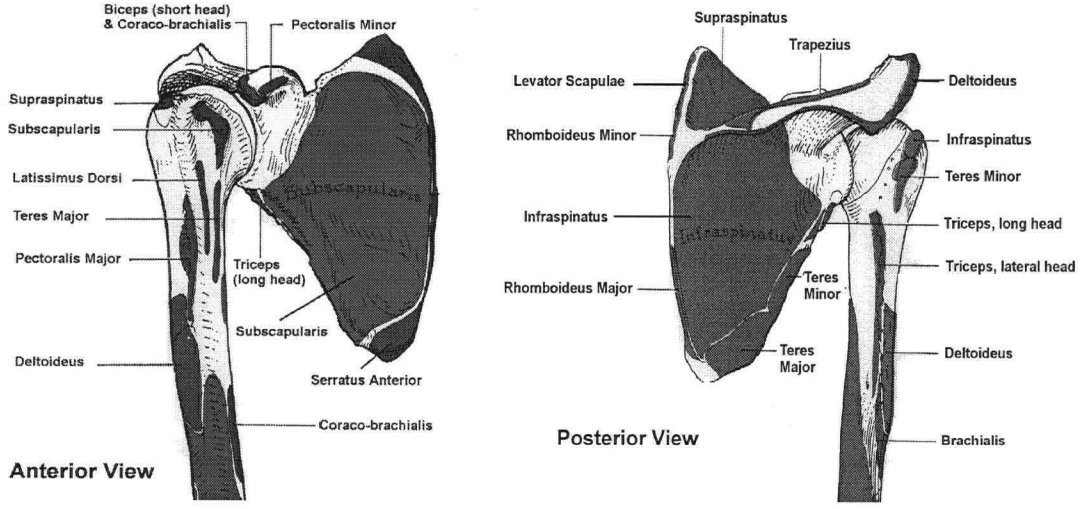

Figure 1-2: Shoulder Muscular Attachments
Muscular attachments of the shoulder joint show the interdependence of the scapula and humurus. Diagrams were modified from Anderson.[5]

Two other stabilizing features are critical in shoulder mechanics. These are the glenoid labrum and the coracoacromial arch. The glenoid labrum is a wedge-shaped fibrous structure that surrounds the shallow glenoid cavity thus forming a deep, yet soft, socket in which the humeral head resides (Figure 1-4). In this way, stability is provided while maintaining high mobility. The coracoacromial arch consists of the acromion, coracoid

process and coracoacromial ligament (Figure 1-4). It acts to contain the humeral head during elevation thus preventing superior migration.

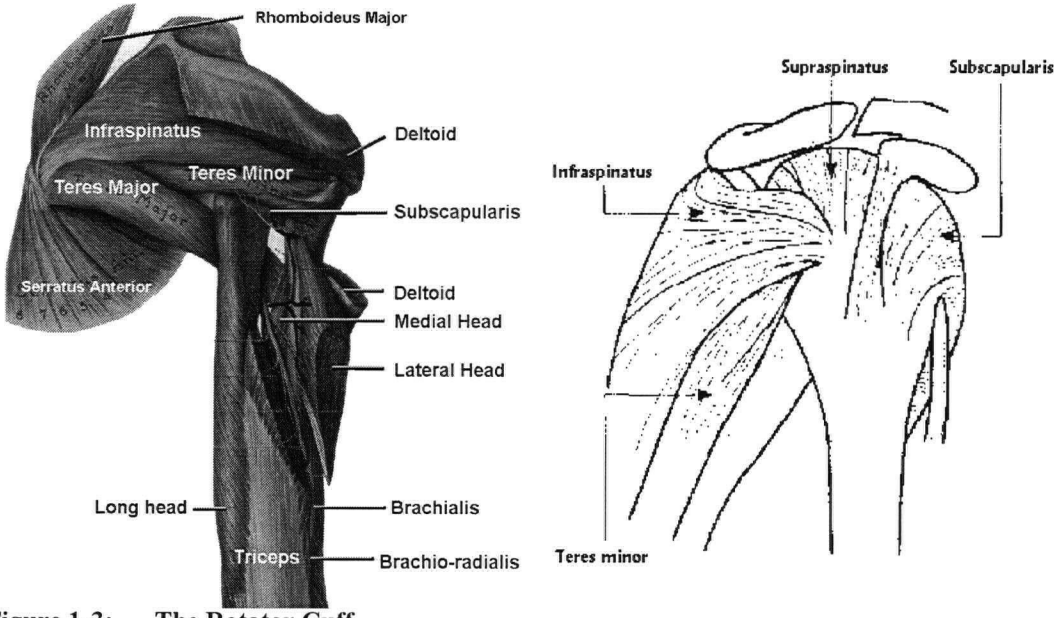

Figure 1-3: The Rotator Cuff

Four muscles comprise the rotator cuff. Subscapularis attaches from the anterior of the scapula to the lesser tubercle of the humerus. Supraspinatus, infraspinatus, and teres minor originate on the posterior surface of the scapula and attach to the greater tuberosity of the humerus. Diagrams were modified from Anderson and Blevins.[5,10]

Finally, although it does not mobilize the glenohumeral joint, the long head of biceps also deserves mention. Its tendon attaches to the superior rim of the glenoid and labrum. It is under high tension because it is a powerful flexor of the forearm. It traverses the glenohumeral joint and consequently is often damaged in glenohumeral injuries.

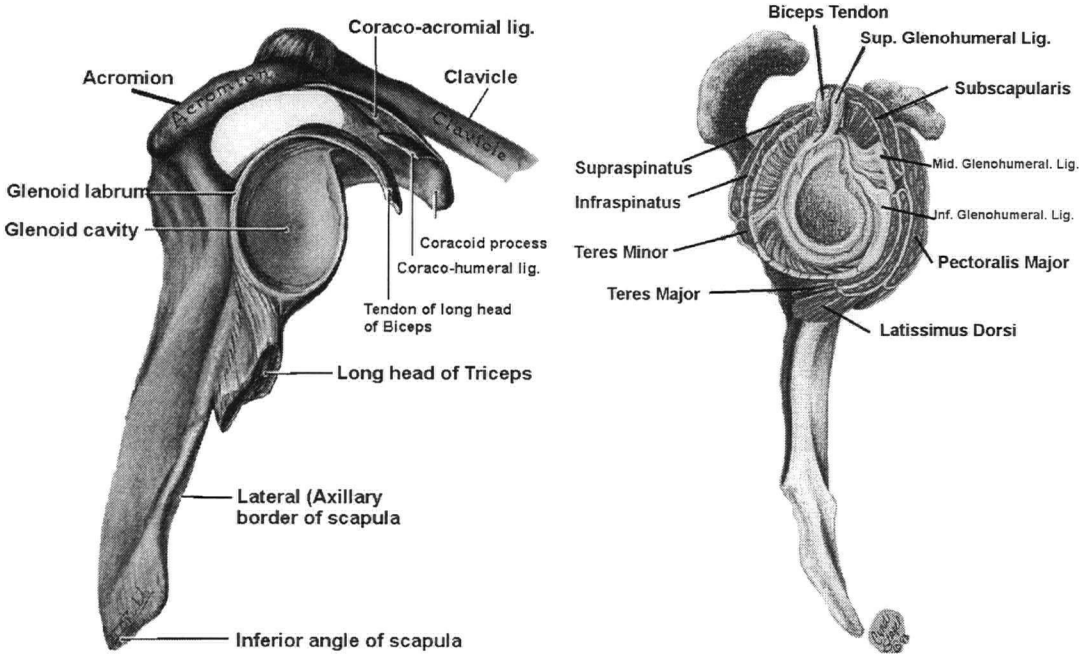

Figure 1-4: The Glenoid
Left: Lateral view of the scapula showing the glenoid fossa. The glenoid is a shallow cup much smaller than the humeral head. It provides little stability hence affording a high range of motion to the humerus. Diagram was modified from Anderson.[5] Right: Lateral view of scapula showing the labrum which surrounds the glenoid. The labrum deepens the socket thus providing stability while still allowing a high range of motion. Diagram was modified from Pettrone.[72]

1.2 Glenohumeral Joint Pathologies

The most fundamental motive for developing a method to measure scapular motion is to assist in the treatment of shoulder pathologies. This section focuses on pathologies at the glenohumeral joint where the scapula interfaces with the humerus. In particular, damage to the glenoid labrum and rotator cuff will be discussed.

1.2.1 Labral Damage and Instability

The role of the glenoid labrum in pathology is still controversial since labral lesions have been observed in many shoulders exhibiting normal function.[3,71] However, the vast majority of authors agree that there is a strong association between labral lesions and instability. [3,69,71]

The location of labral tears may be anterior, posterior, superior or inferior. By far the most common pathology is damage to the superior labrum.[72] These "SLAP" (superior labrum anterior and posterior) lesions may be further classified into four types (Figure 1-5). Type I lesions involve fraying of the superior labrum. Type II lesions are similar to type I, however the superior labrum and biceps tendon are stripped off the glenoid rim. Type III lesions involve a vertical tear through the superior labrum. The fragment often displaces into the joint resulting in mechanical impingement. In a type III lesion, the biceps tendon is intact. A type IV lesion is similar to a type III lesion, except it includes tearing of the biceps tendon.

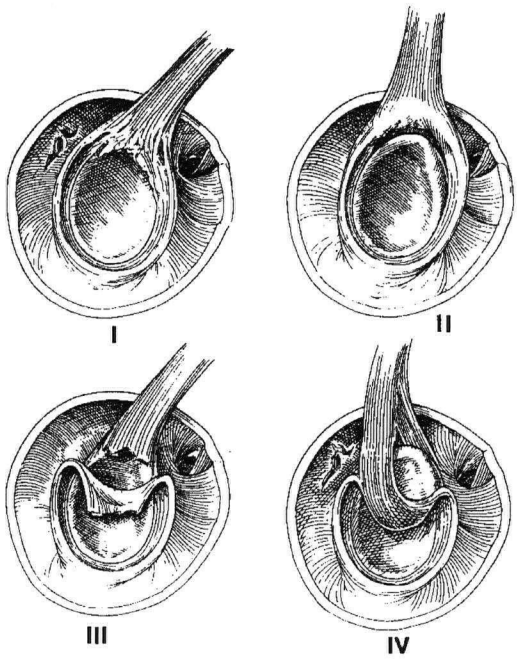

Figure 1-5: SLAP Lesions
I) superior labral fraying, II) superior labrum and biceps tendon stripped off glenoid rim, III) vertical tear with fragment displaced into the joint; biceps tendon still intact, IV) tearing of biceps tendon. Diagrams are from Warner & Fu.[90]

Several factors are usually considered in classifying instability.[29] These are:

1. direction of instability (anterior, posterior or multidirectional)
2. degree of instability (subluxation, dislocation)
3. etiology (traumatic, atraumatic, overuse)
4. timing (acute, recurrent, fixed)

Figure 1-6 shows the Hospital for Special Surgery grading system. As described by Tomlinson and Glousman, a 1+ laxity indicates the examiner can translate the humeral head farther than the contralateral shoulder. A 2+ score indicates the examiner can subluxate the humeral head over the glenoid rim. A 3+ laxity indicates the examiner can lock the humeral head over the glenoid rim.

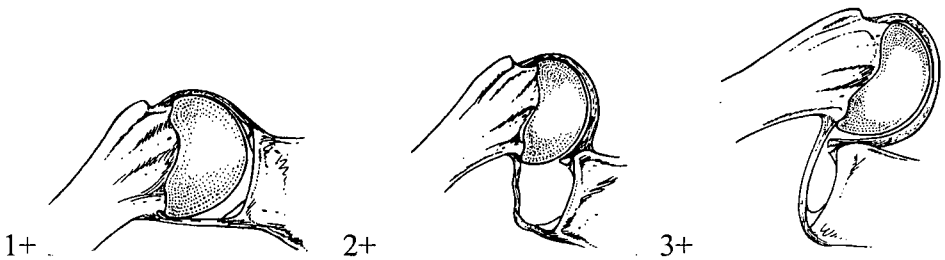

Figure 1-6: Hospital for Special Surgery Instability Grading
Diagrams are from Altchek, Warren et al.[3]

Even moderate instability may cause progressive degeneration over time. An unstable glenohumeral joint may cause excessive tension on the rotator cuff muscles which causes them to tear. Payne et al. have discussed how primary instability may be the cause of secondary Bankart lesions (anterior labral lesions). Alternatively, primary cuff and/or labral damage may result in secondary joint instability. Of particular clinical significance are SLAP lesions and/or damage to the supraspinatus that result in excessive superior migration of the humeral head. The migration causes impingement of the supraspinatus as described in section 1.2.2. Superior migration may also result in degeneration of the articulating surface on the humeral head. Defects on the

articulating surface, typically caused during dislocation, are refered to as Hill-Sachs lesions.

1.2.2 Cuff Damage and Impingement

Similar to labral lesions, it has been reported from cadaver studies that 72% of all rotator cuff tears are asymptomatic.[12] Burkhart found that normal shoulder function is possible even with massive (greater than 5-cm) rotator cuff tears as long as the muscles were balanced.[60] It is not only damage to the tissue, but also its effect on the relative motion between the humerus and scapula that result in pain and abnormal function.

Nevertheless, rotator cuff pathology is a significant problem particularly in overhead athletes. It has been reported that 66% of elite swimmers, 57% of baseball players, and 44% of college volleyball players all suffer from shoulder pathology related to the rotator cuff. [72]

Of the four cuff muscles, supraspinatus is the most commonly torn.[12] Supraspinatus is a relatively weak muscle hence its function is easily compromised through overuse. Its anatomic position, traversing the glenohumeral joint superiorly, makes it easily susceptible to impingement during abduction. During overhead abduction, supraspinatus performs a critical function of keeping the humeral head centred on the glenoid. This prevents superior migration and potential impingement of cuff and labral fragments between the greater tuberosity of the humerus and the superior structures (acromion and coracoacromial ligament). To exacerbate the pathology, damage to the rotator cuff has been found to not heal well. This has been attributed to the sparse vascular supply on the articular side.[12,55]

Impingement results in pain during arm elevation and is the most obvious symptom of patients with shoulder pathology. Impingement typically refers to the trapping of the supraspinatus between the greater tuberosity and the superior structures (acromion and coracoacromial ligament) during elevation. In 1972 Neer published a study where he showed cuff contact with the anterior aspect of the acromion resulting in spurs causing

impingement.[20] Bigliani added to these findings and classified the morphology of the acromion into three types (Figure 1-7). A type II or III acromion results in a smaller subacromial space contributing to impingement (Figure 1-8).

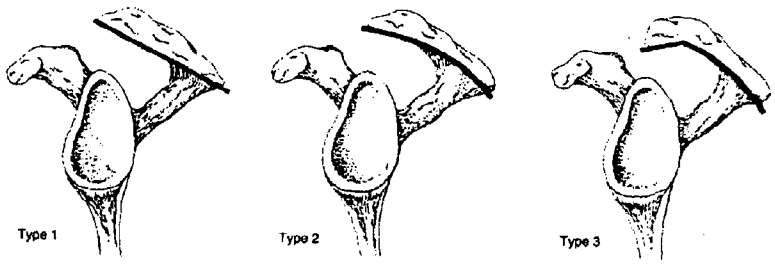

Figure 1-7: Acromion Morphology
Classification of acromion shape from Blevins.[10]

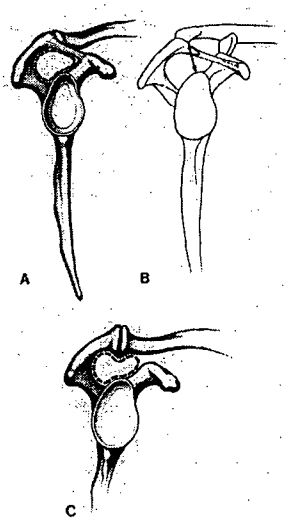

Figure 1-8: Reduced Subacromial Space
In type II and II acromions, the reduction in subacromial space contributes to impingement of supraspinatus tendon. Diagrams are from Marks et al.[57]

Neer classified impingement into three stages.[91] Stage I involved hemorrhage and edema. Stage II impingement was indicated by fibrosis and tendonitis of the cuff muscles. Finally, stage III impingement involves tearing of the rotator cuff.

1.2.3 Diagnosis and Rehabilitation – The Role of Scapular Measurement

Labral and cuff tears need not appear independently. A labral tear may lead to instability that causes a subsequent cuff lesion. Conversely, damage to cuff muscles may cause superior migration of the humeral head resulting in a labral tear. The mechanics of injury may be even more complex. Figure 1-9 shows a state transition diagram of glenohumeral joint pathology. Tracing these pathways shows that a wide range of hypothetical cascading mechanisms of injury are possible. The inability to reliably diagnose the mechanism of injury presents challenges for appropriate identification of optimal surgical procedures and analysis of the outcomes.[6,19,58,82] One of the problems with current diagnostic techniques, pre- and post-op, involves the inability to accurately track the motion of the scapula.

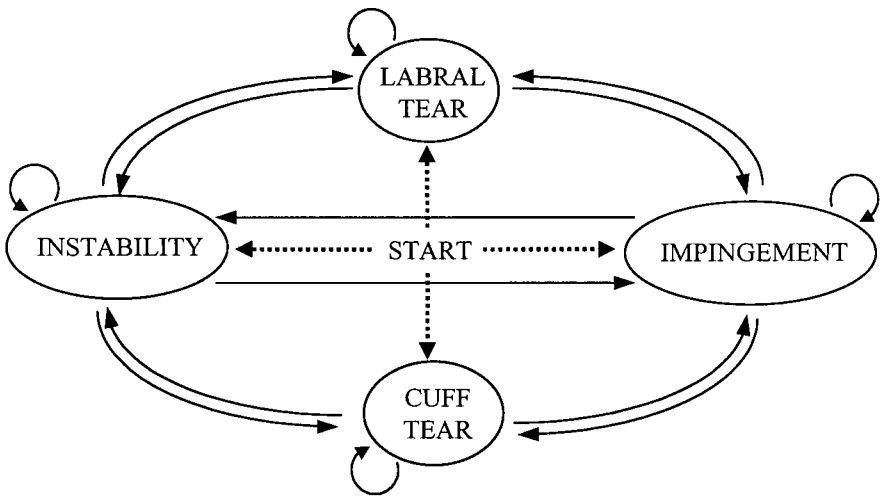

Figure 1-9: State Transition Diagram of Glenohumeral Pathology
Beginning at the start, no pathology, the arrows trace the possible routes of pathologic degeneration. A simple case may be the isolated presence of a small labral tear. The arrow loops back to the same state indicating the pathology does not progress. Frequently, the pathology progresses – the tear follows the arrow to instability. In severe cases, labral tearing, cuff tearing, instability and impingement may all be present. The pathway to arrive at this state is often ambiguous thus contributing to the difficulty in diagnosing the underlying causes of pathology.

A number of tools are used in the diagnosis of labral and rotator cuff tears. In terms of medical imaging, MRI is the primary modality of choice.[19] Large tears are reliably identifiable with modern MRI systems. However, MRI is still unreliable in detecting

partial and small tears. Ultrasound has also been used for detecting soft tissue damage but its reliability is generally regarded as poor. Lateral radiographs are useful in assessing occlusion of the subacromial space. Insufficient space beneath the acromion contributes to impingement of the supraspinatus.

A wide variety of manual tests exist for assessing glenohumeral pathology. Some common tests are listed in Table 1-1.[3,10,55,58]

Test	Details	Significance
Shrug sign	Excessive scapular elevation & rotation during elevation	Supraspinatus pathology
Lift-off test	Inability to actively lift internally rotated arm	Subscapularis pathology
Relocation test	Reduction of pain with posteriorly directed force on abducted, externally rotated shoulder	Anterior laxity
Apprehension test	arm abducted at 90°, fully externally rotated, & anteriorly directed force – patient senses imminent dislocation	Anterior laxity

Table 1-1: Manual Tests
Adapted from Blevins.[10]

In addition to imaging and manual tests, clinical test scores are useful tools for assessing patients before and after surgery. The most commonly used score is the University of California at Los Angeles (UCLA) rating scale (Table 1-2).[12] The maximum score in this scale is 35 points. A score of 34 or 35 is rated as excellent. A score between 28 and 33 is considered good. A score between 21 and 27 is fair. A score below 21 points is poor.

The UCLA scale is not used in all studies. There are several deficiencies in the rating system. First, active flexion to 150° is rated as 5/5. This leaves 30° of elevation unaccounted for in the score. Second, the scale is for overall shoulder function and some authors feel it is not specific enough for rotator cuff pathology.[12] Third, satisfaction is a highly subjective measure and accounts for 5 points in the 35-point scale. These problems limit the objectivity of the UCLA rating.

Items	Points
Pain	
Present always and unbearable; strong medication needed frequently	1
Present always but bearable; strong medication needed occasionally	2
None or little at rest; present during light activities; salicylates needed frequently	4
Present during heavy or particular activities only; salicylates needed occasionally	6
Occasional and slight	8
None	10
Function	
Unable to use limb	1
Only light activities possible	2
Able to do light housework and most activities of daily living	4
Most housework, shopping, and driving possible; able to brush hair and to dress and undress, including fastening of brassiere	6
Slight restriction only; able to work above shoulder level	8
Normal activities	10
Active flexion	
> 150°	5
121° – 150°	4
91° – 120°	3
46° – 90°	2
30° – 45°	1
< 30°	0
Strength of flexion (on manual muscle-testing)	
Grade 5	5
Grade 4	4
Grade 3	3
Grade 2	2
Grade 1	1
Grade 0	0
Satisfaction of patient	
Satisfied and better	5
Not satisfied and worse	0

Table 1-2: **UCLA Rating Scale**
Adapted from Budoff et al.[12]

The shoulder score index of the American Shoulder and Elbow Surgeons is another commonly used system (Table 1-3).[22,66] Still, many studies have been published which do not conform to either of these two standards making their results difficult to compare.[3,6,27,71,75,87]

Description	Points
Pain (36% of total score)	
None	5
Slight	4
After unusual activity	3
Moderate	2
Marked	1
Complete disability	0
Stability (36% of total score)	
Normal	5
Apprehension	4
Rare subluxation	3
Recurrent subluxation	2
Recurrent dislocation	1
Fixed dislocation	0
Function (28% of total score)	
Normal	4
Mild limitation	3
With difficulty	2
With aid	1
Unable	0

Table 1-3: American Shoulder and Elbow Surgeons Scale
Adapted from O'Neill.[66]

Prior to any surgical management, a minimum of three to six months of intense rehabilitation is prescribed for patients presenting with the shoulder pathologies described.[69] The importance of rehabilitation is critical since it is pre-operative and post-operative rehabilitation that ultimately restores proper function. Rehabilitation programs generally consist of strengthening exercises for the scapular stabilizers (trapezius, serratus anterior, rhomboids, levator scapulae) and the rotator cuff. The majority of exercises are performed in the scapular plane or in forward elevation. Exercises in the coronal plane are avoided since this is generally not the plane of

motion for most activities. In addition, this plane of motion often results in impingement and excessive loads to the rotator cuff.

Although it is acknowledged that the scapula and humerus function in concert to produce the overall motion of the shoulder joint, the inability to practically measure scapular motion during rehabilitation has been a barrier to the development of scapular rehabilitation regimes more complex and specific than "overall strengthening". Ice and analgesics are used to control pain. If response is slow, steroid injections may be employed to help with the strengthening of the muscles. Only on the failure of rehabilitation is surgical intervention explored.

1.2.4 Literature on Scapular Motion

The significance of scapular motion as indicated by the anatomy and demonstrated through pathology has been studied by various authors for over a hundred years. It is generally recognized that it was Cathcart in 1886 who first articulated the synchronization of the scapula and humerus in achieving overall arm elevation.[4,7,53] It was Codman in 1934 that coined the term scapulohumeral rhythm to describe the consistent relationship between scapular and humeral motion that contributes to overall arm elevation. Rhythm has since become the focus of numerous investigations into the motion of the shoulder joint complex.

The most often referenced work in describing scapular kinematics is that of Inman, Saunders and Abbott in 1944.[40] Their paper was a comprehensive discussion of the shoulder joint complex. They used x-rays (two-dimensional) to study scapulohumeral rhythm. From 0° to 30° of arm elevation in the coronal plane, there was no identifiable relationship between the scapula and humerus and hence termed this portion of elevation the setting phase. From 30° to 170° of arm elevation, they reported the often-quoted 2:1 ratio of glenohumeral to scapulothoracic rotation.

In 1966 Freedman and Munro studied fifty-two shoulders by x-raying them at 30° anterior to the coronal plane.[21] This was defined as the "scapular plane" and has

become the standard in scapular motion analysis. They were the first authors found in the literature to use the orientation of the glenoid cavity (in two dimensions) to describe the scapula's orientation. They found a wide confidence interval when performing regression analysis on the shoulder rhythm. This indicates that there is a high variability of shoulder rhythm across subjects. They did not report a setting phase but instead reported a 3:2 glenohumeral to scapulothoracic ratio over the entire range of motion.

In 1976, Poppen and Walker published another two-dimensional x-ray study on the shoulder.[73] They were the first found in the literature to analyze the instantaneous centre of rotation (ICR) and excursion of the humeral head in shoulder kinematics. They found that an ICR greater than 10-mm from the ball centre of the humeral head was indicative of abnormal shoulder function. In addition, they reported pathologic excursions on the order of 3-mm. They were not able to relate the excursion results to instability. They were also the first to show, at least qualitatively, that the scapula did not move in a flat plane but also "tipped" out of plane during elevation.

Since these early "classical studies", there has been a move towards three-dimensional analysis to gain a more complete understanding of scapular motion. In 1991 Hogfors' group published a study using roentgen stereophotogrammetric analysis (RSA) to measure shoulder kinematics in three dimensions.[33] They advocated the use of a more general definition of shoulder rhythm. They felt that shoulder rhythm is a consistent relationship between the humerus and scapula during arbitrary arm motions rather than strictly elevation which had been used in the past. To accomplish this, they used RSA while three subjects elevated their arms to 30^{o} and mobilized the arm in a conical motion about a lateral axis. They used polynomial expressions to describe the relationship between the humerus and scapula. The expressions were then used to predict hypothetical scapulohumeral rhythm in elevation. The simulated results were comparable to that reported by Inman and others throughout the years. Although the results were not compelling, the concept of a general definition for rhythm and the use

of RSA for measuring highly accurate three-dimensional shoulder kinematics were found to be important benchmarks in their study.

In 1998 McQuade used a single transmitter placed on the flat portion of the acromion to measure scapular motion.[59] By comparing with x-rays, their group addressed the issue of skin slippage. Although the x-rays were only two-dimensional, they reported skin slippage on the scapula to be 4.2-mm and on the humerus to be 3.1-mm.

In recent years several investigators have gone to surface palpation and digitization of scapular features to track its motion.[53,54,61] Lukasiewicz's group found a high repeatability (intraclass correlation coefficients ranging from 0.88 to 0.99) in identifying scapular features. They used this technique to compare an impingement versus a non-impingement group. They measured posterior tilt as the angle between the vector passing through C7 and T7, and a vector passing through the inferior angle and the root of the scapular spine. As the inferior angle moves anteriorly, posterior tilt increases. They found significantly less posterior tilt of the scapula for patients suffering from impingement. In addition, the impingement group also had greater superior translation of the scapula although the difference was not significant with the patients' contralateral side. Upward rotation and internal rotation of the scapula was not found to be significantly different between the groups.

A current paper by Karduna, accepted for publication but as yet unreleased, compares non-invasive skin marker methods against bone pins drilled into the scapula.[42] One skin marker method, the acromion tracker, uses a single transmitter placed on the acromion region similar to the method used by McQuade. The other uses a jig that contacts the skin at the mid-spine of the scapula and the acromion. The jig, or scapular tracker, has essentially two components. The medial portion is a hinge that conforms to the scapular spine. The lateral component is an adjustable footpad that contacts the acromion region. They found a high discrepancy between the skin and bone markers at high humeral elevations. For upward rotation, the scapular tracker underestimated the

rotation whereas the acromion tracker overestimated the rotation. The RMS errors reported for the scapular tracker in measuring upward rotation, internal/external rotation, and scapular tipping were 8.0^{o}, 3.2^{o}, and 4.7^{o} respectively.

In summary, the literature shows that the problem of tracking the scapula in a clinically practical sense remains unresolved. With the exception of Karduna, there has been an overall lack of gold standard accuracy validation. Only Freedman et al. and Poppen et al. have placed the scapular coordinate system at the site of pathologic pain – the glenohumeral interface. There has been a clear move towards three-dimensional scapular motion measurement, not only because it has been shown scapular motion is non-planar, but moreover, the out of plane motions such as insufficient scapular tipping has been related to pathologies such as impingement. Although scapulohumeral rhythm in elevation is a clear starting point, it is recognized that there is a high variability across subjects and hence other parameters should be considered. Among these are a more general notion of shoulder rhythm, the instantaneous centre of rotation and humeral head excursion. The latter two have received little attention although they are at the critical site of pathology. Finally, the move towards non-invasive surface skin methods has been a clear trend as it is recognized that the method to measure scapular motion must eventually be applied clinically if it is to have any impact in aiding the rehabilitation of shoulder pathology.

1.3 Motivation, Objective & Scope

The scapula, along with the humerus and clavicle are the three bones which comprise the human shoulder joint complex (SJC). The shoulder joint is the most mobile joint in the human body. Pathology of this joint results in the compromise of this high range of motion. There is currently a great deal of controversy regarding the mechanics of shoulder pathology, its proper diagnosis, treatment and rehabilitation.

Motivation

Understanding scapular motion is essential in the overall understanding and treatment of pathologies affecting the shoulder joint complex. Clinically, in a rehabilitation setting, there is no practical method to reliably quantify three-dimensional scapular motion. This lack of quantitative information contributes to the variability between patient groups in clinical studies. The development of a method to clinically measure scapular motion in a large patient group would help answer a vast array of questions regarding shoulder mechanics, the effects of surgery, and the impact of rehabilitation regimes.

Objective

The objectives of this thesis project are to:

1. develop a non-invasive, clinically practical, method for measuring scapular kinematics
2. compare the new method to a gold standard for measuring scapular kinematics
3. identify the limitations of both methods
4. identify strategies for the improvement of the new method

Scope

This thesis is a preliminary exploration into measurement techniques for scapular motion. The current focus is on establishing a *benchmark* for a non-invasive *measurement* technique. This thesis reports on the development of the measurement methods and the performance of the methods in an in vitro, pre-clinical, cadaveric test.

Due to the diversity of topics covered in this thesis, the chapters have been organized such that each is essentially a self-contained study. As a result, the chapters may be read independently. When read as a whole, the chapters lay out the building blocks used to achieve the overall goal of measuring scapular kinematics.

In the greater picture beyond the scope of this thesis, the measurement methods would be further refined and eventually applied in vivo to answer clinical questions regarding shoulder mechanics and pathology.

1.4 Overall Study Design

1.4.1 Requirements

To meet the objective of measuring the scapula, the following requirements were set.

1. the measurement method yield three-dimensional data
2. the method is non-invasive
3. the method is clinically practical in a rehabilitation setting
4. the method is repeatable and hence can be used to assess patient progress over time

1.4.2 Questions

The following questions were asked:

1. What is the accuracy and precision of the method?
2. Under what conditions (actions, range of motions, etc...) are the results considered valid?
3. What are the limitations of the method?
4. Which limitations can be overcome and how?

1.4.3 Measurement Methods - RSA & Skin Markers

Two measurement methods were used to track scapular motion. The first method, roentgen stereophotogrammetric analysis (RSA), has been used by others and has been shown to be a gold standard in measuring in vivo skeletal motion.[41,44,46,64,76] This gold standard was then used to assess the validity of the second method. The second method involves the use of skin markers to track the flexible skin surface. Whereas other authors have reported the use of skin markers, this method differs in that it uses an array of markers forming a grid on the surface of the skin.[43,53,54] The advantage of this method over other authors has been the ability to track the regional variations over the shoulder surface. By measuring the regional variation, a more complete description of the correlation between the skin surface and the underlying bone can be made.

Chapter 2

PILOT STUDY

2.1 Objective

The purpose of this study was to assess the potential for using an array of skin markers, forming a grid on the surface of the shoulder, to measure the scapular motion beneath.

2.2 Methods

The right shoulder region of one 24-year-old male subject was used. The subject was muscular. The taunt skin surface had neither wrinkles nor "sags" due to excessive loss of subcutaneous tissue. The subject had a history of shoulder instability although it is reasoned that this should have no impact in whether the skin surface can be used to infer the underlying scapular motion.

The skin surface posterior to the scapula was divided into ten patches (Figure 2-1). Infrared optoelectronic markers (Optotrak 3020, Northern Digital, Waterloo, Canada) were placed at each patch corner to track the motion of the patch (Figure 2-2). Abduction was performed in the coronal plane over a range of 145° (initial offset of ~25° at rest position to maximum abduction at ~170°). Ten abduction-adduction cycles were performed.

Also visible in Figure 2-2 are two marker carriers in the lumbar spine region used to track the thoracic frame. Two carriers were used in order to assess the magnitude of relative motion between carriers. Ideally, there would be no relative motion indicating a rigid thoracic frame with no skin motion artifact. The final carrier seen in Figure 2-2 was used to track humeral motion.

Figure 2-3 shows the anatomical coordinate systems used for motion analysis. The base frame was defined as the thoracic frame, oriented in a plane parallel to the initial position of the scapular spine (i.e. motions were computed in the scapular plane).

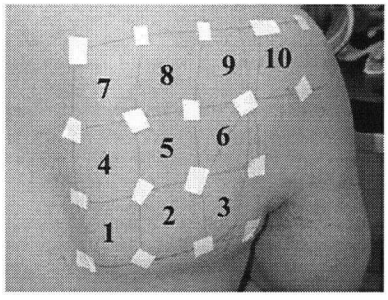

Figure 2-1: Pilot Patches
Skin surface was divided into ten patches roughly 4-cm × 4-cm each.

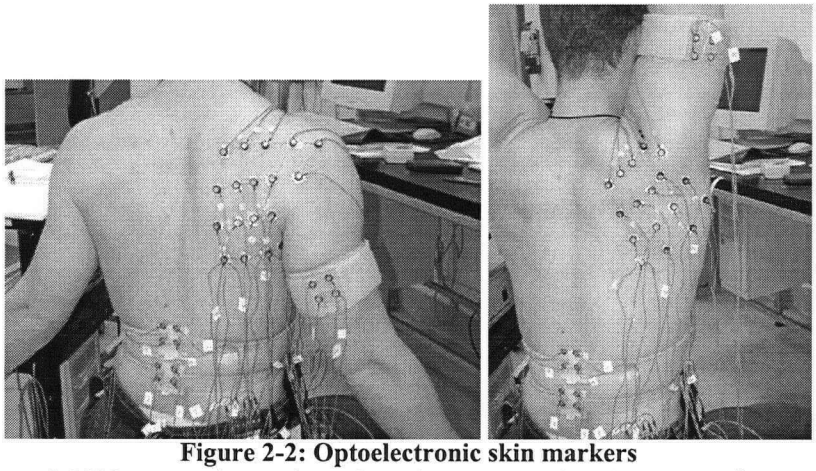

Figure 2-2: Optoelectronic skin markers
Optotrak 3020 was used to track patch motion. Range of motion was ~25° to ~170°.

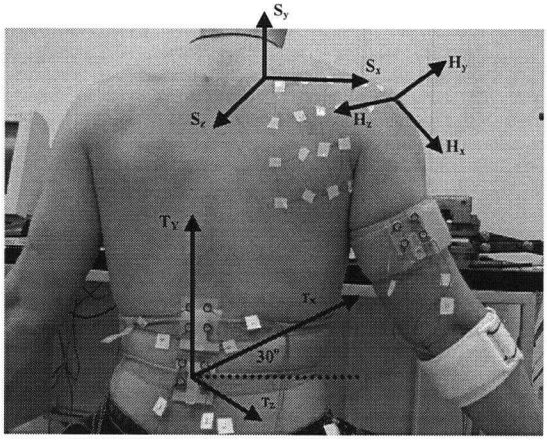

Figure 2-3: Anatomical coordinate systems

2.3 Results

Scapular tipping, or rotation about an axis parallel to the scapular spine, was found to change by ~12° (patch 10) to ~35° (patch 6) relative to the scapula's rest position (Figure 2-4). Humeral rotations about this axis showed approximately 15° of internal rotation at maximum elevation. The motion of the humerus as well as some of the patches was asymmetric (i.e. abduction pattern different from adduction pattern).

Scapular internal/external rotation, about the "vertical" y-axis, showed a more complex pattern than scapular tipping (Figure 2-5). Excluding patch 10, the range of rotation varied between ~10° (patch 9) and ~20° (patch 5). Patch 10 rotated to a maximum of ~4° and a minimum of ~-20°. Its pattern indicated that it was folding in the opposite direction of the other patches during elevation. Humeral rotations about this axis ranged over ~25°. The pattern indicated that the humerus swayed posteriorly during abduction. Near maximum abduction, the humerus swayed anteriorly before reaching maximum abduction. The adduction phase for the humerus mirrored the abduction phase. The humerus swayed posteriorly then anteriorly towards the resting position. The magnitude of the sway was not consistent between cycles. During adduction, the humerus moved at a slightly higher velocity. The motion of the patches was asymmetric.

In this test, rotation about the z-axis corresponded to the upward rotation that has been reported by Inman and others. The rotation patterns appeared the simplest and most consistent of the three axes (Figure 2-6). The change in scapular rotation ranged from ~8° (patch 1) to 65° (patch 10). The change in humeral elevation was approximately 145° from an initial 25° offset (170° maximum absolute elevation). The patterns from all patches appeared similar. Patch 1, at the most medial-inferior corner of the grid, showed very little upward rotation.

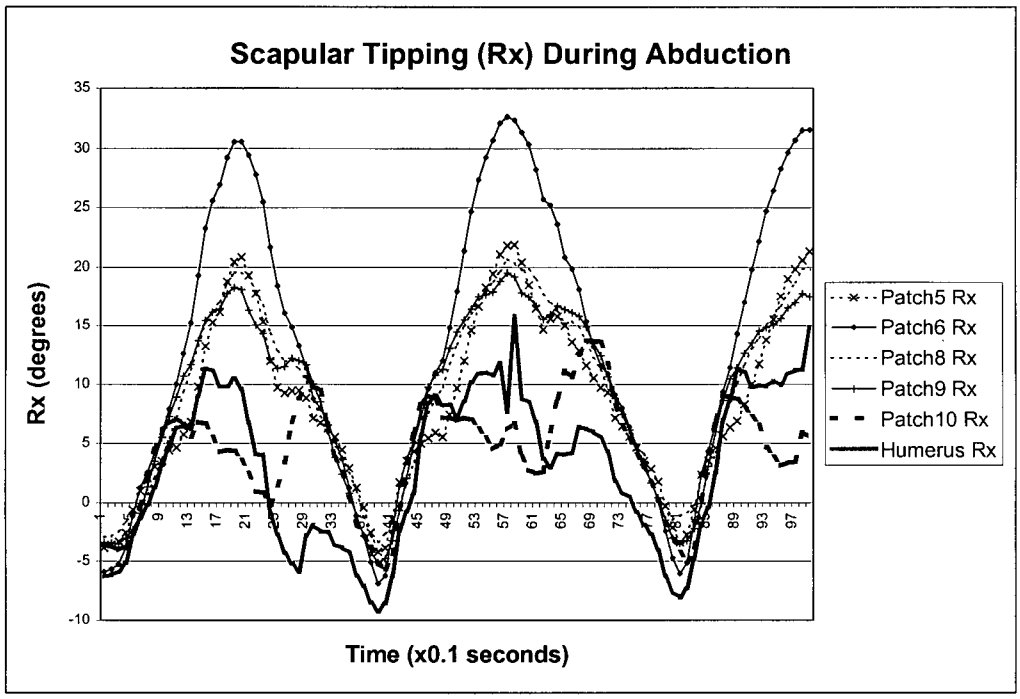

Figure 2-4: Scapular Tipping
Patch and humeral rotations about x-axis during arm elevation cycles

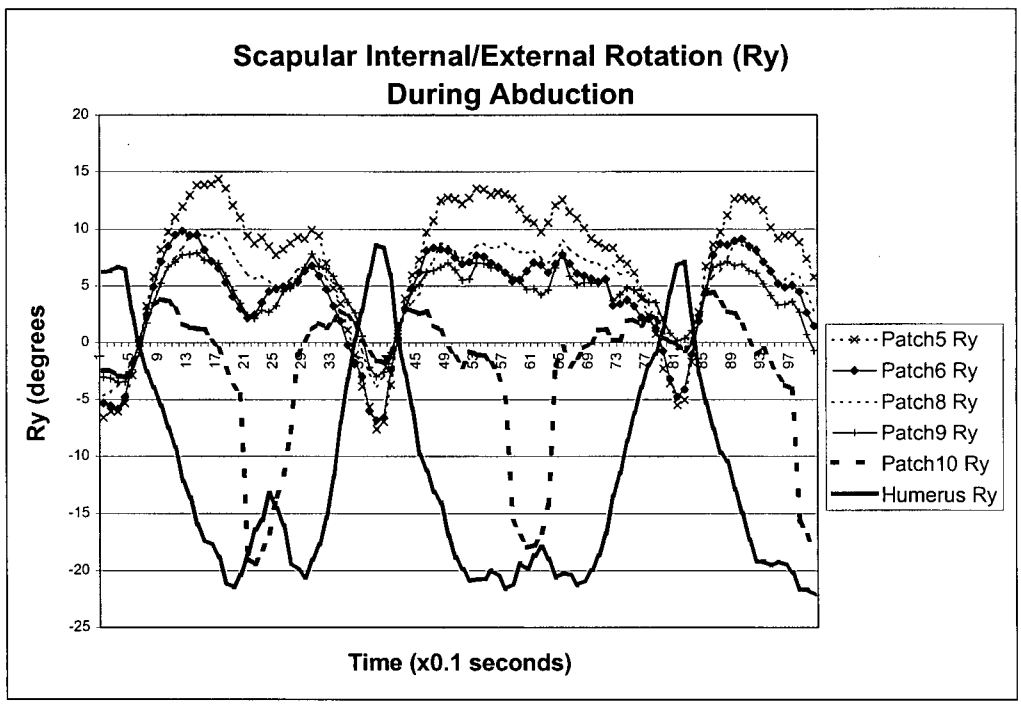

Figure 2-5: Scapular Internal/External Rotation
Patch and humeral rotations about y-axis during arm elevation cycles

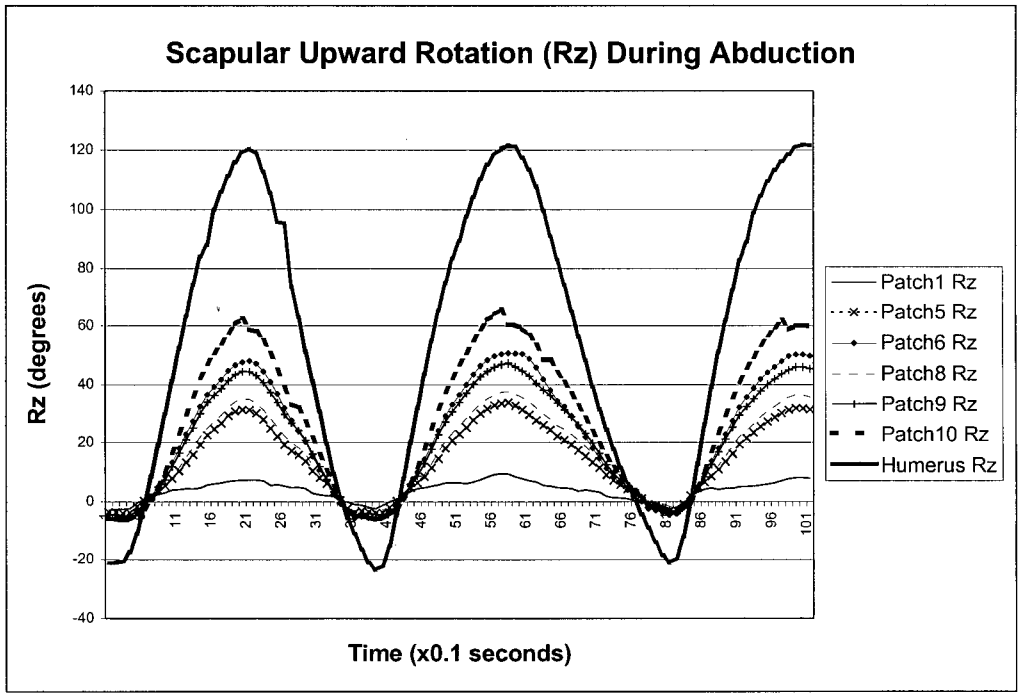

Figure 2-6: Scapular Elevation
Patch and humeral rotations about z-axis during arm elevation cycles

Figure 2-7 shows a scatter plot of scapulothoracic angles versus humeral elevations reported in the literature.[7,21,40,53,59,73,86] The *slopes* of the scatters are accurate although their exact y-intercept is imprecise due to literature inconsistencies. In Figure 2-8, the rotation angles from the inferior patches (1, 2 & 3) are plotted along with the data from the literature. By visual inspection it can be seen that the rhythm (slope) from the patches underestimated the rhythm reported in the literature. Figure 2-9 shows a similar plot from the patches on the medial and lateral periphery (patches 4, 7, 10). The rhythms from the medial patches were also lower than that reported in the literature. The rhythm from patch 10, which lay along the line of action of the humerus, significantly overestimated the ratio of glenohumeral to scapulothoracic angles. Figure 2-10 shows the rhythms from the four centrally located patches. Patches 8 and 9, which lay over the mid-section of the scapular spine, exhibited rhythms very close to that reported in literature. Patches 5 and 6 lay over the central portion of the scapular body and also showed rhythms in the ballpark of those in the literature.

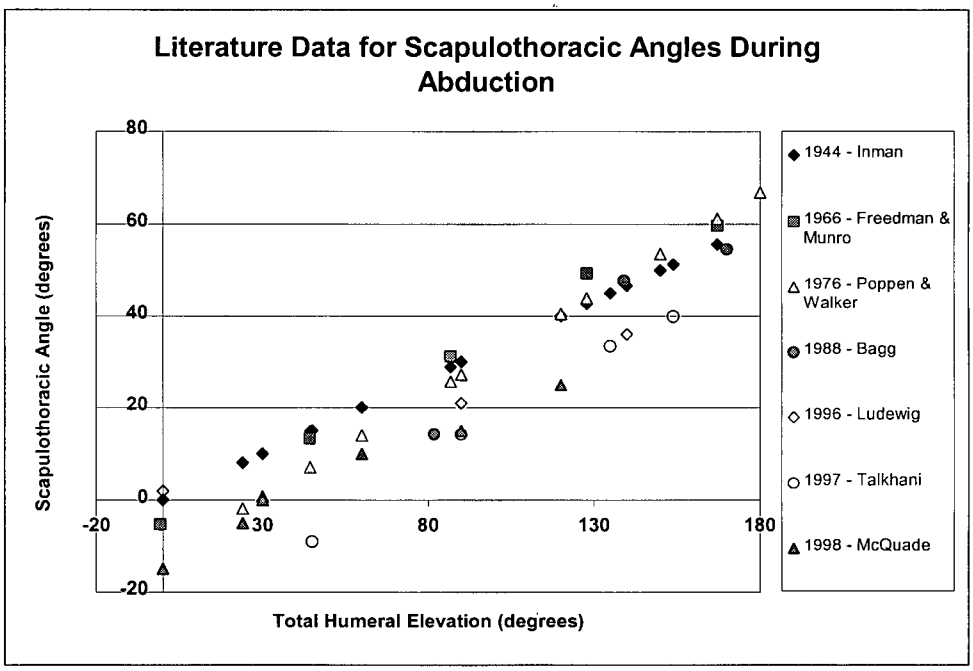

Figure 2-7: Composite of Literature Data
Scatter plot of scapulothoracic angles reported in the literature. Variations in methodology make defining the y-intercept unreliable although the slopes ("rhythms") are as reported.

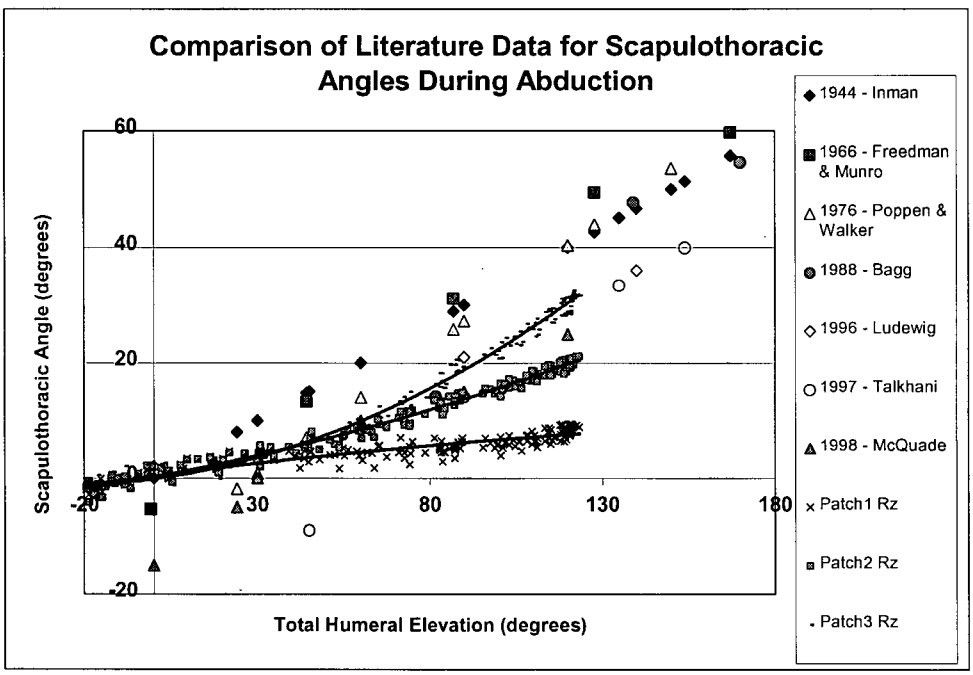

Figure 2-8: Inferior Patches & Literature
Comparison of literature data with inferior skin patches (1, 2, & 3). Scapulothoracic "rhythm" is underestimated.

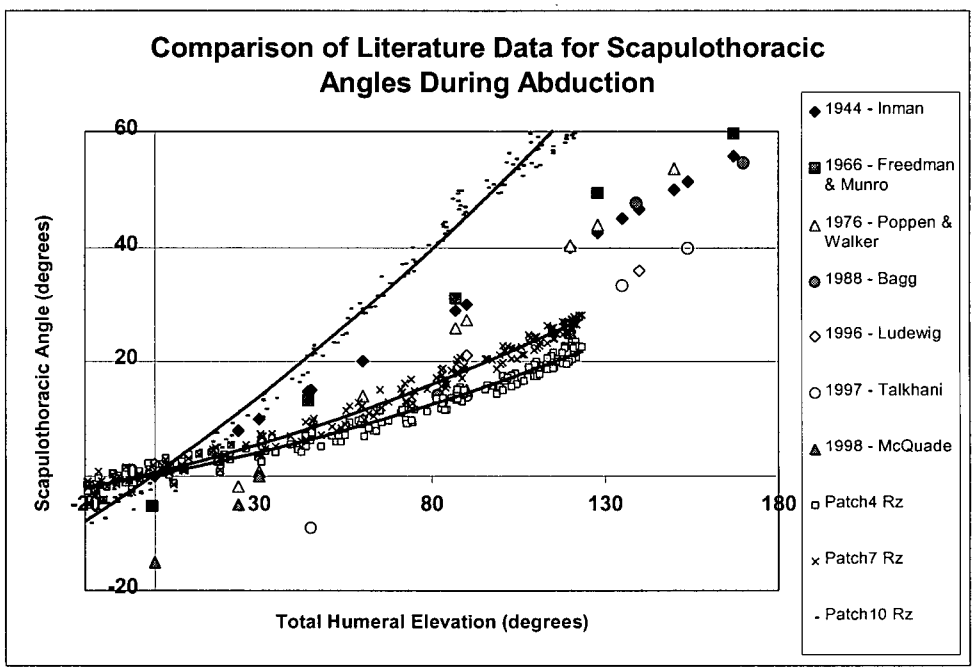

Figure 2-9: Peripheral Patches & Literature
Comparison of literature data with peripheral skin patches (4, 7, & 10). Medial patches (4 & 7) underestimated scapulothoracic "rhythm". Patch 10 on the acromion region highly overestimated upward rotation.

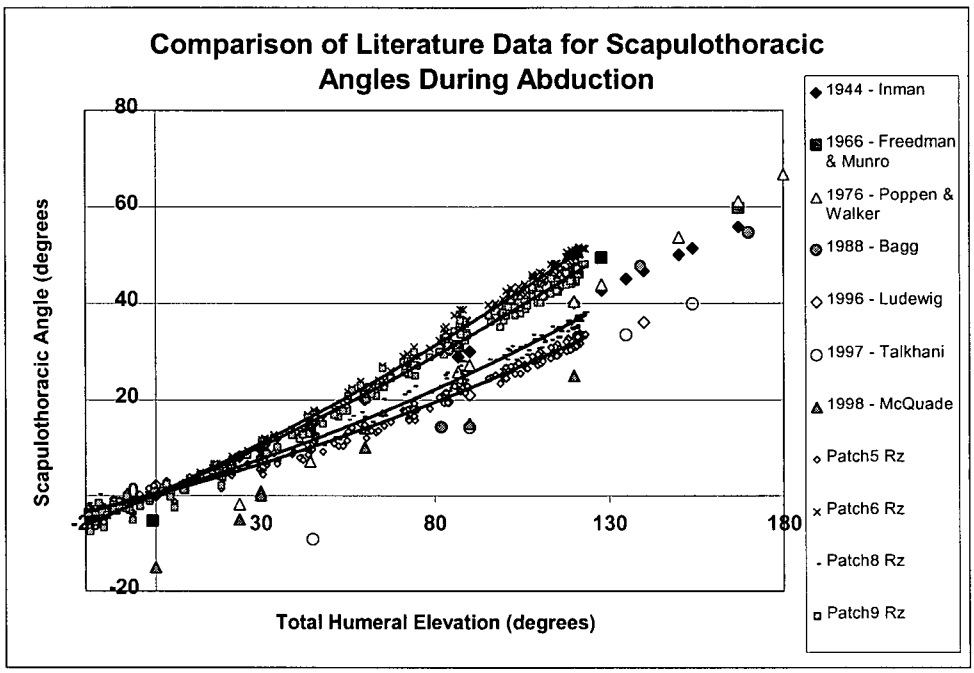

Figure 2-10: Ideal Patches & Literature
Skin patches located centrally (5 & 6) and over the scapular spine (8 & 9) exhibited scapulothoracic "rhythm" similar to that reported in the literature.

2.4 Discussion

Ranges and Patterns of Motion

Although recent studies have aimed to look at scapular motion in three-dimensions,[30,53,54,59] only Ludewig's and Lukasiewicz's groups reported the values found for all three rotation axes. Both groups used digitization of palpable surface features to measure scapular positions. Table 2-1 compares the range of motions of this pilot study to these results. The results for the range of motion about all three axes do not appear similar. There are several reasons for this. As with the study by Lukasiewicz's group, the initial position was chosen as a "rest" position. This position was several degrees of abduction (~25° in this study). Consequently, the results are difficult to compare. This phenomenon is true for all studies reported in the literature. For example, Inman and Saunders reported an upward rotation rhythm of 2:1 for abduction *beyond* the initial setting phase of 30°. Poppen and Walker conducted a similar study as Inman and Saunders but reported a 5:4 rhythm but over the *entire* range of motion. The choice of dissimilar measurement intervals further compounds the problem.

Study	Humerus ROM	tipping	internal/external rotation	upward rotation
Ludewig	0° to140°	15°	-13°	34°
Lukasiewicz	rest to rest +140°	22.8°	-7.1°	28.2°
Pilot – Patch5	rest to rest + 145°	22°	22°	37°
Pilot – Patch6	rest to rest + 145°	38°	17°	55°
Pilot – Patch8	rest to rest + 145°	12°	15°	40°
Pilot – Patch9	rest to rest + 145°	20°	12°	50°
Pilot – min	rest to rest + 145°	12°	10°	8°
Pilot – max	rest to rest + 145°	35°	20°	65°

Table 2-1: Comparison of 3-D rotations
Rotations found in this pilot study did not appear "similar" to those reported in studies by Ludewig and Lukasiewicz.[53,54] Difficulty lay in the inconsistency of coordinate definitions and the specific motions used across different studies.

The highly complex patterns observed in internal/external rotation does not support the findings of Ludewig's group. They found two internal/external rotation patterns. First, 84% of their subjects exhibited a progressive pattern of increased external rotation

during elevation. For the remaining 16%, the scapula remained in the same plane. In this study, the scapula's attitude in internal/external rotation appeared to fluctuate throughout the elevation. Ludewig's group found internal/external rotation to exhibit the highest variability across subjects. Hence the pattern in this study is probably not a general pattern but one unique to this particular subject.

Rhythm

As expected, the patches on the periphery of the scapula showed poor correlation with the literature. In the region located centrally on the scapula, the results showed reasonable correlation with data reported in the literature.

It should be noted that the correlation is based on the similarity of the rhythm (slope) and not on whether the data points overlap. The reason for this is due to the shortcomings of the literature data reported. In general, although rhythm is often discussed, inconsistent initial positions, inconsistent coordinate definitions, normalization of data to maximum elevation, and varying intervals of measurement, make the precise y-axis intercept of the rhythm data unreliable.

The measured scapulohumeral rhythm showed a high variability depending on the patch's position on the skin surface. This demonstrates the regional sensitivity of a skin marker approach. An analysis of regional sensitivity of skin markers has not been addressed in the literature found to date.

Most of the palpation approaches have used the scapular spine as a major landmark. The results support this intuitive notion that since there is minimal soft tissue covering the scapular spine, skin motion artifact is also minimized in this region making it a very reliable location for surface markers (Figure 2-11).

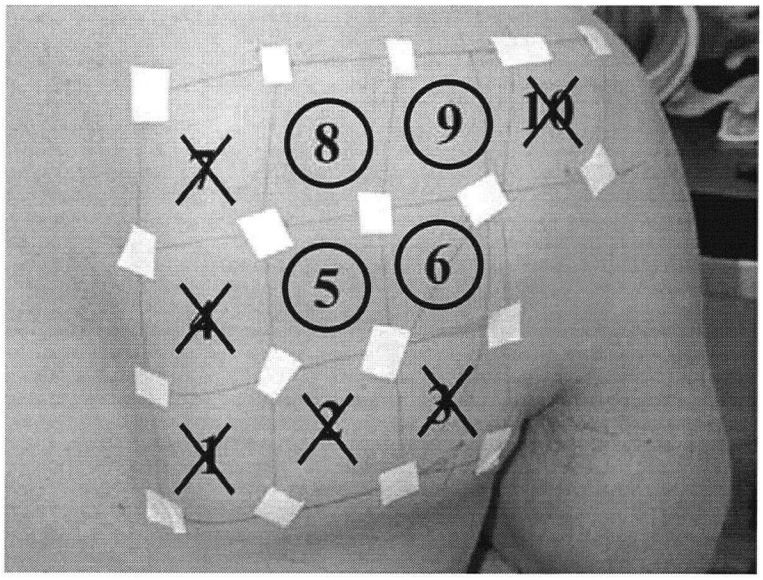

Figure 2-11: Skin Patch Summary
Skin markers in region of the mid-section of the scapular spine and centrally on the scapular body showed scapulohumeral rhythm which correlated well with that reported in the literature.

Studies by McQuade and Karduna have used acromion surface markers to track the scapula. The results from patch 10, which lay along the line of action of the humerus, showed significant overestimation of the scapular upward rotation. A similar result was also observed by Karduna's group in a recent study accepted for publication.[42]

A number of future questions remain to be answered.

1. What is the deformation of each patch?
2. What is the influence of the patch dimensions?
3. At what patch dimensions are deformations negligible?
4. What is the relationship between soft tissue thickness and skin artifact?

2.5 Conclusion

This pilot study showed that scapulohumeral rhythm obtained from tracking grids of skin markers was able to yield data in the ballpark of that reported in the literature. The results also showed that skin marker methods are highly sensitive to surface location.

The study also confirmed that the mid-section of the scapular spine was probably the best location for skin marker placement. Finally, a number of standards are still required in the investigation of scapular motion for meaningful comparisons of data to be achieved.

Chapter 3

STEREOPHOTOGRAMMETRY

The purpose of stereophotogrammetry is to reconstruct three-dimensional coordinates from two or more two-dimensional images. The images may be obtained in various ways - cameras, video cameras or in the case with roentgen stereophotogrammetry, x-rays. There are several methods of analyzing the two-dimensional images to determine the object's three-dimensional coordinates. This chapter discusses a few methods described in the literature. Of these methods, two will be emphasized. First is the direct linear transformation (DLT) frequently used in camera and video stereophotogrammetry. Second is the method described by Selvik and used frequently in roentgen stereophotogrammetry.

3.1 Stereophotogrammetric Preliminaries

Stereophotogrammetric analysis involves two steps – calibration and reconstruction. Initially, prior to calibration, there are two sets of unknowns. First is the relationship (distance, orientation) between the three-dimensional laboratory coordinate system and the two-dimensional image coordinate systems. The second set is the three-dimensional coordinates of the object being imaged. If one set is known, the other set can be calculated (schematic in Figure 3-1).

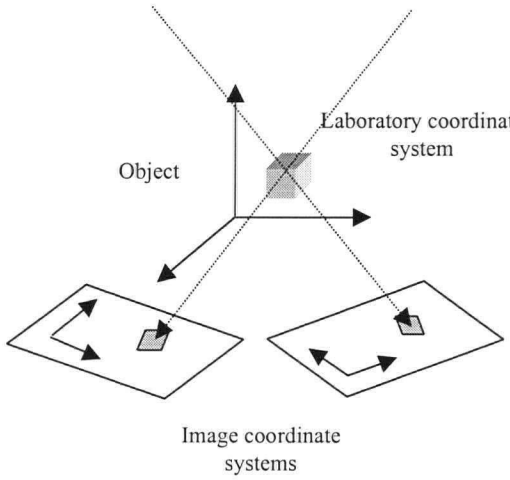

Figure 3-1: Stereophotogrammetric Setup
Schematic showing three-dimensional laboratory coordinate system, a pair of two-dimensional image coordinate systems, and a three-dimensional object.

In calibration, an object with *known coordinates* is imaged and hence the *unknown relationship* between the laboratory and image coordinate systems can be calculated. The object used during calibration is called a calibration cage and typically consists of a balanced distribution of markers called calibration points (or calibration markers).

In reconstruction, an arbitrary object is imaged and the *known relationships* previously solved in calibration are used to calculate the *unknown coordinates* of the object. As long as the relationships remain unchanged, the same calibration data can be used in subsequent reconstruction procedures. If for example the camera/film were moved, it would be necessary to recalibrate.

3.2 Overview of Stereophotogrammetric Analysis Methodologies

The relationship between the laboratory and image coordinate systems can be formally described by the ten parameters listed in Table 3-1. There are nine geometrical parameters and a scale factor (λ) between the laboratory space and image plane. The node point is defined as the point that is collinear with all object and image points. The principal distance is the perpendicular distance from the node point to the image plane. Figure 3-2 shows the definitions of the parameters in Table 3-1.

Factor	Significance
d	principal distance (perpendicular distance between node point and the image plane)
u_o,v_o	coordinates of the node point in image coordinate system
X_o,Y_o,Z_o	coordinates of the node point in lab coordinate system
$\phi_x, \phi_y, \phi_z,$	orientation between the lab coordinate system and the image coordinate system
λ	scale factor between the lab coordinate system and the image coordinate system

Table 3-1: Stereophotogrammetric Parameters
Ten unknown parameters in an object/image pair. Varying stereophotogrammetric analysis algorithms use different approaches to determine these parameters which are subsequently use in 3-D reconstruction.

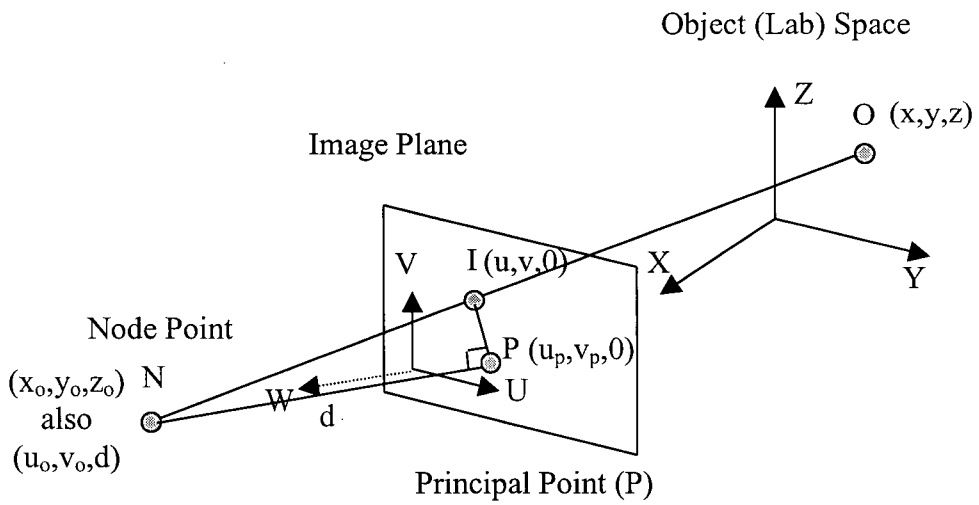

Figure 3-2: Image and Lab Coordinate Systems
This figure shows the relationship between the image and laboratory coordinate systems. Node point (N) is, by definition, the point that is collinear with both the image point (I) and the object point (O). The node point can be represented in the laboratory coordinate system (x_o,y_o,z_o) and the image coordinate system (u_o,v_o,d). The principal distance (d) is the shortest distance from the node point to the image plane and hence intersects the plane perpendicularly at the principal point (P). [ii]

The varying methodologies differ in their approach of computing the parameters and the subsequent, inverse computation, used in the reconstruction phase. Some goals are

[ii] DLT diagrams and derivations in this chapter are based on the information found at www.kwon3d.com.

to improve computational speed and to increase flexibility in terms of the imaging setup. Others are to improve accuracy by compensating for lens distortion or by reducing the need to extrapolate beyond the calibrated space.

Several methods have been used to perform three-dimensional reconstruction in camera/video stereophotogrammetry.[11,15,23-25,28,32,63] One of the earliest methods, and also the most often currently used, is the direct linear transformation (DLT) proposed by Abdel-Aziz and Karara in 1971. The basic DLT method involves converting the ten unknown parameters into eleven intermediate parameters. The equations for the eleven intermediate parameters are linear and easily solved by least-squares optimization. The DLT is discussed in more detail in section 3.3.

Other methods have also been proposed. In 1988, Hatze presented a modified DLT (MDLT) algorithm. The modification involved the addition of one constraint equation to remove the over-determinacy of the original DLT. The results showed an order of magnitude improvement in accuracy. One major flaw in Hatze's method was that the reconstruction object was identical to the calibration object. Other authors since then have discussed problems with using the same object for calibration and reconstruction.[16,32] The calibration parameters are tuned to the calibration object. Using this same object to test reconstruction accuracy causes overly optimistic results. The performance of MDLT for unknown object coordinates has not been reported.

A method called non-linear transformation (NLT) has been proposed as a flexible alternative to DLT.[32] Instead of a fixed calibration cage, NLT uses a pole moved throughout a given space to calibrate it. This method allows for the calibration of an arbitrarily large volume without the need to manufacture large, costly and impractical calibration cages. Hinrichs and McLean compared NLT to DLT in 1995. They found that below 50% of extrapolation, DLT was clearly more accurate than NLT. NLT showed comparable or better results than DLT beyond 100% extrapolation.

A self-calibration method called maximum likelihood estimation was presented by Muijtjens et al. in 1999.[63] No calibration cage is required. The method assumes a symmetric stereo camera setup and solves for the calibration parameters using the symmetry as a guide. The major drawback in this approach is that it requires prior knowledge of the image distance and the intersection of the camera axis with the image plane. These quantities are generally difficult to determine to a high accuracy and hence greatly limit the usefulness of this method.

Although others have shown improvements with new algorithms the results have not been compelling. Consequently, DLT remains the most widely used technique in video stereophotogrammetry. For this reason, DLT and another method developed specifically for *roentgen* stereophotogrammetry will be the focus of the remainder of this chapter.

3.3 Direct Linear Transformation

DLT was initially presented by Abdul-Aziz and Karara in 1971. In 1975, Marzan and Karara presented an algorithm for its computation. The relationship between the laboratory space and the image plane begins with the node point previously defined. Recall the node point (also called the projection centre) is defined as the point that is collinear with the image point and the object point. Figure 3-3 shows how the node point satisfies the collinearity condition with all image/object point pairs.

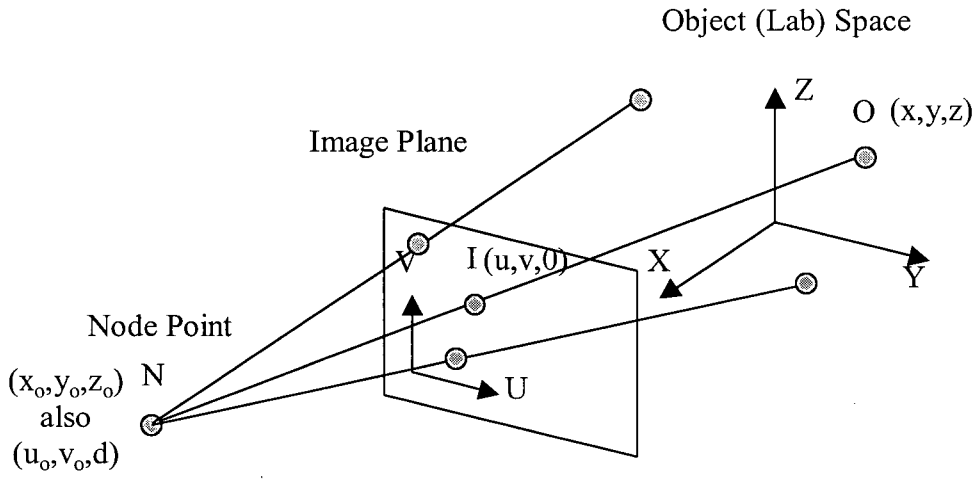

Figure 3-3: The Node Point
The node point (or projection centre) is collinear with each pair of object and corresponding image points. This is the collinearity condition.

Consider the object frame.[iii] (Figure 3-4) The node point has unknown coordinates $(x_0,y_0,z_0)_{object\_frame}$. A vector, **A**, can be defined in the laboratory coordinate system from the node point to the object point $(x-x_0,y-y_0,z-z_0)_{object\_frame}$.

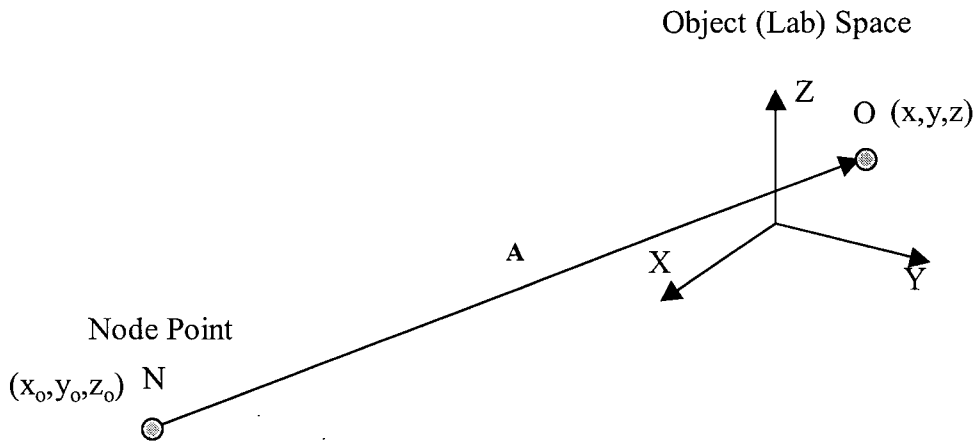

Figure 3-4: Vector A
Define a vector, **A**, in the lab coordinate system (x,y,z) that goes from the node to the object point.

[iii] In this text, frame, frame of reference, and coordinate system are used interchangeably.

Consider the image frame (Figure 3-5). Recall the principal point is the point of perpendicular intersection of a line from the node point to the image plane. Now the node point can be defined with unknown coordinates $(u_0,v_0,d)_{image\_frame}$. A vector, **B**, can be defined in the image frame from the node point to the image point $(u-u_0,v-v_0,-d)_{image\_frame}$.

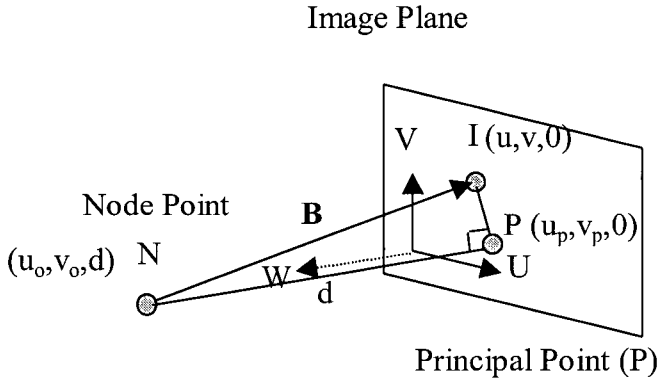

Figure 3-5: Vector B
Define a vector, **B**, in the image coordinate system (u,v,w) that goes from the node to the image point.

The critical step in relating the two frames is the recognition that vectors **A** and **B** are related by just a scale factor. Hence we have:

$$\mathbf{B}_{\text{image\_frame}} = c\mathbf{A}_{\text{image\_frame}} = c\mathbf{R}_{\text{image\_frame\_to\_object\_frame}}\mathbf{A}_{\text{object\_frame}}$$

where **R** is a rotation matrix describing the relative orientation between the image and object frames of reference. Substituting for the vectors and expanding gives:

$$\begin{bmatrix} U-U_o \\ V-V_o \\ -d \end{bmatrix} = c\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{bmatrix} x-x_o \\ y-y_o \\ z-z_o \end{bmatrix}$$

$$\begin{aligned} U-U_o &= c\left[r_{11}(x-x_o + r_{12}(y-y_o) + r_{13}(z-z_o)\right] \\ V-V_o &= c\left[r_{21}(x-x_o + r_{22}(y-y_o) + r_{23}(z-z_o)\right] \\ -d &= c\left[r_{31}(x-x_o + r_{32}(y-y_o) + r_{33}(z-z_o)\right] \end{aligned}$$

where c is a scale constant:

$$c = \frac{-d}{r_{31}(x-x_o) + r_{32}(y-y_o) + r_{33}(z-z_o)}$$

Accounting for the differing units between digitizing units and real-life units:

$$\begin{aligned} U-U_o &= \lambda_u(u-u_o) \\ V-V_o &= \lambda_v(v-v_o) \end{aligned}$$

The final step involves algebraic manipulation of the variables into the form:

$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}$$
$$v = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1}$$

where L_1 to L_{11} are the intermediate "DLT Parameters"

$$L_1 = \frac{u_o r_{31} - d_u r_{11}}{D} \;,\; L_2 = \frac{u_o r_{32} - d_u r_{12}}{D} \;,\; L_3 = \frac{u_o r_{33} - d_u r_{13}}{D}$$
$$L_4 = \frac{(d_u r_{11} - u_o r_{31})x_o + (d_u r_{12} - u_o r_{32})y_o + (d_u r_{13} - u_o r_{33})z_o}{D}$$
$$L_5 = \frac{v_o r_{31} - d_v r_{21}}{D} \;,\; L_6 = \frac{v_o r_{32} - d_v r_{22}}{D} \;,\; L_7 = \frac{v_o r_{33} - d_v r_{23}}{D}$$
$$L_8 = \frac{(d_v r_{21} - v_o r_{31})x_o + (d_v r_{22} - v_o r_{32})y_o + (d_v r_{23} - v_o r_{33})z_o}{D}$$
$$L_9 = \frac{r_{31}}{D} \;,\; L_{10} = \frac{r_{32}}{D} \;,\; L_{11} = \frac{r_{33}}{D}$$
$$D = -(x_o r_{31} + y_o r_{32} + z_o r_{33})$$
$$d_u = \frac{d}{\lambda_u} \;,\; d_v = \frac{d}{\lambda_v}$$

The equations can now be easily expressed in matrix form:

$$\mathbf{Ax} = \mathbf{B}$$

Ideally,

$$\mathbf{Ax}\text{-}\mathbf{B} = 0$$

However, due to measurement errors and other sources of uncertainty,

$$\mathbf{Ax}\text{-}\mathbf{B} \neq 0$$

The solution for **x** that minimizes $(\mathbf{Ax-B})^2$ is the least-squares solution. (i.e. it is the solution that is as close as possible to the ideal solution of zero)

Each calibration point contributes two equations hence a minimum of six points are needed in calibration to solve for the eleven unknown DLT parameters ($L_1...L_{11}$). During reconstruction, each image of a particular object point contributes two equations hence a minimum of two images are required to solve the three unknown coordinates (x,y,z) of the object.

Marzan and Karara have proposed DLT algorithms of up to sixteen parameters. The purposes of the additional parameters are to account for non-linearities in the lens and camera systems. As this project deals with x-rays, which have been shown by Hallert and Hollender to not undergo significant refraction, the 11-parameter DLT method has been used.[44,79]

3.4 Selvik's Method for RSA

During the time that DLT was under development, another three-dimensional reconstruction method was developed by Selvik and applied to roentgen stereophotogrammetric analysis or RSA. Since then, RSA's use in orthopaedics as a highly accurate, in vivo, three-dimensional measurement method has become widespread. In a review by Karrholm, it was reported that between 1972 and 1989, roughly 20,000 tantalum markers had been implanted in about 2000 patients with no adverse effects.[1,44] The implantation of spherical (typical diameters: 0.5mm, 0.8mm, 1.0mm) tantalum markers into bone serve two purposes. First, their high radio-opacity make them accurately identifiable landmarks from image to image. Second, tantalum has an atomic number of 73 making it almost ten times more radio-opaque than bone. This large differential allows the x-ray dose to the patient to be reduced without a significant loss of image contrast. Refer to the appendix on x-ray radiography for x-ray imaging considerations.

3.5 Selvik's Methodology

Selvik's method begins with the principle of a central projection from the x-ray source, through the object space onto the image plane. The method is similar to DLT discussed in the previous section. The central projection is the node point discussed previously.

The calibration points are divided into two groups called fiducial points (marks) and control points (marks). The fiducial points lie in a plane called the fiducial plane. The control points lie in a plane called the control plane. The x-ray source, a fiducial point, and the image of the fiducial point all lie on the same line.

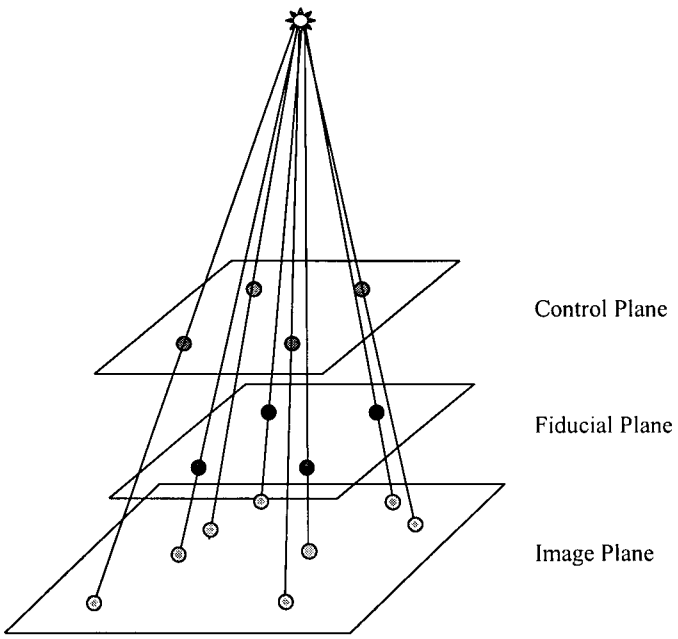

Figure 3-6: Central Projection
The central projection from the source through two planes of calibration points to the image plane. The x-ray source, calibration point (control or fiducial) lie on the same line – the collinearity condition.

The method thus far appears identical to that of DLT with the exception of some terminology. It is in fact the same collinearity condition as before and hence the same resultant relationship holds and is repeated below.

$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}$$
$$v = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1}$$

However, if only the fiducial plane is considered, or only the control plane is considered, the control points in the laboratory coordinate system lie strictly in a plane and it is possible to set z = 0. The expression thus simplifies to:

$$u = \frac{L_1 x + L_2 y + L_3}{L_7 x + L_8 y + 1}$$
$$v = \frac{L_4 x + L_5 y + L_6}{L_7 x + L_8 y + 1}$$

where the subscripts have been renumbered to reflect the reduction in the number of parameters from 11 to 8. This is the form of the collinearity condition presented by Selvik.[79] It is a two-dimensional direct linear transformation. At this point the two methods diverge.

The use of the 2D-DLT means that the image coordinates can only be transformed back to a two-dimensional plane. In order to recover the three-dimensional coordinates, additional steps are required in both the calibration and reconstruction process.

The following describes Selvik's calibration process using a calibration cage with markers in a fiducial plane and a control plane. First, known coordinates of the fiducial markers are combined with their images to compute the eight 2D-DLT parameters which describe the relationship between the object plane and the image plane. Second, the known 2D-DLT parameters are used with the image of the control points to

reconstruct (note: this is still part of the overall calibration) the intersection of the control point x-rays with the fiducial plane. Third, the coordinates of the control points in the fiducial plane are combined with the known coordinates of the control points in the control plane to solve for the coordinates of the x-ray source with respect to the fiducial plane. The x-ray source is the intersection of the bundle or rays passing through the control points in the control plane and their transformed locations in the fiducial plane. The intersection is solved by least-squares optimization. The relationship between the image coordinate system and the laboratory coordinate system is now known (2D-DLT parameters) as well as the location of the node point (i.e. the x-ray source).

In reconstruction, the image points are first transformed to the fiducial plane. The three-dimensional coordinates of a point are determined by finding the intersection of the lines, again solved by least-squares, from the image point in the two fiducial planes to their respective x-ray sources.

3.6 Discussion of DLT and Selvik's Method

Selvik's method can be considered a less general version of DLT. Although Selvik's expression has only eight parameters and thus appears simpler, it is at the expense of a more complicated overall methodology. DLT uses two least square-optimization steps – once in calibration, once in reconstruction. Selvik's method uses four least-squares optimization steps – two in calibration (2D-DLT and determination of x-ray source location) and two in reconstruction (inverse 2D-DLT and determination of three-dimensional intersection).

What is the influence of performing the calibration and reconstruction in more optimization steps? This question, along with the issue of calibration cage geometry, is the focus of the next section, RSA Simulation.

Chapter 4

RSA SIMULATIONS

There were two key motivations for conducting RSA simulations. The first was to determine which of two commercially available calibration cages (Figure 4-1) produced more accurate reconstruction results. The second was to compare the reconstruction accuracy of DLT and Selvik's stereophotogrammetric analysis method.

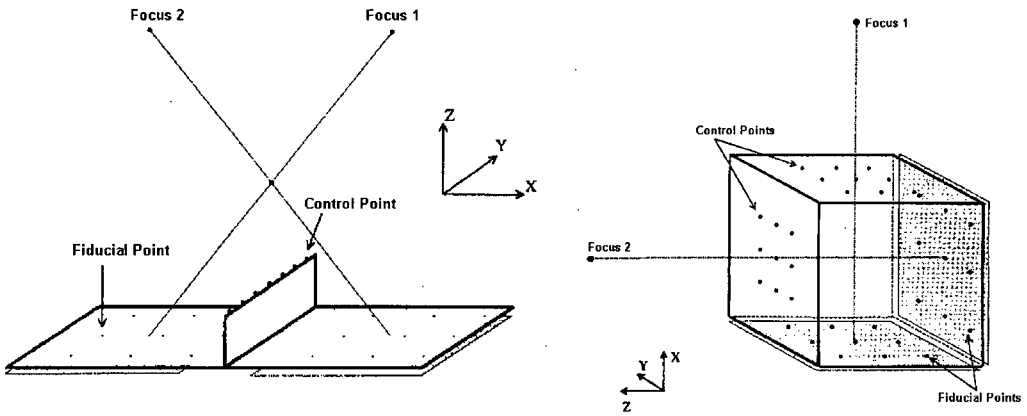

Figure 4-1: Calibration Cages
Commercially available calibration cages designed for RSA using Selvik's stereophotogrammetric analysis methods. Projective cage (left) where the two film cassettes lie in the same plane and biplanar cage (right) where the two film cassettes are at 90° to each other. Diagrams were modified from Yuan & Ryd.[100]

In addition, the simulations could be used to address other issues such as the influence of various parameters on reconstruction accuracy. Stereophotogrammetric exposures involve a number of variables relating to the geometry of the setup, the configuration of the calibration markers on the cage, and the measurement error. Specific geometric and cage variables are listed below.

Geometric Variables

- distance of x-ray source to film
- distance of calibration cage to film
- distance/position of object to calibration cage
- orientation of films

Calibration Cage Variables

- # calibration markers (also called control and fiducial points) on cage
- distribution of calibration markers
- volume enclosed by calibration markers
- accuracy of knowing calibration marker coordinates

The simulation was divided into two parts. Part I dealt with the primary objective of comparing the two cages and the two algorithms. Part II assessed the influence of various parameters on reconstruction accuracy.

4.1 Part I – Calibration Cage and Analysis Algorithm

The calibration cages were designed for RSA using Selvik's stereophotogrammetric analysis methods. The first cage is referred to as a projective cage. The cage is designed for RSA of the hip, spine, and shoulder. The cage is typically placed beneath the subject and is visible in all stereo exposures. The second cage is referred to as a biplanar cage. The cage is designed for RSA of the knee. The knee is typically place inside the cage and is thus surrounded by the calibration markers.

For the projective cage, where the subject is above the essentially flat cage, the question arises as to how much inaccuracy is introduced by extrapolation beyond the region surrounded by the calibration markers? It is often noted in general stereophotogrammetric literature (i.e. non-RSA) that for highest accuracy, the calibration markers should evenly surround the volume being imaged and extrapolation beyond this volume should be avoided.[16,18,24,28,32]

4.1.1 Part I – Objective

The objective of this set of simulations was to determine which commercial calibration cage should be acquired for the RSA system under development and to determine

which algorithm, DLT or Selvik's, should be used to perform the stereophotogrammetric analysis. Reconstruction accuracy was used to compare the combinations of cage and analysis method.

4.1.2 Part I – Methods

4.1.2.1 RSA Graphic User Interface (RSAGUI)

A simulation environment was implemented in Matlab (The MathWorks Inc., Natick, MA). The environment provided functionality for both simulation and actual analysis of real data. Some general features are listed below:

1. define calibration cages and hypothetical objects
2. generate simulated x-ray exposures of cage and objects
3. specify x-ray source and film positions and orientations
4. control measurement error of calibration points and measured image points
5. render the stereophotogrammetric setup in three-dimensions for visual inspection
6. apply DLT and Selvik's analysis methods for calibration and reconstruction

Figure 4-2 and Figure 4-3 show some of the interface of RSAGUI. The Matlab code for RSAGUI is found in the appendices.

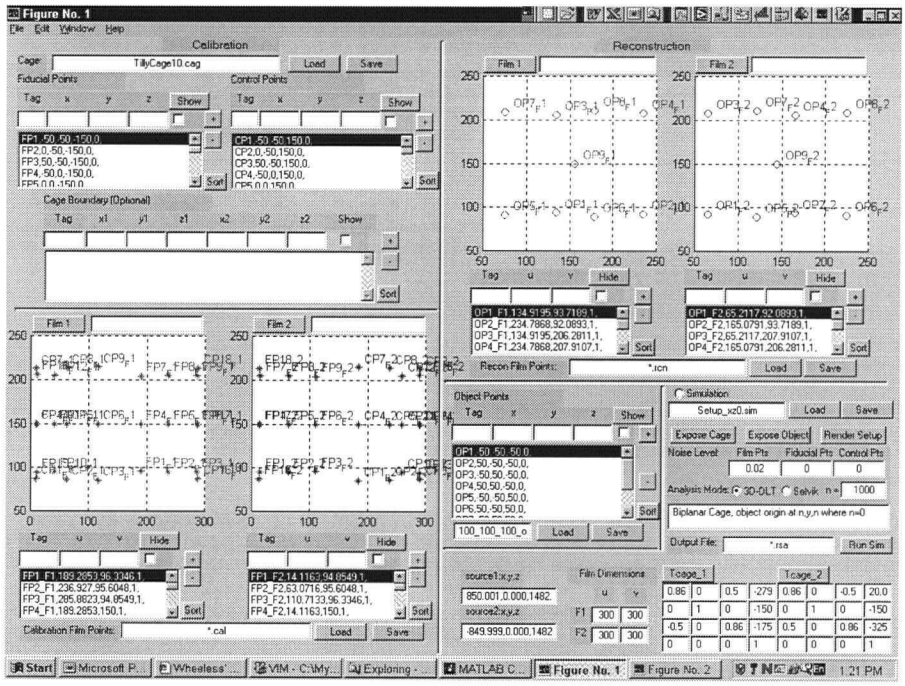

Figure 4-2: RSAGUI Window

Functionality for simulation and real data analysis.

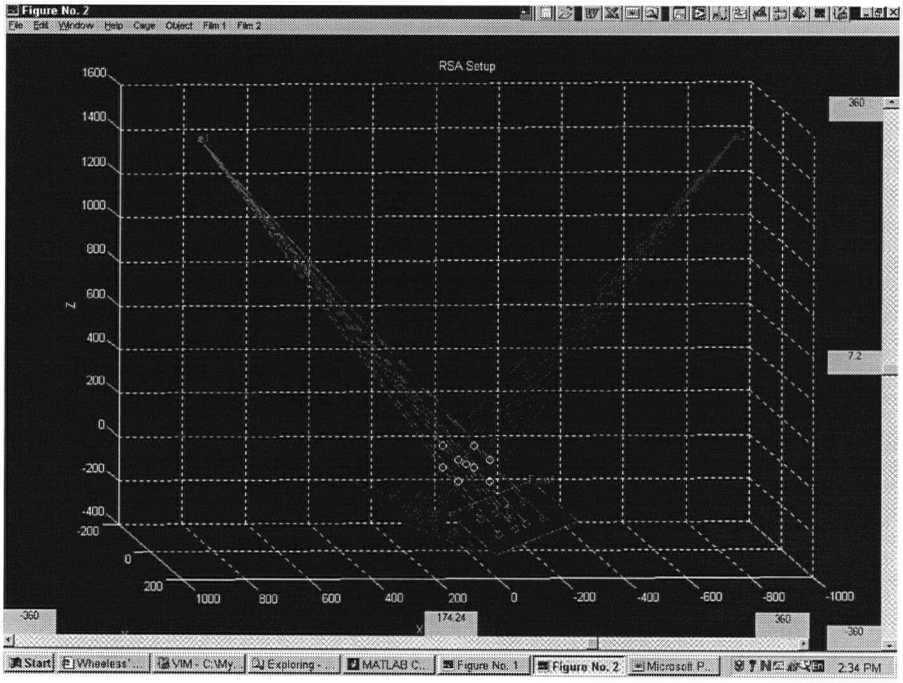

Figure 4-3: RSAGUI Rendering

RSAGUI provides three-dimensional rendering of stereophotogrammetric setup for visual inspection.

Two separate underlying analysis engines were integrated into RSAGUI. The core of the DLT analysis was performed by the Kinmat toolbox available from the International Society of Biomechanics (http://isb.ri.ccf.org). Selvik's analysis method was implemented using 2D-DLT components also available from isb.ri.ccf.org and custom routines written to complete the analysis method.

4.1.2.2 Simulated Calibration Cages and Object

Two virtual calibration cages were created for the simulation. They approximated the projective and biplanar cages available commercially. The projective cage (Figure 4-4) was designed such that the two x-ray cassettes lay in the same plane. There were nine fiducial markers, covering a 30.5 cm^2 x 16.5 cm^2 area, associated with each film. There were seven control markers lying symmetrically between the two films 21-cm above the fiducial plane. Figure 4-5 shows how the oblique exposure would image the control markers and the fiducial markers.

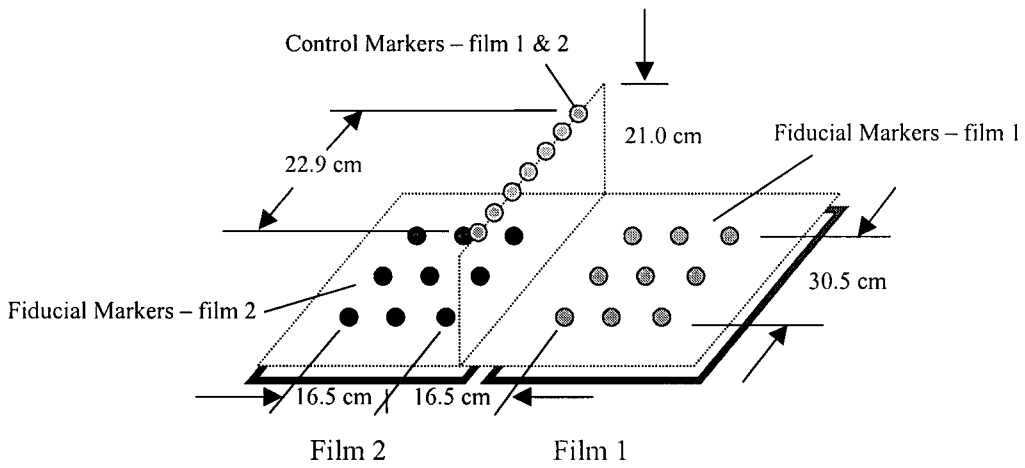

Figure 4-4: Virtual projective cage
X-ray cassettes lie in the same plane beneath the fiducial markers.

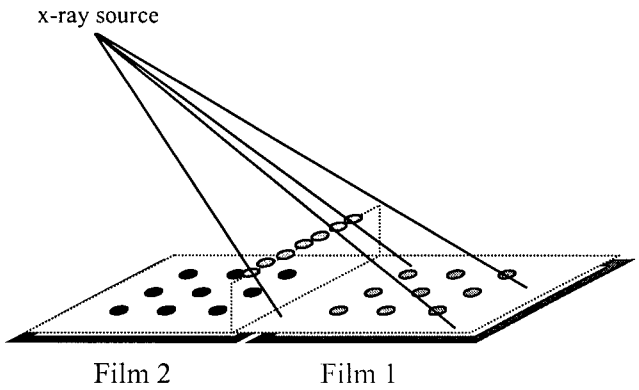

Figure 4-5: Projective Cage Exposure
Schematic of x-rays from source 1 imaging the control markers and the fiducial markers.

The biplanar cage (Figure 4-6) was designed such that the two x-ray sources and films were at 90° to each other. There were nine calibration markers on each of the two fiducial planes and each of the two control planes. In Selvik's methodology, the exposure is such that only control plane 1 markers and fiducial plane 1 markers appear on film 1 and only control plane 2 markers and fiducial plane 2 markers appear on film 2 (Figure 4-7). This exposure of only two planes can also be analyzed with DLT. However, DLT is not restricted to a planar analysis. For this cage, the exposure of all thirty-six calibration markers on film 1 and all thirty-six calibration markers on film 2 was also analyzed with DLT.

To compare the DLT and Selvik methods, a virtual object consisting of nine points was imaged in the simulations. The points were distributed at the vertices of a 10-cm x 10-cm cube with one marker at the centroid of the cube (Figure 4-8). The dimensions of the cube were chosen such that its position could be varied within the calibration space. This allowed a continuum of reconstruction accuracy to be established from interpolation within the calibration space to extrapolation beyond the calibration cage's boundaries.

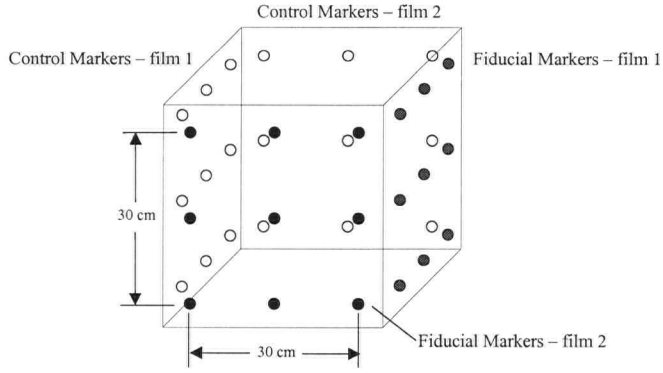

Figure 4-6: Virtual Biplanar Cage
X-ray cassettes (not shown) lie behind the two fiducial planes at 90° to each other.

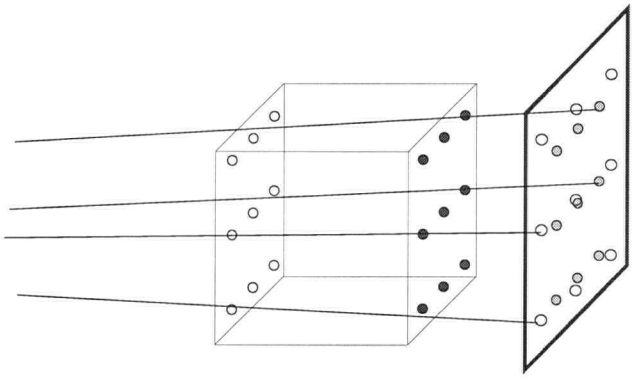

Figure 4-7: Biplanar Cage Exposure
Sample path of x-rays from source 1 imaging control and fiducial markers. In Selvik's analysis method, only 2 planes (control and fiducial) are used with a given film. Markers for source 2 have been omitted for clarity.

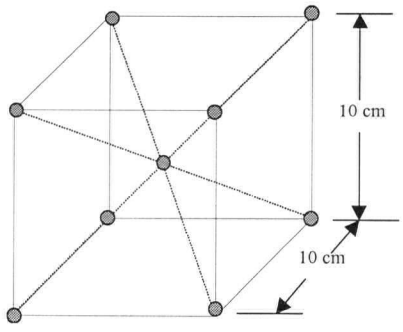

Figure 4-8: Virtual object
Dimensions were selected such that object's position could be varied inside the calibrated space. This enabled interpolation and extrapolation reconstruction accuracy to be compared.

4.1.2.3 Simulation Trials

An "ideal test" using the biplanar cage and zero measurement error was used to ensure confidence that the analysis engines were functioning properly. Zero measurement error should result in zero reconstruction error.

A total of five sets of simulations were conducted to compare the two calibration cages and two analysis methods (Figure 4-9). The projective cage was used with Selvik's analysis method (set 1) and DLT (set 2). The biplanar cage was used with Selvik's analysis (set 3) and DLT (set 4). To understand the distinction of the fifth set, recall that in Selvik's analysis method, only two planes of calibration markers (fiducial and control) are used for calibration. DLT can also use this biplanar data for calibration (set 4). However, DLT can also use all calibration markers on the entire cage for calibration. (i.e. markers on all four planes) This set has been labeled "Full Cage + 3D-DLT" (set 5).

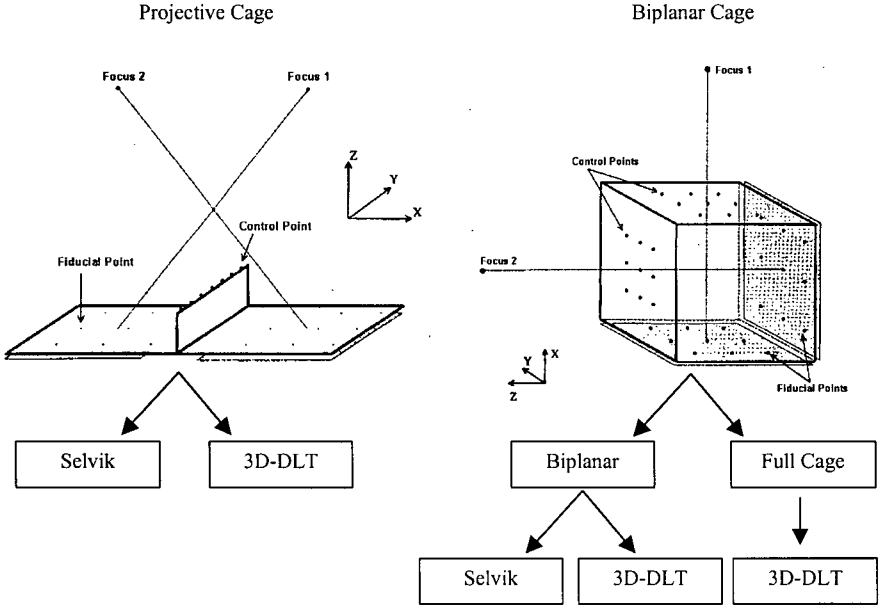

Figure 4-9: Simulation Sets
Five simulations were conducted to compare the two calibration cages and two analysis algorithms. For the biplanar cage, Selvik's analysis method only uses two planes for calibrating each film. 3D-DLT calibration can use this biplanar data, however, it can also use the full cage (i.e. four planes in 3D) for calibration. This special case constitutes the fifth simulation set. Diagrams were modified from Yuan & Ryd.[100]

For each simulation, the object was positioned at various locations along the line of symmetry between the two x-ray sources. At each position, the reconstruction of the object was simulated 1000 times using a zero mean normal distribution of measurement error with a standard deviation of 0.02-mm.[100] The geometry of the simulations was as would typically be used for each respective cage. For the biplanar cage, the sources were at 90° to each other. For the projective cage, the angle between the sources was approximately 24°. Consequently, the setup conditions used for the two cages were not exactly analogous but the geometry was consistent a practical imaging geometry for shoulder kinematics.

4.1.3 Part I – Results

The "ideal test" produced essentially zero reconstruction error (Figure 4-10). The slight deviation from exactly zero (i.e. < 0.0003-mm) was attributed to round-off errors introduced when the simulation uses trignometric functions to mathematically project the object points onto the image planes. For each test condition, the results using DLT and Selvik's method overlapped.

The results for the projective cage showed a continuous increase in error as the object was moved further from the fiducial plane towards the x-ray sources (Figure 4-11). At 350-mm from the fiducial plane, the RMS error was 0.14-mm (SD = 0.05). At 1100-mm from the fiducial plane, the RMS error was 0.51-mm (SD = 0.30). The results using DLT and Selvik were equal to three decimal places.

$RMSE_z$ dominated the overall RMSE. The z-axis was in the direction towards the x-ray sources. The RMS errors in the x-y plane remained below 0.1-mm at 1100-mm above the fiducial plane. An increase in extrapolation distance of 900-mm resulted in an $RMSE_y$ increase of 0.04-mm and $RMSE_x$ of 0.06-mm.

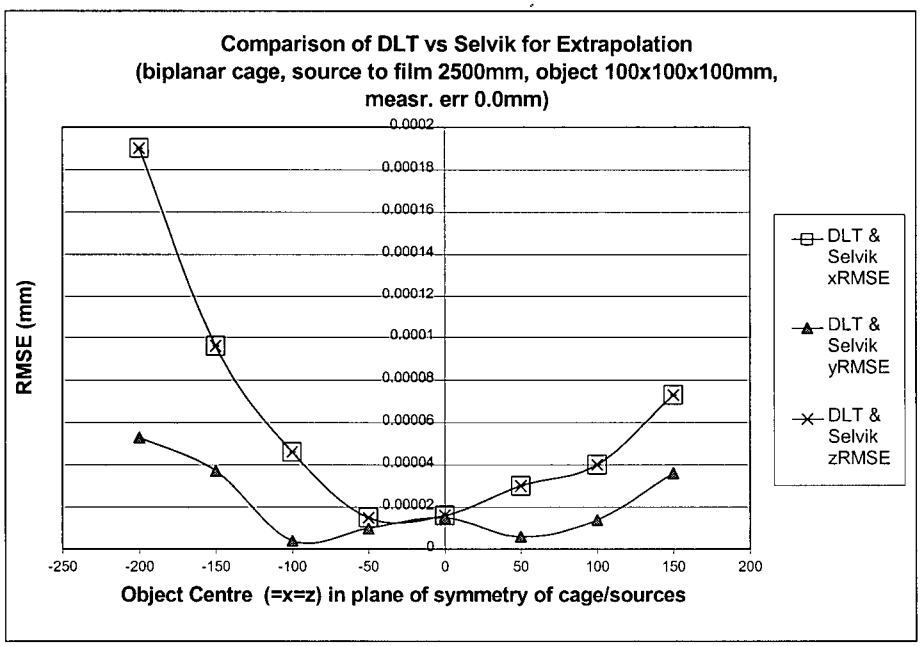

Figure 4-10: Zero Simulation

Essentially zero reconstruction error was found when measurement error was set to zero. This demonstrates correct functionality of MATLAB code. Deviation from exact zero was attributed to rounding errors introduced when trignometric functions were used to mathematically project object points onto the image plane.

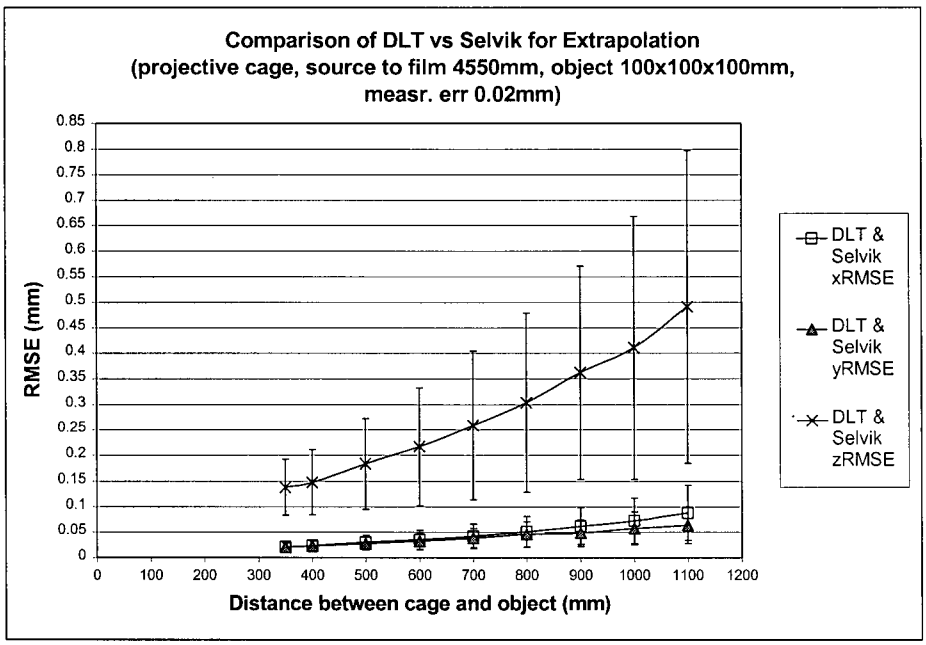

Figure 4-11: Projective Cage Simulation

Simulation prediction of RMS error of object reconstruction using the projective calibration cage. Reconstruction was performed using Selvik's method and DLT. Results were identical to three decimal places.

For the biplanar calibration cage, the reconstruction error, as expected, was lowest at the centre of the cage and increased towards the calibration space boundary (Figure 4-12). At the centre of the calibrated space, there was little difference between the three reconstruction methods (Table 4-1). The RMS error at the centre of the calibrated space ranged from 0.026-mm for DLT using a full cage to 0.031-mm when using Selvik's method. With the object centred on the cage boundary, the RMS error ranged from 0.030-mm when using DLT with a full cage to 0.129-mm when using Selvik's method.

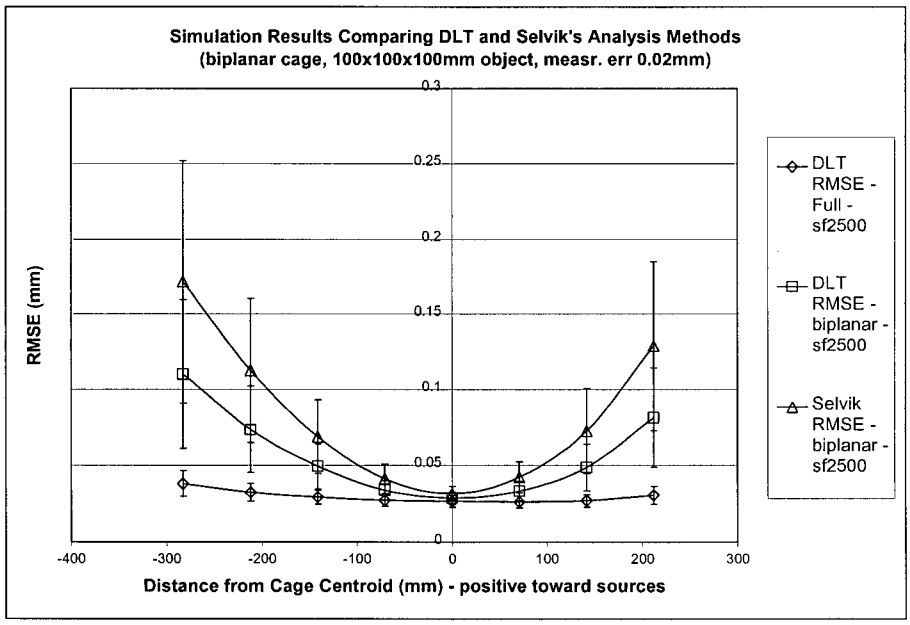

Figure 4-12: Biplanar Cage Simulation
Simulation prediction of RMS error of object reconstruction using the biplanar calibration cage. Reconstruction was performed using Selvik's method, DLT using two planes for calibrating each film (biplanar), and DLT using the entire cage for calibrating each film (full). Source to film distance was fixed at 2500-mm.

Analysis Method	Partial Extrapolation 212-mm nearer film RMSE (SD)-mm	Interpolation at Centre RMSE (SD)-mm	Partial Extrapolation 212-mm nearer sources RMSE (SD)-mm
Selvik – biplanar	0.113 (0.048)	0.031 (0.005)	0.129 (0.056)
DLT – biplanar	0.074 (0.029)	0.028 (0.004)	0.082 (0.033)
DLT – full	0.032 (0.006)	0.026 (0.004)	0.030 (0.006)

Table 4-1: Comparison of Methods
Comparison of Selvik and DLT reconstruction accuracy for object centred on cage boundary and at centre of calibrated space.

4.1.4 Part I – Discussion

The results of the "ideal test" indicated that the analysis software was functioning correctly. Deviation from exactly zero cannot be avoided since round-off errors were introduced into the simulation when the object points were mathematically projected onto the image plane.

The projective cage results showed significant overall RMS errors in extrapolation. Typical usage of the projective cage places the subject in this extrapolation space. However, the typical usage of this cage is to detect planar motions. In the plane parallel to the fiducial plane, the RMS errors ($RMSE_x$, $RMSE_y$) were significantly lower (~0.05-mm vs. >0.15-mm) and remained more constant than in the normal direction ($RMSE_z$). Yuan and Ryd reported similar results.[100] They used Selvik's method in simulation and reported the z-axis error to be 2.7× greater than x-axis or y-axis error for the projective cage. The lower error in the x-y plane presents a reasonable argument for the use of the projective cage in situations when very little out of plane motion is expected. It is expected that the scapula will exhibit significant non-planar motion.

The accuracy in the axial direction ($RMSE_z$) is dependent on the camera angle. In the limit as the camera angle approaches zero, the two images become identical and the z position cannot be determined. This dependence was not quantified in this study. The

choice of camera angle was consistent with practical imaging of the shoulder region in a rehabilitation setting.

A comparison of the overall RMSE between the two cages shows that the best accuracy can be achieved with the biplanar cage (Figure 4-13). Yuan and Ryd also found similar results.[100] However, the biplanar cage confines the object of interest and is poorly suited for large planar motions.

To circumvent this, the object may be placed outside the cage boundaries. However in this case the RMS errors are very similar to those of the projective cage. Alternatively, the cage may be removed after calibration to allow a greater range of subject motion in subsequent exposures while avoiding extrapolation. The region of interest could be centred in the calibrated space to ensure highest accuracy.

There is a drawback to this second alternative. The change of x-ray film cassettes introduces slight displacements in the imaging plane. In addition, an image plane coordinate system is defined for each x-ray film. The image points are measured with respect to this image plane coordinate system. Naturally, there is variability in the definition of this local coordinate system. When the cage is present in each exposure, each stereo pair is self-calibrated. By removing the cage from subsequent exposures, the calibration relies on the initial-calibration and does not account for shifts due to x-ray cassette replacement or measurement errors in defining the image plane coordinate system.

Extrapolation beyond cage boundaries increases RMS error.[17,98] The removal of the calibration cage increases RMS error. Which introduces greater error is a question to be resolved.

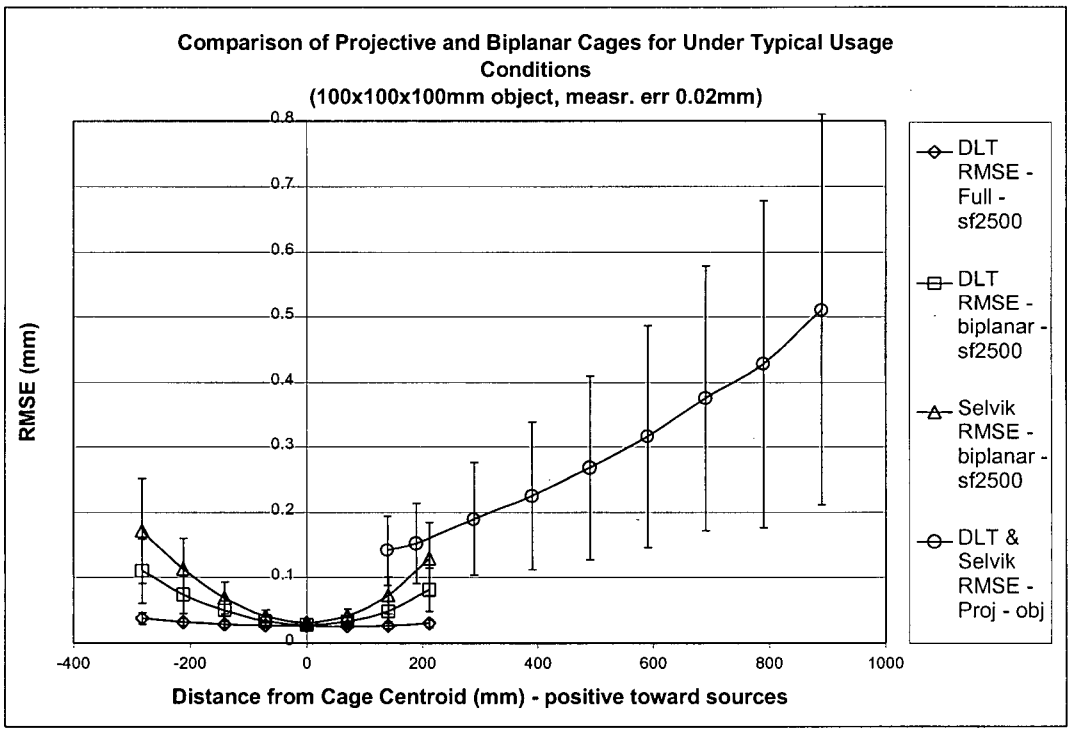

Figure 4-13: Comparison of overall RMSE
Comparison between projective and biplanar calibration cages. Note that the planar RMS errors (x,y) for the projective cage are much lower than in the normal direction (z). The overall RMSE for both cages are dominated by $RMSE_z$.

The comparison between the two analysis methods yielded curious results. For the projective cage, the results were essentially identical. For the biplanar cage, the results were identical only for the "ideal test" where the measurement error was zero (Figure 4-10). In the subsequent tests using 0.02-mm of measurement error, DLT showed higher accuracy than Selvik's method. In addition, DLT using all calibration points (full cage) was more accurate than the biplanar method.

The key geometric difference between the cages is in the control plane. For the projective cage, the control markers are collinear. For the biplanar cage, the control markers form a plane. No analytical explanation resulting from this geometric distinction alone was found.

Consider the two calibration methods. For both techniques, the reconstructions are simply inverse operations. DLT solves for the relationship between the lab and image

frames in a single least-squares optimization step. Selvik's method uses two steps. A 2D-DLT solves for orientation, via least-squares, while the projection centre (node point) is solved in a subsequent least-squares optimization.

It is difficult to visualize an n-dimensional least-squares optimization. Instead consider the simpler least-squares optimization of fitting a straight line to a set of data. What is the effect of dividing the optimization into two steps? A hypothetical example is shown in Figure 4-14. Assume for this example that all the data used in the optimization are equally weighted. If the optimization is broken into two steps, the weighting is redistributed. For example, five inputs may be used to solve for a single intermediate value. This single value holds less weight than the original five in the final line fit. The difference in the two solutions becomes more pronounced as the errors of the data increase.

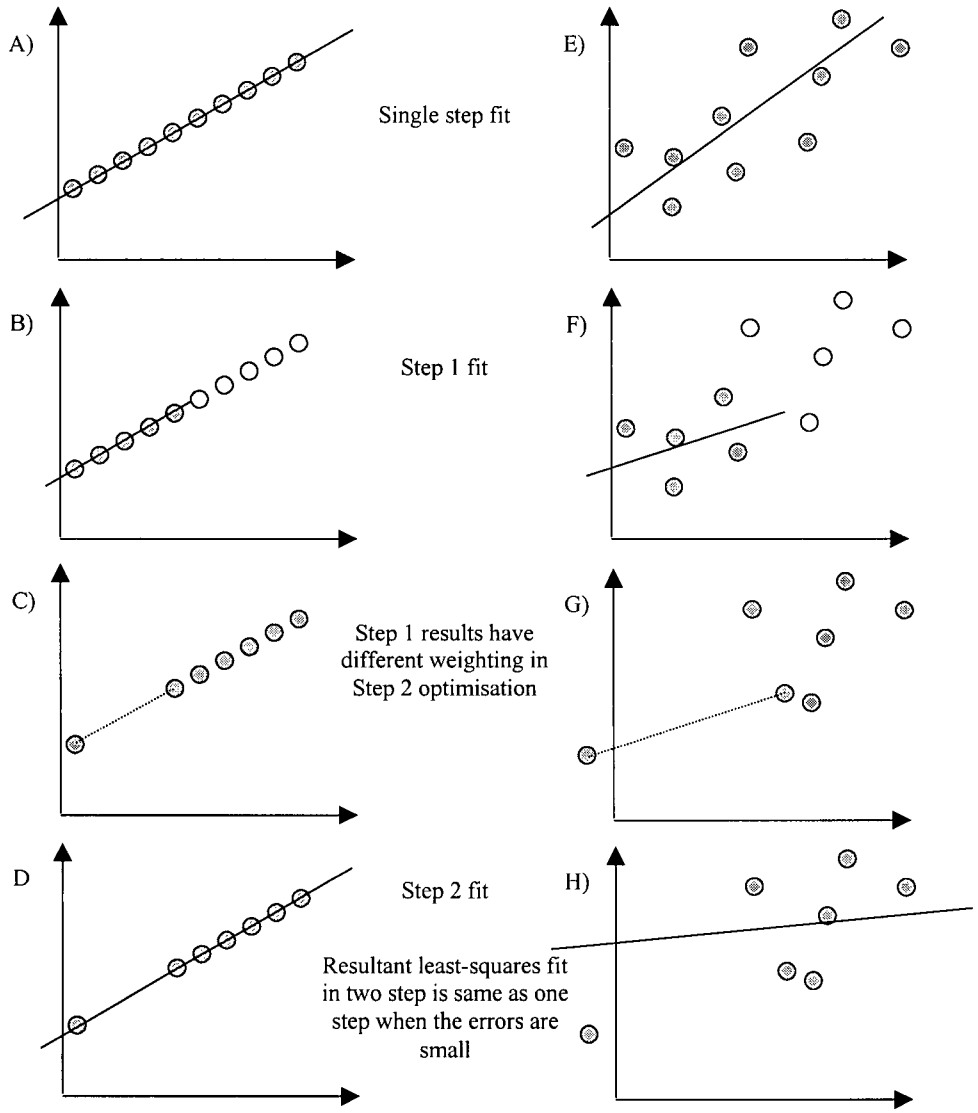

Figure 4-14: Optimization - Single versus Multiple Steps
Least-squares optimization for two variables. In the two step optimization, the results of the first step are no longer weighted as they would have been originally. When the errors are small, the two step optimization is the same as the single step. When the deviations are larger, the two step fit may not be the same as that obtained from the single optimization.

This graphical argument implies that there is an intermediate step in Selvik's method that has essentially no error in the projective cage, but has a more significant error in the biplanar cage. The source of this error may be the control markers.

During step 2 of Selvik's calibration method, the control markers are projected onto the fiducial plane (Figure 4-15). For the projective cage, this inverse 2D-DLT projects the markers inside the area covered by the fiducial markers. This is a region of interpolation. For the biplanar cage used in this simulation, the control markers are projected outside the area covered by the fiducial markers. This is a region of extrapolation and hence is subject to higher errors. The control points are no longer used after this step and this intermediate error is similar to the scenario depicted in Figure 4-14. DLT does not have this step and hence does not incur the error associated with it.

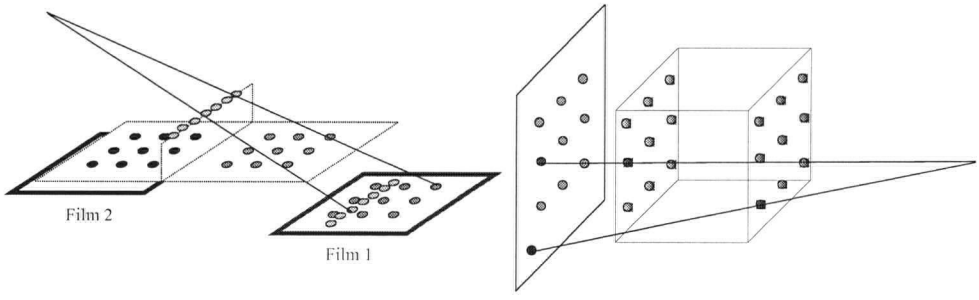

Figure 4-15: Control Point Extrapolation
Using Selvik's analysis method, the control markers are projected into the calibrated region of the fiducial plane for the projective cage. For the biplanar cage used in this simulation, the control markers are projected into the extrapolation region of the fiducial plane. The extrapolation error propagates through the remainder of the analysis. Is this the source of the discrepancy between Selvik and DLT in the biplanar cage but not the projective cage?

More formally, for a result that is a function of several variables, $R = R(x_1,x_2,x_3,\ldots,x_n)$, where $e_1,e_2,e_3,\ldots,e_n$ are the respective uncertainties in $x_1,x_2,x_3,\ldots,x_n$, then the uncertainty in R is:

$$e_R = \left[\left(\frac{\partial R}{\partial x_1} e_1 \right)^2 + \left(\frac{\partial R}{\partial x_2} e_2 \right)^2 + \ldots + \left(\frac{\partial R}{\partial x_n} e_n \right)^2 \right]^{1/2}$$

Applying this to compare the calibration result from DLT and Selvik:

$$U_{Cal\_DLT} = U_{Cal\_DLT}(P) \qquad U_{Cal\_Selvik} = U_{Cal\_Selvik}(P,C)$$

$\boldsymbol{U}$ denotes the collective unknown variables discussed in the chapter on stereophotogrammetry (i.e. node point, relationship between image and laboratory coordinate). DLT and Selvik's method both solve for $\boldsymbol{U}$ in calibration.

The accuracy of $\boldsymbol{U}$ depends on the image points and calibration points, $\boldsymbol{P}$, which are the same for both DLT and Selvik's method. However, the accuracy of Selvik's method also depends on the projection of the control points onto the fiducial plane. Hence $\boldsymbol{C}$ denotes this projection solution. Solving for the expected error:

$$e_{U_{Cal\_DLT}} = \left[\left(\frac{\partial U_{Cal\_DLT}}{\partial P} e_P\right)^2\right]^{1/2} \text{ and}$$

$$e_{U_{Cal\_Selvik}} = \left[\left(\frac{\partial U_{Cal\_Selvik}}{\partial P} e_P\right)^2 + \left(\frac{\partial U_{Cal\_Selvik}}{\partial C} e_c\right)^2\right]^{1/2}$$

The image points and calibration points used in Selvik's analysis are also used in DLT. Hence, $\boldsymbol{e_P}$ are the same for both expressions. The relationships $\frac{\partial U_{Cal\_DLT}}{\partial P}$ & $\frac{\partial U_{Cal\_Selvik}}{\partial P}$ are complicated because they depend on least-squares solution of matrix equations. They are not necessarily equal. However, in the "ideal test" where $\boldsymbol{e_c}$ is zero, $e_{U_{Cal\_DLT}} = e_{U_{Cal\_Selvik}}$ are known to be equal because the results for the two analysis methods overlap (Figure 4-10). Thus from the "ideal test" results, it can be concluded that for $\boldsymbol{e_c}$ small, $\frac{\partial U_{Cal\_DLT}}{\partial P} = \frac{\partial U_{Cal\_Selvik}}{\partial P}$.

For the projective cage, $\boldsymbol{e_c}$ is small because the control points are projected into the interpolation region of the fiducial markers. For the biplanar cage, $\boldsymbol{e_c}$ is larger due to

extrapolation. These graphical and error analysis arguments explain why the two analysis methods produce the same results in the projective cage but differing results for the biplanar cage.

The arguments suggest that the control markers in the fiducial plane should occupy a smaller area so as to be projected into the interpolation region of the fiducial plane. Upon acquisition of the biplanar cage, this was observed to be the case.

4.1.5 Part I – Conclusions

In summary, the biplanar cage was found to have a higher overall accuracy. For planar motions in the extrapolation region, both cages appeared comparable. It remains to be answered whether removing the biplanar cage to allow planar motion in the calibrated space yields higher accuracy than extrapolation.

The analysis methods gave identical results when used with the projective cage and different results when used with the biplanar cage. A graphical and error analysis explanation was presented but an analytical proof has not yet been found. For biplanar calibration, the use of the full cage with DLT yields the highest accuracy. DLT analysis performed as good, or better, than Selvik's analysis under all conditions.

4.2 Part II – Influence of Parameters

4.2.1 Part II – Objective

The setup of a stereophotogrammetric exposure can be highly variable. In addition to the calibration cage and the analysis method, there are geometric and cage parameters that can be varied. This series of simulations aimed to address the following questions.

1. What is the effect of calibration cage manufacturing accuracy? This question was asked to determine the feasibility of manufacturing the cage "in-house". "In-house" manufacturing allows flexibility in the design of the cage's geometry.

2. What is the effect of the x-ray source to film distance? This question was posed to address a potential practical experimental issue. It is anticipated that the source to film distance is somewhat difficult to control to a high accuracy during experimentation.
3. What is the effect of asymmetric object positioning in the exposure? This question was asked to address another experimental issue. Due to the high range of motion in the shoulder joint, it is anticipated that the motion of the scapula and humerus will be highly asymmetric with respect to the stereophotogrammetric setup.
4. What is the interpolation accuracy of the projective cage? This was a hypothetical question posed in an attempt to assess the influence of projective versus biplanar cage geometry. The methods and results in part I do not indicate if the projective cage's poorer performance was a result of extrapolation or cage geometry or both. In part I, the virtual object was always outside the volume of the projective cage while the biplanar cage accuracy was assessed both inside and outside the calibrated volume. This difference followed naturally from the typical usage of each of the cages.

4.2.2 Part II – Methods

The simulation environment used in part I, RSAGUI, was used to run this series of simulations. Unless otherwise stated, the calibration cages, the virtual object, and the object's positions were identical to that used in part I.

The manufacturing accuracy of the calibration cage was assessed by comparing two virtual biplanar cages with different manufacturing accuracies. The placement accuracy of the calibration markers on one cage was 0.01-mm while on the other cage it was 0.2-mm.

The effect of source distance was assessed using the biplanar cage. The two distances compared were 2500-mm and 4500-mm.

The effect of asymmetric object position was assessed using the biplanar cage. The object was moved along the axis of x-ray source 1.

The interpolation accuracy of the projective cage geometry was examine by creating a hypothetical cage, similar to the projective cage in part I. The only difference was that the control markers were at a height of 1000-mm, instead of 210-mm, above the fiducial plane. This allowed a range of interpolation positions to be tested. Due to geometric limitations of the angle of x-ray exposure, only a single object point was used. The object point was moved along the z-axis toward the x-ray sources along the setup's line of symmetry.

4.2.3 Part II – Results

The increase in manufacturing uncertainty, from 0.01-mm to 0.2-mm resulted in a significant increase in reconstruction error (Figure 4-16). At the cage centre where the errors are lowest, the cage with calibration markers accurate to 0.01-mm had a reconstruction RMS error of 0.03-mm compared to 0.11-mm for the cage manufactured to a 0.2-mm accuracy. At 200-mm from the cage centre, the difference was almost an order of magnitude (0.04-mm versus 0.33-mm).

An increase in source distance from 2500-mm to 4500-mm resulted in a small increase in error. The effect was much smaller than the choice of analysis method used (Figure 4-17).

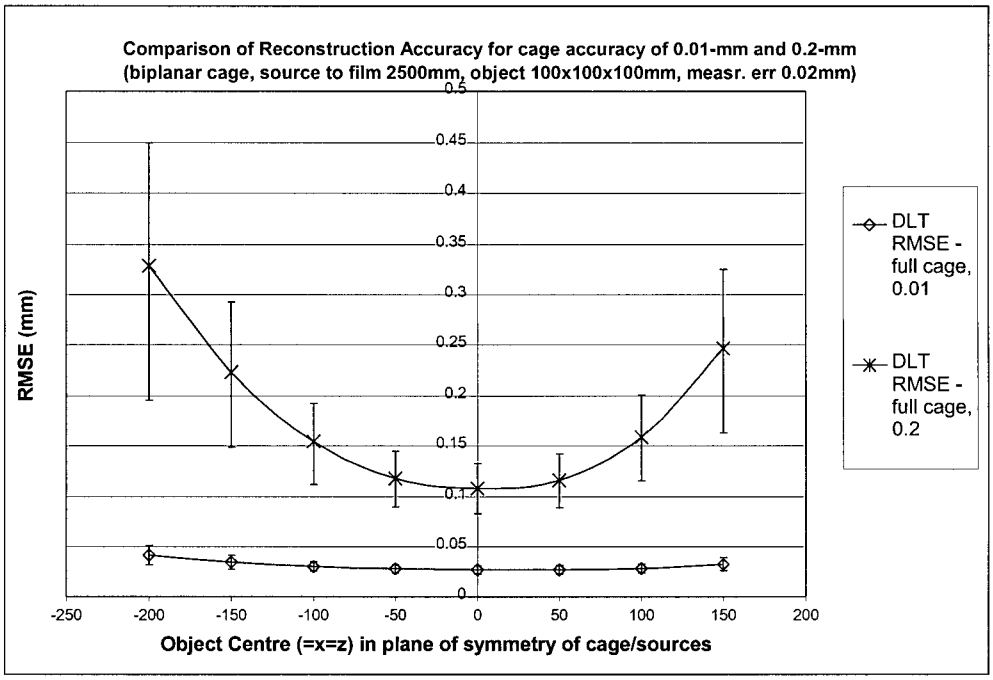

Figure 4-16: Manufacturing Accuracy
Comparison of reconstruction error for biplanar cage manufactured to an accuracy of 0.01-mm versus 0.2-mm. The cage boundary was at 75-mm from the cage centre.

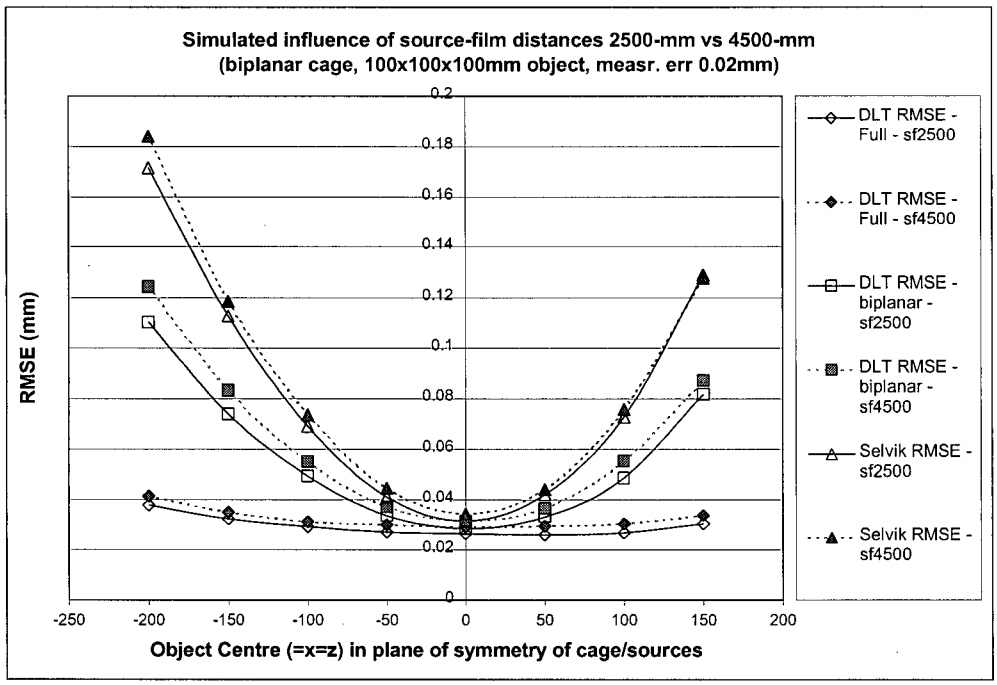

Figure 4-17: Source Distance
Influence of source distance was less than the effect of analysis method used.

Using DLT with the full cage for calibration, there was no difference between the accuracy of symmetric and asymmetric object positioning. For biplanar calibration data used with DLT and Selvik's method, there was an increase in errors in the direction away from the x-ray sources. Towards the sources, there was no difference (Figure 4-18).

The interpolation accuracy of the projective cage was highest at the control plane (Figure 4-19). The hypothetical cage with the control plane at 1000-mm from the fiducial plane had a higher accuracy than the cage with the control plane at 210-mm.

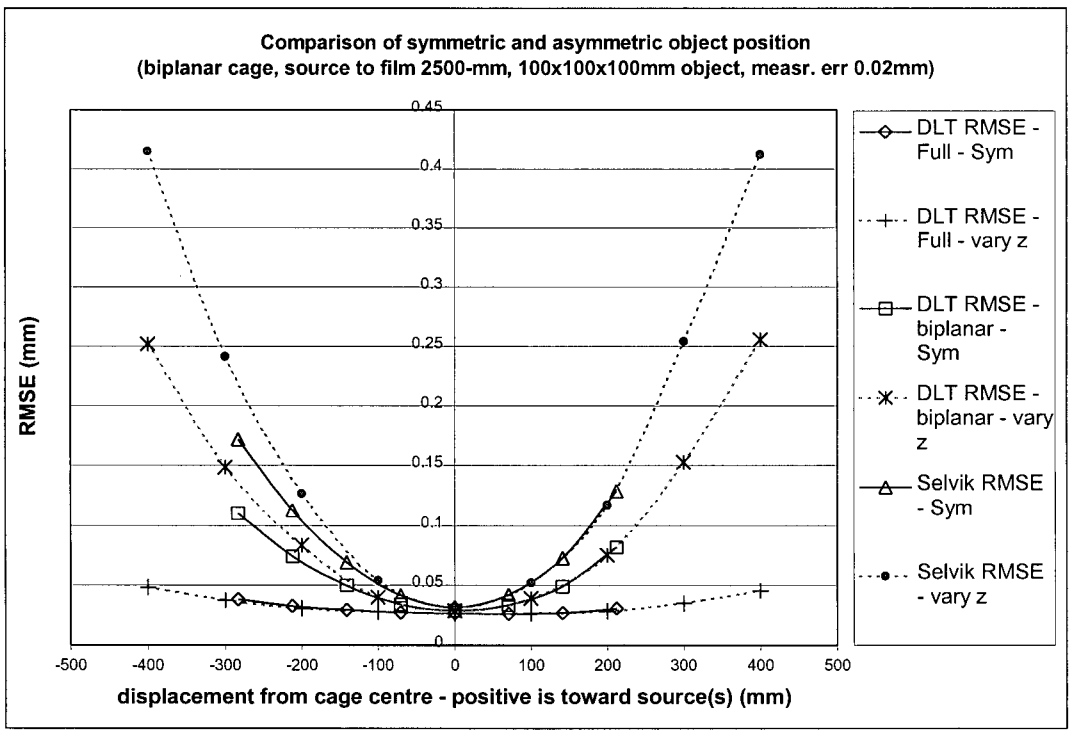

Figure 4-18: Asymmetric Object Position
Influence of asymmetric object position within a symmetric stereophotogrammetric setup. Object was moved towards source 1.

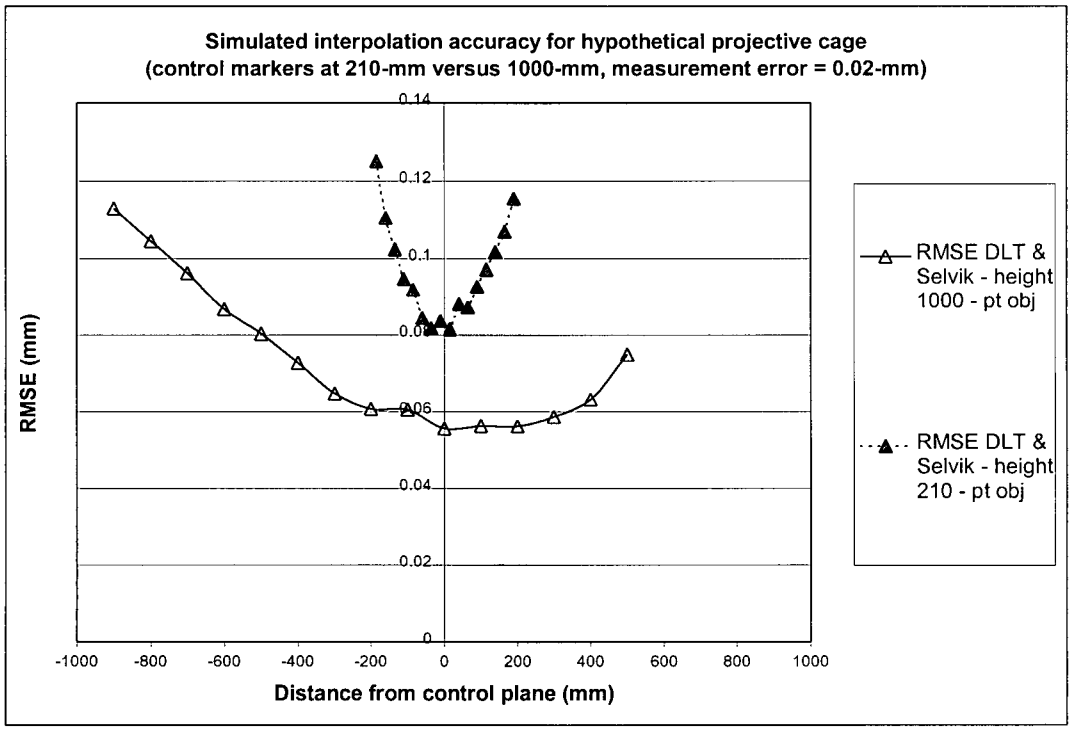

Figure 4-19: Projective Cage Interpolation Accuracy
Highest accuracy was at the control plane. A higher control plane resulted in higher accuracy.

4.2.4 Part II – Discussion

The data emphasized the importance of accuracy in manufacturing the calibration cage. With a manufacturing accuracy of 0.2-mm, the RMS error at 20-cm from the cage centre was predicted to be 0.33-mm. This is a significant error if the cage is to be used to detect micromotions on the order of 0.1-mm. In addition, the measurement accuracy used in the simulation was a very optimistic 0.02-mm.

The negligible effect of source distance indicates that there is reasonable flexibility in the stereophotogrammetric setup. This is an important result for RSA of the shoulder where future investigations may require the use of rehabilitation equipment and hence a more distant positioning of the x-ray source from the film plane.

DLT using the full cage for calibration negates any effect of asymmetry in the object positioning. This is an extremely useful result since it is very difficult to create a

perfectly symmetric stereophotogrammetric setup. In addition, for large motions such as in the shoulder, asymmetric object positions will be unavoidable.

It was an unexpected result that the highest accuracy for the projective cage was at the control plane. This is an asymmetric position. No analytical expressions were found to explain this result. The question was not pursued further because the projective cage was only hypothetical and intended to extend geometric comparisons with the biplanar cage.

4.2.5 Part II – Conclusions

As expected, accuracy in cage manufacturing was found to significantly influence reconstruction accuracy. The results suggest there is flexibility in the sterephotogrammetric setup with respect to source to film distance as well as asymmetric positioning of the object in the field of view. The projective cage exhibited highest reconstruction accuracy at the control plane.

4.3 RSA Simulation Conclusions

The accuracy of a stereophotogrammetric exposure depends on a wide variety of variables. A software environment has been developed to analyse arbitrary stereophotogrammetric setups and allow the influence of various parameters to be assessed. The simulations on the calibration cages show that the biplanar cage has the potential to be more accurate than the projective cage. Hence the biplanar cage was acquired for the RSA system developed. The simulations also showed that DLT performed as good or better than Selvik's method for all trials. Hence, DLT was selected as the analysis method for the RSA system. There is flexibility in setting up the stereophotogrammetric exposures since the source to film distance as well as asymmetric object positioning had little effect on overall accuracy.

Chapter 5

RSA ACCURACY STUDIES

RSA has been considered a gold standard for measuring in vivo skeletal positions for over twenty-five years. Its accuracy has been reported to range anywhere between 0.01-mm and 0.25-mm for reconstruction/translations, and between 0.03° and 0.6° for rotations.[44] The variation in accuracy presumably stems from differences in how the data are acquired and analysed. This chapter chronicles the systematic benchtop assessment of a variety of factors that influence the accuracy of the RSA system used in this thesis. The analysis was necessary because it identified the current capabilities of the RSA methods used. Scapulothoracic motions are large and hence an extremely high accuracy (say 0.01-mm) was probably not essential for this specific thesis. Nevertheless, the identification of the system's limitations will assist in the refinement and ultimate realization of an RSA system accurate enough to measure in vivo micromotions on the order of 0.1-mm.

5.1 Measurement Accuracy

The reconstruction accuracy of the RSA system is dependent on the quality of the data input to the analysis. A typical RSA image consists of circular images of the calibration cage markers and implanted tantalum markers. The coordinates of these marks must be extracted and fed into the RSA analysis software. There are several methods available to digitize the coordinates of the markers on the x-ray images. These can be grouped into manual coordinate measuring tables and software coordinate measurement.

5.1.1 Objective

This investigation aimed to determine software measurement accuracy of scanned x-ray images. Specifically, the effect of sphere radius and image contrast was analysed.[iv]

5.1.2 Methods

Matlab (The MathWorks Inc., Natick, MA) was used to generate hypothetical x-ray images. Each image contained ten circles representing the simulated exposure of spherical markers. Five sphere radii (0.5,1.0,1.5,2.0,2.5-mm) and five contrast levels (10,30,50,70,90% transmission of x-rays through sphere[v]) were tested for a total of 25 images and 250 circles. The images were generated with a resolution of 150 pixels per inch (ppi).[vi] This is the same x-ray scanner resolution that would subsequently be used on real x-rays. The images were in tiff format with an 8-bit grayscale resolution. The size of each image was 750×750 pixels. This was chosen to allow a sparse distribution of circles while maintaining a reasonable file size (550KB). The exact coordinates of the centre of each sphere's image was output to a file and used as a gold standard for accuracy. All images had Poisson noise added to them to simulate x-ray noise. The circles in the images were numbered in Adobe Photoshop (Adobe Systems Inc., San Jose, CA) and inverted. The final results were reasonable approximations of scanned x-rays (Figure 5-1).

[iv] Contrast has been defined here as 1 – T, where T is the x-ray transmissibility. T is the variable presented throughout this chapter.

[v] A perfectly radio-opaque object has 0% x-ray transmissibility and hence has a high contrast. Conversely, a perfectly radio-lucent object has 100% x-ray transmissibility and hence has no contrast.

[vi] Pixels per inch (ppi) refers to the input resolution (e.g. from scanner) whereas dots per inch (dpi) refers to the output resolution (e.g. to printer). The two are often used interchangeably.

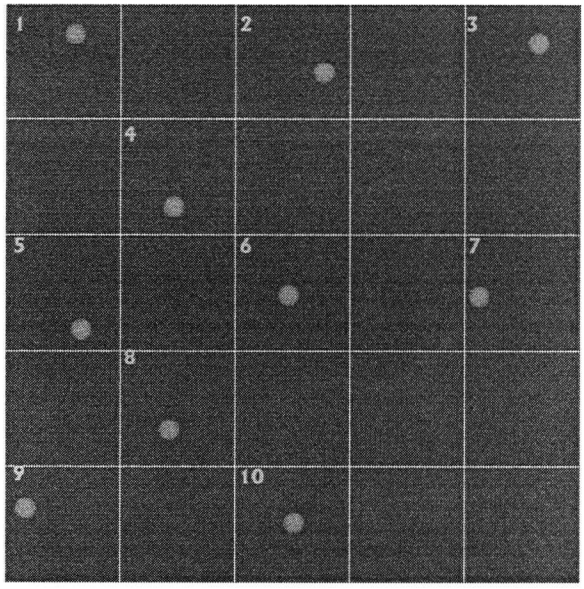

Figure 5-1: Measurement Accuracy Image
Example of x-ray image generated in Matlab and used to assess measurement accuracy.

The circle centres were identified using the crosshair tool in Scion Image for Windows B403 (based on NIH Image for Macintosh adapted by Scion Corporation, www.scioncorp.com). The monitor used was a 17" Acer 77e (Acer Inc., Taipei, Taiwan) running at a resolution of 1024×768, True Color (32-bit), with a refresh rate of 75-Hz. A Microsoft mouse using IntelliPoint software allowing it to run at 30% maximum speed with maximum acceleration was used for digitization (Microsoft Corp., Redmond, WA). The gold standard file was not opened until all digitization was completed. The measurement error for point, p_i, was calculated in pixels as $e_i = [(x_{gold} - x_i)^2+(y_{gold} - y_i)^2]^{1/2}$.

Two-way analysis of variance was used to compare the effects of sphere radius and percent x-ray transmission. A p-value less than 0.05 was considered significant.

5.1.3 Results

A constant error of 1.4-pixels, 1-vertical and 1-horizontal, was observed for most of the measurements. The most significant difficulties were encountered digitizing the image of 1-mm diameter spheres at 90% x-ray transmission. This difficulty was evident in the results with this test having a mean measurement error of 62.7-pixels (SD = 27.6-

pixels). Essentially the circles were not reliably visible. The mean error for measuring the imaged centre of a 2-mm sphere at 90% transmission was 16.54-pixels (SD = 35.4-pixels). At higher diameters, it was again difficult to align the crosshair at the centre of the circles. The mean error for a sphere of diameter 5-mm at 90% was 1.81-pixels (SD = 1.1-pixels). Table 5-1 shows the mean error and standard deviation for each of the 25 test images.

Radius (mm) & Transmission (%)	Mean Error (pixels)	Standard Deviation (pixels)
0.5, 10	1.414213538	0
0.5, 30	1.414213538	0
0.5, 50	1.414213538	0
0.5, 70	1.414213538	0.259893179
0.5, 90	**62.68143082**	**27.64332962**
1.0, 10	1.414213538	0
1.0, 30	1.414213538	0
1.0, 50	1.414213538	0
1.0, 70	1.372792244	0.130985826
1.0, 90	**16.53697968**	**35.39190674**
1.5, 10	1.372792244	0.130985826
1.5, 30	1.372792244	0.130985826
1.5, 50	1.414213538	0
1.5, 70	1.231370807	0.451818347
1.5, 90	1.771477699	0.42408973
2.0, 10	1.372792244	0.130985826
2.0, 30	1.248528123	0.213898957
2.0, 50	1.289949536	0.200084165
2.0, 70	1.348528147	0.300872564
2.0, 90	1.453663111	0.873602748
2.5, 10	1.331370831	0.174647778
2.5, 30	1.537163138	0.390341222
2.5, 50	1.307106733	0.318803072
2.5, 70	1.165028095	0.61708039
2.5, 90	**1.814218283**	**1.058621883**

Table 5-1: Measurement Accuracy Results
Mean measurement error and standard deviation for measuring the centre of an x-ray image of a sphere. 1.41 error corresponded to an offset error of 1 pixel in x-direction and 1 pixel in y-direction. The small standard deviation showed that this was essentially a constant systemic offset of the software measurement.

The 2-way ANOVA showed a significant effect for both radius and transmission ($p<0.001$ for both main effects). There was also a significant interaction between the two variables ($p<0.001$). These results are summarized in Figure 5-2 through Figure 5-4.

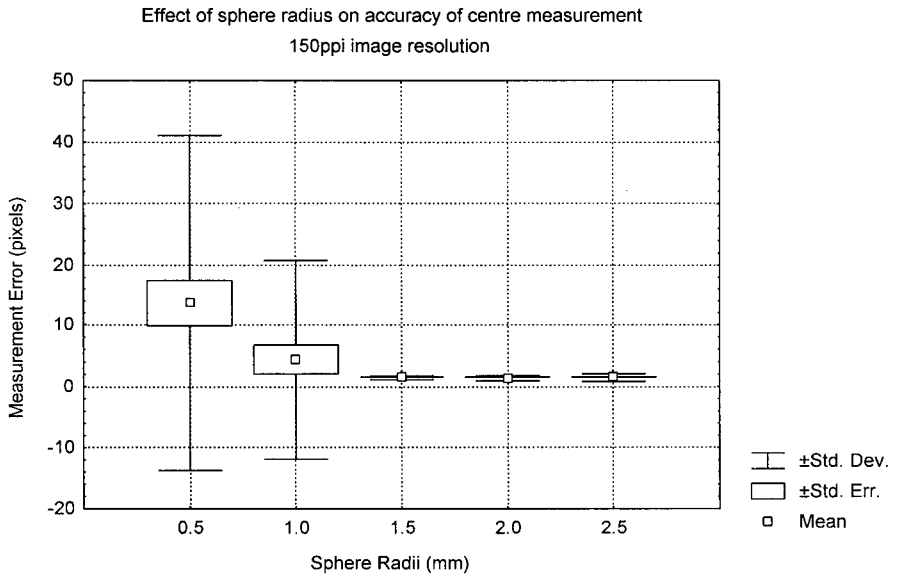

Figure 5-2: **Sphere Radius Effect**
Small sphere radii had a significant effect ($p<0.001$) on the accuracy of measuring the centre of the image when scanned at 150-ppi. The percent of x-ray transmission through the sphere had a significant interaction ($p<0.001$).

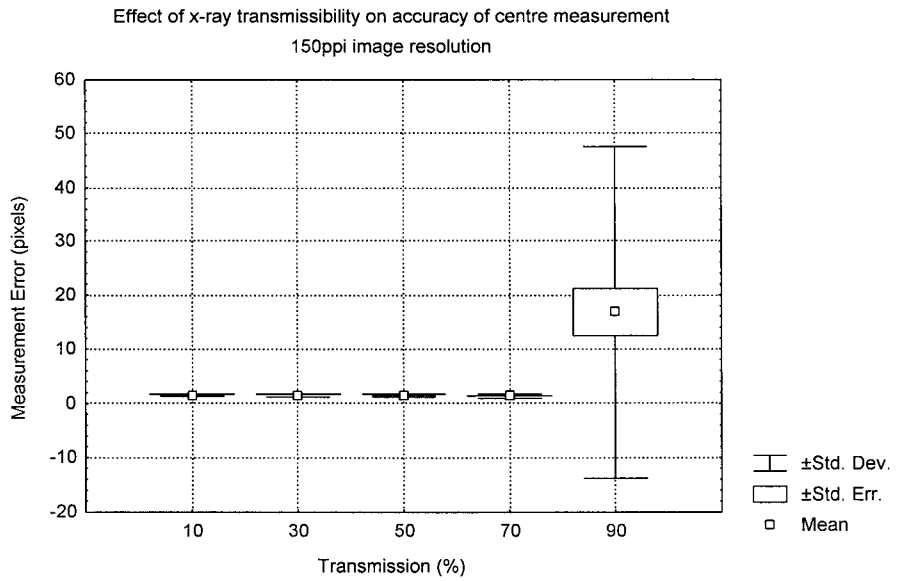

Figure 5-3: **X-ray Transmissibility Effect**
Very high x-ray transmission through a sphere had a significant effect ($p<0.001$) on measurement accuracy of the centre of the image when scanned at 150-ppi. The sphere radius had a significant interaction ($p<0.001$).

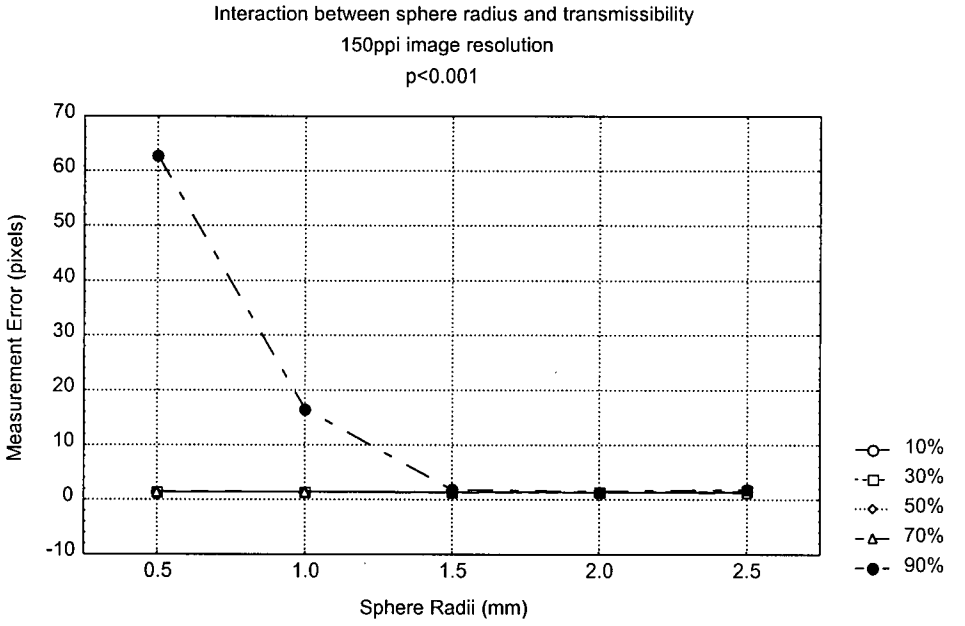

Figure 5-4: Radius and Transmissibility
The combination of small sphere diameter and very high x-ray transmission resulted in significant measurement error ($p<0.001$). Images were generated at 150-ppi.

Post-hoc SNK test showed that there was a significant difference between R = 0.05-mm, T = 90% and all other combinations (all p's <0.001). There was also a significant difference between R = 1.0-mm, T = 90% and all other combinations (all p's <0.03). Between all other remaining combinations, p was greater than 0.9.

5.1.4 Discussion

Inspection of the mean absolute error and standard deviations of the nominal values shows the measurement technique had a consistent 1×1-pixel offset. This corresponds to the centre being marked in the upper left-hand corner of the crosshair centre. A constant offset in itself does not decrease RSA accuracy. The definition of the image coordinate system on each radiograph is also subjected to this offset. Thus the resulting measured coordinates are optimal. The measurement is optimal, but still not exactly perfect because the pixels are discrete representations of continuous positions.

The results of the ANOVA agreed with intuition. Very small sphere diameters combined with high x-ray transmissions resulted in poor image contrast and hence high

measurement errors. The post-hoc comparison showed that for the most part, measurement error was constant irrespective of sphere radii between 1.5-mm and 2.5-mm with transmissions below 70%.

The data was used to estimate the measurement accuracy of actual RSA exposures. Actual exposures of 0.8-mm diameter tantalum markers fall in the range between T = 10% and T = 50% (Figure 5-5). The simulated accuracy for 1.0-mm diameter spheres in this transmission range was 1.41-pixels. At a resolution of 150-ppi, this corresponded to a measurement accuracy of 0.24-mm (SD = 0-mm). Actual exposures of 3.0-mm markers correspond to T = 10% (Figure 5-6). The simulated accuracy for this case was 1.37-pixels. At a resolution of 150-ppi, the inferred measurement accuracy was 0.23-mm (SD = 0.02-mm).

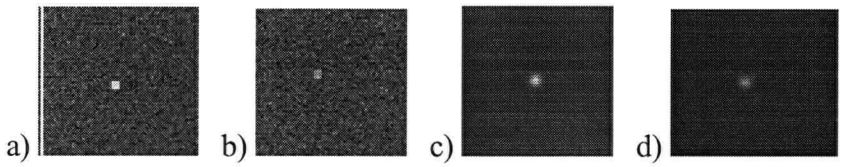

Figure 5-5: Simulated and Actual 0.8-mm Exposures
a) simulated exposure, D = 1.0-mm, T = 10%, b) simulated exposure, D = 1.0-mm, T = 50%, c & d) actual exposure of 0.8-mm diameter chromium sphere

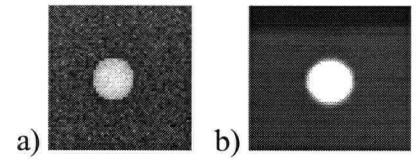

Figure 5-6: Simulated and Actual 3-mm Exposures
a) simulated exposure, D = 3-mm, T = 10%, b) actual exposure of 3.0-mm diameter chromium sphere.

5.1.5 Conclusion

A wide range of sphere diameters may be used as RSA markers provided the transmissibility is below 70%. The measurement accuracy using digital techniques was within one pixel in x and y. At 150-ppi scanning resolution, this corresponded to an accuracy of better than 0.24-mm.

5.2 RSA Reconstruction Accuracy

5.2.1 Objective

There were two objectives to this experiment. First was to quantify the reconstruction accuracy of the RSA system using the biplanar cage with only an initial-calibration method. Initial-calibration is used with the biplanar cage when it is necessary to remove the cage from subsequent exposures to provide space for a greater range of motion and to avoid extrapolation. Second was to assess the variability introduced into the initially calibrated system by changing the x-ray cassettes. When the x-ray cassette is changed, it is impossible to exactly place the new cassette in the identical axial plane.

5.2.2 Methods

A biplanar cage (model 10, Tilly Medical, Lund, Sweden) was used for calibration. The cage had a manufacturing accuracy of ± 0.01-mm and an overall dimension of 26-cm×26-cm×30-cm (Figure 5-8). The fiducial markers covered an area of 192-cm^2 while the control markers covered an area of 108-cm^2. The distance between the two planes was 24.9-cm.

It was not possible to place the x-ray films in the exact same transverse position within the x-ray cassette. To account for this, stationary alignment markers (3.0-mm markers) rigidly attached to the imaging setup were used to align the local x-ray coordinate systems (Figure 5-7). Axial displacements of the x-rays were not accounted for.

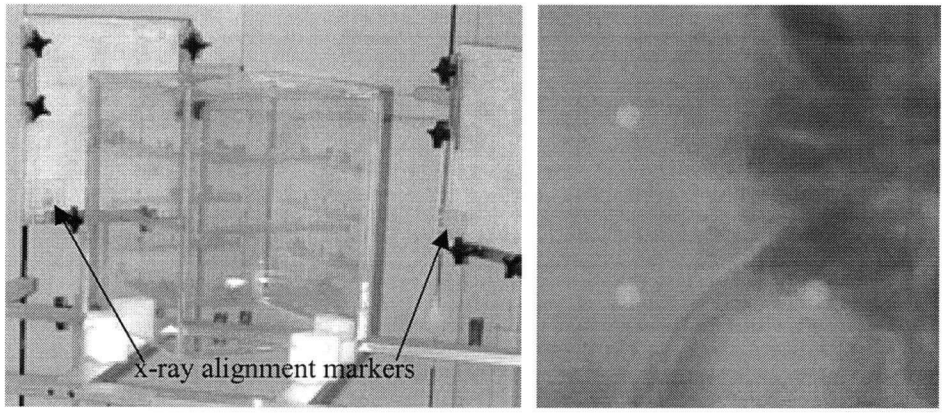

Figure 5-7: Alignment Markers
Stationary alignment markers rigidly attached to the imaging setup were used to align the local x-ray coordinate systems.

The cage was imaged with two mobile x-ray sources operating at 110-kV and 5-mAs. Leg-length cassettes were used in order to image a volume larger than the cage dimensions. The cassettes were angled at 20° to each other allowing the entire calibration cage to be in the imaging field. The sources were approximately 2-m from their respective film cassettes. A total of seven stereo exposure pairs were made. For six of the exposures (EXP_1 … EXP_6), the cage was left stationary. For the seventh exposure (EXP_7), the cage was rotated by an arbitrary angle.

EXP_7 was used for calibration and the cage in EXP_1 through EXP_6 was reconstructed. The reconstructed coordinate system (EXP_7) was in a different orientation than the cage in EXP_1 through EXP_6. This was intentional. When the calibration and reconstruction points are in identical positions, the accuracy is overly optimistic since the DLT parameters are biased to those coordinates.[16]

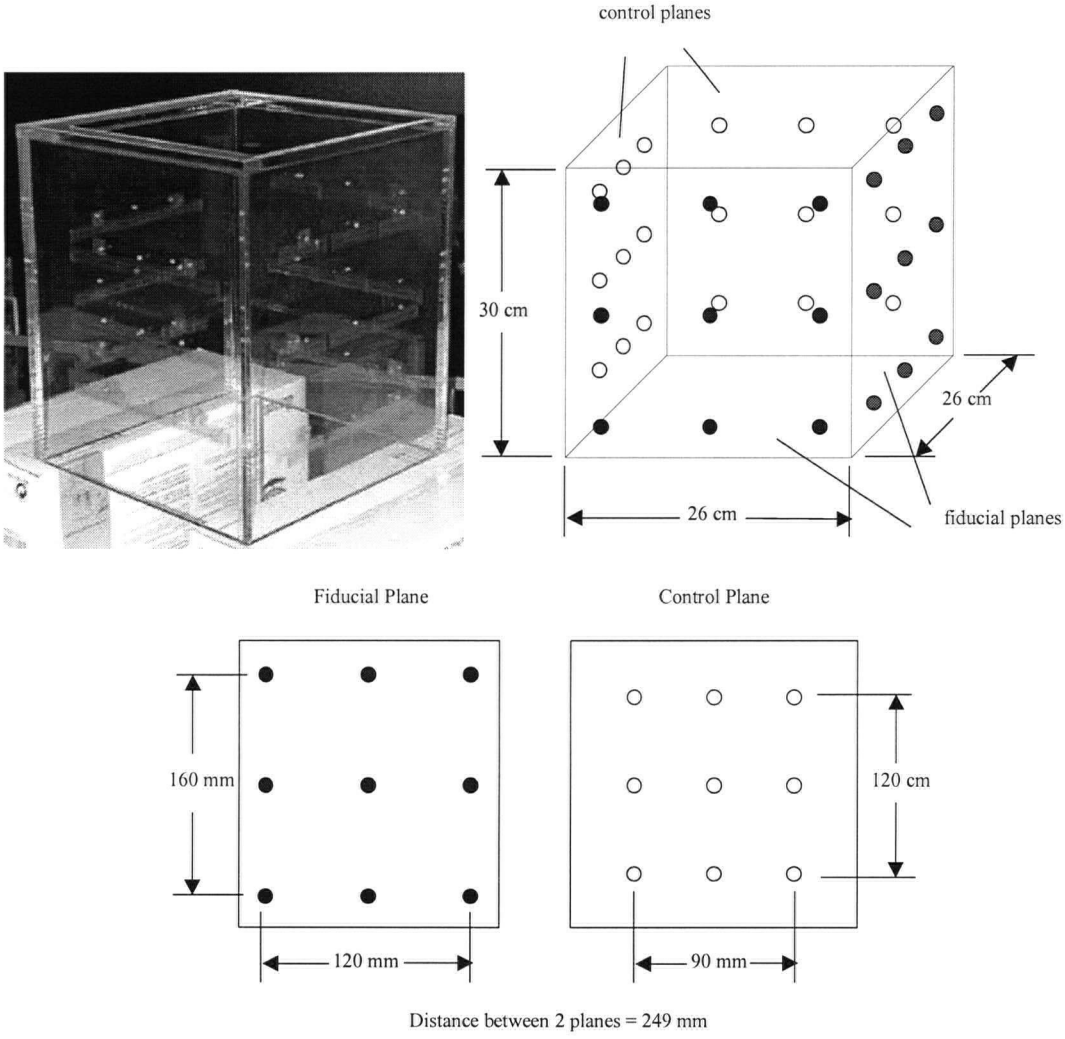

Figure 5-8: Biplanar Cage Dimensions

To determine accuracy, the Matlab routine "soder" from the Kinmat toolbox was used. The routine calculated the optimum transformation matrix for a body between two poses. The routine also computed a residual error that was a measure of rigidity. The residual error is defined as the root mean square of the sum of the distances between the points of the ideal rigid body and the reconstructed rigid body.

The transformation and residual error was computed between the ideal cage and the six reconstructed cages (EXP_1 … EXP_6). Note that it was not appropriate to arbitrarily

align the ideal cage and reconstructed cages because the analysis would have been sensitive to how the coordinates were aligned (Figure 5-9).

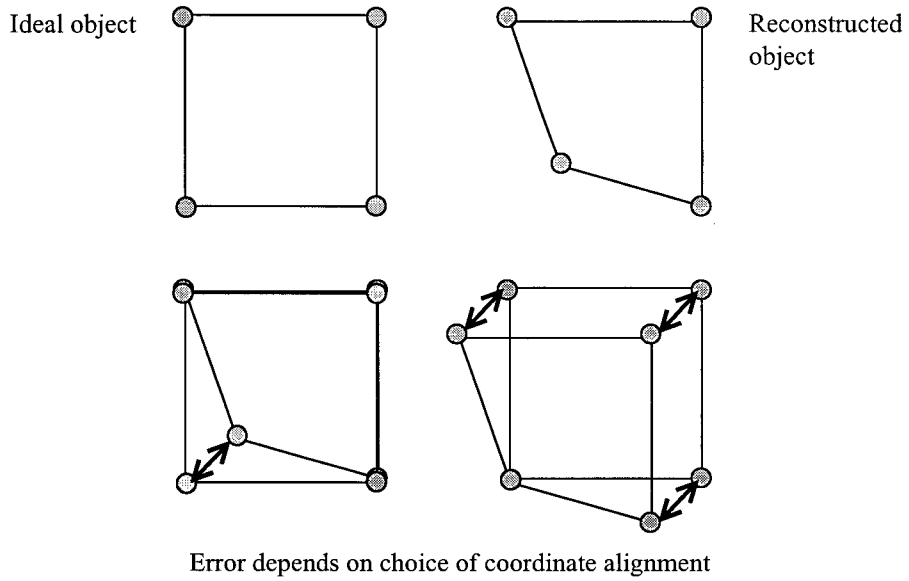

Figure 5-9: Coordinate Comparisons
Demonstration of why direct coordinate comparison is a poor indicator of reconstruction accuracy. The use of an optimal transformation followed by a computation of residuals errors is more consistent.

The effect of cassette changes was assessed by analyzing the variability of the reconstructed coordinates in EXP_1 through EXP_6. In these exposures, the cage was stationary hence the variability in the reconstructed coordinates was due to measurement error during digitization and the exchange of the x-ray cassettes. The former variable was already addressed in the section on measurement accuracy. A one-way ANOVA was performed on the variability of the reconstructed cage coordinates in EXP_1 through EXP_6. The dependent variable was the standard deviation of the coordinate. The data were grouped by axes (x, y, z).

5.2.3 Results

The accuracy of reconstructing the calibration points on the biplanar cage was found to be 0.43-mm (SD = 0.03) (Table 5-2). The cassette changes introduced a variability of 0.78-mm (SD = 0.19) in the x-coordinates, 0.55-mm (SD = 0.19) in the y-coordinates,

0.97-mm (SD = 0.17) in the z-coordinates (Figure 5-10). The x and y axes were transverse to the x-ray sources and the z-axis was axial to the x-ray sources. The variability introduced into the coordinate values was larger than the reconstruction accuracy and is addressed in the following discussion.

Exposure	Residual (mm)
1	0.3768
2	0.4360
3	0.4550
4	0.4431
5	0.4626
6	0.4144
Mean (std)	0.4313 (0.0315)

Table 5-2: Reconstruction Accuracy
Residuals computed as measure of rigidity between known calibration cage coordinates and reconstructed calibration cage coordinates.

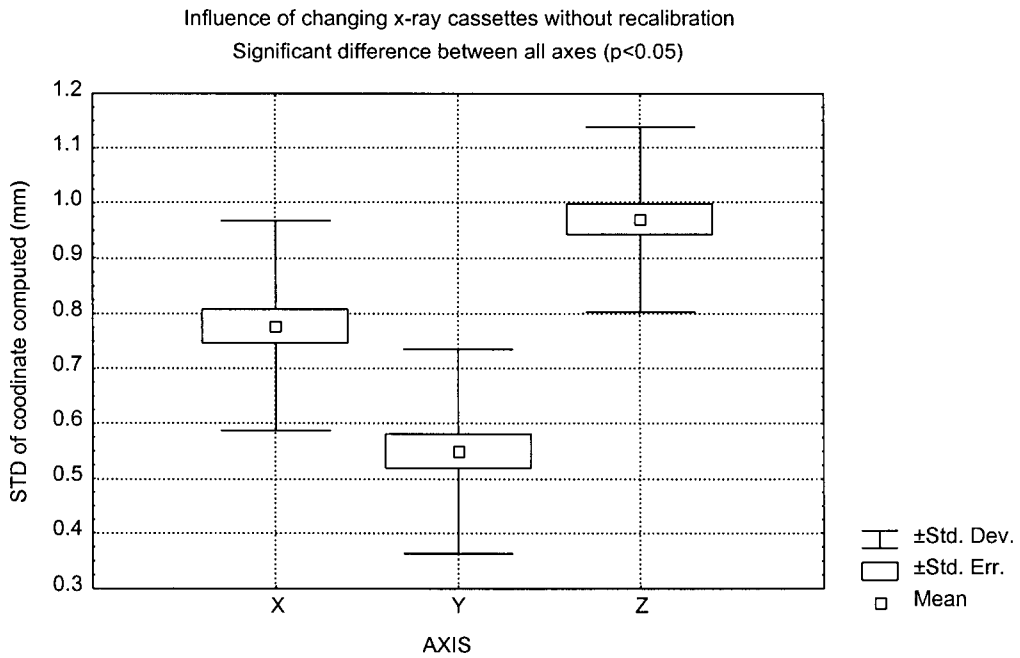

Figure 5-10: Cassette Changes
Influence of changing x-ray cassettes without re-calibration.

5.2.4 Discussion

The reconstructing accuracy was found to be better than the variability introduced by the change of x-ray cassettes. This seems to indicate that the RSA system was able to reconstruct the *shape* of the object to a higher accuracy than it was able to reconstruct its absolute *position*. A possible explanation for this may lie in the different effects of object marker error and alignment marker error. Although both sets of markers are subject to the same measurement error, alignment marker measurement error introduces an offset to all other markers on the x-ray. Hence it is hypothesized that the *shape* of the object was affected by the relative error of the object points with respect to each other. The *position* of the object was affected by this error, plus the added error accumulated from the variability of alignment marker measurement from film to film.

For intersegmental rigid body analysis, where the *relative* position of kinematic markers in each instance is more important than their absolute positions, it is appropriate to report 0.43-mm as the accuracy of the system.

RSA accuracy reported in the literature ranges from 0.01-mm to 0.25-mm and 0.03° to 0.6°.[18,44,76,80] It is often unclear how the accuracy is measured in the literature. For instance, the accuracy of measuring the distance between two points (or angle between two line segments) on a single image will not be the same as measuring the translation of a point in two separate images. The latter case incurs the error of the variability between the two exposures (Figure 5-11).

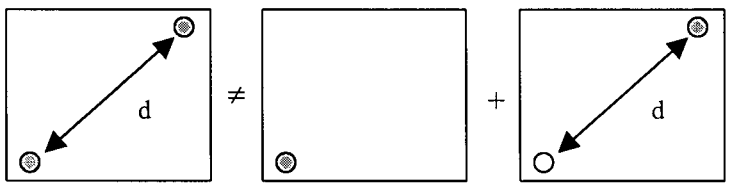

Figure 5-11: Distance Accuracy
The accuracy of measuring a distance, d, in one image, is not the same as measuring the distance from two separate images. The latter case has a lower accuracy due to variability between the images. It is unclear which accuracy was reported in literature. It is also often unclear if precision, or repeatability, has been confused with accuracy – deviation from the true value.

In addition, accuracy – the deviation of the measurement from the true value – is frequently confused with precision – the deviation of the measurements from each other.

Some studies have computed accuracy in the following way.[68,79] The coordinates of a set of implanted markers were measured from the average of two exposures. A third exposure was taken and the difference with the first average was taken as the accuracy. All three exposures were equally inaccurate and hence this method was dubious. The difficulty lies in finding a gold standard that is at least an order of magnitude more accurate. The discussion is perhaps academic since RSA has been used for over 25 years and appears to have been able to detect micromotions on the order of 0.1-mm.

The performance of the current RSA system was inferior to that reported by others. Two factors can improve the system's accuracy. First, the scanning resolution of 150-ppi yields a measurement accuracy of 0.24-mm. Measurement accuracy in the literature has been reported at 0.005-mm with manual digitizing tables. However, the best digitizing tables found to date have a resolution of 0.003-in corresponding to 0.08-mm.

Second, this experiment was conducted with an initial-calibration of the space followed by cage removal. A self-calibration approach where the cage remains visible in all exposures can be expected to improve accuracy.

5.2.5 Conclusion

The current RSA system has a *relative* reconstruction accuracy of 0.43-mm (SD = 0.03). The effect of x-ray cassette changes appears to increase variability in the *absolute* coordinates of the markers being imaged. Higher scanning resolution and self-calibration exposures are needed if the RSA system is to perform as well as those reported in the literature.

5.3 RSA Translation Accuracy

5.3.1 Objective

The objective of this experiment was to determine the RSA system's accuracy in measuring translation. In addition, this test aimed to quantify the difference in translation accuracy when self-calibration and initial-calibration were used. Traditional RSA in the shoulder uses a self-calibration method where a projective cage appears in each stereo exposure. The subject is in the extrapolation region of the flat projective cage resulting in extrapolation error. It was shown in the chapter on RSA simulations that the biplanar cage has a higher theoretical accuracy. However, for shoulder kinematics, it is necessary to initially calibrate the space with the biplanar cage and then remove it in subsequent exposures. This allows for a wide range of motion within the calibrated space while avoiding extrapolation.

5.3.2 Methods

This study used the same biplanar cage as the one in the reconstruction study. The cage was imaged with two mobile x-ray sources operating at 110-kV and 5-mAs. Leg-length cassettes were used in order to image a volume larger than the cage dimensions. The cassettes were angled at 20^{o} to each other allowing the entire calibration cage to be in the imaging field. The sources were approximately 2-m from their respective film cassettes. Six stereo exposure pairs were made (EXP_1 ... EXP_6). In all exposures, the cage was left stationary while sixteen object points (eight internal + eight external) were translated using an x-z table. The object points were 3.0-mm chromium spheres embedded in radiolucent acrylic (Figure 5-12). A dial gauge with a resolution of 0.0254-mm (1/1000 inch) was used as a gold standard to measure the translations.

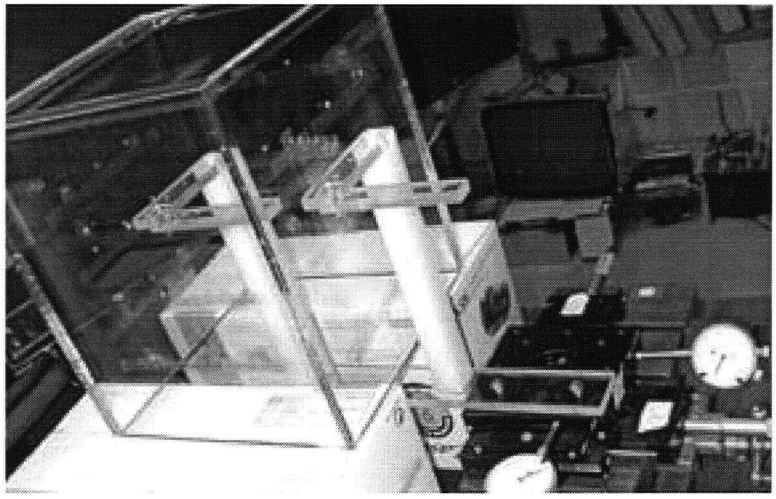

Figure 5-12: Translation Markers
Translation markers were moved in the x-z plane to generate an accuracy map inside and outside of the calibration cage. Translation was measured with dial gauges (1/1000 inch resolution).

In EXP_1 through EXP_3, the object points were translated 0, 12.7, 25.4-mm (0, ½, 1-inch) along the transverse x-axis (Figure 5-13). In EXP_ 4 through EXP_6, the object points were translated 0, 12.7, 25.4-mm (0, ½, 1-inch) along the axial z-axis (Figure 5-14). Figure 5-15 shows the coordinates sampled by the moving object points. The variation in the vertical y-axis translation was not measured and assumed to be the same as the x-axis since both axes are transverse to the x-ray sources. Figure 5-16 shows the setup.

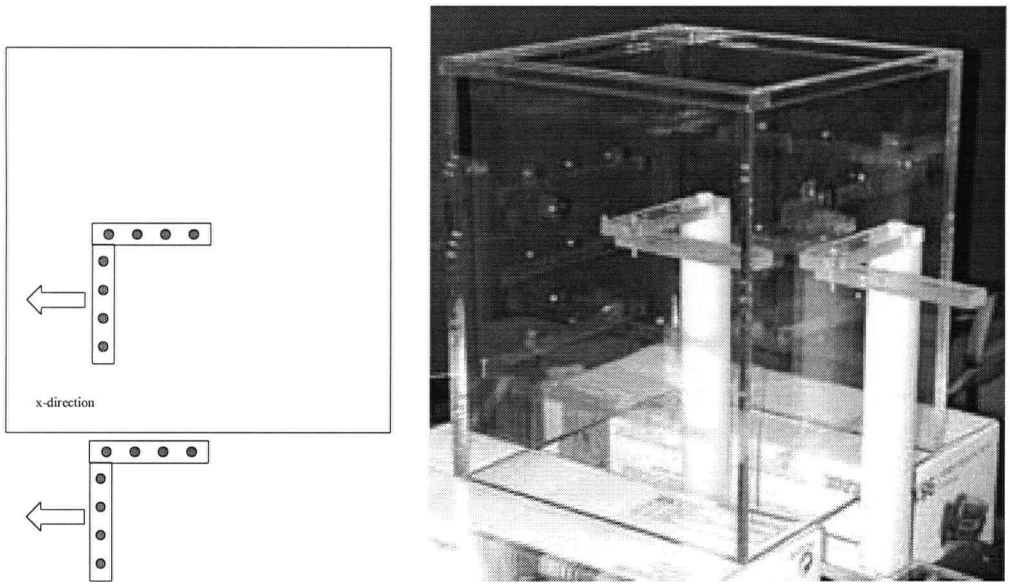

Figure 5-13: Translation x-axis
Translation markers moved along x-axis at 0, 12.7, 25.4-mm (0, ½, 1-inch) intervals.

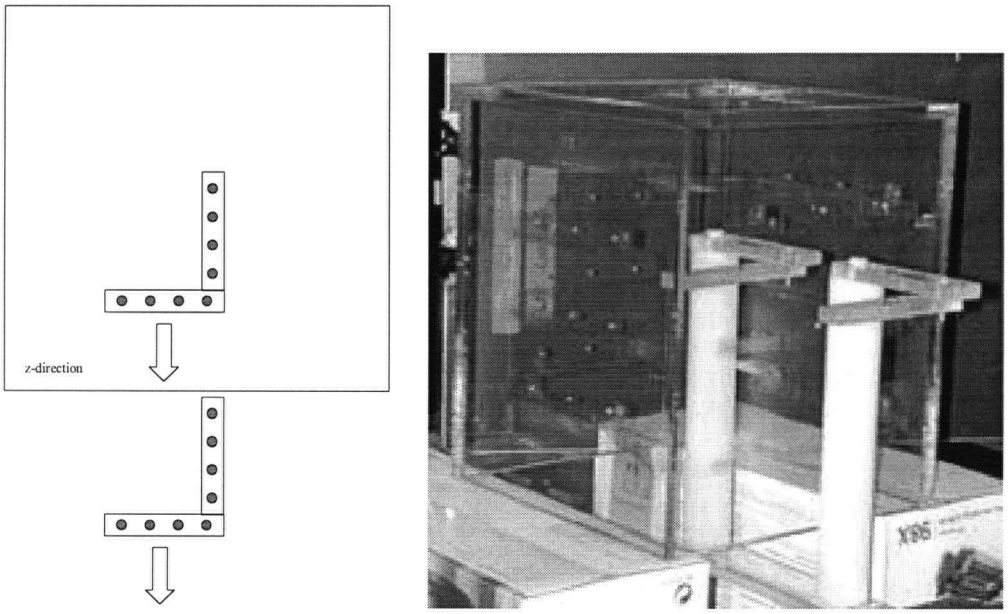

Figure 5-14: Translation z-axis
Translation markers moved along z-axis at 0, 12.7, 25.4-mm (0, ½, 1-inch) intervals.

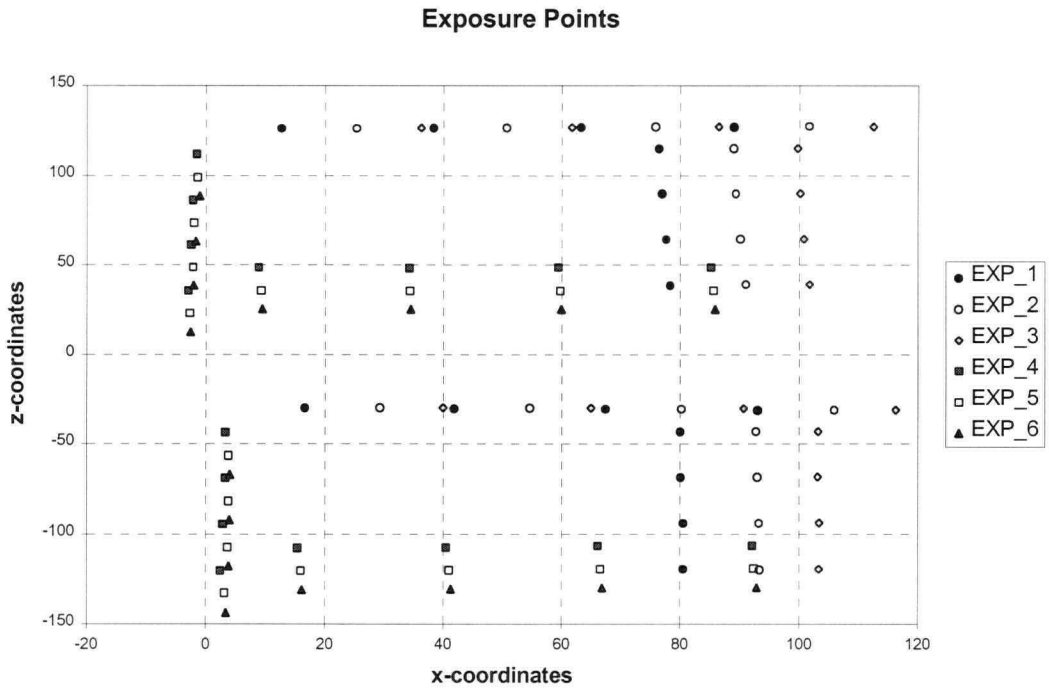

Figure 5-15: Sample Points
Distribution of object points from exposures 1 through 6. Refer to Figure 5-13 and Figure 5-14 for x-axis and z-axis schematics.

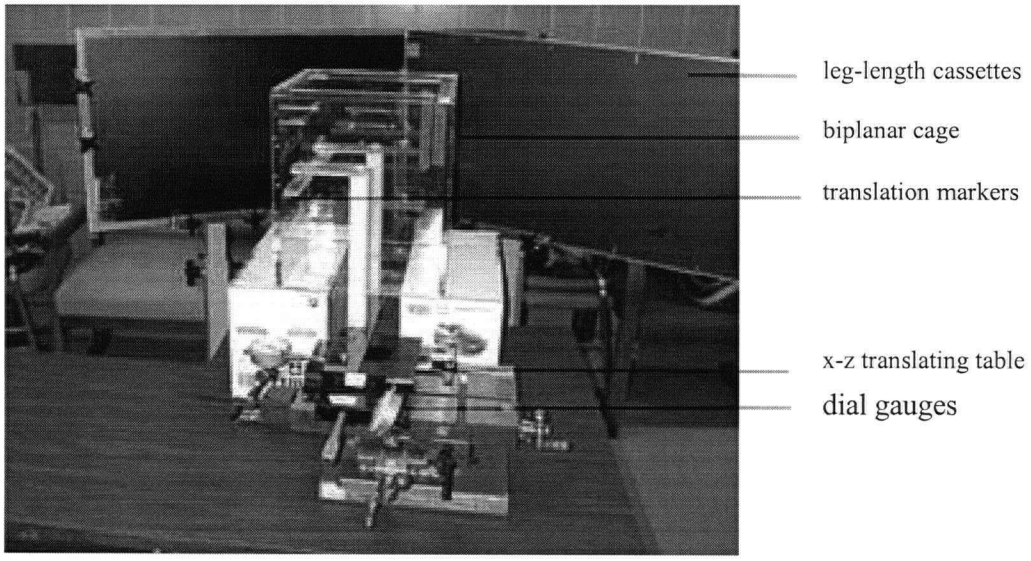

Figure 5-16: Translation Accuracy Setup
Translation accuracy setup. Dial gauges with 1/1000-inch resolution were used to measure translation. 3.0-mm chromium markers embedded in acrylic was used to map out the region inside and outside of calibration cage. Films were at 120° to each other.

EXP_1 to EXP_3 were used to assess translation accuracy in the transverse direction (relative to the line of symmetry of the x-ray sources). EXP_4 to EXP_6 were used to assess translation accuracy in the axial direction.

To estimate the one-dimensional coordinate error (e_x, e_y, or e_z), the one-dimensional translation error was divided by two. Hence, it was assumed that the translation error was equally distributed between the two points used to determine the translation. The one-dimensional coordinate errors were used to estimate a three-dimensional point reconstruction error. The following expressions demonstrate the calculation.

$$e_{\text{Point\_Reconstruction}} = \sqrt{e_x^{\,2} + e_y^{\,2} + e_z^{\,2}}$$

$$e_{\text{Point\_Reconstruction}} = \sqrt{\left(\frac{e_{x\_translation}}{2}\right)^2 + \left(\frac{e_{y\_translation}}{2}\right)^2 + \left(\frac{e_{z\_translation}}{2}\right)^2}$$

$$e_{\text{Point\_Reconstruction}} = \sqrt{2 \times \left(\frac{e_{transverse\_translation}}{2}\right)^2 + \left(\frac{e_{axial\_translation}}{2}\right)^2}$$

In this way, EXP_1 to EXP_3 were used to estimate the transverse coordinate error (e_x, e_y) and EXP_4 to EXP_6 were used to estimate the axial coordinate error (e_z).

The object points were calculated using self-calibration and initial-calibration methods. Table 5-3 and Table 5-4 show the exposure used for calibration and the corresponding exposure used for object point reconstruction in both these scenarios. By definition, for self-calibration, there is only one set of object points. For initial-calibration, the number of sets depends on the combinations used. In this study, there were three sets of object points for transverse translation data and three sets of object points for axial translation data.

Transverse direction analysis		Axial direction analysis	
Calibration Source	Object Points Source	Calibration Source	Object Points Source
EXP_1	EXP_1	EXP_4	EXP_4
EXP_2	EXP_2	EXP_5	EXP_5
EXP_3	EXP_3	EXP_6	EXP_6

Table 5-3: Self-calibration Pairs
In self-calibration, the calibration cage and object points were from the same exposure pair.

Transverse direction analysis		Axial direction analysis	
Calibration Source	Object Points Source	Calibration Source	Object Points Source
EXP_1	EXP_1	EXP_4	EXP_4
EXP_1	EXP_2	EXP_4	EXP_5
EXP_1	EXP_3	EXP_4	EXP_6
EXP_2	EXP_1	EXP_5	EXP_4
EXP_2	EXP_2	EXP_5	EXP_5
EXP_2	EXP_3	EXP_5	EXP_6
EXP_3	EXP_1	EXP_6	EXP_4
EXP_3	EXP_2	EXP_6	EXP_5
EXP_3	EXP_3	EXP_6	EXP_6

Table 5-4: Initial Calibration Pairs
In initial calibration, the same calibration image was used to reconstruct all subsequent points. This was analogous to removing the cage after calibration and having only object points in the subsequent exposures.

Two-way repeated measures analysis of variance was used to compare the effects of direction of translation (transverse, axial), and calibration method (self, initial).

5.3.3 Results

The two-way repeated measures ANOVA showed a significant difference in translation accuracy depending on calibration method and direction of translation ($p<0.001$). Figure 5-17 shows the mean translation error when all results are grouped by calibration method. When self-calibration was used, the mean absolute translation error was 0.25-mm (SD = 0.18) for transverse translation and 0.23-mm (SD = 0.17) for axial translation. When initial-calibration was used, the mean absolute translation error increased to roughly 0.86-mm (SD = 0.14-mm) for transverse translation and 0.54-mm (SD = 0.40) for axial translation.

Figure 5-18 shows the mean translation error when all results were grouped by the direction of translation. The mean error was lower for axial translations ($p<0.001$).

There was also a significant interaction between calibration method and direction of translation ($p<0.001$). When self-calibration was used, there was essentially no difference between the translation accuracy in the transverse and axial directions (Figure 5-19).

Post-hoc SNK tests showed a significant difference in translation error between self-calibration and all initial-calibration combinations (for all $p<0.001$). There was no significant difference between each initial-calibration trails ($p = 0.24$ to 0.49).

For self-calibration, the transverse coordinate error was estimated to be 0.125-mm (0.25-mm ÷ 2) and the axial coordinate error was estimated to be 0.115-mm (0.23-mm ÷ 2). The resultant three-dimensional point reconstruction accuracy was 0.21-mm.

For initial-calibration, the transverse coordinate error was estimated to be 0.43-mm (0.86-mm ÷ 2) and the axial coordinate error was estimated to be 0.27-mm (0.54-mm ÷ 2). The resultant three-dimensional point reconstruction accuracy was 0.67-mm.

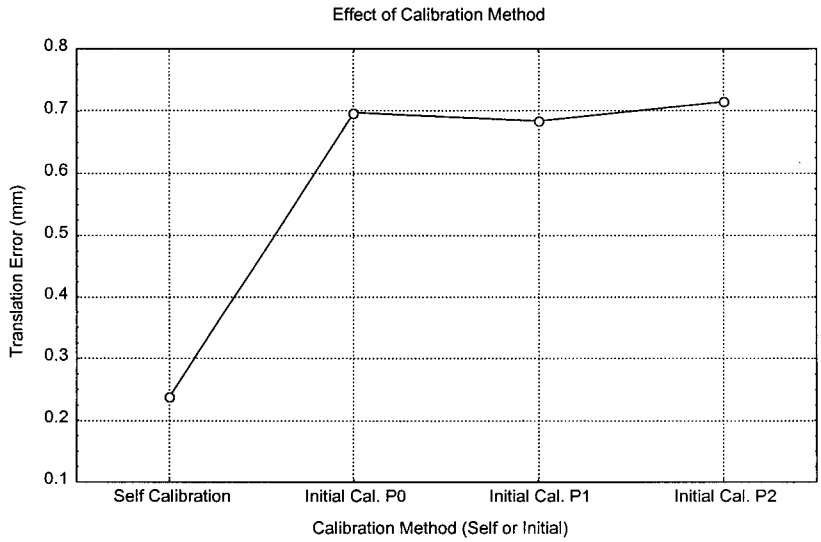

Figure 5-17: Effect of Calibration Method

There was a significant impact on the ability to accurately measure translation when the calibration cage was not present in each exposure. The increase in inaccuracy from the initial-calibration method stemmed from variability introduced by x-ray cassette changes and measurement error. Post-hoc SNK tests between self-calibration and all other initial-calibration combinations yielded $p<0.001$. Between all initial-calibration combinations, p ranged from 0.24 to 0.49.

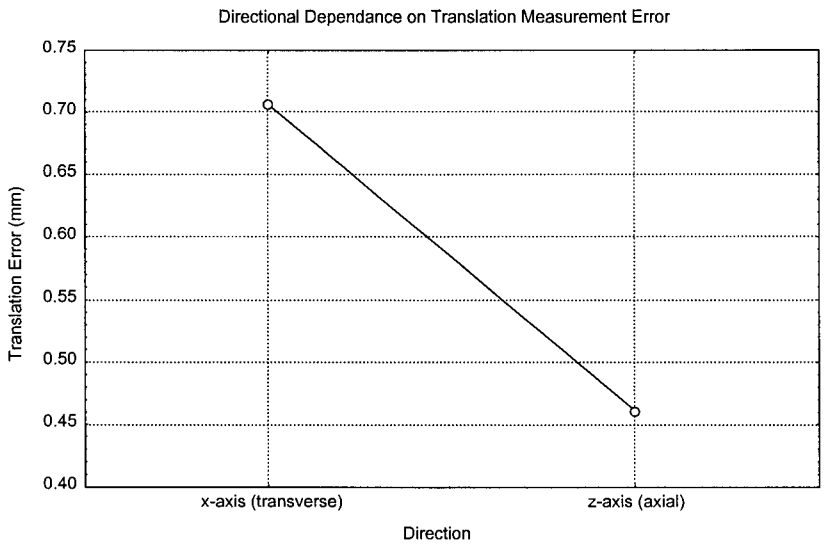

Figure 5-18: Directional Dependence

There was a higher error in the translation error measured transversely as compared to measurements in the axial direction. Post-hoc SNK tests showed $p<0.001$. This result was in disagreement with simulation and results reported in the literature. Increased transverse error may be attributed to the use of leg-length cassettes which were composed of separate x-ray films attached together.

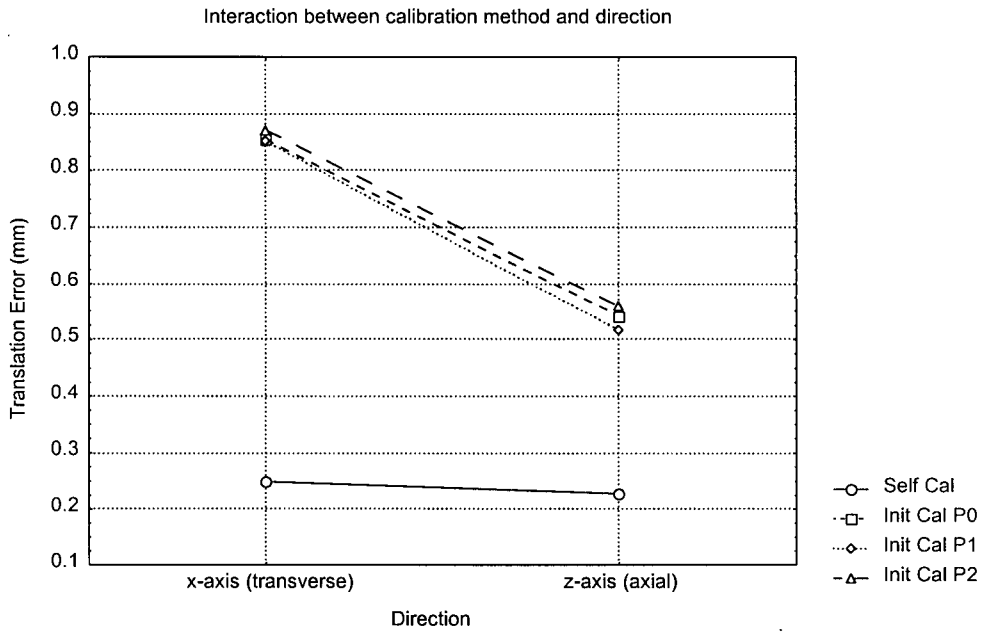

Figure 5-19: Calibration Method & Direction
Interaction between calibration method and direction showing that using self-calibration, the error difference between the transverse and axial direction was small. The interaction was significant ($p<0.001$).

5.3.4 Discussion

It was found during RSA simulations that the best possible point reconstruction accuracy for the biplanar cage was 0.026-mm at the cage centre. For the projective cage using extrapolation, the point reconstruction accuracy at 350-mm from the fiducial plane was 0.14-mm. The projective cage was more than five times less accurate (5.38×).

In this study, when self-calibration was used, the reconstruction accuracy was estimated to be 0.21-mm. When initial-calibration was used, the reconstruction accuracy was estimated to be 0.67-mm. Initial calibration was more than three times less accurate (3.19×).

This result indicates that using the biplanar cage with initial-calibration may offer a 41% [= (5.38 – 3.19)/5.38] improvement in accuracy over using the projective cage with self-calibration and extrapolation.

5.3.5 Conclusion

The removal of the calibration cage from subsequent stereo exposures reduced the accuracy of the RSA system with the biplanar cage by about three times. This was lower than the projective cage extrapolation error found in the RSA simulations. Hence removal of the biplanar cage is a viable alternative for large motions in three-dimensions. Accuracy in the transverse direction was unexpectedly found to be worse and requires further investigation.

5.4 RSA Rotational Accuracy

5.4.1 Objective

This experiment aimed to determine the accuracy of rotational measurements for the specific distribution of markers to be used in the proximal humerus and scapula.

5.4.2 Methods

Six 3-mm chromium spheres were implanted into an acetyl marker carrier. Tests in the RSA accuracy chapter showed no significant difference in measurement error between using 3-mm and 1-mm markers. The arrangement of the spheres was similar to the distribution of markers to be eventually used for future clinical testing (Figure 5-20). To assess the effect of distance between carriers on rotation accuracy, three marker carriers were rigidly mounted onto a rotation table used as a gold standard with a resolution of $1/3600^{o}$ (Figure 5-21). The carrier at the centre of the rotating table was referred to as the scapula carrier (SC). The middle carrier was referred to as proximal humerus (PH) and was located at approximately 8-cm from SC. The outer carrier was referred to as the distal humerus (DH) was located at approximately 16-cm from SC. Although all carriers contained six chromium markers, only the markers appropriate for that carrier (scapular or humeral) were used in the rigid body calculations. Initial-calibration was used. Stereo exposures were taken at 0^{o}, 90^{o}, 180^{o}, 270^{o} and 360^{o}. These intervals covered the entire calibrated volume enabling the detection of possible spatial inhomogeneity in accuracy. Rotation was about the y-axis of the cage coordinate system.

Matlab kinematics routines from the Kinmat toolbox previously described were used to compute the cardan angles for the rotation. The cardan angle sequence RyRxRz was used to avoid the gimbal-lock that would result if the 90° rotation occurred about the second axis in the sequence. (i.e. RxRyRz or RzRyRx)[94,95]

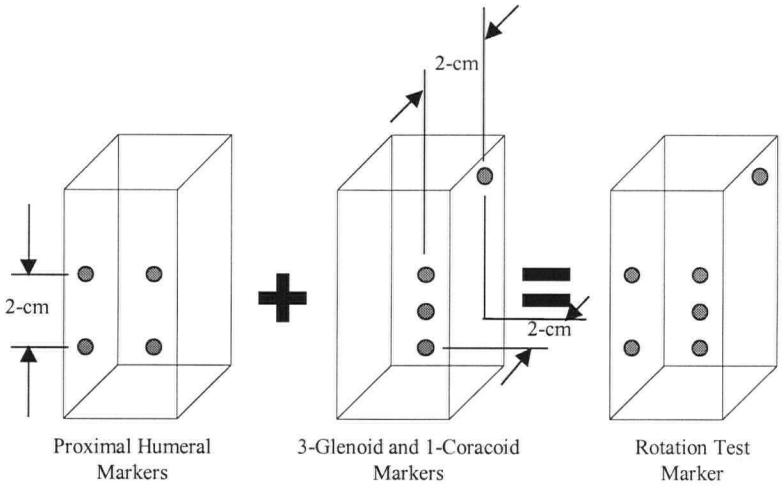

Figure 5-20: Rotation Marker Carriers
Arrangement of 3.0-mm diameter chromium markers used for rotation accuracy test.

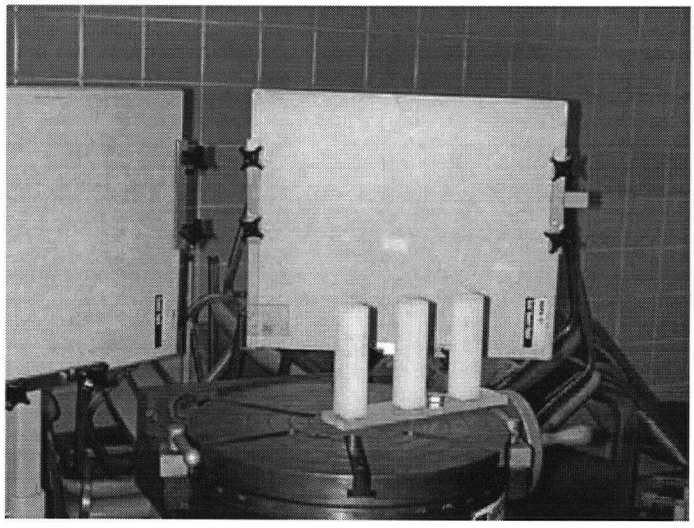

Figure 5-21: Rotation Test Setup
Photo of rotation test setup. Six 3.0-mm chromium markers were embedded into acetyl marker carriers. From the centre of the rotating table moving radially outward the carriers are labeled SC, PH, and DH. The distribution of the markers was similar to what will be used in vivo in the humerus, glenoid, and coracoid. The table was rotated 0°, 90°, 180°, 270°, 360°.

To validate that the RSA system had correctly reconstructed the object positions, the rotation of PH and DH were computed relative to SC's initial position. To assess rotation accuracy, the intersegmental rotation of PH and DH was computed relative to SC. The three bodies moved rigidly on the table and hence the intersegmental motion should ideally be zero. Two-way analysis of variance was used to assess the effect of radial distance (PH vs. DH) and axes (x,y,z) on rotation accuracy.

5.4.3 Results

The rotation of PH and DH relative to the initial position of SC is shown in Table 5-5. Almost all rotations about the y-axis were within half a degree of the value indicated by the rotating table. The only exception was the rotation of the outer carrier (DH) at 180°. The RSA system detected a rotation of only 177.95°.

Proximal Humerus			Distal Humerus		
Rx	Ry	Rz	Rx	Ry	Rz
-0.793°	90.626°	-0.223°	-1.190°	90.065°	0.039°
-0.850°	-179.582°	0.775°	-0.480°	177.950°	0.867°
-0.487°	-89.402°	0.321°	-0.716°	-90.330°	0.589°
0.087°	0.392°	-0.081°	0.008°	-0.383°	-0.015°

Table 5-5: Absolute Rotations
Rotation of PH and DH carriers relative to initial position of SC carrier. Ideal rotations would be 90°, 180°, 270°, 360° (or 0°).

The intersegmental motion of PH and DH relative to SC is shown in Table 5-6. The mean absolute error for PH was 1.05°. The mean absolute error for the DH was 0.856°.

Proximal Humerus			Distal Humerus		
Rx	Ry	Rz	Rx	Ry	Rz
0.126°	1.263°	0.439°	-0.277°	0.707°	0.710°
-0.133°	0.755°	0.387°	0.253°	-1.710°	0.510°
-0.020°	0.642°	-0.284°	-0.240°	-0.289°	-0.008°
-0.487°	1.529°	0.591°	-0.575°	0.755°	0.650°
Mean of Absolute errors			**Mean of Absolute errors**		
0.192°	**1.047°**	**0.425°**	**0.336°**	**0.856°**	**0.470°**

Table 5-6: Relative Rotations
Rotation of PD and DH carriers relative to SC carrier. There was no intersegmental motion and ideally, all rotations should have been zero.

Analysis of variance showed no significant effect of radial distance ($p<0.99$, 8-cm vs. 16-cm). A significant difference was found between the three axes with the y-axis having a significantly larger error than either x or y ($p<0.002$). There was no significant interaction ($p<0.64$) between radial distance and axes (Figure 5-22).

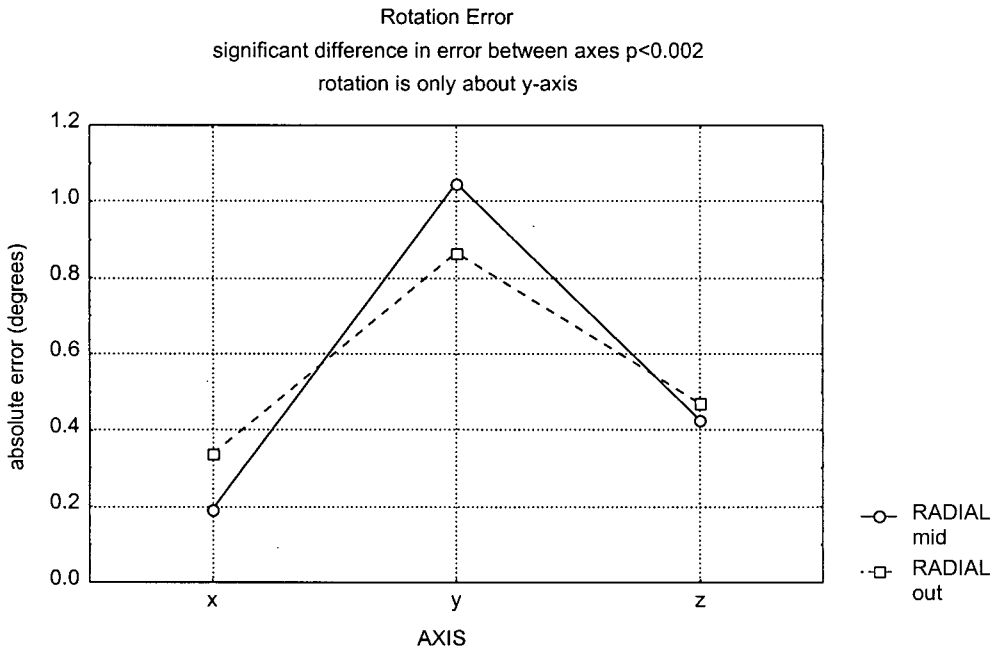

Figure 5-22: Rotation Error
A significant increase in rotation error was found about the y-axis. This was the axis about which there was rotation. There was no significant effect of radial distance (8-cm vs. 16-cm) from the centre (reference) object. There was no significant interaction.

5.4.4 Discussion

The RSA system's accuracy of measuring rotation was poorer than expected. An intersegmental rotation of 1.7° was found in one instance when there should have been zero intersegmental rotation.

Rotation about the y-axis involved motion in the axial direction relative to the x-ray sources. Although it was not supported by the results in the translation accuracy test, in theory and in simulation, the axial direction has the poorest reconstruction accuracy.

This may be the reason for the high errors observed about the y-axis. Vrooman's group reported a similar phenomenon.[89]

Another contributing factor may be the configuration of the scapula markers. These markers are very close to being collinear and others have discussed that the resultant accuracy is decreased.[81] Nevertheless, surgical exposure conditions limit the flexibility of scapular marker placement and these results indicate the errors that should be expected for future clinical trials.

5.4.5 Conclusion

The mean absolute accuracy for measuring rotation was 0.5° about the primary axis of rotation. Significant errors were found – one as large at 1.7°. Marker configuration and axis of rotation may be the cause of larger than expected errors. There was no significant effect on accuracy for the intersegmental distances used (8-cm and 16-cm).

5.5 Summary Conclusions for RSA Accuracy Studies

Issue	Calibration Method	Quantity Assessed	Value
Digital Measurement Accuracy	n/a	error from centre of circle	0.24-mm to 0.23-mm (0.0 to 0.02)
		effect of sphere diameter & contrast	negligible for 1.0-mm and 3.0-mm with T < 70%
Reconstruction Accuracy	initial	***pt reconstruction accuracy***	***0.43-mm (0.03)***
Influence of Cassette Changes	initial	mean variability in x-coord	0.78-mm (0.19)
		mean variability in y-coord	0.55-mm (0.19)
		mean variability in z-coord	0.97-mm (0.17)
		difference between axes	p < 0.001
Translation Accuracy	self	translation measurement error when translation in transverse direction	0.25-mm (0.18)
		inferred x-axis and y-axis error (ex and ey)	*0.125-mm*
		translation measurement error when translation in axial direction	0.23-mm (0.17)
		inferred z-axis error (ez)	*0.115-mm*
		inferred pt reconstruction accuracy sqrt(ex^2+ey^2+ez^2)	***0.21-mm***
	initial	translation measurement error when translation in transverse direction	0.86-mm (0.15)
		inferred x-axis and y-axis error (ex and ey)	*0.43-mm*
		translation measurement error when translation in axial direction	0.54-mm (0.47)
		inferred z-axis error (ez)	*0.27-mm*
		inferred pt reconstruction accuracy sqrt(ex^2+ey^2+ez^2)	***0.67-mm***
	n/a	*effect of calibration method, direction of translation, and interaction*	*p < 0.001*
Rotation Accuracy	initial	Rx (r = 8-cm, r = 16-cm)	0.192 to 0.336 degrees
		Ry (r = 8-cm, r = 16-cm)	0.856 to 1.047 degrees
		Rz (r = 8-cm, r = 16-cm)	0.425 to 0.470 degrees
		no significant effect of radial distance	p > 0.05
		significant difference between axes	p = 0.002

Table 5-7: RSA Accuracy Summary

Self-calibration showed the best reconstruction accuracy. The initial-calibration results from the reconstruction accuracy test and translation accuracy test were not the same (0.43-mm versus 0.67-mm). This was likely due to different methods of measuring accuracy discussed previously (Figure 5-11). In the reconstruction test, distances were calculated for points on the same image. In the translation test, the distances were calculated between points on different images. The reduction in accuracy was due to additional variability between images.

Chapter 6

SCAPULAR KINEMATICS IN A CADAVER

6.1 Objective

The primary objective of this test was to further assess the feasibility of using skin surface markers to measure scapular kinematics by comparing skin marker data to that obtained from bone embedded RSA markers. The hypothesis was that the motion - as approximated from a series of static positions - of the skin surface could be used to infer the motion of the underlying bone. A threshold of 5^{o} was pre-selected as a reasonable margin of error below which the data would be considered promising. The secondary objective was to identify areas in the experimental methods that require refinement before the study could be advanced to in vivo tests. The measurement of scapulothoracic or scapulohumeral rhythm, which has been the topic of many studies, was not within the scope of this study. The concept of rhythm implies synchronized movement controlled by active muscles. This was not applicable in this cadaveric study.

6.2 Methods

Specimen

The measurements were performed on the right shoulder of a fresh cadaver (24-hrs. to 48-hrs. postmortem). The subject was a 70-year old male, approximately 170-cm in height and 70-kg in weight. The length of the scapular spine was 14.5-cm. The medial border was 15.0-cm. The skin and soft tissue thickness was measured with a caliper. The values measured were 1.0-cm on the acromion, 1.5-cm on the mid-scapular spine, 1.5-cm on the medial edge of the scapular spine and 1.5-cm at the inferior angle of the scapula. The specimen was lean with qualitatively good skin tension hence minimizing skin motion artifact.

Measurement System

Roentgen stereophotogrammetric analysis (RSA) was used to accurately measure the skeletal positions of the thorax, scapula and humerus. The system was previously shown to have accuracy ranging from 0.54-mm to 0.86-mm for translation, and 0.192° to 1.047° for rotations. (refer to chapter on "RSA Accuracy Studies") RSA was used to measure the positions of embedded bone markers and also the skin surface markers. A wide variety of methods can be used to acquire skin marker data such as the Optotrak 3020 system used in the pilot study. The use of RSA to measure the skin markers in this test was for convenience since radiation exposure to the subject was not an issue.

Implanted Bone Markers

Five 0.8-mm chromium spheres were implanted into the scapula. Four of these were placed into the acromion. The fifth was placed in the coracoid process. Five 0.8-mm chromium spheres were implanted into the proximal humerus. Although the intended geometry was a 2-cm × 2-cm square, the hardness of cortical bone resulted in damage to one of the two implant guides. Subsequently, the resulting marker placement was dictated by the ability to easily penetrate cortical bone. Figure 6-1 shows photos of the implant instruments used.

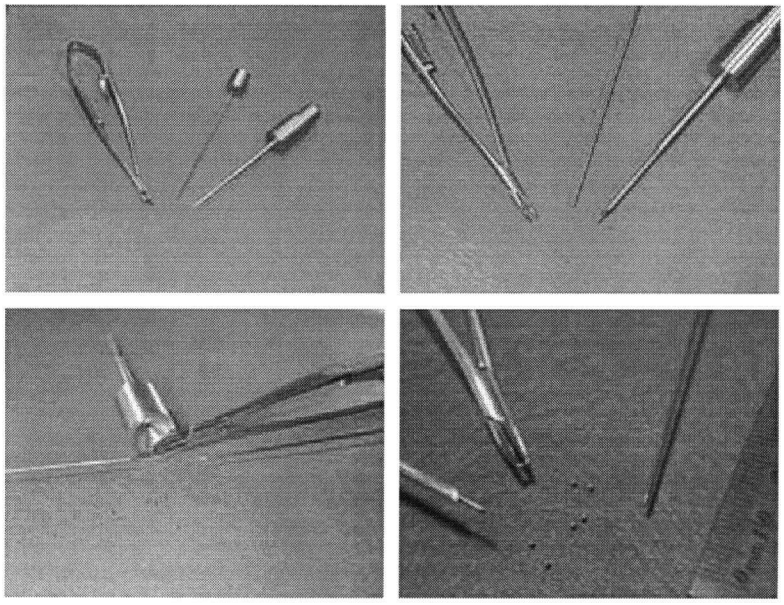

Figure 6-1: Implantation Instruments

The instruments were made of stainless steel. The guide consisted of three concentric tubes (0.051"OD×0.038"ID, 0.072"OD×0.054"ID, 0.109"OD× 0.077"ID) silver soldered. The tip was beveled to ~60°. A castroveijo locking needle holder was used to manipulate the 0.8-mm balls.

A thoracic marker carrier was rigidly screwed to the thoracic spine (region of T7 to T9). This carrier served as the anatomical base coordinate system and accounted for motion of the cadaver body as a whole during the experiment (Figure 6-2).

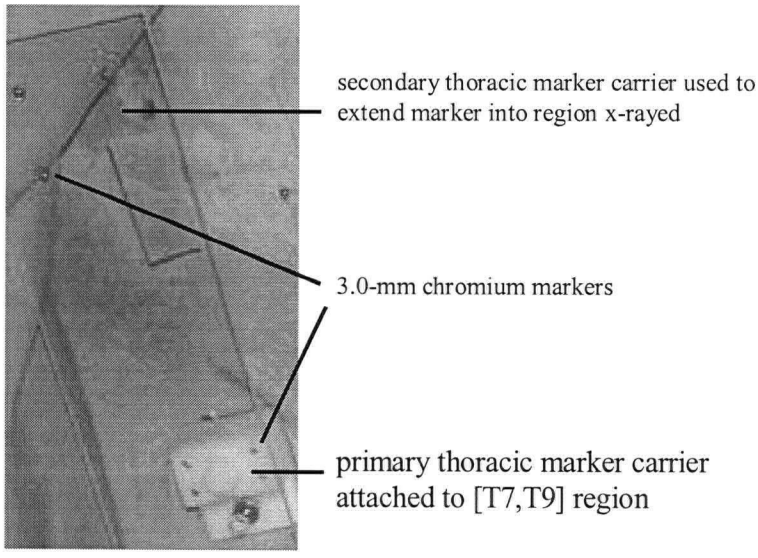

Figure 6-2: Thoracic Marker Carrier

Skin Surface Markers

An arm marker carrier consisting of four 3-mm chromium spheres was strapped to the lateral surface of the proximal arm. Trial exposures showed this carrier to be out of the field of view and it was removed. Alternate means of defining the humeral local coordinates were used during the analysis.

An *array* of sixty-eight 3-mm spheres was attached to the skin surface with super glue. Minimal glue was used and did not qualitatively appear to affect the elasticity or motion of the skin surface. The resulting array of skin markers formed fifty-one grid patches of approximately 2-cm × 2-cm. The interval was chosen to maximize surface sampling, while minimizing the risk of spheres obstructing each other on the x-ray images. The exact geometry is shown in Figure 6-3. The first horizontal row of patches lay superior to the scapular spine. The second row lay approximately over the scapular spine, and the third row lay inferior to the scapular spine. The markers extended medially beyond the medial boarder of the scapular spine and inferiorly almost to the inferior angle. Surface markers were not placed on the acromion to prevent obstruction of the bone markers implanted there. The results of the pilot study suggested that skin on the acromion overestimated scapular upward rotation due to its position along the line of action of the humerus.

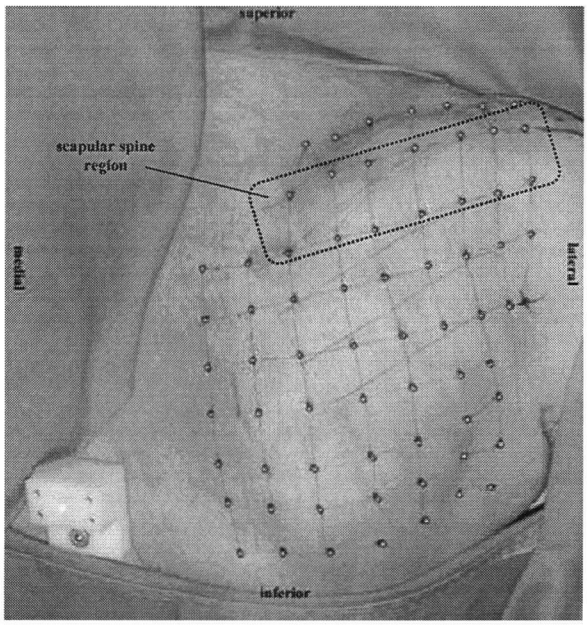

Figure 6-3: Grid on Skin Surface
A total of 68 3-mm stainless steel spheres were used as skin surface markers. The markers were attached with super glue. The glue did not qualitatively appear to influence the motion of the skin surface.

Experiment

The x-ray sources were initially oriented at 60°. Trial exposures of the cadaver were taken at 0° and maximum elevation. It was found that this geometric setup was not suitable because at certain humeral elevations, the humerus would be imaged axially resulting in significant obstruction of bone and skin markers. The final orientation of the sources was at approximately 30°, which resulted in some asymmetry of the setup. With this orientation, the entire calibration cage was barely within the imaging field of the 14"×17" x-ray cassettes.

The imaging space was initially calibrated. The specimen was secured sideways and moved into the calibrated volume (Figure 6-4). Straps were tethered to the ceiling and used to hold the arm in its various positions. Trial exposures were taken to ensure the markers were visible at the extremes of the range of motion. Humeral elevation exposures were taken at 0°, 45°, 90°, 135° and maximum (~180°) elevation. Humeral retraction exposures were taken at 0°, 45°, 75° and 90° where 0° was defined as 90° of forward arm elevation in the sagittal plane. The space was re-calibrated.

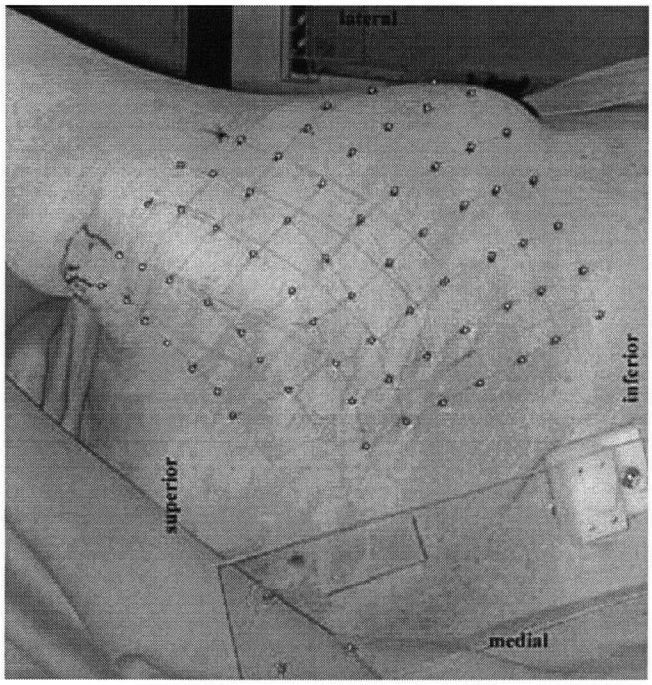

Figure 6-4: Specimen on Side
The specimen was secured on the side and tethers were attached to the ceiling and used to mobilize the arm.

The x-ray voltage was set at 120-kV – the maximum voltage on most mobile units. The setting was chosen to generate maximum contrast between the bone and metallic spherical markers. In an in vivo subject, this high kV also ensures a higher proportion of the x-rays will pass through the subject without interacting with tissue thus minimize radiation dosage. The milliamp-seconds was varied and found to be best at 5-mAs.

Digitization

The x-rays were scanned at 150-ppi using a Vision Ten x-ray scanner. Scion Image (www.scioncorp.com) was used to digitize the scanned x-ray images. The residual error of reconstructing each marker's three-dimensional position was used to resolve pairing ambiguity of points in any given stereo pair of images (Figure 6-5).[65] Ambiguity of assigning labels to markers between sequential exposures was resolved by computing the distance between the markers (Figure 6-5).[65] For implanted markers in rigid bodies, the distance between markers remains unchanged from exposure to exposure. Both these processes were implemented in Matlab with custom scripts (refer to appendix).

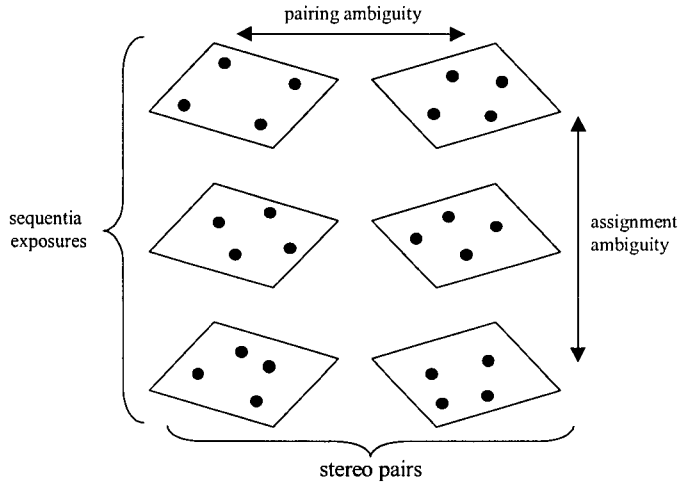

Figure 6-5: Pairing and Assignment Ambiguity
Pairing ambiguity was resolved by assessing the residual error of three-dimensional reconstruction. Appropriate pairing was indicated by a low error. Assignment ambiguity was resolved by considering the lengths between 3-D markers. Appropriate assignment resulted in consistent lengths between reconstructed markers from one exposure to the next.

Analysis

The anatomical coordinate systems were defined using the stereo exposure at 0° of humeral elevation (Figure 6-6). The thoracic coordinate system was defined in the scapular plane as has been reported by other authors.[21,54,73] The origin was set at thoracic marker 2. The z-axis was defined normal to the scapular plane directed posteriorly. The x-axis was directed along the scapular spine from the medial angle towards the acromion.

The scapular coordinate system was defined as proposed by the International Shoulder Group (http://www.wbmt.tudelft.nl/mms/dsg/intersg/isg.html). The origin was set at the medial angle of the scapula. The x-axis was directed along the scapular spine towards the acromion. The z-axis was normal to the scapular plane directed posteriorly.

The humeral external marker carrier was discarded during the exposures because it was not in the field of view. To define a humeral coordinate system, anatomical features were digitized on the two radiographs. The origin was set at the superior prominence

of the greater tuberosity. The y-axis was defined from the lateral prominence of the deltoid tuberosity to the greater tuberosity. A temporary humeral z-axis used to define the y-z plane was defined pointing posteriorly from the lesser tuberosity to the greater tuberosity. The reconstruction residual error for anatomical features was 11.3-mm for the greater tuberosity, 15.0-mm for the lesser tuberosity, and 2.2-mm for the deltoid tuberosity. For comparison, the average residual error was 4.2-mm for bone embedded markers and 5.4-mm for skin markers.

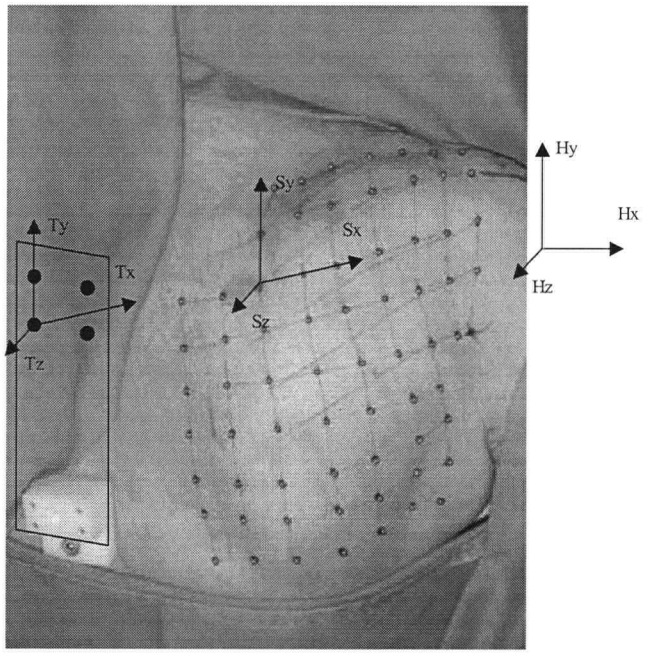

Figure 6-6: Anatomical Coordinate Systems
The thoracic coordinate system was defined in the scapular plane and its origin was set to thoracic marker #2. (The thoracic marker carrier has been digitally added. Refer to Figure 6-2 for a photo.) The scapular coordinate system was located at the medial angle of the scapula. Due to imaging difficulties, the humerus coordinate system was defined from bony landmarks observed on the radiographs.

Standard kinematic analysis[35,39,70,78,83,88,93,96,97,101] was performed in Matlab using the Kinmat toolbox previously described. The function "soder.m" solved for the transformation of a set of markers using singular value decomposition described by Soderkvist and Wedin.[81] The function "rxyzsolv.m" solved for the cardan angles in the sequence $R_zR_yR_x$. The residual error of rigid body fitting was used as a measure of deformation.

6.3 Results

Of the five implanted markers into the scapula, only four markers were visible on the radiographs. Of the five implanted markers into the humerus, only four were visible on the radiographs. The fate of the two missing markers was unresolved.

Roughly six to eight trial stereo exposures (twelve to sixteen x-rays) were taken of the specimen in attempts to ensure all markers were in the field of view and not obstructed.

The purpose of the two calibration exposures, initial and final re-calibration, was to provide a check of setup rigidity throughout the test. However, between the initial calibration and the fine adjustment of the specimen in the imaging volume, x-ray source 1 was slightly bumped. As a consequence, all reconstruction was performed using the final re-calibration exposure only, and no check of system rigidity was conducted.

Difficulties were encountered trying to control the axial orientation of the humerus using the tethered straps. In addition, maintaining the humerus in the formal anatomical position restricted humeral elevation to 90^{o} of abduction in the coronal plane. Consequently, for 135^{o} and maximum elevation, the humerus was allowed to internally rotate about its axis and it rotated about the glenohumeral joint to assume a position in the scapular plane. Conversely, it was not possible to use the scapular plane for the full range of motion. For exposures at 45^{o} and 90^{o} of forward elevation in the scapular plane, the humerus would have been imaged axially thus obstructing many markers.

The presence of soft tissue made the x-ray images of the markers more complex than modelled in the measurement accuracy test and the RSA accuracy tests where there was no soft tissue surrounding the markers. Figure 6-7 shows examples of poor images. The effect of soft tissue and non-uniform x-ray beam resulted in image asymmetry that complicated centre identification. When necessary, images were enhanced in Adobe Photoshop by reducing the input gray scale.

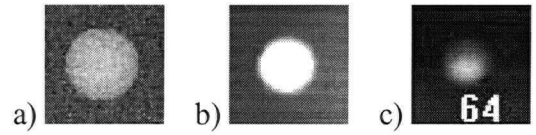

Figure 6-7: Examples of Poor X-rays
a) Image of a marker used in the measurement accuracy test. The image was symmetric. b) Image of a marker used in the RSA accuracy tests where no soft tissue was present. The image was also symmetric. c) Particularly poor image of a marker in the cadaver study. Image was enhanced in Adobe Photoshop. The effect of soft tissue and non-uniform x-ray beam resulted in image asymmetry making centre identification problematic.

At 0° of retraction the humerus was imaged axially resulting in significant obstruction of many skin and all bone markers. Consequently, this stereo exposure was discarded and the retraction positions were calculated with respect to 0° of humeral elevation.

At 0° of humeral elevation, it was not possible to detect one of the four thoracic markers. Since this was the initial exposure, relative to which all subsequent exposures were compared, only three markers were used to track the thoracic frame. This did not appear to have a detrimental impact on the error of calculating rigid body motion for the thoracic frame marker carriers. The average error of rigid body fitting for the thoracic marker carrier was 0.15-mm. For comparison, the average errors for the scapular and humeral marker carriers were 0.33-mm and 0.38-mm respectively. The average rigid body fitting error for the deformable skin patches was 1.74-mm.

No difficulties were encountered matching stereo pairs of skin markers and correctly assigning the same label to each reconstructed marker between subsequent stereo images. However, ambiguities were encountered for the humeral and scapular bone markers. Table 6-1 shows an example of digitization data for the stereo pair at 45° of humeral elevation. The points were labeled and paired. Table 6-2 shows the point reconstruction between every possible pair along with the reconstruction error. The pairings were ranked to determine correct point pairs. In this example, the coracoid marker was correctly paired "p8-8" whereas points 5, 6 and 7 were initially paired incorrectly. The result is most easily understood by tracing the descriptions in Table 6-1 and Table 6-2.

Point	F1_u	F1_v	Point	F2_u	F2_v
Origin	**68**	**1631**	**Origin**	**80**	**1604**
x-axis	**168**	**1630**	**x-axis**	**180**	**1604**
y-axis	**68**	**1536**	**y-axis**	**78**	**1510**
1 **HD**	1090	556	1 **HD**	525	529
2 **HX**	986	537	2 **HX**	406	500
3 **HL**	937	522	3 **HL**	377	484
4 **HM**	944	567	4 **HM**	396	532
5 **AA**	633	593	5 **AA**	135	655
6 **AP**	652	651	6 **AP**	185	545
7 **AM**	666	698	7 **AM**	231	610
8 **CR**	949	902	8 **CR**	392	886

Table 6-1: Example Digitized Bone Markers

Film points were tentatively labeled and paired. Pixel coordinates from film 1 (F1_u,F1_v) and film 2 (F2_u,F2_v) were measured. Coordinates were transformed to local film coordinates (Origin, x-axis, y-axis) and 3D coordinates were reconstructed. Refer to Table 6-2 for correction of pairings.

Pair	x	y	z	Error
p7-5	*115.78*	*108.74*	*196.68*	*0.5089*
p8-8	78.2836	77.5177	215.343	1.42065
p2-2	76.4897	132.796	224.244	1.49141
p5-6	*110.22*	*123.72*	*179.84*	*1.5137*
p6-7	*104.09*	*114.73*	*175.47*	*1.7208*
p4-4	78.5253	127.894	215.553	2.41583
p1-1	59.4678	129.302	225.987	2.92564
p3-3	81.1098	134.811	217.896	3.06433
above, all points matched – below, unused pairings				
p4-1	62.7769	127.077	190.278	4.92555
p2-3	80.1699	134.235	230.207	10.1779
p1-4	75.5011	130.182	252.561	10.594
p3-2	77.4492	133.379	212.02	14.5868
unused pairings deleted for brevity				
p8-2	76.9197	105.47	210.704	305.754
p2-8	75.8825	104.354	220.205	310.479
p8-3	80.4914	106.758	216.208	318.722
p3-8	76.7692	105.132	207.718	321.251

Table 6-2: Example Point Pairing

Data from Table 6-1 was reconstructed with all possible point pairings between film 1 and film 2. The reconstruction error was used as a measure of correct pairings. Rows in *italics* highlight points that were mistakenly paired in Table 6-1.

An example of resolving the assignment problem is shown in Table 6-3 and Table 6-4. Table 6-3 shows the bone marker points in the initial position at 0° of humeral elevation. The distances between the coracoid marker, denoted "CR", and all other markers on the scapula are shown. The distances between one of the humeral markers, denoted "HD", and all other bone markers are also shown. In this exposure, the scapular and humeral markers were clearly distinct. Table 6-4 shows the results of similar distance calculations. The characteristic distances in Table 6-3 were used to identify ("New Tag" column) the reconstructed markers in Table 6-4. Note that the "HD" marker was only misidentified once. In that particular case, none of the lengths, except one, corresponded to the lengths in Table 6-3. The one matching length was correctly inferred to be between the misidentified "HD" and the true "HD".

Distance between points 0° of humeral elevation							
F1_ID	F2_ID	x	y	z	D_HD	D_CR	Tag
1 HD	HD	42.81466	131	256.8795	n/a	-	H0
2 HX	HX	56.03647	142	253.5716	17.52172	-	H1
3 HL	HL	59.26794	146	246.7068	23.91423	-	H3
4 HM	HM	60.76051	138	245.4859	22.18623	-	H2
5 AA	AA	102.8926	143	218.9653	-	66.35744	A2
6 AP							
7 AM	AP	106.3833	129	237.2838	-	52.27757	A1
8 CR	CR	70.91058	91	245.3918	-	n/a	CR

Table 6-3: Reference Distance between Markers
Distance between points at 0° elevation. The lengths in this table were used as a reference to correctly assign marker labels to subsequent stereo exposures. Note that marker A3 was missing and its appropriate distance from the coracoid marker (CR) was determined from other exposures.

Distance between points 45° of humeral elevation							
F1_ID	F2_ID	x	y	z	D_HD	D_CR	New Tag
1 HD	HD	59.46779	129	225.9868	n/a	-	H0
2 HX	HX	76.48973	133	224.2437	17.46407	-	H1
3 HL	HL	81.1098	135	217.8961	23.75267	-	H3
4 HM	HM	78.52535	128	215.5534	21.77222	-	H2
5 AA	AP	110.2221	124	179.8423	-	66.44326	A2
6 AP	AM	104.0931	115	175.4725	-	60.33671	A3
7 AM	AA	115.7753	109	196.6806	-	52.23766	A1
8 CR	CR	78.28362	77.5	215.3435	-	n/a	CR

Table 6-4: Marker Distances at 45° Elevation
Comparison with reference distances (Table 6-3) indicates correct marker assignment.

The results for all fifty-one patches for rotations about R_z, R_y, R_x, for humeral elevation and retraction can be found in the appendices. The main results are summarized here along with graphs for the best patches.

The data for three-dimensional humeral poses did not correspond to the intuitive two-dimensional target poses of 0°, 45°, 90°, 135° and 180° of humeral elevation. (0°, 45°, 75°, 90° of humeral retraction) The reasons for this are addressed in the discussion. To make the data simpler to interpret, the rotations of the scapular bone markers and skin markers were plotted with respect to the targeted humeral pose rather than the calculated humeral angles about the respective axes.

Figure 6-8 shows the RMS error for each patch in measuring scapular upward rotation. Only patches 45 and 51 on the lateral and lateral-superior aspect of the scapular spine had an overall error less than 5°. However, for many patches, a high error was observed at maximum (~180°) elevation. Figure 6-9 shows that in addition, patches 31, 38, 43, and 44 also measured scapular upward rotation to within 5°, deviating only at maximum elevation. These skin patches represented the region of the lateral aspect of the scapular spine, the lateral-superior aspect of the scapular spine, and a region lateral-inferior to the scapular spine.

The scapular attitudes in Figure 6-9 suggest that in this cadaver model, where muscle activation was not present, the scapula did not exhibit a strict continuous upward rotation below 90° of elevation. There was little change between 45° and 90°.

For internal/external rotation of the scapula, the RMS error for virtually all patches was below 15°. Ten patches had an overall error less than 5° (Figure 6-10). A clustering of good patches was observed on the scapular spine – patches 41, 42, 43, 44 and 45.

The scapular attitudes indicate a complex pattern of motion. The scapula remained externally rotated relative to its rest position, however, there appeared to be fluctuations in the extent of rotation throughout the motion.

For scapular tipping, the scapular spine patches showed poor results. The RMS errors for patches 41 to 45 ranged from 15.9° to 22.2°. The best results were observed for patches inferior to the lateral scapular spine (Figure 6-11). Patches 30, 37, 38 and 39 had RMS errors ranging between 3.3° and 5.3°.

The scapular attitudes suggest a pattern of motion where the inferior angle progressively tipped anteriorly.

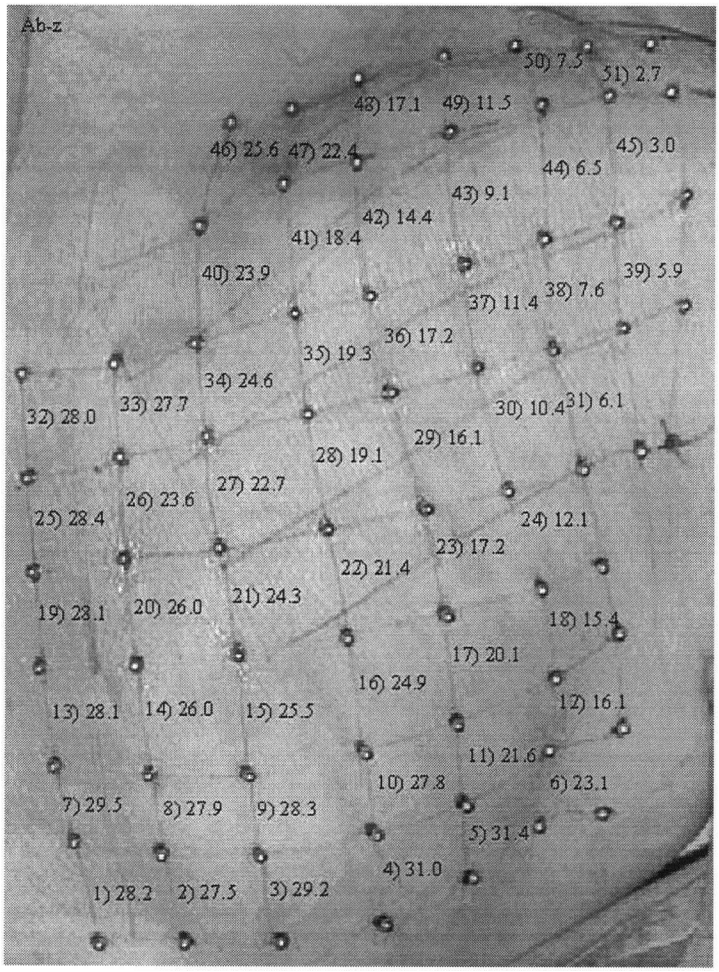

Figure 6-8: Scapular upward rotation RMSE during humeral elevation
RMSE (°) computed for 45°, 90°, 135°, 180° of targeted humeral elevation using rotations measured with embedded bone RSA markers as the gold standard.

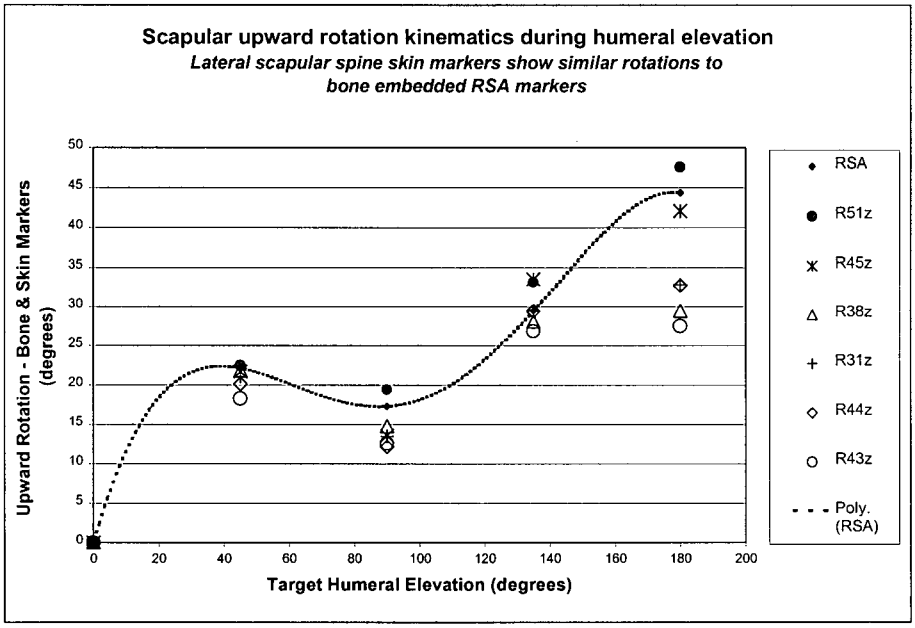

Figure 6-9: Scapular upward rotation - lateral scapular spine patches
Below 135° of elevation, the markers on the lateral region of the scapular spine and the lateral inferior region of the scapular spine showed very similar rotations compared to the bone markers. Deviation between skin and bone markers increased at maximum elevation.

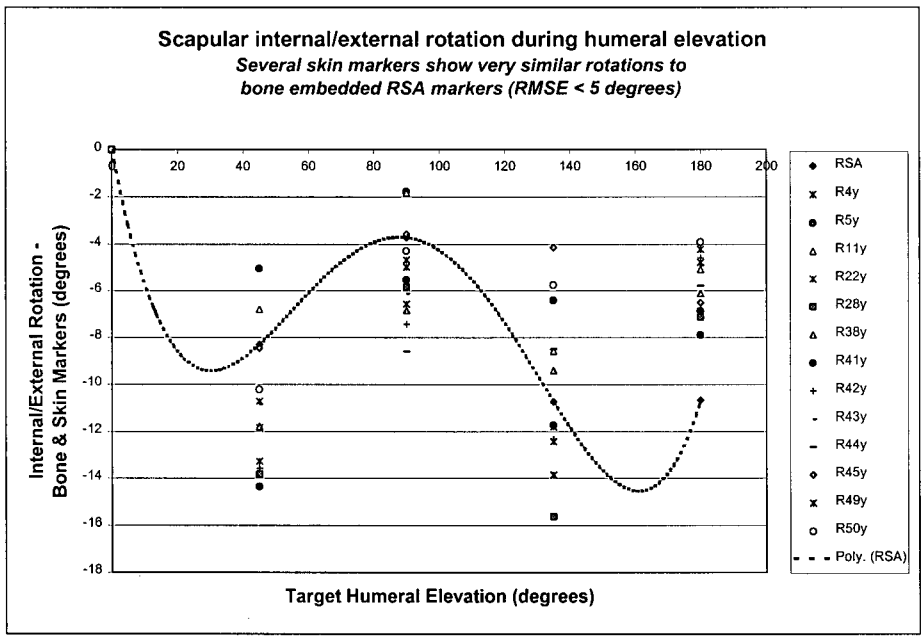

Figure 6-10: Scapular internal/external rotation - RMSE < 5°
No general trend was observed indicating a particular region of skin markers was the best candidate for measuring internal/external scapular rotation. Patches on the scapular spine showed good results. This graph shows the best skin patches for which the RMSE was below 5°.

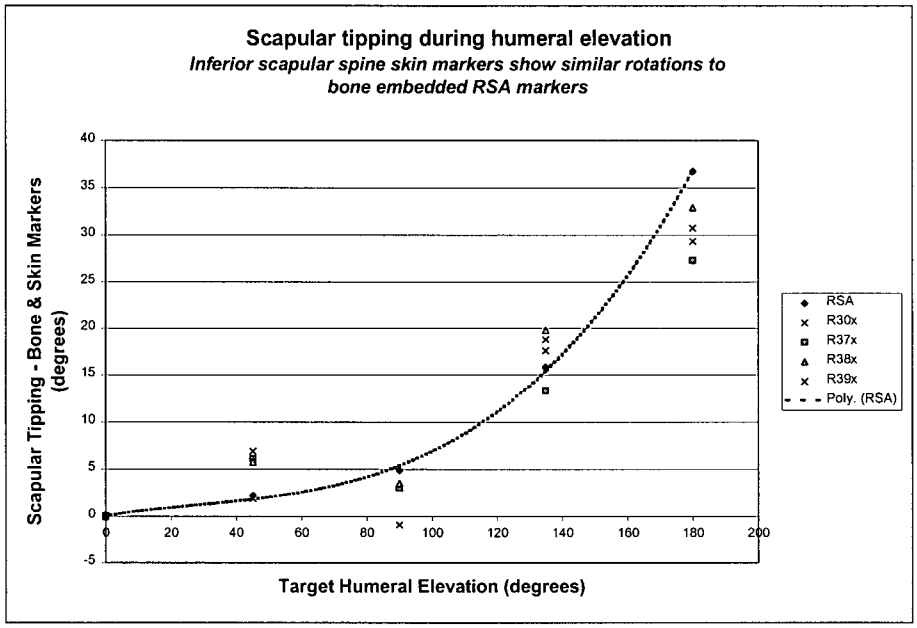

Figure 6-11: Scapular tipping - inferior scapular spine patches
Patches 30, 37, 38 and 39 lay in the region just inferior to the scapular spine. These patches showed the most similar rotations about the x-axis (axis along the scapular spine). The patches directly on the scapular spine showed poorer results for rotation about this (scapular spine) axis. This intuitively made sense since the scapular spine patches were measuring the rotation of a cylindrical-like structure beneath the skin.

The results for humeral retraction were similar to elevation. For scapular upward rotation, the best patches were on the lateral aspect of the scapular spine, along with patches directly superior and inferior to the lateral scapular spine. For internal/external rotation, many more patches were observed to have low RMS errors. There was no discernable pattern of "good" versus "poorer" patches. As with humeral elevation, the patches on the scapular spine all showed errors below 5^{o}. For scapular tipping, the scapular spine patches again showed poor results. The best results were again observed for the patches inferior to the lateral aspect of the scapular spine.

The suggested patterns of motion for retraction were different than for humeral elevation (Figure 6-12, Figure 6-13, Figure 6-14). A range of 10^{o} of downward rotation was observed. Internal/external rotation remained nearly constant at approximately -9^{o} even though this is the axis of retraction. Scapular tipping showed the greatest range of

rotation. At 45° of retraction, the inferior angle was tipped anteriorly by roughly 25°. At 90° or retraction, the tipping had reduced to approximately 5°.

At a retraction angle of 45°, one skin marker from the medial boarder was obstructed. The loss of the marker significantly reduced the accuracy of the two patches associated with the marker. The rotations for the two affected patches was calculated with the remaining three markers (Figure 6-15).

Inspection of the rigid body fitting errors showed no relationship between the rigidity of the patch and its accuracy in measuring scapular motion.

Patch 38, lying inferior to the lateral scapular spine, showed the best overall results for all rotations. For humeral elevation, the RMS errors for rotations about the z, y, and x axes were 7.6°, 3.3 °, and 3.3 ° respectively. For retraction, the errors for rotations about the z, y, and x axes were 5.1 °, 3.7 °, and 3.5 ° respectively. Note that the error, E_{Rz} = 7.6 ° was primarily contributed by the discrepancy at maximum elevation.

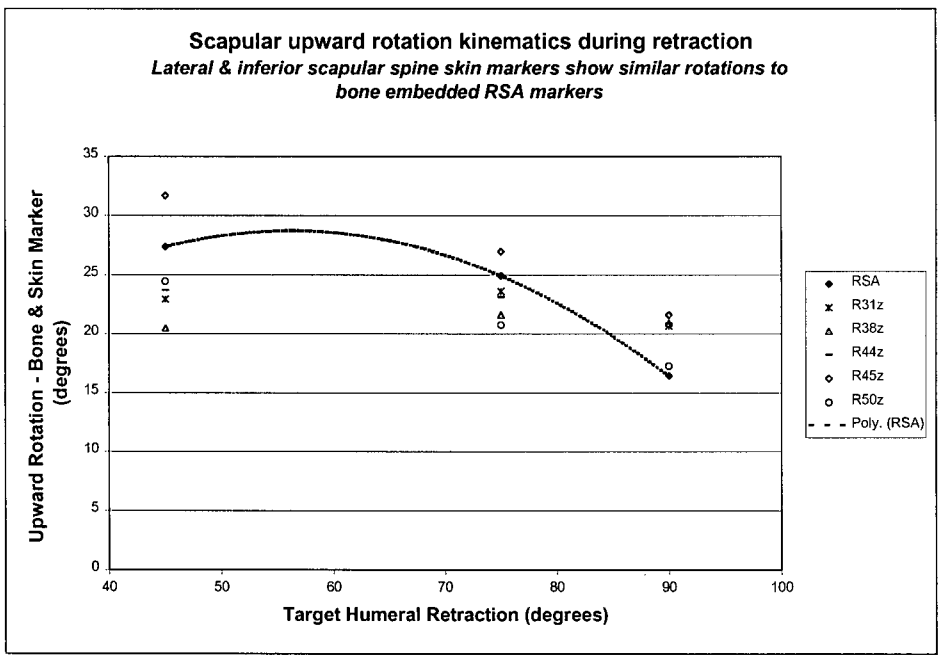

Figure 6-12: Scapular upward rotation – patches near scapular spine
Rotations measured with skin markers on lateral scapular spine and lateral-inferior to scapular spine

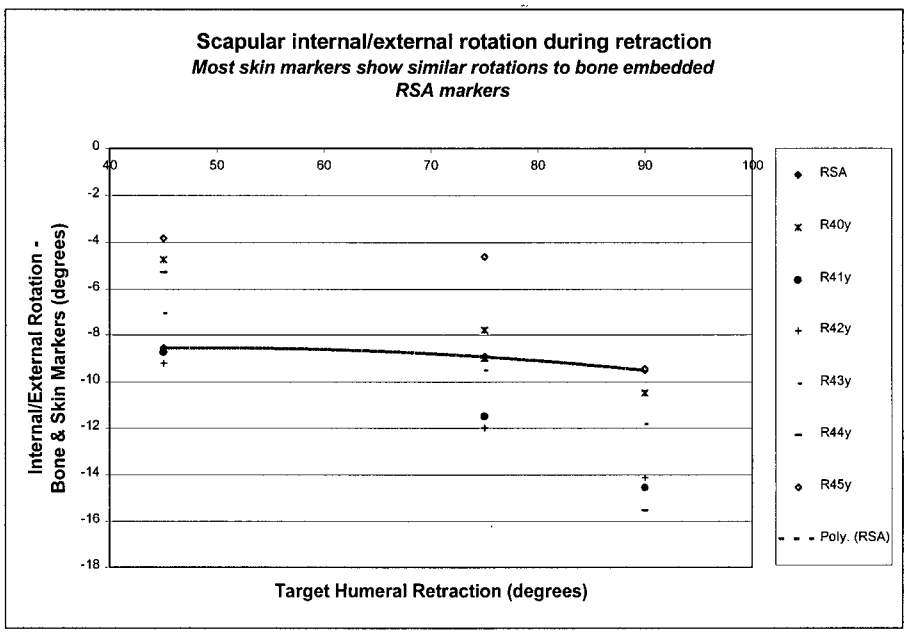

Figure 6-13: Scapular internal/external rotation – patches on scapular spine
Most skin patches showed similar results with RSA. As with humeral elevation, the patches on the scapular spine showed good results.

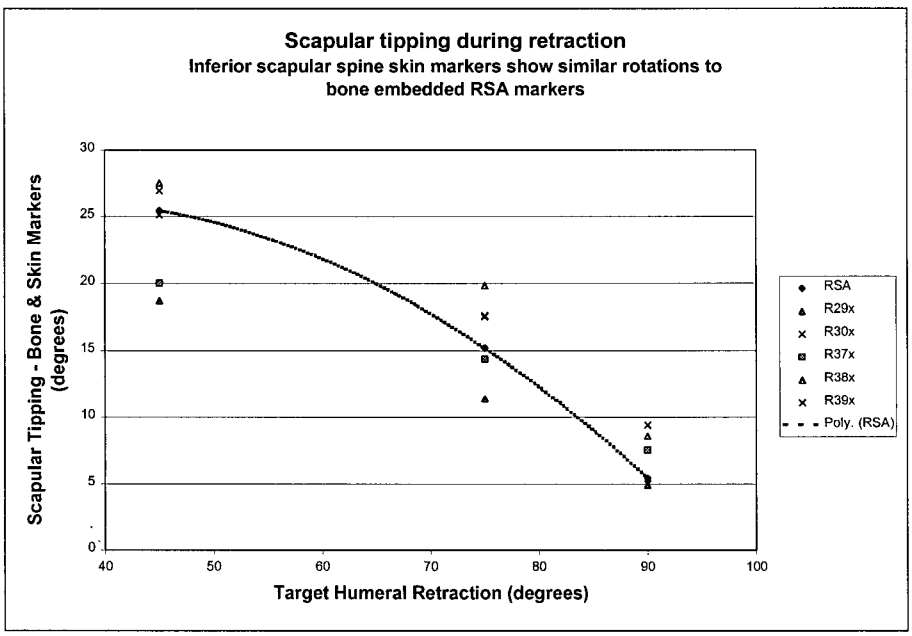

Figure 6-14: Scapular tipping - inferior scapular spine patches
Patches 29, 30, 37, 38 and 39 lay in the region just inferior to the scapular spine. These patches showed the most similar rotations about the x-axis (axis along the scapular spine). The patches directly on the scapular spine showed poorer results for rotation about this (scapular spine) axis. This intuitively made sense since the scapular spine patches were measuring the rotation of a cylindrical-like structure beneath the skin.

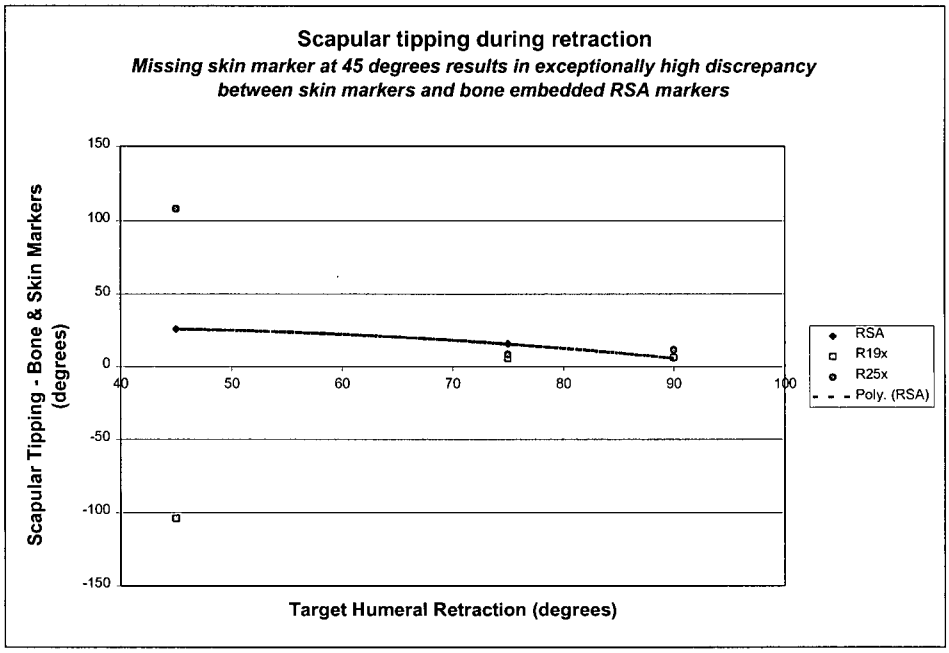

Figure 6-15: Effect of missing skin marker
A missing skin marker affecting patch 19 and 25 resulted in a significant discrepancy between the rotation calculated with RSA bone markers and the rotation calculated for each patch with their remaining three markers.

6.4 Discussion

There are a number of issues to resolve before advancing this project to in vivo testing. The loss of two implanted markers, one in the acromion and one in the proximal humerus, must be addressed. A combination of implantation practice and instrument refinement is necessary. An increase in the hardness of the metal used in the instrument will assist in the puncture of the cortical bone and ensure the instrument is not damage if it is tested in a region of thick cortical bone.

In addition, the excessive number of trial exposures is also a concern. A key difficulty was orienting the stereo setup such that the cage and specimen were contained in the field of view over the entire range of motion. Reducing the size of the region of interest may alleviate this problem. This can be achieved by using optoelectronic markers to track the skin surface, and using RSA strictly in the region of the glenohumeral joint.

The difficulties encountered with control of humeral orientation should not appear during in vivo clinical testing since the subject's muscles will be actively controlling scapular and humeral motion. In this study, glenohumeral orientation was a secondary issue to whether the skin patches were representative of the scapula's position.

Finally, the images in Figure 6-7 demonstrate that if larger RSA markers are used with soft-tissue, it is important that the size be kept to a minimum. Poor image quality has been noted as one of the principal factors affecting overall RSA performance.[33,44,68] In the 1991 study by Hogfors' group, anywhere from 40% to 80% of their RSA data was lost due to the inability to identify markers on the radiographs.[33]

The relationship between the skin patches and scapular upward rotation was more definitive than that found during the pilot study. In the pilot study, the general region of the scapular spine was shown to have results similar to that found in the literature. The region near the acromion was previously found to over estimate rotations because it lay along the line of action of the humerus. In this study, the most lateral region of the scapular spine showed the best results. This region is qualitatively perceived to be covered by less soft tissue. Skin slippage is intuitively a greater issue in the plane of upward rotation. Hence these results agree with the notion that the markers should be placed in the region with the least soft tissue. The issue of measuring upward rotation appears unresolved beyond 135^{o} of elevation where more significant discrepancies were found between the patches and the RSA markers. Karduna's group also recently noted this problem.[42]

No authors in the literature have addressed the issue of skin markers in tracking scapular internal/external rotation. This study showed that the ideal region for measuring upward rotation was not necessarily the same as for internal/external rotation. In this case, the entire length of the scapular spine, perhaps excluding the most medial portion, was a good predictor. The internal/external rotation curves indicate a pattern of motion that was more complex than upward rotation or tipping.

Similar results were observed during the pilot study. However, in the pilot study, the scapula was observed to internally rotate whereas here external rotation was observed. This may be partially due to variations in the definition of the scapular plane since in both studies, the scapula's direction of rotation about the y-axis was observed to fluctuate. Alternatively, it may represent an actual difference in the pattern of motion between strict abduction in the coronal plane (pilot study), and the mixed coronal/scapular plane elevation performed in this study.

It was an interesting result that the scapular spine patches poorly tracked scapular tipping. No authors have ever suggested skin markers be placed anywhere but on the scapular spine and acromion. The finding that the region inferior to the scapular spine was the optimal skin location to measure scapular tipping made sense. The curved surface of the scapular spine introduces orientation ambiguity. As the scapula rotates about the axis of the scapular spine, skin slippage along with this ambiguity increases skin patch measurement error. The patches just inferior to the scapular spine do not have the same geometric ambiguity and hence produce more accurate measurements (Figure 6-16).

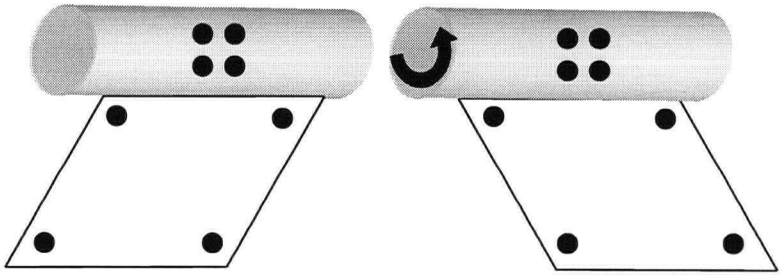

Figure 6-16: Scapular tipping
In this simplified diagram, the scapular spine is modeled as a cylinder with the scapular body attached inferiorly. Rotation about the axis of the scapular spine is difficult to measure with skin surface markers due to skin slippage and the ambiguity in the axial orientation of the cylinder. The makers inferior to the scapular spine do not suffer from the same ambiguity resulting in more accurate rotation data.

The patches, which showed good results for humeral elevation, also showed good results during retraction. This demonstrates that skin patches have some degree of robustness for tracking the scapula during different motions. The repeatability of the analysis for different specimens is still to be determined.

The translation data was not analysed in this study. The primary reason was the lack of clear significance for the translation data. The standard coordinates recommended by the international shoulder group[vii] (Figure 6-17) were used for the scapula in this study and allows for intuitive interpretation of rotation data. However, the significance of the translation data was not readily interpreted clinically. The shoulder group's choice of coordinates system has been based on easily palpable structures such as the medial angle and the acromion.

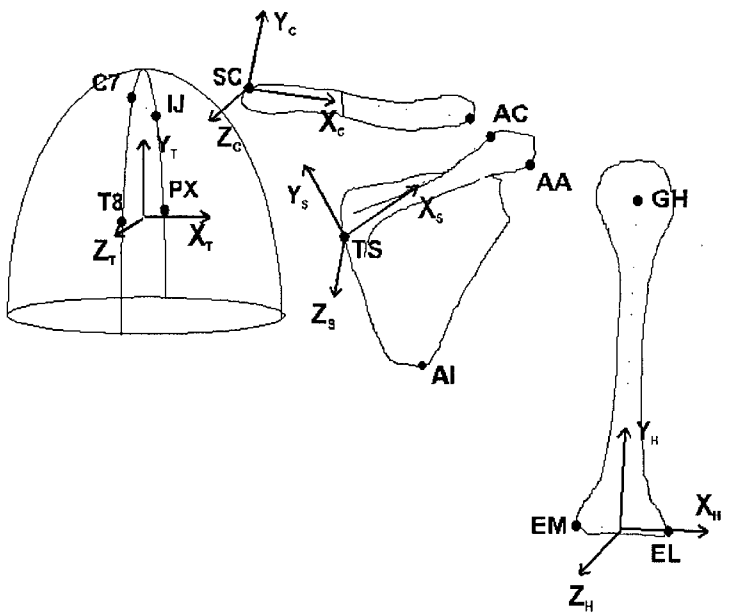

Figure 6-17: Shoulder coordinates of the international shoulder group

[vii] http://www.fbw.vu.nl/research/Lijn_A4/shoulder/isg/proposal/protocol.html

A more useful coordinate system would place the scapular origin at the centre of the glenoid fossa with the y-z plane defined parallel to the glenoid. The humerus origin could be set at the centre of the hemispherical humeral head in an orientation defined by the anatomical neck of the humerus. These structures are not easily palpable, however, they are features that may be identified stereo photogrammetricaly from radiographs.

Using this new coordinate convention, the superior translation of the humerus with respect to the scapula has the immediate interpretation of superior migration of the humeral head within the glenoid socket.

The skin thickness measured over the region of the scapula was constant. Hence this quantity may not be as useful a predictor of patch accuracy as intuitively thought. In addition, the patch rigidity was also found to be a poor predictor of patch accuracy. This was a somewhat non-intuitive result but is a promising sign that the method may perhaps be robust enough to measure motion on specimens with more substantial soft tissue.

6.5 Conclusions

Several experimental issues, such as reducing the field of view and change of coordinate conventions, were identified. The results continue to show promise for the use of skin marker patches. A patch of skin inferior to the lateral scapular spine was found to best track three-dimensional scapular rotations. In general, it can be concluded that scapular rotations about different axes can be inferred from its effect on different regions of skin. Upward rotation was best measured on the lateral aspect of the scapular spine. Markers along the axis of the scapular spine best measured internal/external rotation. Markers in the region inferior to the lateral scapular spine best measured scapular tipping. Contrary to conventional rigid body motion analysis methods, the results support the notion that rigid body kinematics can indeed be inferred from non-rigid measurements.

Chapter 7

SLIP ESTIMATION USING FOURIER ANALYSIS

7.1 Background

The use of Fourier analysis to estimate planar translations has been known for some time.[8,48,74] More recently, methods have been proposed to also measure planar and three-dimensional rotations using Fourier analysis.

A complete discussion of Fourier analysis is beyond the scope of this thesis. Sources may be found in the references.[56,67,85] In brief, Fourier's theorem states that any arbitrary modulation can be decomposed into the superposition of an infinite number of sinusoidal modulations of infinite extent. Figure 7-1 shows how a sinusoid can be plotted versus time, versus frequency, or as a polar plot. In this way, one-dimensional Fourier analysis is a means to move between time and time frequency domains. Similarly, two-dimensional Fourier analysis relates space and spatial frequency. (A chessboard is an example where spatial periodicity is apparent in two directions.) The two-dimensional continuous Fourier transform (FT) and discrete Fourier transform (DFT) between space and spatial frequency are:

$$X(f_u, f_v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(u,v)e^{-j2\pi(f_u u + f_v v)}\,du\,dv$$

$$X(k_u, k_v) = \frac{1}{N^2}\sum_{n_u=0}^{N-1}\sum_{n_v=0}^{N-1} x(n_u, n_v)e^{-j2\pi\frac{(k_u n_u + k_v n_v)}{N}}, \text{ for } u = 0,1,..N-1, v = 0,1,...N-1$$

f denotes continous frequency, u & v denotes continous spatial position

k denotes discrete frequency, n_u & n_v denotes discrete spatial position (i.e. sampled values)

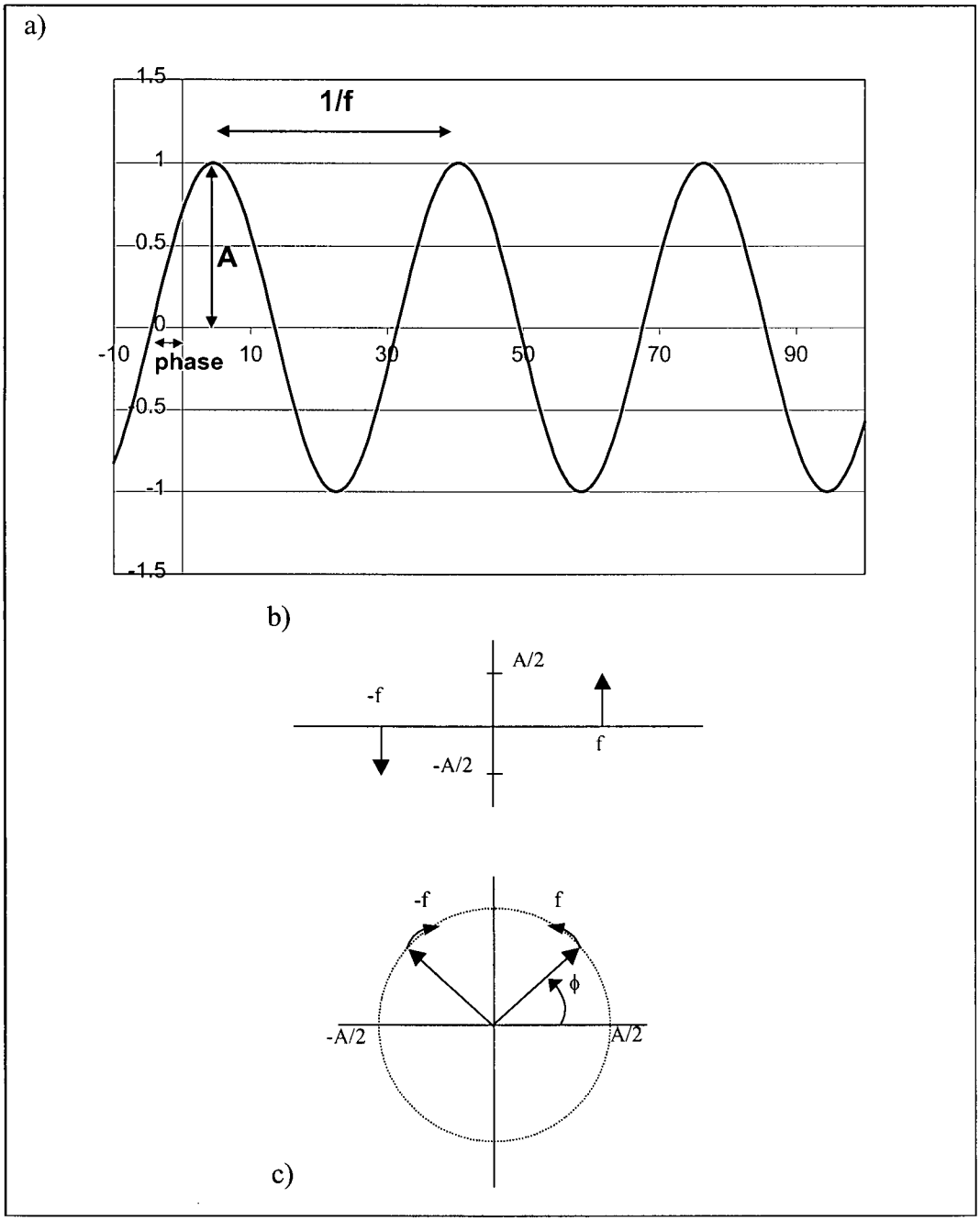

Figure 7-1: Representation of a signal

a) Time domain representation of a typical sinusoid of frequency f, amplitude A and phase angle ϕ. b) Alternate representation in the frequency domain. This "magnitude plot" shows frequency f and amplitude A/2 of the sinusoid, but does not show phase angle ϕ. c) Polar diagram of sinusoid showing magnitude A/2 and phase angle ϕ. Frequency of revolution is implied.

As will be shown, there are properties of working in the frequency domain that can be manipulated to hypothetically determine the amount of scapular slippage beneath the skin. In a series of papers by Lucchese and his colleagues, it was shown that rotation and translation could be decoupled in the spatial frequency domain.[47-52] Rotation affects the magnitude plot of the spatial frequencies. Translation affects the phase (offset) of these frequencies.

Summarizing from Lucchese's group, let $L_1(\mathbf{k})$ denote the DFT of an image. Let $L_2(\mathbf{k})$ denote the DFT of the same image rotated by the rotation matrix, $\mathbf{R}(\theta)$ and translated by the vector $\mathbf{t}$. They are related by,

$$L_2(\mathbf{k}) = L_1(\mathbf{R}(\theta)^{-1}k)e^{-j2\pi k^T \mathbf{R}(\theta)\mathbf{t}}$$

where,

$$\mathbf{k} = \begin{bmatrix} k_u \\ k_v \end{bmatrix}, \ \mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \in SO(2), \mathbf{t} = \begin{bmatrix} t_u \\ t_v \end{bmatrix}$$

and their magnitudes are related by,

$$\left|L_2(\mathbf{k})\right| = \left|L_1(\mathbf{R}(\theta)^{-1}\mathbf{k})\right|$$

This shows the decoupling of the rotation from the translation. The rotation only affects the magnitude of the Fourier spectrum.

The method of determining translations is referred to as phase correlation. A delay in time for the 1-D case, or a translation in space for the 2-D case, is equivalent to an offset in the angle of the oscillator. This angle is referred to as phase. Translation was not assessed in the cadaveric study and hence the details of phase correlation are left to the references.

It is a property of the 2D-DFT that the first and third quadrants are reflections of each other as are the second and fourth. This Hermitian symmetry can be used to measure the rotation between the images. To identify the rotation θ, define:

$$\Delta(\mathbf{k}) = \frac{|L_1(\mathbf{k})|^2}{L_1^2(\mathbf{0})} - \frac{|L_2(\mathbf{Hk})|^2}{L_2^2(\mathbf{0})}$$
$$= \frac{|L_1(\mathbf{k})|^2}{L_1^2(\mathbf{0})} - \frac{|L_1(\mathbf{R}(\theta)^{-1}\mathbf{Hk})|^2}{L_1^2(\mathbf{0})}$$

where, $\mathbf{H} = \begin{bmatrix} \pm 1 & 0 \\ 0 & \mp 1 \end{bmatrix}$ is used to reflect the spectrum about either of the frequency axes.

It was proven by Lucchese's group that the set of points, locus, where $\Delta(\mathbf{k}) = 0$ includes two perpendicular lines which are rotated from the principal axes by θ/2. Hence, the problem of determining the rotation was solved by calculating the angle of these perpendicular lines from the axes. This was accomplished by defining a threshold radial distance from the origin, ρ, and determining the arctangent of each zero point in this region of interest. A histogram of the angles revealed a sharp peak corresponding to the high density of points representing the two perpendicular lines.

The method of determining the rotation is best demonstrated by an example as shown in the following sequence of figures.

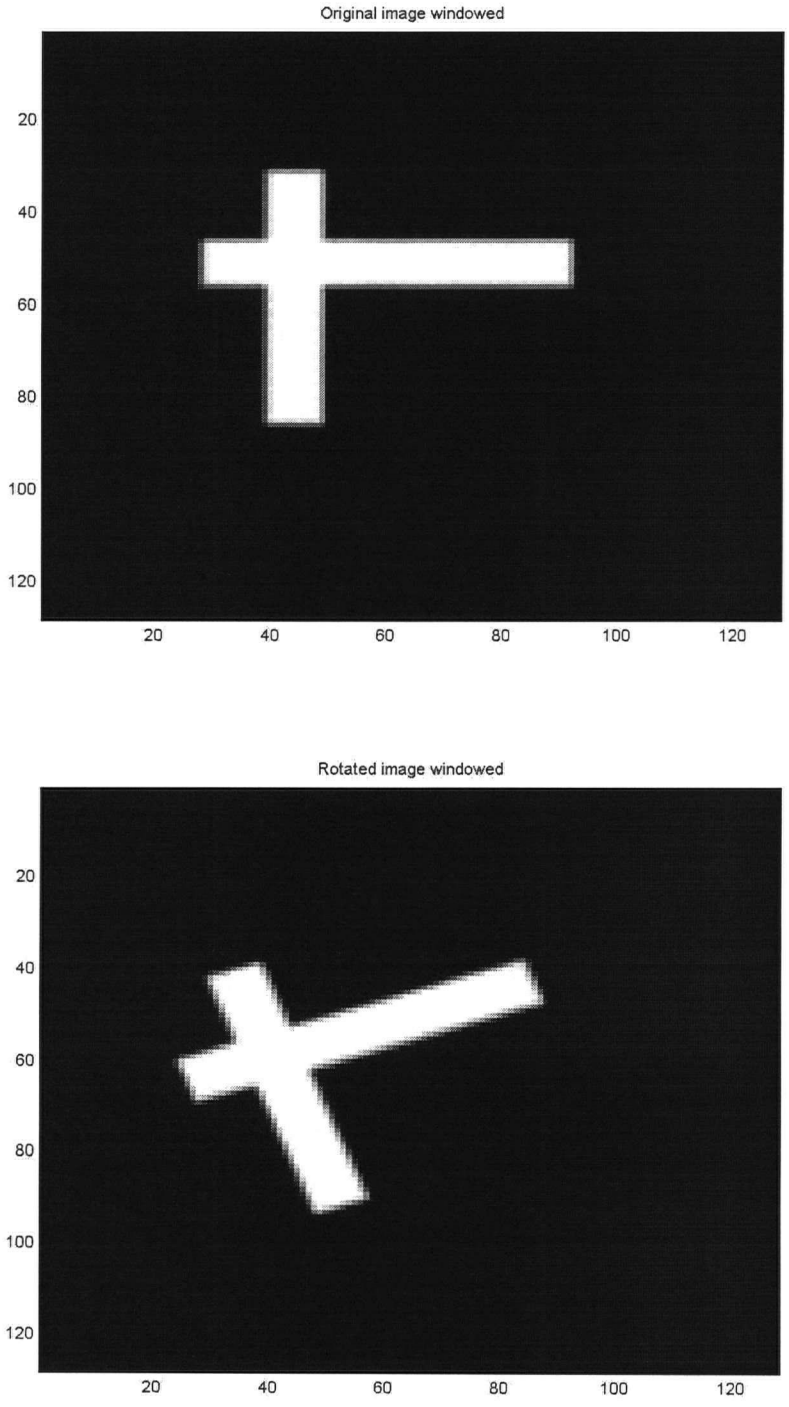

Figure 7-2: Two images with relative rotation
Top: The image used as a reference. Bottom: The same image rotated 20°. Hamming window was used to improve performance of discrete Fourier transform.

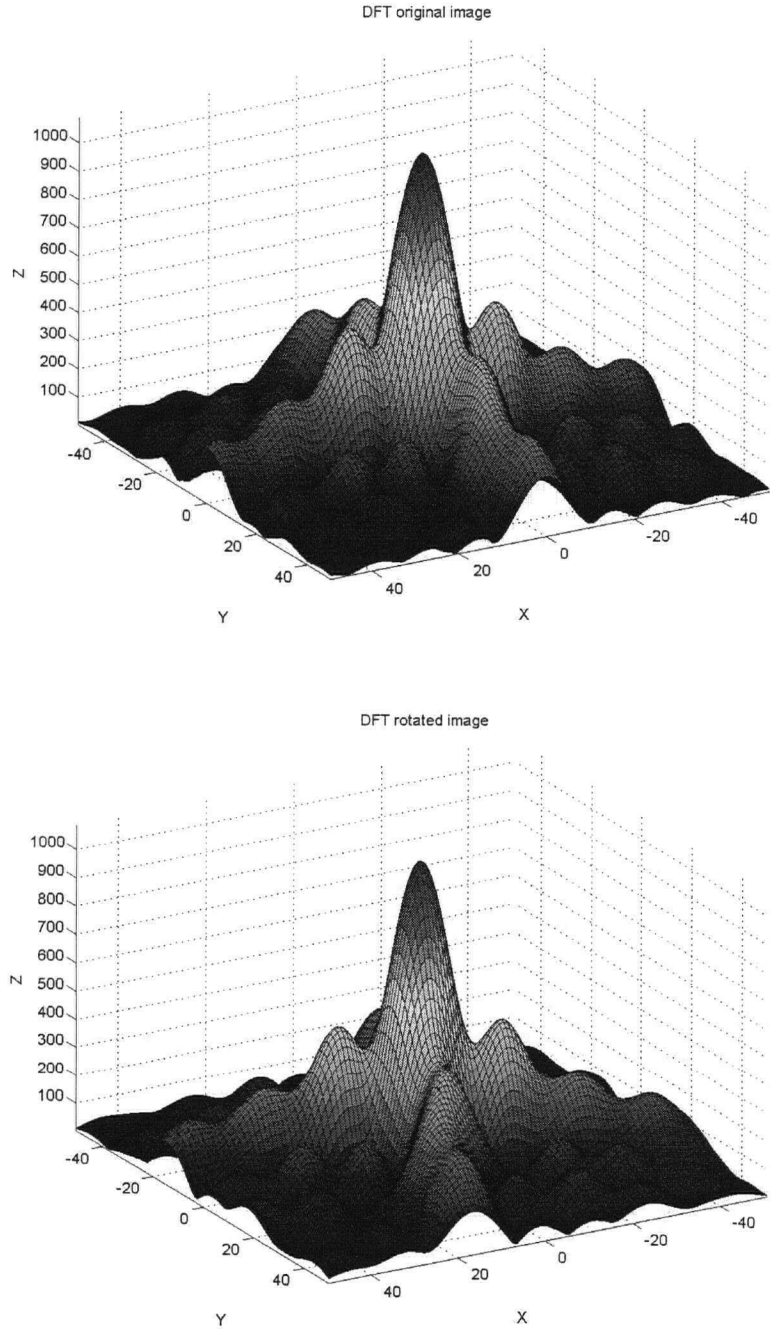

Figure 7-3: 3-D Magnitude of Fourier spectra
Top: Fourier spectrum of original image, $L_1(\mathbf{k})$. Bottom: Fourier spectrum of rotated image, $L_2(\mathbf{k})$. The spectra are the same but rotated.

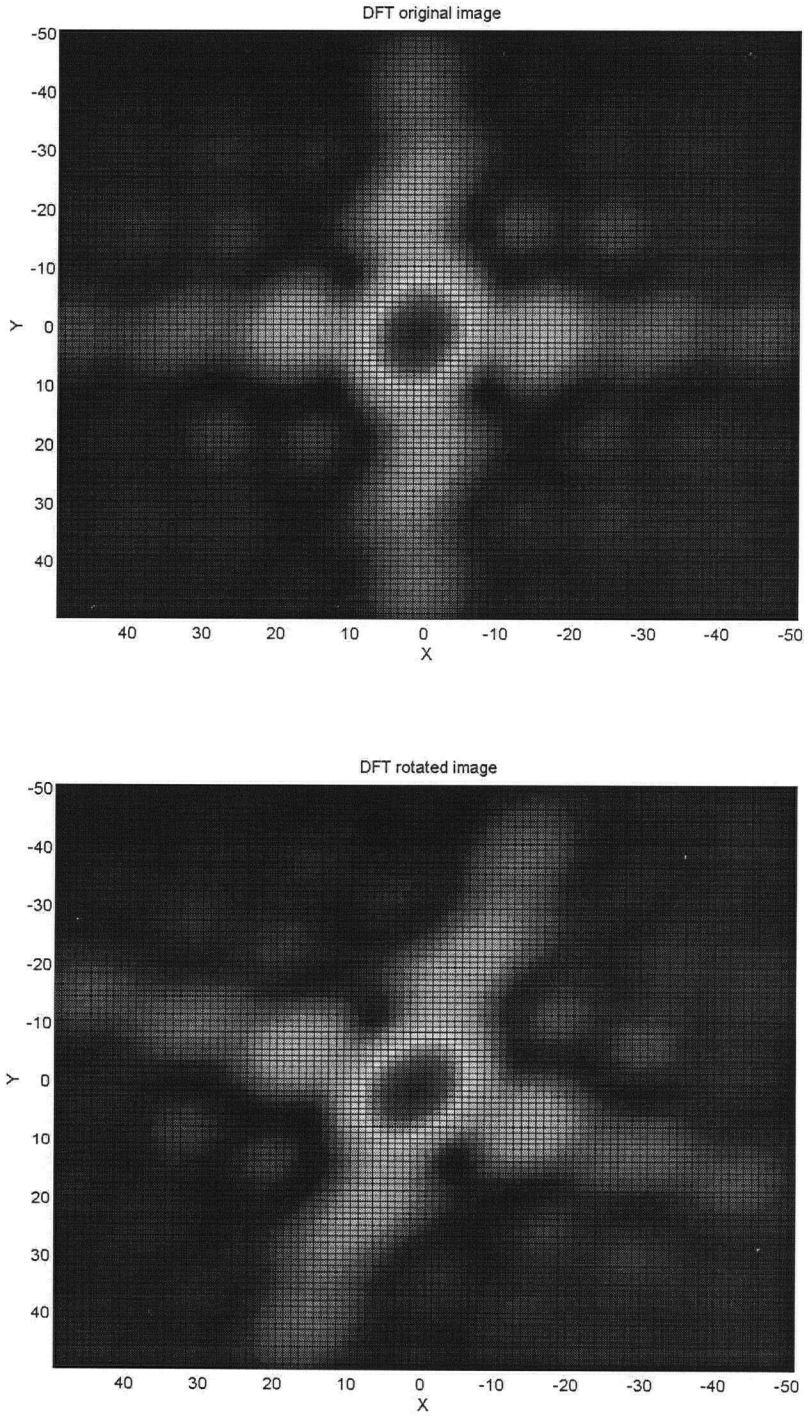

Figure 7-4: 2-D Magnitude of Fourier spectra
Top: Fourier spectrum of original image, $L_1(\mathbf{k})$. Bottom: Fourier spectrum of rotated image, $L_2(\mathbf{k})$. The spectra are the same but rotated.

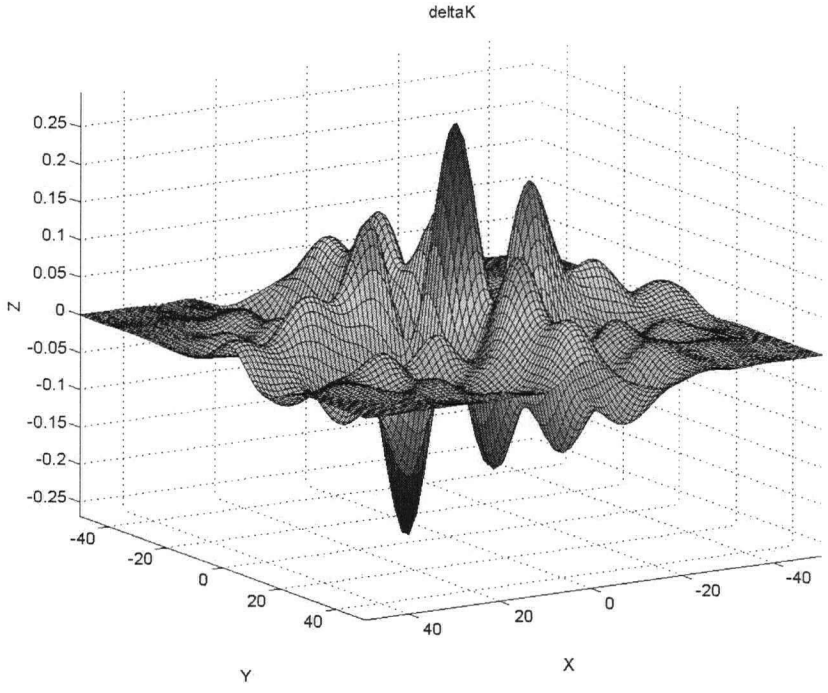

Figure 7-5: Δ(k) function

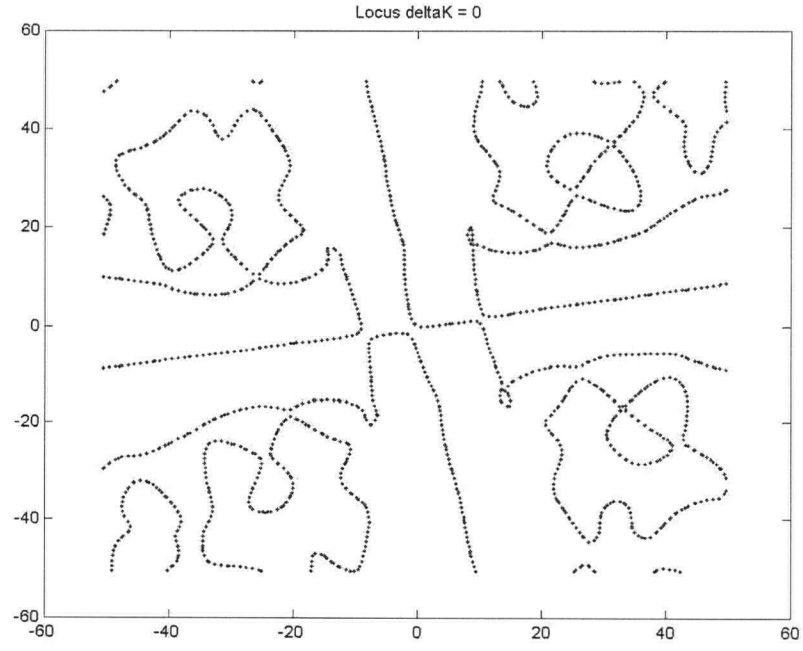

Figure 7-6: Locus where Δ(k) = 0

Perpendicular lines passing through origin describe the relative rotation between the images. The angle is equal to half the actual rotation. The other zero points are located where $L_1(\mathbf{k})$ and $L_2(\mathbf{k})$ equal zero.

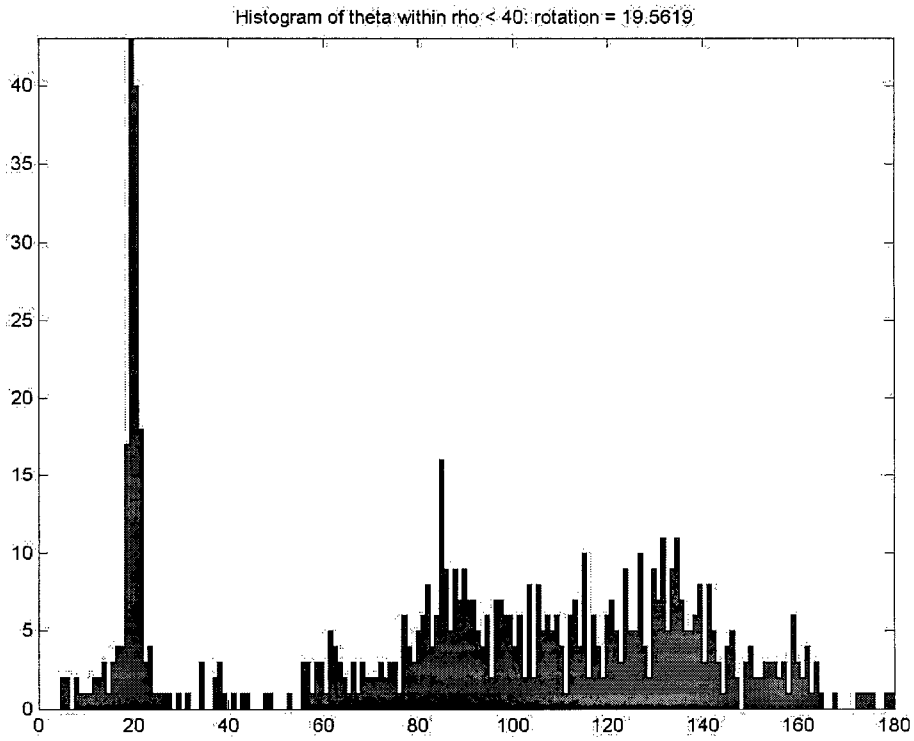

Figure 7-7: Histogram of locus $\Delta(k) = 0$
The angle of the perpendicular lines relative to the principal axes is found to be 19.56° compared with the actual rotation of 20°. Discrepancy is due to the finite frequency resolution of the discrete Fourier transform. Rho (ρ) denotes the distance from the centre of the locus for which points were considered.

During the pilot and pre-clinical studies, the skin markers were directly used for kinematic analysis. The accuracy of skin markers in measuring scapular kinematics was reduced by relative motion between the skin surface and the scapula underneath. An alternate solution may be to use the skin markers as a means of sampling a surface, under which the scapula slips. The scapula's position should be detectable through its prominent features such as the scapular spine and medial margin. The Fourier methods described in this chapter may then be used to analyse these surface features instead of the skin markers themselves. It is on this premise that this preliminary study proceeds.

7.2 Objective

The purpose of this study was to investigate the feasibility of using two-dimensional Fourier analysis to measure the rotation of scapular features beneath the skin surface and hence estimate the magnitude of scapular upward rotation.

7.3 Methods

The data from the cadaveric study was used. The 68 points from the skin surface were mapped to the scapular plane. Cubic interpolation was used to fit a 128×128 surface to the skin marker coordinates. To clarify, once the surface was fitted, the 68 skin marker points were no longer used in the subsequent analysis. The resultant surface image was padded to 700×700 and the two-dimensional DFT was taken. The padded image size was restricted by hardware limitations. The analysis was conducted on PC equipped with PentiumII-450MHz processor with 128MB of RAM. The Fourier methods described in the background were used to estimate the upward rotation of surface features.

Ideally, in order to assess upward rotation, the overall boundaries of the skin surface should remain stationary while the surface fluctuates in response to the scapular motion beneath (Figure 7-8). The surface data from the cadaver study did not cover a region large enough to completely satisfy this idealization (Figure 7-9). Attempts were made to compensate for the boundary edges using windowing techniques that gradually brought the skin boundary to zero. Windowing has the effect of improving the performance of the DFT by reducing a phenomenon called spectral leakage. However, due to the odd geometry of the skin surface boundaries, no consistent window was found that evenly brought all skin edges to zero without truncating a large portion of the surface.

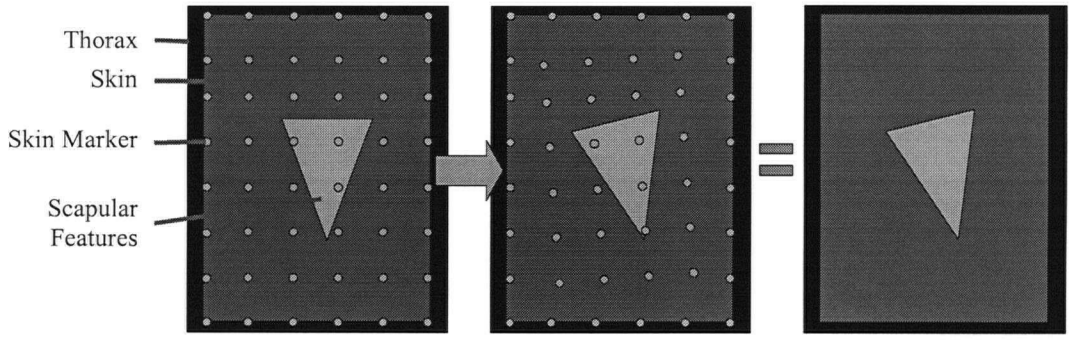

Figure 7-8: Ideal Skin Boundary
Ideally, skin markers would extend to a stationary region such as the skin over the vertebral column. Skin markers are used to sample the surface. Fourier analysis would be conducted on the interpolated surface – not on the skin marker data itself. Analysis of the images detects the slippage in upward rotation of the scapular features.

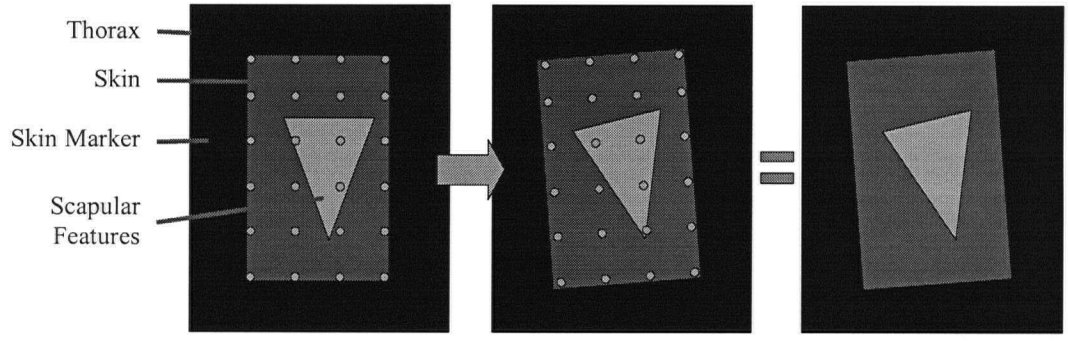

Figure 7-9: Actual Skin Boundary
In reality, the skin markers did not extend to a stationary region. Consequently, analysis of the images detects the combined effect of feature slippage rotation and skin boundary rotation.

To reduce the influence of the fluctuating boundaries, the analysis was conducted between successive images (elevation 90° and 135°, elevation 135° and 180°) instead of using the initial image as a constant reference.

Without any other currently viable alternatives to smooth the skin boundary, this aspect was neglected for this initial trail and the results would be assessed accordingly.

As implemented, the methods of Lucchese's group cannot distinguish between positive and negative rotations. Instead, rotations on the locus of $\Delta(\mathbf{k}) = 0$ that were found to be, say $\theta/2 = 88^{\circ}$ ($\theta = 176^{\circ}$), were interpreted as $\theta/2 = -2^{\circ}$ ($\theta = -4^{\circ}$). This was considered justifiable based on physical knowledge of the system under investigation.

7.4 Results

Examples of the skin surface after interpolation are shown in Figure 7-10. The fluctuating skin boundary was visually apparent due to the large elevation intervals (~45°) between exposures. The lack of pure transverse rotation and translation was also readily observable. This deviation from the ideal case (example in the background section) altered the appearance of the perpendicular lines in the loci of $\Delta(\mathbf{k}) = 0$. The perpendicular lines were limited to a smaller region near the origin (Figure 7-11). In addition, the vertical axis was found to be slightly offset from centre. The two effects forced the histogram analysis (Figure 7-12) to be limited to a relatively tight region near the origin. In this case, ρ was less than 15 or 10 sample units as opposed to 50 sample units used in the example for the background section.

The summary results from the histograms are shown in Table 7-1. For each exposure, a range of slip angles is reported. The range, as opposed to an exact value, was a consequence of the slight curving of the perpendicular lines even near the origin. The slip angle at 45° of elevation was found to range from –69° to –55°. (Negative slip indicates downward rotation.) The remaining slip angles ranged between –10° to 8°.

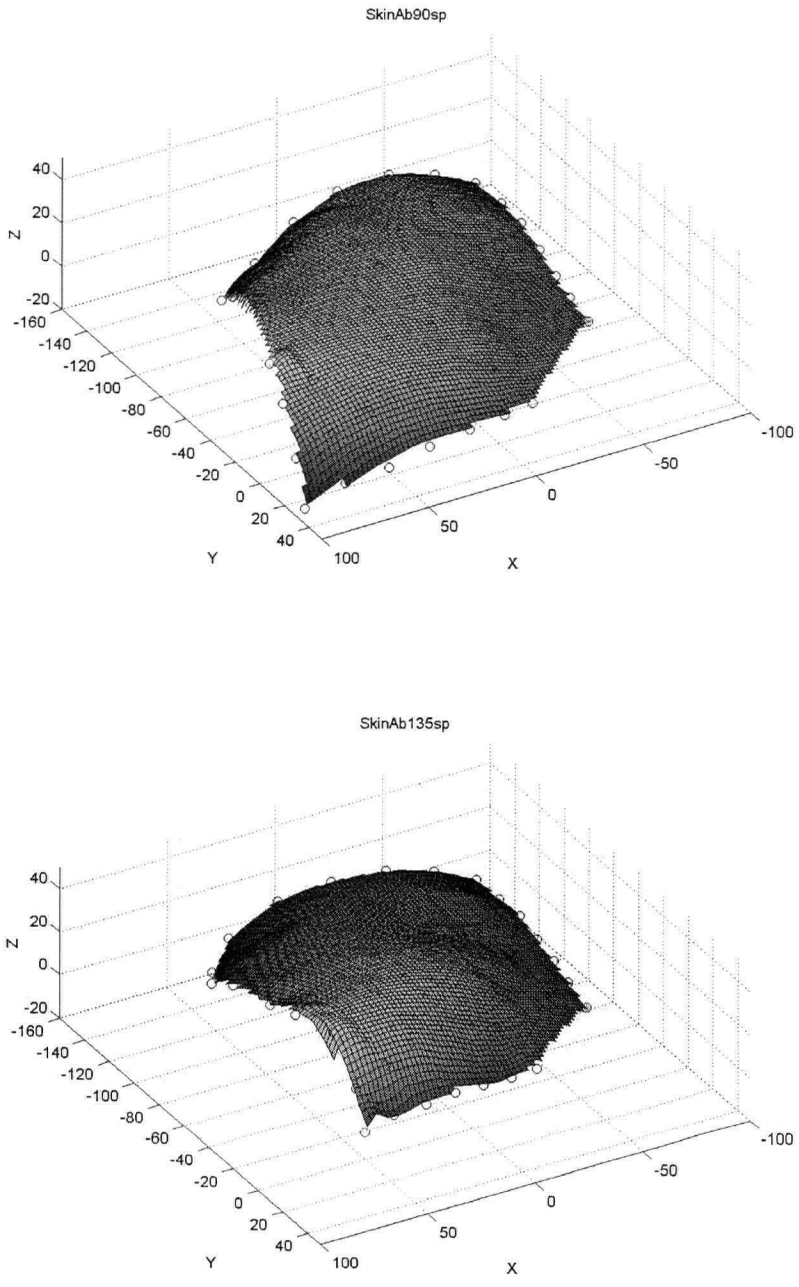

Figure 7-10: Interpolated Skin Surfaces

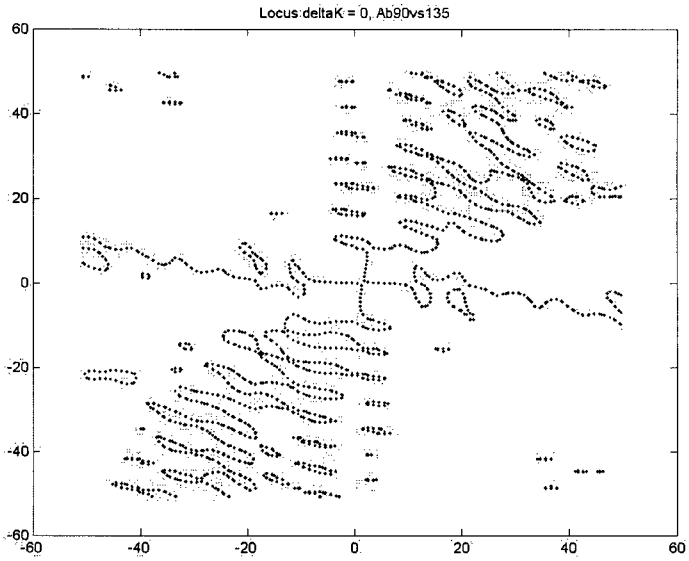

Figure 7-11: Locus of $\Delta(k) = 0$ between 45^o & 90^o of elevation
The effect of surface variation, aside from transverse rotation and translation, distorts the locus at higher spatial frequencies.

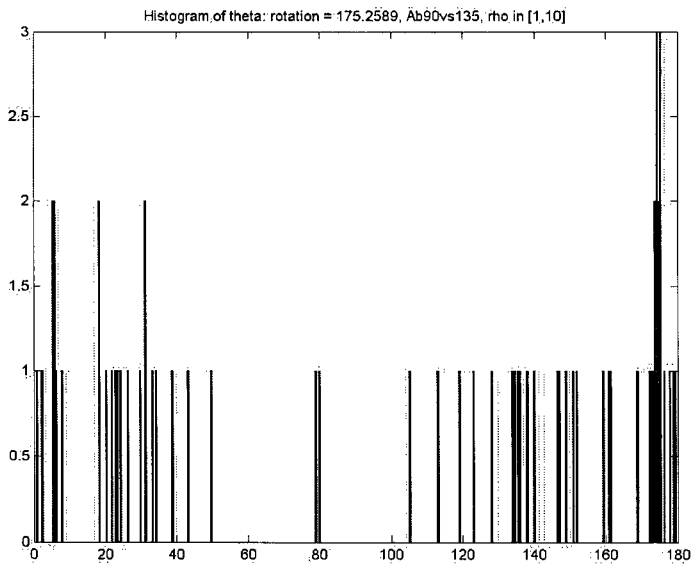

Figure 7-12: Histogram of $\Delta(k) = 0$ between 45^o & 90^o of elevation
The small number of samples was due to the reduction in the region of interest around the origin. Further from the origin, the perpendicular lines curved. The effect was due to surface variation not resulting from transverse rotation and translation.

Target Humeral Angle	RSA Incremental Scapular Upward Rotation (°)	Incremental Scapular Feature Slip Range (°)
Elevation		
45°	22.1	[-69.2, -54.8]
90°	-4.8	[4.6, 8]
135°	12.1	[-5.7, -4.8]
180°	14.9	[5.1, 6.5]
Retraction		
45°	-	n/a
75°	-2.4	[-10.0, -9.5]
90°	-8.5	[-6.9, -5.4]

Table 7-1: Scapular Feature Upward Rotation
Scapular feature slippage in upward rotation is compared to the scapular upward rotation angles found using RSA.

7.5 Discussion

Distortions in the spectrum are evident by the curling of the points on the various loci of $\Delta(\mathbf{k}) = 0$. This phenomenon is similar to the distortion introduced by noise in the image and has been discussed by Lucchese and Cortelazzo.[48]

The slippage of the scapular features beneath the skin surface was inconsistent with the gold standard values found using RSA. The effect of the skin boundary is hypothesized to be the source of the discrepancy. To resolve this, the skin boundary must be removed. One possibility would be to alter the method of measuring the skin surface. Currently, the skin surface was measured in a Lagrangian sense where fixed points on the skin surface were measured throughout the motion. This resulted in a moving boundary that has now become problematic. To avoid this, the surface could be measured in an Eulerian sense where the measurement intervals are fixed irrespective of the skin's position (Figure 7-13). Alternatively, the Lagrangian markers could be extended to a region where little skin motion is present (e.g. the skin posterior to the spinal column).

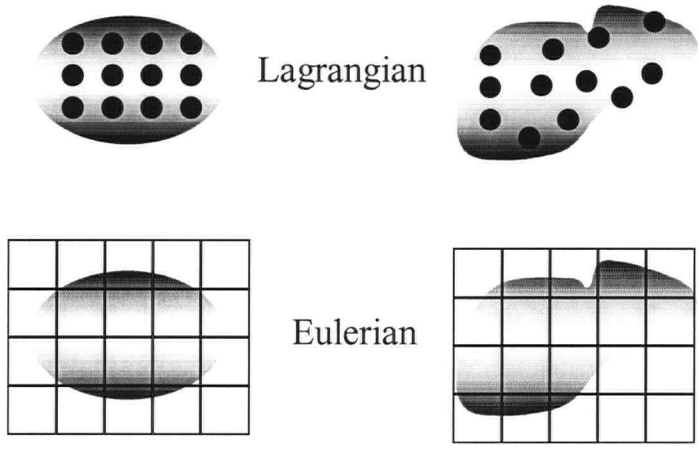

Figure 7-13: Lagrangian versus Eulerian descriptions
Skin position was measured in a Lagrangian sense. This rendered the analysis sensitive to varying skin boundary effects. By using an Eulerian method of measuring skin motion, it may be easier to ensure a consistent boundary condition. Alternatively, a wider region of Lagrangian markers may be used to ensure a perimeter that is not affected by skin motion.

The "slippage" measured was not purely feature slippage in upward rotation, but instead, some type of combined effect of slip and changing skin boundary. The perpendicular lines appeared only in the region close to the origin of $\Delta(\mathbf{k}) = 0$. Was there sufficient information contained in this region to determine the rotation of the scapula? The question was difficult to resolve completely with only one data set. The results were not highly conclusive but the methods used require refinement. The magnitude of scapular slippage appears reasonable and suggests further investigation may be fruitful.

7.6 Conclusions

A preliminary study into the use of Fourier analysis for measuring scapular feature slippage has shown inconclusive results. A key problem was the presence of a varying skin boundary that remains unresolved. The results showed feature slippage on the order of the upward rotation angles found using RSA. The analysis thus far has not been in-depth and this methodology warrants further investigation before more conclusive statements can be made.

Chapter 8

SUMMARY, FUTURE WORK & RECOMMENDATIONS

8.1 Summary

This thesis began with a discussion of the overall clinical motivation for studying and measuring scapular motion. The glenohumeral joint is the focal point of a great deal of shoulder pathology. Understanding the kinematics at this interface requires the three-dimensional tracking of both the humerus and the scapula. Scapular kinematics has been the subject of several papers over the last fifty years. The methods have evolved from two-dimensional x-rays to current attempts at non-invasive three-dimensional techniques. A clinically practical method of measuring scapular motion is yet to be established.

This thesis has focused primarily on the technical challenges involved in developing methods to measure scapular kinematics. A roentgen stereophotogrammetric analysis system was used as a gold standard to assess the validity of a novel non-invasive skin marker approach for scapular tracking.

The skin marker approach involved tracking a grid of surface markers placed on the skin over the scapular region. The initial feasibility of this method was demonstrated in a pilot study. Data from skin patches located centrally and on the scapular spine showed comparable results with that reported in the literature.

To quantify the accuracy of the skin marker method, an RSA system was set up. A number of questions were raised during this process. In order to answer them, a simulation environment was developed in Matlab. A flat projective cage was compared

to a biplanar cage in the shape of a cube. The original RSA analysis method developed by Selvik was compared with the DLT analysis method commonly used in other areas of stereophotogrammetry. Other parameters affecting accuracy were also assessed. The simulation showed that the most accurate three-dimensional RSA system should use a biplanar cage with DLT analysis.

An experimental investigation of the RSA system's accuracy was undertaken. The digital measurement accuracy was found to be within 1×1 pixel. The accuracy was found to be independent of marker diameter (1, 2, 3, 4, 5-mm tested) provided there was adequate contrast in the image (x-ray transmissibility < 70%). The reconstruction accuracy was found to be 0.43-mm. Using self-calibration, the translation accuracy was found to be 0.25-mm in the transverse direction and 0.23-mm axially. Using initial-calibration, the translation accuracy was found to be 0.86-mm in the transverse direction and 0.54-mm axially. Using initial-calibration, the rotation accuracy ranged from 0.192^{o} about the x-axis to 1.047^{o} about the y-axis. The x-y plane was transverse to the axis of symmetry between the x-ray sources.

The skin marker method was tested in a fresh cadaver. The rotations measured with the skin markers were compared to those measured with bone embedded RSA markers. A number of experimental issues, such as implantation techniques and excessive trial exposures, were identified for refinement and should be addressed before advancing this study to the in vivo stage. A finer grid was used in this study and the overall results were encouraging. A single patch located inferior to the lateral scapular spine was found to best track the scapula in all three-dimensions. More generally, the rotation of the scapula could be determined by considering different patch regions along the spine of the scapula and in the region superior and inferior to the lateral scapular spine. Patch accuracy appeared to be independent of patch deformation and soft tissue thickness.

Through the pilot study and the cadaver study, this thesis has demonstrated the feasibility of using a grid of skin markers to track scapular motion. In addition to

upward rotation, the results have quantitatively shown complex internal/external rotation and scapular tipping. These latter two have not been fully described in the literature and hence constitute a significant deficiency in the overall understanding of scapular motion and how it affects the environment at the glenohumeral joint.

8.2 Future Work & Recommendations

The next stage in this ongoing study is to test the performance of the skin marker grid in vivo. Ethical approval was applied for and obtained to proceed with this next stage. A few revisions are required to the ethics application. Primarily, the geometry of the setup should be modified to allow more space for motion. This will have an impact on the radiation dose calculations. The method for computing radiation dose is described in the appendix on x-ray radiography.

To date, no subjects have been tested in vivo. Difficulties have been encountered recruiting patients. One reason for this may be the lack of a rehabilitation component to the experimental protocol. An amendment to the protocol may involve a concurrent study to assess the effect of a novel rehabilitation regime on glenohumeral orientations. Exactly what are favorable glenohumeral orientations remains to be investigated in the literature. The concurrent study would involve two imaging sessions – one prior to rehabilitation and one after a designated time period of rehabilitation. The number of stereo exposures in a single session could be reduced to maintain a similar overall radiation dose.

There are also a number of technical issues that should be addressed. The methods used in the RSA accuracy study should be tested with the RSA simulator to further demonstrate the validity of the simulation system. At present, the validity of the simulator is based on the assumption that measurement accuracy is the primary source of error outweighing variables such as film distortion and x-ray beam inhomogeneity. The results of Yuan and Ryd using Selvik's analysis methods support the validity of the RSA simulation environment.[100]

Although the RSA GUI environment can be used to perform actual data analysis, its utility could be improved. Additional functionality for circle detection,[89] integrated stereo point matching, integrated point assignment, and integrated kinematics analysis along with other minor revisions would vastly enhance the utility of the current version of RSA GUI.

Finally, although the skin patches have been shown to exhibit rotations similar to that of the scapula, they are not exactly equal. A preliminary investigation was conducted into using two-dimensional Fourier analysis to measure upward rotation slippage of scapular features beneath the skin surface. The method found slippage on the order of the rotations found using RSA. The input data were far from ideal resulting in skin boundary effects. Refinement of the method has been left to future investigations.

A wide variety of surface measurement techniques remain unexplored.[9,99] The fields of range imaging, stereophotogrammetry, and image analysis are rich with tools that may be adapted for accurate close range measurement.

This thesis has hopefully demonstrated some measurement techniques, identified some issues, and presented some novel ideas that will eventually contribute to the long-term development of a clinically practical, non-invasive, accurate method for measuring human scapular kinematics.

Appendix A

X-RAY RADIOGRAPHY

A.1 General X-Ray Setup

Figure A-1 shows the typical setup of an x-ray imaging system. Electrons are accelerated through a potential between the cathode and the tungsten anode. When the electrons strike the anode a variety of interactions may occur such as compton and rayleigh scattering. The interaction of interest is the energy released when the electron interacts with the nucleus of the tungsten atom. As the electron decelerates, energy is released in the form of x-rays. This "breaking" release of energy is referred to as Bramstrulung radiation. The x-ray photons strike the subject where some interact (scatter) and others do not. Ideally, the film should capture only non-interacting x-rays since it is not possible to know the origin of those x-rays that have undergone scattering. The beam passes through a grid to remove as many scattered x-rays as possible before striking the film cassette.

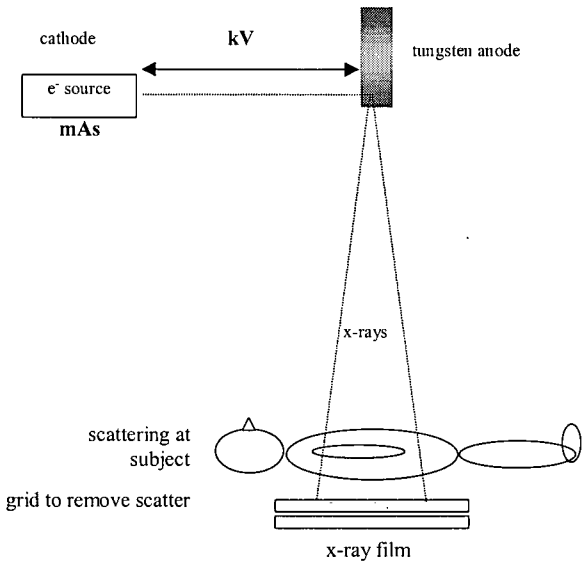

Figure A-1: X-ray Imaging Setup

A.2 Control of Image Quality

A comprehensive description, including extensive examples, of controlling radiographic image quality may be found in *Fuchs's Principles of Radiographic Exposure, Processing and Quality Control.*[14]

A.2.1 Density and Contrast

Density refers to the darkness of the x-ray image, it is measured by taking the logarithm of the opacity of the x-ray film where opacity is defined as the light stopping power of the film. Contrast refers to the difference in density between light and dark regions of an image. An ideal x-ray image has high contrast between the structures of interest, and adequate density for the human eye to observe this contrast.

A.2.2 kV

The kilovoltage, kV, refers to the potential energy change through which the electrons from the cathode are accelerated towards the tungsten anode. The energy of the electron is directly proportional to the kV and subsequently, the maximum energy of the x-ray spectrum is limited by this value. The absorption of x-rays by a material is roughly proportional to the cubed of its atomic number. Different materials have

different capacities to stop x-rays hence kV is used to generate contrast in the x-ray image. The kV should not be used to control density although clearly, a more penetrating beam will produce a denser x-ray image. As a rule of thumb, a 15% increase in kV will increase density by a factor of 2.

A.2.3 mAs

The mAs is the product of the current measured in milliamperes (coulombs per second) and exposure time (seconds) and refers to the total number of electrons (coulombs) striking the tungsten anode. The density of the x-ray image is directly proportional to the mAs. Doubling the number of electrons results in double the number of x-ray photons, which results in double the image darkness.

A.3 Control of Dose

The International Commission of Radiological Protection (ICRP) presents regulations regarding the control of dose. X-ray interaction is a random process and a vast number of studies have been published addressing the issue of determining radiation dose.[2,13,26,31,34,36-38,45,62,77,92,92]

A.3.1 Half-Value Layer (HVL)

The half value layer is a measure of the beam penetration. It is defined as the number of millimeters of aluminum the beam must pass through to reduce the beams intensity by half. If a beam's half value layer is too thin, that is an indication that the beam has poor penetration and consequently, a significant amount of radiation will be absorbed by the subject before the beam exits. The minimum value of HVL at various kV is specified by standard regulations[viii].[84]

A.3.2 Filtration

The purpose of filtration is to remove low energy x-ray photons from the beam. As indicated earlier, the beam contains a spectrum of x-ray energies. Very low energy

[viii] Bureau of Radiological Health name changed in 1982 to Center for Devices and Radiological Health

photons which will be uniformly absorbed by the patient and thus not contribute to the image contrast. Filtration removes these components and thus prevents unnecessary dosage to the patient.

A.3.3 Entrance Skin Exposure

The entrance skin exposure (ESE) was previously used to determine the patient dose. However, the measure is deficient. A given ESE to the hand is clearly not equivalent to an equivalent ESE to the abdomen where there is a greater amount of soft tissue which absorbs significantly more ionizing radiation.

A.3.4 Effective Dose

The effective dose (E) was introduced in the 1990 recommendations of the ICRP. It is a measure of detriment that accounts for the specific organs irradiated and characteristics of the x-ray beam. It is an updated quantity from the effective dose equivalent (E_H) presented in 1977. Whereas before it was mentioned that a given ESE to the hand and abdomen was not the same, a given E to the hand and abdomen may be considered to have the same overall detriment to the patient. Of course, the specific exposure conditions to the hand and abdomen would be different to produce the same E.

A.3.5 Methods of Determining Effective Dose

Several authors have discussed the problem of computing the effective dose to a patient.[37,45,92] Below are the relevant equations used to determine the effective dose.

Variables:

kV – kilovoltage
mAs - milliamperes×seconds
EA – exposure area
t_s – subject thickness
m_s – subject mass
d_s – subject distance
TO – tube output
d_o – distance tube output measured at
HVL – half value layer

Entrance Skin Exposure – [mR] (using inverse square law)

$$ESE = TO \times \left(\frac{d_o}{d_s}\right)^2 \times mAs$$

Exposure Area Product – [mR cm^2]

$$EAP = ESE \times EA$$

Energy Imparted per Unit Area Exposure Product: ω [J/R·cm^2][26]

$$\omega = \alpha \times HVL + \beta$$

where, $\alpha = f(t_s, kV)$ and $\beta = f(t_s, kV)$ found from empirical charts

Energy Imparted – [mJ]][26]

$$\varepsilon = \omega \times EAP$$

Dose Ratio – [mSv/mJ][37]

$Dose\_Ratio \equiv \left(\frac{E}{\varepsilon}\right)_i$ is found from empirical tables for body part i

Effective Dose – [mSv][37]

$$E = \varepsilon \times Dose\_Ratio \times \frac{70.9}{m_s}$$

or combined,

$$E = (\alpha \times HVL + \beta) \times TO \times \left(\frac{d_o}{d_s}\right)^2 \times mAs \times EA \times Dose\_Ratio \times \frac{70.9}{m_s}$$

Example

Parameters:

$kV = 140\text{kV}$
$mAs = 1\text{mAs}$
$EA = 300\text{cm}^2$
$t_s = 10\text{cm}$
$m_s = 45\text{kg}$

$d_s = 40\text{ cm}$
$TO = 10\text{ mR/mAs at 100kV}$
$d_o = 101.6\text{ cm}$
$HVL = 0.27\text{ cm}$

From charts and tables:

$$\alpha = 8.36 \cdot 10^{-6},\ \beta = 5.05 \cdot 10^{-5}$$
$$\omega = 5.276 \cdot 10^{-5}\ \text{J/R} \cdot \text{cm}^2$$
$$Dose\_Ratio = 8.6\text{ mSv/J for left shoulder}$$

Result:

$$E = 0.014\text{ mSv}$$

Appendix B

RSA GUI - MATLAB CODE

B.1 RSAGUI_CB

```
% RSAGUI_CB - CallBack function for the RSA GUI
%
% All CallBacks for RSAGUI are directed to this function for execution.
%
% Inputs:
%       CB_ID - callback identifier string
%       varargin - variable input arguments
%
% Revision History:
%       July 5, 2000 - Anthony Choo - initial version
%
function rsagui_cb (CB_ID, varargin)

% Constants
LIST_TAG = 1;
AXES_TAG = 2;
LIST_FIELDS = 2;
COL_LABELS = 2;
NUM_CHECKBOXES = 3;
COLOUR = 3;
FILE_TAG = 4;
SYMBOL = 4;
GROUP_LABEL = 5;
WHICH_ITEM = 5;
MEMBER_TAGS = 6;

[curObj guiFig] = gcbo;

switch CB_ID
  % cases for adding and deleting points
  case {'FP_add', 'CP_add','CB_add', ...
    'Calib_Film1_add', 'Calib_Film2_add', ...
    'Recon_Film1_add', 'Recon_Film2_add', ...
    'OP_add'}
    Add2List(varargin{LIST_TAG},varargin{LIST_FIELDS}, ...
        varargin{NUM_CHECKBOXES});
  case {'FP_delete','CP_delete','CB_delete', ...
    'Calib_Film1_delete','Calib_Film2_delete', ...
    'Recon_Film1_delete', 'Recon_Film2_delete', ...
    'OP_delete'}
    DeleteFromList(varargin{LIST_TAG},varargin{LIST_FIELDS}, ...
        varargin{NUM_CHECKBOXES});

  case {'SortList'}
    ProcessSortList;

  % manage lists
  case {'FP_list','CP_list','CB_list', ...
    'Calib_Film1_list','Calib_Film2_list', ...
    'Recon_Film1_list','Recon_Film2_list', ...
    'OP_list'}
    MakeListItemCurrent(varargin{LIST_TAG},varargin{LIST_FIELDS}, ...
        varargin{NUM_CHECKBOXES});

  % cases for Homogeneous Transformation Tool
  case {'HTtool_Tc1', 'HTtool_Tc2'}
    ProcessHTtool (CB_ID, guiFig);
```

```
% cases for saving and loading points
case {'Cage_save', 'Calib_FilmPts_save', 'Recon_FilmPts_save', 'OP_save'}
  SaveData(CB_ID, ...
      varargin{LIST_TAG}, varargin{COL_LABELS}, ...
      varargin{NUM_CHECKBOXES}, varargin{FILE_TAG}, ...
      'w');
case {'Cage_load', 'Calib_FilmPts_load', 'Recon_FilmPts_load', 'OP_load'}
  LoadData(CB_ID, varargin{LIST_TAG}, varargin{LIST_FIELDS}, ...
      varargin{NUM_CHECKBOXES}, varargin{FILE_TAG});

% case for saving and loading simulation
case 'SaveSimulation'
  SaveSimulation;
  %SaveData(CB_ID, ...
  %    varargin{LIST_TAG}, varargin{COL_LABELS}, ...
  %    varargin{NUM_CHECKBOXES}, varargin{FILE_TAG}, ...
  %    'w',varargin{GROUP_LABEL}, varargin{MEMBER_TAGS});
case 'LoadSimulation'
  LoadSimulation;
  %LoadData(CB_ID, ...
  %    varargin{LIST_TAG}, varargin{LIST_FIELDS}, ...
  %    varargin{NUM_CHECKBOXES}, varargin{FILE_TAG}, ...
  %    varargin{GROUP_LABEL}, varargin{MEMBER_TAGS});

case {'Expose_Cage', 'Expose_Object'}
  Expose(CB_ID);

case {'ShowUnshowCalibFilm1', 'ShowUnshowReconFilm1', ...
  'ShowUnshowCalibFilm2', 'ShowUnshowReconFilm2', ...
  'ShowUnshowFP', 'ShowUnshowCP', 'ShowUnshowOP' }
  ProcessShowUnshow (varargin{LIST_TAG},varargin{AXES_TAG}, ...
    varargin{COLOUR},varargin{SYMBOL},varargin{WHICH_ITEM});

case {'Render_Setup'}
  RenderRSASetup;

case {'Menu_ShowFP','Menu_ShowFP_txt', ...
  'Menu_ShowCP', 'Menu_ShowCP_txt' ...
  'Menu_ShowOP', 'Menu_ShowOP_txt' ...
  'Menu_ShowCalib1', 'Menu_ShowCalib1_txt', 'Menu_ShowCalib1_rays' ...
  'Menu_ShowCalib2', 'Menu_ShowCalib2_txt', 'Menu_ShowCalib2_rays' ...
  'Menu_ShowRecon1', 'Menu_ShowRecon1_txt', 'Menu_ShowRecon1_rays' ...
  'Menu_ShowRecon2', 'Menu_ShowRecon2_txt', 'Menu_ShowRecon2_rays'}
  ProcessMenuEvent(CB_ID);

case {'CloseRSASetup'}
  RSASetupClosed;

case {'RunSimulation'}
  RunSim;

case {'Recon_OPs'}
  ReconOPs;

case {'Close_Progress'}
  figHandle = findobj('Tag','ProgressFig');
  if not(isempty(figHandle))
    close(figHandle);
  end
otherwise
  errordlg ('Callback for action is not defined','RSAGUI_CB Error');
end

% ------------------------------------------------------------------------
% Add2List
% ------------------------------------------------------------------------
function Add2List (listTag, listFields, numCheckBoxes)

DELIM = ',';

[curObj curFig] = gcbo;

numFields = size(listFields,1);

% get contents of each of the listFields and concatenate them
newString = [];
for i = 1:numFields - numCheckBoxes
  fieldHandle = findobj(curFig,'Tag',char(listFields(i)));
  fieldString = get(fieldHandle,'String');
  newString = [newString fieldString ','];
end
```

```
% handle checkBoxes
for i = numFields - numCheckBoxes + 1: numFields
   fieldHandle = findobj(curFig,'Tag',char(listFields(i)));
   checkValue = get(fieldHandle,'Value');
   newString = [newString num2str(checkValue) DELIM];
end

% add concatenated string to list
listHandle = findobj(curFig,'Tag',listTag);
listString = get(listHandle,'String');
   % listString will be cells because consistently using cells in listboxes

if isempty(deblank(listString))
   listString = {newString};
else
   % consistently use cells in listboxes
   listString = [listString; {newString}];
end
set(listHandle,'String',listString);
% end Add2List
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% DeleteFromList
% ------------------------------------------------------------------------
function DeleteFromList (listTag, listFields, numCheckBoxes)

[curObj curFig] = gcbo;

listHandle = findobj(curFig,'Tag',listTag);
deleteElement = get(listHandle,'Value');
listString = get(listHandle,'String');
   % listString will be cells because consistently using cells in listboxes
sizeList = size(listString,1);
if sizeList == 1
   listString = {};
elseif deleteElement == 1
   listString = listString(2:sizeList);
elseif deleteElement == sizeList
   % Set the value to something valid since Matlab will try
   % to advance to next one.
   set(listHandle,'Value',sizeList-1);
   listString = listString(1:sizeList-1);
else
   listString = [listString(1:deleteElement-1); ...
         listString(deleteElement+1:sizeList)];
end
set(listHandle,'String',listString);
% Update list field
MakeListItemCurrent(listTag, listFields, numCheckBoxes)
% end DeleteFromList
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% MakeListItemCurrent
% ------------------------------------------------------------------------
function MakeListItemCurrent(listTag, listFields, numCheckBoxes)

DELIM = ',';

[curObj curFig] = gcbo;

% find the list and extract out the line that is highlighted
listHandle = findobj(curFig,'Tag',listTag);
currentValue = get(listHandle,'Value');
listString = get(listHandle,'String');
   % listString will be cells because consistently using cells in listboxes

if isempty(deblank(listString))
   % nothing in listbox
   return;
end

currentString = listString{currentValue};
   % {} dereferences string contents

% parse currentString and store contents in listFields
% check that string is terminated with DELIM
LAST = size(currentString,2);
if strcmp(currentString(LAST:LAST),DELIM) ~= 1
   currentString = [currentString DELIM];
end
```

```
delimAt = findstr(currentString,DELIM);

numFields = size(listFields,1);

fieldHandle = findobj(curFig,'Tag',listFields{1});
if numFields > numCheckBoxes
   % special case: store the first field
   set(fieldHandle,'String',currentString(1:delimAt(1)-1));

   % store the other fields
   for i = 2:numFields - numCheckBoxes
      fieldHandle = findobj(curFig,'Tag',listFields{i});
      set(fieldHandle,'String', ...
         currentString(delimAt(i-1)+1:delimAt(i)-1));
   end

   % handle checkboxes
   for i = numFields - numCheckBoxes + 1 : numFields
      fieldHandle = findobj(curFig,'Tag',listFields{i});
      set(fieldHandle,'Value', ...
         str2num(currentString(delimAt(i-1)+1:delimAt(i)-1)));
   end

else % numFields = numCheckBoxes
   set(fieldHandle,'Value',str2num(currentString(1:delimAt(1)-1)));
   % store the other checkboxes
   for i = 2:numFields
      fieldHandle = findobj(curFig,'Tag',listFields{i});
      set(fieldHandle,'Value', ...
         str2num(currentString(delimAt(i-1)+1:delimAt(i)-1)));
   end
end
% end MakeListItemCurrent
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ProcessHTtool
% ------------------------------------------------------------------------
% Process call to open the homogeneous transformation tool.
function ProcessHTtool(ID, guiFig)

httool;
htHandle = gcf;

AddUserData(htHandle,[{'rsaGuiFig'} {guiFig}]);
switch ID
case 'HTtool_Tc1'
   AddUserData(htHandle,[{'matrixTagPrefix'} {'Tcage_1_'}; ...
         {'sourceTag'} {'source1'}]);
   TPrefix = 'Tcage_1_';
case 'HTtool_Tc2'
   AddUserData(htHandle,[{'matrixTagPrefix'} {'Tcage_2_'}; ...
         {'sourceTag'} {'source2'}]);
   TPrefix = 'Tcage_2_';
end
for i = 1:4
   for j = 1:4
      Tcage_element = findobj(guiFig,'Tag',[TPrefix num2str(i) num2str(j)]);
      Telement = findobj(htHandle,'Tag',['T_' num2str(i) num2str(j)]);
      Tcage_element_string = get(Tcage_element,'String');
      set(Telement,'String',Tcage_element_string);
   end
end
% end ProcessHTtool
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% SaveData
% ------------------------------------------------------------------------
function SaveData(saveID, listTag, colLabels, numCheckBoxes, fileTag, saveMode, ...
      groupLabel, memberTags)

% constants
TRUE = 1;
FALSE = 0;
DELIM = ',';

if nargin == 8
   GROUPS_ENABLED = TRUE;
else
   GROUPS_ENABLED = FALSE;
```

```
end

[curObj curFig] = gcbo;

fileNameHandle = findobj(curFig,'Tag',fileTag);
fileName = get(fileNameHandle,'String');

[saveFile savePath] = uiputfile(fileName, [saveID ' Save As']);
if saveFile == 0
   return;
else
   set(fileNameHandle,'String',saveFile);
end

% open file
FID = fopen([savePath saveFile],saveMode);

% save lists
for i = 1:size(listTag,1)
   % write list title
   fprintf(FID,'%s%s\n','% ',listTag{i});

   % write column titles
   fprintf(FID,'%s','% ');
   for j = 1:size(colLabels{i},1)
      fprintf(FID,'%s,',colLabels{i}{j});
      %fprintf(FID,'%s,',colLabels{j,i});
   end
   fprintf(FID,'\n');

   % find and write values
   listHandle = findobj(curFig,'Tag',listTag{i});
   listString = get(listHandle,'String');
   for j = 1:size(listString,1)
      % remove Show
      outString = listString{j};
      delimAt = findstr(outString,DELIM);
      showAt = size(delimAt,2) - 1;
      outString = outString(1:delimAt(showAt)-1);
      fprintf(FID,'%s\n',outString);
   end
   fprintf(FID,'%sEND_LIST\n\n','% ');
end

if GROUPS_ENABLED == TRUE

% Save Grouped Items
for i = 1:size(groupLabel,1)
   % write group title
   fprintf(FID,'%s%s\n','% ',groupLabel{i});
   % write group members, write members on the same row to one line
   for j = 1:size(memberTags{i},1)
      for k = 1:size(memberTags{i},2)
         % find and write values
         memberHandle = findobj(curFig,'Tag', ...
               memberTags{i}{j,k});
         memberString = get(memberHandle,'String');
         fprintf(FID,'%s%s',memberString,DELIM);
      end
      fprintf(FID,'\n');
   end
   fprintf(FID,'%sEND_GROUP\n\n','% ');
end

end

fclose(FID);
% end SaveData
% ---------------------------------------------------------------------------


% ---------------------------------------------------------------------------
% LoadData
% ---------------------------------------------------------------------------
function LoadData(loadID, listTag, listFields, numCheckBoxes, fileTag, ...
      groupLabel, memberTags)

% constants
DELIM = ',';
TRUE = 1;
FALSE = 0;
```

```
if nargin == 7
   GROUPS_ENABLED = TRUE;
else
   GROUPS_ENABLED = FALSE;
end

CBs = [];
for i = 1:numCheckBoxes
   CBs = [CBs '0' DELIM];
end

[curObj curFig] = gcbo;

fileNameHandle = findobj(curFig,'Tag',fileTag);
fileName = get(fileNameHandle,'String');

[loadFile loadPath] = uigetfile(fileName, [loadID ' Load']);
if loadFile == 0
   return;
else
   set(fileNameHandle,'String',loadFile);
end

% open file
FID = fopen([loadPath loadFile],'r');

% read in lists
for i = 1:size(listTag,1)
   listString = {};
   listTagFound = FALSE;
   % search for listTag, ignore blank lines and comments
   while not(feof(FID))
      curLine = fgetl(FID);
      if listTagFound == TRUE
         if isempty(findstr(curLine,'END_LIST'))
            if not(isempty(deblank(curLine))) & (strncmp(curLine,'%',1)==0)
               % ensure the line is terminated with a delimiter
               if curLine(size(curLine,2)) ~= DELIM
                  curLine = [curLine DELIM];
               end
               % append zero for Show value
               curLine = [curLine CBs];
               listString = [listString; {curLine}];
            end
         else % END_LIST or END_LIST_SECTION found
            break; % new list
         end
      else
         if findstr(curLine,listTag{i}) > 0
            listTagFound = TRUE;
         end
      end
   end

   listHandle = findobj(curFig,'Tag',listTag{i});
   set(listHandle,'String',listString);
end

if GROUPS_ENABLED == TRUE

% load Grouped Items
for i = 1:size(groupLabel,1)
   % search for group label, ignore blank lines
   while not(feof(FID))
      curLine = fgetl(FID);
      if findstr(curLine,groupLabel{i}) > 1
         break; % group label found
      end
   end

   if findstr(curLine,groupLabel{i}) > 1
      % group label found
      % get each line in the group
      for j = 1:size(memberTags{i},1)

         % skip comments and blank lines
         while not(feof(FID))
            curLine = fgetl(FID);
            if curLine(1,1) ~= '%'
               break;
            end
         end
```

```
        % ensure the line is terminated with a delimiter
        if curLine(size(curLine,2)) ~= DELIM
          curLine = [curLine DELIM];
        end

        % parse each line
        delimAt = findstr(curLine,DELIM);
        for k = 1:size(memberTags{i},2)
          if k > 1
            memberString = curLine(delimAt(k-1)+1:delimAt(k)-1)
          else % special case, k=1
            memberString = curLine(1:delimAt(1)-1);
          end
          % find and write values
          memberHandle = findobj(curFig,'Tag', ...
              memberTags{i}{j,k});
          set(memberHandle,'String',memberString);
        end
      end

    end % skip line till group label found
end % for groupLabels

end % GROUPS_ENABLED

fclose(FID);
% end LoadData
% --------------------------------------------------------------------------------


% --------------------------------------------------------------------------------
% Expose
% --------------------------------------------------------------------------------
function Expose (Expose_What)

DELIM = ',';

% Constants for tagPos,x,y,z positions in lists
tagPos=1;tX=2;tY=3;tZ=4;
X=1;Y=2;Z=3;
U=1;V=2;

% NOTE:
%  _c denotes in cage frame
%  _1 denotes in film1 frame (with z axis perpendicular to plane)
%  _2 denotes in film2 frame (with z axis perpendicular to plane)

[curObj curFig] = gcbo;

% get the transformation matricies from the cage frame to the two film planes
Tc1 = GetTMatrix('Tcage_1_',curFig);
Tc2 = GetTMatrix('Tcage_2_',curFig);
T1c = inv(Tc1);
T2c = inv(Tc2);

listString = {};
% get the points to be expose
if strcmp(Expose_What,'Expose_Cage')
  listHandle = findobj(curFig,'Tag','FP_list');
  listString1 = get(listHandle,'String');
  listHandle = findobj(curFig,'Tag','CP_list');
  listString2 = get(listHandle,'String');
  listString = [listString1; listString2];
    % listString is a column of cells 'cause listString1 and listString2
    % are cells
  f1ListTag = 'Calib_Film1_list';
  f2ListTag = 'Calib_Film2_list';
  f1Suffix = '_CLF1';
  f2Suffix = '_CLF2';
elseif strcmp(Expose_What,'Expose_Object')
  listHandle = findobj(curFig,'Tag','OP_list');
  listString = get(listHandle,'String');
  f1ListTag = 'Recon_Film1_list';
  f2ListTag = 'Recon_Film2_list';
  f1Suffix = '_RCF1';
  f2Suffix = '_RCF2';
end

pts_c = [];
ptsTags = [];
for j = 1:size(listString,1)
```

```
    coords = ParseDelimited(listString{j},[tX tY tZ],'NUM');
    pts_c = [pts_c; coords'];
    ptsTags = strvcat(ptsTags,ParseDelimited(listString{j},tagPos));
end

% get sources
sourceHandle = findobj(curFig,'Tag','source1');
sourceString = get(sourceHandle,'String');
source1_c = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
sourceHandle = findobj(curFig,'Tag','source2');
sourceString = get(sourceHandle,'String');
source2_c = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
% transform to source frames
source1_1 = MultHomogeneous(source1_c,T1c);
source2_2 = MultHomogeneous(source2_c,T2c);
pts_1 = MultHomogeneous(pts_c,T1c);
pts_2 = MultHomogeneous(pts_c,T2c);

% get dimensions of films
dimHandle_u = findobj(curFig,'Tag','Dim_F1_u');
dimHandle_v = findobj(curFig,'Tag','Dim_F1_v');
u = get(dimHandle_u,'String');
v = get(dimHandle_v,'String');
if isempty(u) | isempty(v)
    errordlg('Film dimensions unspecified. Operation cancelled.','RSA Error');
    return;
end
f1Dim = [str2num(u(1,:)) str2num(v(1,:))];

dimHandle_u = findobj(curFig,'Tag','Dim_F2_u');
dimHandle_v = findobj(curFig,'Tag','Dim_F2_v');
u = get(dimHandle_u,'String');
v = get(dimHandle_v,'String');
if isempty(u) | isempty(v)
    errordlg('Film dimensions unspecified. Operation cancelled.','RSA Error');
    return;
end
f2Dim = [str2num(u(1,:)) str2num(v(1,:))];

% call xpose
f1Suffix = '_F1';
f2Suffix = '_F2';
[film1, f1Tags] = xpose(pts_1,source1_1,f1Dim,ptsTags,f1Suffix);
[film2, f2Tags] = xpose(pts_2,source2_2,f2Dim,ptsTags,f2Suffix);

% generate list
f1List = {};
for i = 1:size(f1Tags,1)
    rowString = [deblank(f1Tags(i,:)) DELIM num2str(film1(i,U)) ...
        DELIM num2str(film1(i,V)) DELIM '0' DELIM];
    f1List = [f1List; {rowString}];
end
f2List = {};
for i = 1:size(f2Tags,1)
    rowString = [deblank(f2Tags(i,:)) DELIM num2str(film2(i,U)) ...
        DELIM num2str(film2(i,V)) DELIM '0' DELIM];
    f2List = [f2List; {rowString}];
end
% display the points
listHandle = findobj(curFig,'Tag',f1ListTag);
set(listHandle,'String',f1List);
listHandle = findobj(curFig,'Tag',f2ListTag);
set(listHandle,'String',f2List);

% end Expose
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ProcessShowUnshow
% ------------------------------------------------------------------------
function ProcessShowUnshow (listTag, axesTag, colour, symbol, whichItem)

% constants
X=1;Y=2;Z=3;

[curObj guiFig] = gcbo;

% get the list handle and listString
listHandle = findobj(guiFig,'Tag',listTag);
listString = get(listHandle,'String');
if isempty(deblank(listString))
    return;
```

```
end

% verify that the RSAxes is plotted
[RSAFig RSAxes] = VerifyRSASetupPlotted (guiFig);

% verify that the sources and film boundaries are plotted
PlotSources (guiFig, RSAxes);
PlotFilmBoundaries (guiFig, RSAxes);

% update the checkValue control and the value for show in the listbox
checkValue = UpdateCheckStatus(guiFig,curObj,listTag,whichItem);

if isempty(axesTag)
   % only plot to the RSAxes
   PlotList(guiFig,listTag,RSAxes,colour,symbol,RSAC('3D'));
   ApplyVisibility(guiFig,listTag,RSAxes);
else
   % plot to the 2D axes on the GUI and the RSAxes
   axes2D = findobj(guiFig,'Tag',axesTag);
   PlotList(guiFig,listTag,axes2D,colour,symbol,RSAC('2D'));
   ApplyVisibility(guiFig,listTag,axes2D);

   % For the 3D plot to the RSAxes
   % 1) determine the homogeneous transformation for the film points
   % 2) determine the sources for the film points
   if findstr(listTag,'1')
      TPrefix = 'Tcage_1_';
      sourceHandle = findobj(guiFig,'Tag','source1');
      sourceString = get(sourceHandle,'String');
      source = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
   else
      TPrefix = 'Tcage_2_';
      sourceHandle = findobj(guiFig,'Tag','source2');
      sourceString = get(sourceHandle,'String');
      source = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
   end
   Tcfilm = GetTMatrix(TPrefix,guiFig);

   % Plot Film Points and Rays
   PlotList(guiFig,listTag,RSAxes,colour,symbol,RSAC('3D'),Tcfilm);
   PlotRays(guiFig,listTag,source,RSAxes,colour,Tcfilm);
   ApplyVisibility(guiFig,listTag,RSAxes);
end

% ProcessShowUnshow
% ---------------------------------------------------------------------------

% ////////////////////////////////////////////////////////////////////////
% Begin: Routines called from ProcessShowUnshow
% \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
%
% ---------------------------------------------------------------------------
% VerifyRSASetupPlotted
% ---------------------------------------------------------------------------
% This function checks and returns the figure user data for
% RSAxes and RSAFig handles.  They are created if not found.
% ---------------------------------------------------------------------------
function [RSAFig, RSAxes] = VerifyRSASetupPlotted (guiFigureHandle)

% get the axes handle for the RSA Setup
RSAxes = ExtractFromUserData(guiFigureHandle,RSAC('UD_RSAXES'));
RSAFig = ExtractFromUserData(guiFigureHandle,RSAC('UD_RSAFIG'));
if isempty(RSAxes)
   % create the figure for the plot
   [RSAFig RSAxes] = RSASetupFigure;

   % store the handles for later use
   AddUserData(guiFigureHandle,[{RSAC('UD_RSAFIG')} {RSAFig}; ...
         {RSAC('UD_RSAXES')} {RSAxes}]);
   AddUserData(RSAFig,[{RSAC('UD_GUIFIG')} {guiFigureHandle}]);
   set(RSAFig,'CloseRequestFcn','rsagui_cb("CloseRSASetup");closereq');
end
% end VerifyRSASetupPlotted
% ---------------------------------------------------------------------------

% ---------------------------------------------------------------------------
% RSASetupClosed
% ---------------------------------------------------------------------------
% This function removes the RSA figure and axes from the user data in the
% gui figure.
% ---------------------------------------------------------------------------
function RSASetupClosed()
```

```
[curObj RSAFig] = gcbo;

guiFig = ExtractFromUserData(RSAFig,RSAC('UD_GUIFIG'));
DeleteUserData(guiFig,[{RSAC('UD_RSAFIG')};{RSAC('UD_RSAXES')}]);

% end RSASetupClosed
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% PlotSources
% ------------------------------------------------------------------------
% This function plots the 2 sources on the RSAxes.
% ------------------------------------------------------------------------
function PlotSources (guiFig, RSAxes)

% constants
LABEL_OS = RSAC('LABEL_OS');

axes(RSAxes);

% do not plot if sources already plotted
sHandle = findobj(RSAxes,'Tag','s1');
if isempty(sHandle)
   % Sources
   source1 = GetString(guiFig,'source1');
   source1 = [ParseDelimited(source1,[1 2 3],'NUM')]';
   source2 = GetString(guiFig,'source2');
   source2 = [ParseDelimited(source2,[1 2 3],'NUM')]';

   sColour = [RSAC('f1COLOUR') RSAC('f2COLOUR')];
   sSymbol = [RSAC('f1SYMBOL') RSAC('f2SYMBOL')];

   pts = [source1;source2];
   ptsTags = [{'s1';'s2'}];

   [pts ptsTags] = AddPts3D( RSAxes, pts, sColour, sSymbol, LABEL_OS, ptsTags);
end

% end PlotSources
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% PlotFilmBoundaries
% ------------------------------------------------------------------------
% This function plots the 2 film boundaries on the RSAxes.
% ------------------------------------------------------------------------
function PlotFilmBoundaries (guiFig, RSAxes)

% constants
LABEL_OS = RSAC('LABEL_OS');

axes(RSAxes);

% do not plot if patch already plotted
patchHandle = findobj(RSAxes,'Tag','Film1');
if isempty(patchHandle)
   % get film dimensions
   f1u = str2num(GetString(guiFig,'Dim_F1_u'));
   f1v = str2num(GetString(guiFig,'Dim_F1_v'));
   f2u = str2num(GetString(guiFig,'Dim_F2_u'));
   f2v = str2num(GetString(guiFig,'Dim_F2_v'));

   % get transformation matricies
   Tc1 = GetTMatrix('Tcage_1_',guiFig);
   Tc2 = GetTMatrix('Tcage_2_',guiFig);

   % convert edges to patch dimensions
   film1 = MakePatch(f1u,f1v,Tc1);
   film2 = MakePatch(f2u,f2v,Tc2);

   corners = [{film1};{film2}];
   colour = [{RSAC('f1COLOUR')} ; {RSAC('f2COLOUR')}];
   [patchTags, labelTags] = AddPatch3D (RSAxes, corners, colour, LABEL_OS,
[{'Film1'};{'Film2'}]);
end
% end PlotFilmBoundaries
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% UpdateCheckedStatus
% ------------------------------------------------------------------------
```

```
function checkValue = UpdateCheckStatus (curFig, curObj, listTag, whichItem)

% get the list handle and listString
listHandle = findobj(curFig,'Tag',listTag);
listString = get(listHandle,'String');

% determine the value of the check box and the loop range for the update
if whichItem == RSAC('ALL')
   % Show/Hide button pressed
   buttonString = get(curObj,'String');
   if strcmp(buttonString,'Show')
      set(curObj,'String','Hide');
      checkValue = RSAC('CHECKED');
   else
      set(curObj,'String','Show');
      checkValue = RSAC('UNCHECKED');
   end
   loopRange = 1:size(listString,1);
   UpdateMenuChecks(curFig,listTag)
else
   % Checkbox clicked
   checkValue = get(curObj,'Value');
   loopRange = get(listHandle,'Value');
end

% Update the check value in the listbox
for i=loopRange
   posCheckBox = size(listString{i},2) - 1;
   listString{i}(posCheckBox) = num2str(checkValue);
end
set(listHandle,'String',listString);

% end UpdateCheckedStatus
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% SetCheckStatus
% ------------------------------------------------------------------------
% This function sets the check status to all items in a list.
% ------------------------------------------------------------------------
function SetCheckStatus(guiFig, listTag, checkValue, items)

% get the list handle and listString
listHandle = findobj(guiFig,'Tag',listTag);
listString = get(listHandle,'String');

if nargin == 3
   % set all items
   loopRange = 1:size(listString,1);
   % update the show button
   showButton = findobj(guiFig,'Tag',[listTag '_show_button']);
   if checkValue == RSAC('CHECKED')
      set(showButton,'String','Hide');
   else
      set(showButton,'String','Show');
   end
else
   loopRange = items;
end

% Update the check value in the listbox
for i=loopRange
   posCheckBox = size(listString{i},2) - 1;
   listString{i}(posCheckBox) = num2str(checkValue);
end
set(listHandle,'String',listString);

% end SetCheckStatus
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% UpdateMenuChecks
% ------------------------------------------------------------------------
% This function is called from Show all or hide all
function UpdateMenuChecks (guiFig, listTag)

% get RSAFig
RSAFig = ExtractFromUserData(guiFig,RSAC('UD_RSAFIG'));

% toggle menu items

switch listTag
```

```
case 'FP_list'
   tagName = [{'Menu_ShowFP'}; {'Menu_ShowFP_txt'}];
case 'CP_list'
   tagName = [{'Menu_ShowCP'}; {'Menu_ShowCP_txt'}];
case 'OP_list'
   tagName = [{'Menu_ShowOP'}; {'Menu_ShowOP_txt'}];

case 'Calib_Film1_list'
   tagName = [{'Menu_ShowCalib1'}; {'Menu_ShowCalib1_txt'};
{'Menu_ShowCalib1_rays'}];
case 'Calib_Film2_list'
   tagName = [{'Menu_ShowCalib2'}; {'Menu_ShowCalib2_txt'};
{'Menu_ShowCalib2_rays'}];


case 'Recon_Film1_list'
   tagName = [{'Menu_ShowRecon1'}; {'Menu_ShowRecon1_txt'};
{'Menu_ShowRecon1_rays'}];
case 'Recon_Film2_list'
   tagName = [{'Menu_ShowRecon2'}; {'Menu_ShowRecon2_txt'};
{'Menu_ShowRecon2_rays'}];

end

for i = 1:size(tagName,1)
   menuHandle = findobj(RSAFig,'Tag',tagName{i});
   umtoggle(menuHandle);
end
% end UpdateMenuChecks
% ---------------------------------------------------------------------------

% ---------------------------------------------------------------------------
% PlotList
% ---------------------------------------------------------------------------
function PlotList (guiFig, listTag, axesHandle, colour, symbol, nD, HT)

TRUE = 1; FALSE = 0;

% check for homogeneous transformation
if nargin == 7
   HomogeneousT = TRUE;
else
   HomogeneousT = FALSE;
end

% constants
LABEL_OS = RSAC('LABEL_OS');

% extract points and tags from the list
[listTagNames listPts] = GetPtsFromList(guiFig,listTag);

% run through points and remove those that are already plotted
pts = [];
tagNames = {};
for i = 1:size(listTagNames,1)
   ptHandle = [];
   ptHandle = findobj(axesHandle,'Tag',listTagNames{i});
   if isempty(ptHandle)
      pts = [pts; listPts(i,:)];
      tagNames = [tagNames; listTagNames(i)];
   end
end

% Check if need to add column of zeros to pts
if size(pts,2) == RSAC('2D') & nD == RSAC('3D')
   pts = [pts zeros(size(pts,1),1)];
end

if size(pts,2) == RSAC('2D')
   AddPts2D (axesHandle, pts, colour, symbol, LABEL_OS, tagNames);
elseif size(pts,2) == RSAC('3D')
   if HomogeneousT == TRUE
      pts = MultHomogeneous(pts,HT);
   end
   AddPts3D (axesHandle, pts, colour, symbol, LABEL_OS, tagNames);
end

% end PlotList
% ---------------------------------------------------------------------------

% ---------------------------------------------------------------------------
% PlotRays
```

```
% ------------------------------------------------------------------------
% This function plots rays from the source to each point in the list.
% ------------------------------------------------------------------------
function PlotRays(guiFig,listTag,source,RSAxes,colour,Tcfilm);

TRUE=1;FALSE=0;

if nargin == 6
   HomogeneousT = TRUE;
else
   HomogeneousT = FALSE;
end

% extract points and tags from the list
[listTagNames listPts] =  GetPtsFromList(guiFig,listTag);

% run through rays and remove those that are already plotted
endPts = [];
rayTagNames = {};
for i = 1:size(listTagNames,1)
   rayHandle = [];
   rayHandle = findobj(RSAxes,'Tag',[listTagNames{i} '_R']);
   if isempty(rayHandle)
      endPts = [endPts; listPts(i,:)];
      rayTagNames = [rayTagNames; {[listTagNames{i} '_R']}];
   end
end

% Check if need to add column of zeros to endPts
if size(endPts,2) == RSAC('2D')
   endPts = [endPts zeros(size(endPts,1),1)];
end

if HomogeneousT == TRUE
   endPts = MultHomogeneous(endPts,Tcfilm);
end

% Plot Rays
rayTags = AddLns3D(RSAxes,source,endPts,colour,rayTagNames);

% end PlotRays
```

```
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ApplyVisibility
% ------------------------------------------------------------------------
% This function runs through the list of points and determines whether to set
% the point to visible or not.
% ------------------------------------------------------------------------
function ApplyVisibility(guiFig,listTag,RSAxes,showTxt,showRays);

TRUE=1;FALSE=0;

if nargin == 3
   showTxt = TRUE;
   showRays = TRUE;
elseif nargin == 4
   showRays = FALSE;
end

listString = GetString(guiFig,listTag);

for i=1:size(listString,1)
   ptTag = ParseDelimited(listString{i},1);

   ptHandle = findobj(RSAxes,'Tag',ptTag);
   txtHandle = findobj(RSAxes,'Tag',[ptTag 'Txt']);

   posCheckBox = size(listString{i},2) - 1;
   if listString{i}(posCheckBox) == num2str(RSAC('CHECKED'))
      setVisible = 'on';
   else
      setVisible = 'off';
   end

   set(ptHandle,'Visible',setVisible);

   if showTxt
      set(txtHandle,'Visible',setVisible);
   else
      set(txtHandle,'Visible','off');
   end
```

```
    % check for a ray if necessary
    rayHandle = findobj(RSAxes,'Tag',[ptTag '_R']);
    if not(isempty(rayHandle))
        if showRays
            set(rayHandle,'Visible',setVisible);
        else
            set(rayHandle,'Visible','off');
        end
    end
end
% end ApplyVisibility
% ------------------------------------------------------------------------
%
% ////////////////////////////////////////////////////////////////////////
% End: Routines called from ProcessShowUnshow
% \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

% ------------------------------------------------------------------------
% RenderRSASetup
% ------------------------------------------------------------------------
function RenderRSASetup()

% constants
X=1;Y=2;Z=3;

[curObj guiFig] = gcbo;

% verify that the RSAxes is plotted
[RSAFig RSAxes] = VerifyRSASetupPlotted (guiFig);

% verify that the sources and film boundaries are plotted
PlotSources (guiFig, RSAxes);
PlotFilmBoundaries (guiFig, RSAxes);

% Update the check status of each list
listTags = [{'FP_list'};{'CP_list'};{'OP_list'}; ...
    {'Calib_Film1_list'}; {'Calib_Film2_list'};
    {'Recon_Film1_list'}; {'Recon_Film2_list'}];

for i = 1:size(listTags,1)
    SetCheckStatus(guiFig, listTags{i}, RSAC('CHECKED'));
    UpdateMenuChecks(guiFig,listTags{i});
end

% 1) PLOT TO THE 2d AXES on the GUI and the RSAxes
axes2D = findobj(guiFig,'Tag','Calib_Film1_axes');
PlotList(guiFig,'Calib_Film1_list',axes2D,RSAC('f1CALIBCOLOUR'),RSAC('f1CALIBS
YMBOL'),RSAC('2D'));
ApplyVisibility(guiFig,'Calib_Film1_list',axes2D);

axes2D = findobj(guiFig,'Tag','Calib_Film2_axes');
PlotList(guiFig,'Calib_Film2_list',axes2D,RSAC('f2CALIBCOLOUR'),RSAC('f2CALIBS
YMBOL'),RSAC('2D'));
ApplyVisibility(guiFig,'Calib_Film2_list',axes2D);

axes2D = findobj(guiFig,'Tag','Recon_Film1_axes');
PlotList(guiFig,'Recon_Film1_list',axes2D,RSAC('f1RECONCOLOUR'),RSAC('f1RECON
SYMBOL'),RSAC('2D'));
ApplyVisibility(guiFig,'Recon_Film1_list',axes2D);

axes2D = findobj(guiFig,'Tag','Recon_Film2_axes');
PlotList(guiFig,'Recon_Film2_list',axes2D,RSAC('f2RECONCOLOUR'),RSAC('f2RECON
SYMBOL'),RSAC('2D'));
ApplyVisibility(guiFig,'Recon_Film2_list',axes2D);

% 2) get transformation matricies
Tc1 = GetTMatrix('Tcage_1_',guiFig);
Tc2 = GetTMatrix('Tcage_2_',guiFig);

% 3) PLOT each list on the 3D RSAxes
PlotList(guiFig,'FP_list',RSAxes,RSAC('fpCOLOUR'),RSAC('fpSYMBOL'),RSAC('3D'));
PlotList(guiFig,'CP_list',RSAxes,RSAC('cpCOLOUR'),RSAC('cpSYMBOL'),RSAC('3D'));
PlotList(guiFig,'OP_list',RSAxes,RSAC('opCOLOUR'),RSAC('opSYMBOL'),RSAC('3D'))
;
PlotList(guiFig,'Calib_Film1_list',RSAxes,RSAC('f1CALIBCOLOUR'),RSAC('f1CALIBS
YMBOL'),RSAC('3D'),Tc1);
PlotList(guiFig,'Calib_Film2_list',RSAxes,RSAC('f2CALIBCOLOUR'),RSAC('f2CALIBS
YMBOL'),RSAC('3D'),Tc2);
PlotList(guiFig,'Recon_Film1_list',RSAxes,RSAC('f1RECONCOLOUR'),RSAC('f1RECO
NSYMBOL'),RSAC('3D'),Tc1);
```

```
PlotList(guiFig,'Recon_Film2_list',RSAxes,RSAC('f2RECONCOLOUR'),RSAC('f2RECO
NSYMBOL'),RSAC('3D'),Tc2);

% 4) PLOT RAYS
%   a) get sources
sourceHandle = findobj(guiFig,'Tag','source1');
sourceString = get(sourceHandle,'String');
source1 = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
sourceHandle = findobj(guiFig,'Tag','source2');
sourceString = get(sourceHandle,'String');
source2 = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
%   b) plot rays
PlotRays(guiFig,'Calib_Film1_list',source1,RSAxes,RSAC('f1CALIBCOLOUR'),Tc1);
PlotRays(guiFig,'Calib_Film2_list',source2,RSAxes,RSAC('f2CALIBCOLOUR'),Tc2);
PlotRays(guiFig,'Recon_Film1_list',source1,RSAxes,RSAC('f1RECONCOLOUR'),Tc1);
PlotRays(guiFig,'Recon_Film2_list',source2,RSAxes,RSAC('f2RECONCOLOUR'),Tc2);

% 5) Apply visibility
for i = 1:size(listTags,1)
  ApplyVisibility(guiFig,listTags{i},RSAxes);
end

% end RenderRSASetup
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ProcessMenuEvent
% ------------------------------------------------------------------------
function ProcessMenuEvent(eventID);

% constants
TRUE = 1; FALSE = 0;
PT_EVENT = 1;
FILM_EVENT = 2;
TXT_ONLY_EVENT = 3;
TXT_RAY_EVENT = 4;

[curObj curFig] = gcbo;

% get the RSAxes
guiFig = ExtractFromUserData(curFig,RSAC('UD_GUIFIG'));
RSAxes = ExtractFromUserData(guiFig,RSAC('UD_RSAXES'));
RSAFig = ExtractFromUserData(guiFig,RSAC('UD_RSAFIG'));

% toggle the menu item
newCheckStatus = umtoggle(curObj);

switch eventID
case 'Menu_ShowFP'
  Event_Type = PT_EVENT;
  listTag = 'FP_list';
  colour = RSAC('fpCOLOUR');
  symbol = RSAC('fpSYMBOL');
  txtMenu = findobj(curFig,'Tag','Menu_ShowFP_txt');
  raysMenu = [];
case 'Menu_ShowCP'
  Event_Type = PT_EVENT;
  listTag = 'CP_list';
  colour = RSAC('cpCOLOUR');
  symbol = RSAC('cpSYMBOL');
  txtMenu = findobj(curFig,'Tag','Menu_ShowCP_txt');
  raysMenu = [];
case 'Menu_ShowOP'
  Event_Type = PT_EVENT;
  listTag = 'OP_list';
  colour = RSAC('opCOLOUR');
  symbol = RSAC('opSYMBOL');
  txtMenu = findobj(curFig,'Tag','Menu_ShowOP_txt');
  raysMenu = [];
case 'Menu_ShowCalib1'
  Event_Type = FILM_EVENT;
  listTag = 'Calib_Film1_list';
  colour = RSAC('f1CALIBCOLOUR');
  symbol = RSAC('f1CALIBSYMBOL');
  txtMenu = findobj(curFig,'Tag','Menu_ShowCalib1_txt');
  raysMenu = findobj(curFig,'Tag','Menu_ShowCalib1_rays');
  axesTag = 'Calib_Film1_axes';
case 'Menu_ShowCalib2'
  Event_Type = FILM_EVENT;
  listTag = 'Calib_Film2_list';
  colour = RSAC('f2CALIBCOLOUR');
  symbol = RSAC('f2CALIBSYMBOL');
```

```
    txtMenu = findobj(curFig,'Tag','Menu_ShowCalib2_txt');
    raysMenu = findobj(curFig,'Tag','Menu_ShowCalib2_rays');
    axesTag = 'Calib_Film2_axes';
case 'Menu_ShowRecon1'
    Event_Type = FILM_EVENT;
    listTag = 'Recon_Film1_list';
    colour = RSAC('f1RECONCOLOUR');
    symbol = RSAC('f1RECONSYMBOL');
    txtMenu = findobj(curFig,'Tag','Menu_ShowRecon1_txt');
    raysMenu = findobj(curFig,'Tag','Menu_ShowRecon1_rays');
    axesTag = 'Recon_Film1_axes';
case 'Menu_ShowRecon2'
    Event_Type = FILM_EVENT;
    listTag = 'Recon_Film2_list';
    colour = RSAC('f2RECONCOLOUR');
    symbol = RSAC('f2RECONSYMBOL');
    txtMenu = findobj(curFig,'Tag','Menu_ShowRecon2_txt');
    raysMenu = findobj(curFig,'Tag','Menu_ShowRecon2_rays');
    axesTag = 'Recon_Film2_axes';


case 'Menu_ShowFP_txt'
    Event_Type = TXT_ONLY_EVENT;
    listTag = 'FP_list';
case 'Menu_ShowCP_txt'
    Event_Type = TXT_ONLY_EVENT;
    listTag = 'CP_list';
case 'Menu_ShowOP_txt'
    Event_Type = TXT_ONLY_EVENT;
    listTag = 'OP_list';

case {'Menu_ShowCalib1_txt', 'Menu_ShowCalib1_rays'}
    Event_Type = TXT_RAY_EVENT;
    listTag = 'Calib_Film1_list';
    txtMenuTag = 'Menu_ShowCalib1_txt';
    rayMenuTag = 'Menu_ShowCalib1_rays';
case {'Menu_ShowCalib2_txt','Menu_ShowCalib2_rays'}
    Event_Type = TXT_RAY_EVENT;
    listTag = 'Calib_Film2_list';
    txtMenuTag = 'Menu_ShowCalib2_txt';
    rayMenuTag = 'Menu_ShowCalib2_rays';
case {'Menu_ShowRecon1_txt','Menu_ShowRecon1_rays'}
    Event_Type = TXT_RAY_EVENT;
    listTag = 'Recon_Film1_list';
    txtMenuTag = 'Menu_ShowRecon1_txt';
    rayMenuTag = 'Menu_ShowRecon1_rays';
case {'Menu_ShowRecon2_txt','Menu_ShowRecon2_rays'}
    Event_Type = TXT_RAY_EVENT;
    listTag = 'Recon_Film2_list';
    txtMenuTag = 'Menu_ShowRecon2_txt';
    rayMenuTag = 'Menu_ShowRecon2_rays';
end

if Event_Type == PT_EVENT
    SetCheckStatus(guiFig,listTag,newCheckStatus);
    PlotList(guiFig,listTag,RSAxes,colour,symbol,RSAC('3D'));
    ApplyVisibility(guiFig,listTag,RSAxes);
    if newCheckStatus ~= umtoggle(txtMenu)
        umtoggle(txtMenu);
    end
elseif Event_Type == FILM_EVENT
    SetCheckStatus(guiFig,listTag,newCheckStatus);

    % Redundant code as in ProcessShowUnshow but...
    %  unable to perform a recursive call because ProcessShowUnshow relies
    %  on [curObj curFig] = gcbo to determine the calling button instead of
    %  calling findobj
    %
    % plot to the 2D axes on the GUI and the RSAxes
    axes2D = findobj(guiFig,'Tag',axesTag);
    PlotList(guiFig,listTag,axes2D,colour,symbol,RSAC('2D'));
    ApplyVisibility(guiFig,listTag,axes2D);

    % For the 3D plot to the RSAxes
    % 1) determine the homogeneous transformation for the film points
    % 2) determine the sources for the film points
    X=1;Y=2;Z=3;
    if findstr(listTag,'1')
        TPrefix = 'Tcage_1_';
        sourceHandle = findobj(guiFig,'Tag','source1');
        sourceString = get(sourceHandle,'String');
        source = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
```

```
    else
        TPrefix = 'Tcage_2_';
        sourceHandle = findobj(guiFig,'Tag','source2');
        sourceString = get(sourceHandle,'String');
        source = [ParseDelimited(sourceString,[X Y Z],'NUM')]';
    end
    Tcfilm = GetTMatrix(TPrefix,guiFig);

    % Plot Film Points and Rays
    PlotList(guiFig,listTag,RSAxes,colour,symbol,RSAC('3D'),Tcfilm);
    PlotRays(guiFig,listTag,source,RSAxes,colour,Tcfilm);
    ApplyVisibility(guiFig,listTag,RSAxes);

    if newCheckStatus ~= umtoggle(txtMenu)
        umtoggle(txtMenu);
    end
    if newCheckStatus ~= umtoggle(raysMenu)
        umtoggle(raysMenu);
    end
elseif Event_Type == TXT_ONLY_EVENT
    if newCheckStatus == RSAC('CHECKED')
        showTxt = TRUE;
    else
        showTxt = FALSE;
    end
    ApplyVisibility(guiFig,listTag,RSAxes,showTxt);
elseif Event_Type == TXT_RAY_EVENT
    txtMenuHandle = findobj(RSAFig,'Tag',txtMenuTag);
    txtCheckStatus = get(txtMenuHandle,'Checked');
    rayMenuHandle = findobj(RSAFig,'Tag',rayMenuTag);
    rayCheckStatus = get(rayMenuHandle,'Checked');
    if strcmp(txtCheckStatus,'on')
        showTxt = TRUE;
    else
        showTxt = FALSE;
    end
    if strcmp(rayCheckStatus,'on')
        showRays = TRUE;
    else
        showRays = FALSE;
    end
    ApplyVisibility(guiFig,listTag,RSAxes,showTxt,showRays);
end
% end ProcessMenuEvent
% ------------------------------------------------------------------------------

% ------------------------------------------------------------------------------
% SaveSimulation
% ------------------------------------------------------------------------------
function SaveSimulation()

[curObj curFig] = gcbo;

CMT = RSAC('CMT');
DELIM = RSAC('DELIM');

fileNameHandle = findobj(curFig,'Tag','sim_file');
fileName = get(fileNameHandle,'String');

[saveFile savePath] = uiputfile(fileName, ['Simulation Save As']);
if saveFile == 0
    return;
else
    set(fileNameHandle,'String',saveFile);
end

% open file
FID = fopen([savePath saveFile],'w');

fprintf(FID,'%sRSA Simulation Input File\n',CMT);
fprintf(FID,'%sCreated %s\n\n',CMT,datestr(now));

% Save Individual Elements
fprintf(FID,'%sIndividual Data\n',CMT);
simParamsTags = [ ...
        {'Simulation Comments'}; ...
        {'Film Points Noise'}; ...
        {'Fiducial Points Noise'}; ...
        {'Control Points Noise'}; ...
        {'Trials'}; ...
        {'% Source Coordinates (x,y,z)'}; ...
        {'source1'}; ...
```

```
      {'source2'}; ...
      {'% Film Dimensions'}; ...
      {'Dim_F1_u'}; ...
      {'Dim_F1_v'}; ...
      {'Dim_F2_u'}; ...
      {'Dim_F2_v'}; ...
      {'Cage_file'}; ...
      {'Calib_FilmPts_file'}; ...
      {'Recon_FilmPts_file'}; ...
      {'OP_file'}];

for i = 1:size(simParamsTags,1)
   % write Tag as heading
   fprintf(FID,'%s%s\n',CMT,simParamsTags{i});

   editHandle = findobj(curFig,'Tag',simParamsTags{i});
   if not(isempty(editHandle))
      fprintf(FID,'%s\n',get(editHandle,'String'));
   end
end
fprintf(FID,'\n');
fprintf(FID,'%sEND Individual Data\n\n',CMT);

% write checkbox/radiobutton info
fprintf(FID,'%sAnalysis Mode\n',CMT);
checkTags = [ ...
      {'Analyse_DLT'}; ...
      {'Analyse_Selvik'}];
for i = 1:size(checkTags,1)
   % write Tag as heading
   fprintf(FID,'%s%s\n',CMT,checkTags{i});
   checkHandle = findobj(curFig,'Tag',checkTags{i});
   fprintf(FID,'%s\n',num2str(get(checkHandle,'Value')));
end
fprintf(FID,'\n');
fprintf(FID,'%sEND Analysis Mode\n\n',CMT);

% Save Transformation Matricies
fprintf(FID,'%sTransformation Matricies: T_cage_filmPlane\n',CMT);
TPrefix = [{'Tcage_1_'};{'Tcage_2_'}];
for i = 1:size(TPrefix,1)
   fprintf(FID,'%s%s\n',CMT,TPrefix{i});
   for j = 1:4
      for k = 1:4
         tagName = [TPrefix{i} num2str(j) num2str(k)];
         editHandle = findobj(curFig,'Tag',tagName);
         fprintf(FID,'%s%s',get(editHandle,'String'),DELIM);
      end
      fprintf(FID,'\n');
   end
   fprintf(FID,'\n');
end
fprintf(FID,'%sEND Transformation Matricies\n\n',CMT);

% Save lists
listTags = [ ...
      {'FP_list'}; ...
      {'CP_list'}; ...
      {'CB_list'}; ...
      {'OP_list'}; ...
      {'Calib_Film1_list'}; ...
      {'Calib_Film2_list'}; ...
      {'Recon_Film1_list'}; ...
      {'Recon_Film2_list'}];

colLabels = [ ...
      {'Tag,x,y,z'}; ...
      {'Tag,x,y,z'}; ...
      {'Tag,x1,y1,z1,x2,y2,z2'}; ...
      {'Tag,x,y,z'}; ...
      {'Tag,u,v'}; ...
      {'Tag,u,v'}; ...
      {'Tag,u,v'}; ...
      {'Tag,u,v'}];

for i = 1:size(listTags,1)
   % write list title
   fprintf(FID,'%s%s\n',CMT,listTags{i});

   % write column titles
   fprintf(FID,'%s%s\n',CMT,colLabels{i});
```

```
    % find and write values
    listHandle = findobj(curFig,'Tag',listTags{i});
    listString = get(listHandle,'String');

    if not(isempty(deblank(listString)))
      for j = 1:size(listString,1)
        % remove Show
        outString = listString{j};
        delimAt = findstr(outString,DELIM);
        showAt = size(delimAt,2) - 1;
        outString = outString(1:delimAt(showAt)-1);
        fprintf(FID,'%s\n',outString);
      end
    end
    fprintf(FID,'%sEND_LIST\n\n',CMT);
end

fclose(FID);
% end SaveSimulation
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% LoadSimulation
% ------------------------------------------------------------------------
function LoadSimulation()

TRUE = 1; FALSE = 0;
DELIM = RSAC('DELIM');

[curObj curFig] = gcbo;

CMT = '% ';
DELIM = RSAC('DELIM');

fileNameHandle = findobj(curFig,'Tag','sim_file');
fileName = get(fileNameHandle,'String');

[loadFile loadPath] = uigetfile(fileName, ['Load Simulation']);
if loadFile == 0
    return;
else
    set(fileNameHandle,'String',loadFile);
end

% open file
FID = fopen([loadPath loadFile],'r');

% Read Individual Elements
simParamsTags = [ ...
      {'Simulation Comments'}; ...
      {'Film Points Noise'}; ...
      {'Fiducial Points Noise'}; ...
      {'Control Points Noise'}; ...
      {'Trials'}; ...
      {'% Source Coordinates (x,y,z)'}; ...
      {'source1'}; ...
      {'source2'}; ...
      {'% Film Dimensions'}; ...
      {'Dim_F1_u'}; ...
      {'Dim_F1_v'}; ...
      {'Dim_F2_u'}; ...
      {'Dim_F2_v'}; ...
      {'Cage_file'}; ...
      {'Calib_FilmPts_file'}; ...
      {'Recon_FilmPts_file'}; ...
      {'OP_file'}];

for i = 1:size(simParamsTags,1)
    if strncmp(simParamsTags{i},CMT,size(CMT,2)) == 0
      readData = SeekData(FID,simParamsTags{i});
    end
    editHandle = findobj(curFig,'Tag',simParamsTags{i});
    set(editHandle,'String',readData);
end

% read checkbox/radiobutton info
checkTags = [ ...
      {'Analyse_DLT'}; ...
      {'Analyse_Selvik'}];
for i = 1:size(checkTags,1)
    readData = SeekData(FID,checkTags{i});
    editHandle = findobj(curFig,'Tag',checkTags{i});
```

```
    set(editHandle,'Value',str2num(deblank(readData)));
end

% Read Transformation Matricies
TPrefix = [{'Tcage_1_'};{'Tcage_2_'}];
for i = 1:size(TPrefix,1)
    readData = SeekData(FID,TPrefix{i});
    rowData = ParseDelimited(readData,[1 2 3 4],'NUM');
    for j = 1:4
        for k = 1:4
            tagName = [TPrefix{i} num2str(j) num2str(k)];
            editHandle = findobj(curFig,'Tag',tagName);
            set(editHandle,'String',num2str(rowData(k)));
        end
        if j<4
            readData = fgetl(FID);
            rowData = ParseDelimited(readData,[1 2 3 4],'NUM');
        end
    end
end

% Read lists
listTags = [ ...
        {'FP_list'}; ...
        {'CP_list'}; ...
        {'CB_list'}; ...
        {'OP_list'}; ...
        {'Calib_Film1_list'}; ...
        {'Calib_Film2_list'}; ...
        {'Recon_Film1_list'}; ...
        {'Recon_Film2_list'}];

colLabels = [ ...
        {'Tag,x,y,z'}; ...
        {'Tag,x,y,z'}; ...
        {'Tag,x1,y1,z1,x2,y2,z2'}; ...
        {'Tag,x,y,z'}; ...
        {'Tag,u,v'}; ...
        {'Tag,u,v'}; ...
        {'Tag,u,v'}; ...
        {'Tag,u,v'}];

% hack solution
CBs = ['0' DELIM];
for i = 1:size(listTags,1)
    listString = {};
    listTagFound = FALSE;
    % search for listTag, ignore blank lines and comments
    while not(feof(FID))
        curLine = fgetl(FID);
        if listTagFound == TRUE
            if isempty(findstr(curLine,'END_LIST'))
                if not(isempty(deblank(curLine))) & (strncmp(curLine,'%',1)==0)
                    % ensure the line is terminated with a delimiter
                    if curLine(size(curLine,2)) ~= DELIM
                        curLine = [curLine DELIM];
                    end
                    % append zero for Show value
                    curLine = [curLine CBs];
                    listString = [listString; {curLine}];
                end
            else % END_LIST or END_LIST_SECTION found
                break; % new list
            end
        else
            if findstr(curLine,listTags{i}) > 0
                listTagFound = TRUE;
            end
        end
    end

    listHandle = findobj(curFig,'Tag',listTags{i});
    set(listHandle,'String',listString);
end

fclose(FID);
% end LoadSimulation
% ---------------------------------------------------------------------------

% ---------------------------------------------------------------------------
% SeekData
% ---------------------------------------------------------------------------
```

```
function [dataLine] = SeekData(FID,dataID)
   dataLine = [];
   % search for dataID
   while not(feof(FID))
      dataLine = fgetl(FID);
      if findstr(dataLine,dataID)
         break; % found item
      end
   end
   % look for first line of data after identifier
   while not(feof(FID))
      dataLine = fgetl(FID);
      if strncmp(dataLine,'%',1)==0
         break;
      end
   end
% end SeekData
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% UTILITY FUNCTIONS
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% RSASetupFigure
% ------------------------------------------------------------------------
function [figureHandle, axesHandle] = RSASetupFigure

figureHandle = figure;
title ('RSA Setup','Color','w');
set(figureHandle,'Color','black');
axesHandle = gca;
set(axesHandle,'Color','black');
set(axesHandle,'XColor','white');
set(axesHandle,'YColor','white');
set(axesHandle,'ZColor','white');
set(axesHandle,'GridLineStyle',':');
xlabel('X'); ylabel('Y'); zlabel('Z');
view(150,30);
grid on; hold on;

%%%%%%%%%%%%%%%%%%%
%% ui controls %%
%%%%%%%%%%%%%%%%%%%

% Current view
vw = get(gca,'View');

% define slider step
sStep = RSAC('SLIDER_STEP');
troughStep = RSAC('TROUGH_STEP');

% Azimuth
Hc_az = uicontrol(gcf,'Style','slider',...
        'Units','normalized',...
        'Position',[0 0 0.9 0.02],...
        'sliderstep',[sStep troughStep], ...
        'Min',-360,'Max',360,'Value',vw(1),...
        'Tag','AZslider',...
        'CallBack','azCallBack("AZslider","AZcur")');
        %'CallBack',[...
        %'set(Hc_cur,"String",num2str(get(Hc_az,"Value"))),'...
        %'set(gca,"View",[get(Hc_az,"Value") vw(2)])']);

Hc_min = uicontrol(gcf,'Style','text',...
        'Units','normalized',...
        'Position',[0 0.02 0.06 0.04],...
        'String', num2str(get(Hc_az,'Min')));

Hc_max = uicontrol(gcf,'Style','text',...
        'Units','normalized',...
        'Position',[0.84 0.02 0.06 0.04],...
        'String', num2str(get(Hc_az,'Max')));

Hc_cur = uicontrol(gcf,'Style','text',...
        'Units','normalized',...
        'Position',[0.47 0.02 0.06 0.04],...
        'Tag','AZcur',...
        'String', num2str(get(Hc_az,'Value')));
```

```
% Elevation
Hc_el = uicontrol(gcf,'Style','slider',...
        'Units','normalized',...
        'Position',[0.98 0 0.02 0.9],...
        'sliderstep',[sStep troughStep], ...
        'Min',-360,'Max',360,'Value',vw(2),...
        'Tag','ELslider',...
        'CallBack','elCallBack("ELslider","ELcur")');
        %'CallBack',[...
        %'set(Hc_cur,"String",num2str(get(Hc_az,"Value"))),'...
        %'set(gca,"View",[get(Hc_az,"Value") vw(2)])']);

EL_min = uicontrol(gcf,'Style','text',...
        'Units','normalized',...
        'Position',[0.92 0 0.06 0.04],...
        'String', num2str(get(Hc_el,'Min')));

EL_max = uicontrol(gcf,'Style','text',...
        'Units','normalized',...
        'Position',[0.92 0.86 0.06 0.04],...
        'String', num2str(get(Hc_el,'Max')));

EL_cur = uicontrol(gcf,'Style','text',...
        'Units','normalized',...
        'Position',[0.92 0.45 0.06 0.04],...
        'Tag','ELcur',...
        'String', num2str(get(Hc_el,'Value')));

%%%%%%%%%%%%%%%%%
%% Menu Items %%
%%%%%%%%%%%%%%%%%

%% Cage %%
cageMenu = uimenu(gcf,'Label','Cage');
cageShowMenu = uimenu(cageMenu,'Label','Show');

CallBackString = [ 'RSAGUI_CB("Menu_ShowFP")' ];
cageShowFPMenu = uimenu(cageShowMenu,'Label','Fiducial Points', ...
          'Checked','off', ...
          'Tag', 'Menu_ShowFP', ...
          'CallBack', CallBackString);

CallBackString = [ 'RSAGUI_CB("Menu_ShowFP_txt")' ];
cageShowFPtxtMenu = uimenu(cageShowMenu,'Label','Fiducial Points Text', ...
          'Checked','off', ...
          'Tag', 'Menu_ShowFP_txt', ...
          'CallBack', CallBackString);


CallBackString = [ 'RSAGUI_CB("Menu_ShowCP")' ];
cageShowFPMenu = uimenu(cageShowMenu,'Label','Control Points', ...
          'Checked','off', ...
          'Tag', 'Menu_ShowCP', ...
          'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowCP_txt")' ];
cageShowCPtxtMenu = uimenu(cageShowMenu,'Label','Control Points Text', ...
          'Checked','off', ...
          'Tag', 'Menu_ShowCP_txt', ...
          'CallBack', CallBackString);

%% Object %%
objMenu = uimenu(gcf,'Label','Object');
objShowMenu = uimenu(objMenu,'Label','Show');
CallBackString = [ 'RSAGUI_CB("Menu_ShowOP")' ];
cageShowOPMenu = uimenu(objShowMenu,'Label','Object Points', ...
          'Checked','off', ...
          'Tag', 'Menu_ShowOP', ...
          'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowOP_txt")' ];
cageShowOPtxtMenu = uimenu(objShowMenu,'Label','Object Points Text', ...
          'Checked','off', ...
          'Tag', 'Menu_ShowOP_txt', ...
          'CallBack', CallBackString);


%% Film1 %%
f1Menu = uimenu(gcf,'Label','Film 1');
f1ShowMenu = uimenu(f1Menu,'Label','Show');
% calib 1
CallBackString = [ 'RSAGUI_CB("Menu_ShowCalib1")' ];
cageShowCalib1Menu = uimenu(f1ShowMenu,'Label','Calibration Points', ...
          'Checked','off', ...
          'Tag', 'Menu_ShowCalib1', ...
```

```
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowCalib1_txt")' ];
cageShowCalib1txtMenu = uimenu(f1ShowMenu,'Label','Calibration Points Text', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowCalib1_txt', ...
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowCalib1_rays")' ];
cageShowCalib1RaysMenu = uimenu(f1ShowMenu,'Label','Calibration Rays', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowCalib1_rays', ...
        'CallBack', CallBackString);
% recon 1
CallBackString = [ 'RSAGUI_CB("Menu_ShowRecon1")' ];
cageShowRecon1Menu = uimenu(f1ShowMenu,'Label','Recon Points', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowRecon1', ...
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowRecon1_txt")' ];
cageShowRecon1txtMenu = uimenu(f1ShowMenu,'Label','Recon Points Text', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowRecon1_txt', ...
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowRecon1_rays")' ];
cageShowRecon1RaysMenu = uimenu(f1ShowMenu,'Label','Recon Rays', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowRecon1_rays', ...
        'CallBack', CallBackString);


%% Film2 %%
f2Menu = uimenu(gcf,'Label','Film 2');
f2ShowMenu = uimenu(f2Menu,'Label','Show');
% calib 2
CallBackString = [ 'RSAGUI_CB("Menu_ShowCalib2")' ];
cageShowCalib2Menu = uimenu(f2ShowMenu,'Label','Calibration Points', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowCalib2', ...
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowCalib2_txt")' ];
cageShowCalib2txtMenu = uimenu(f2ShowMenu,'Label','Calibration Points Text', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowCalib2_txt', ...
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowCalib2_rays")' ];
cageShowCalib2RaysMenu = uimenu(f2ShowMenu,'Label','Calibration Rays', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowCalib2_rays', ...
        'CallBack', CallBackString);
% recon 2
CallBackString = [ 'RSAGUI_CB("Menu_ShowRecon2")' ];
cageShowRecon2Menu = uimenu(f2ShowMenu,'Label','Recon Points', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowRecon2', ...
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowRecon2_txt")' ];
cageShowRecon2txtMenu = uimenu(f2ShowMenu,'Label','Recon Points Text', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowRecon2_txt', ...
        'CallBack', CallBackString);
CallBackString = [ 'RSAGUI_CB("Menu_ShowRecon2_rays")' ];
cageShowRecon2RaysMenu = uimenu(f2ShowMenu,'Label','Recon Rays', ...
        'Checked','off', ...
        'Tag', 'Menu_ShowRecon2_rays', ...
        'CallBack', CallBackString);


% end RSASetupFigure
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% GetPtsFromList
% ------------------------------------------------------------------------
% tagNames - column cell array of strings
% pts - column where each row is the x,y(,z) coordinates of the points
%
function [tagNames, pts] = GetPtsFromList(figHandle,listTag,specificItems)

% constants
DELIM = RSAC('DELIM');
TWO_D = 4;
THREE_D = 5;
LINES = 7;
```

```
TAG_POS = 1;
X_POS = 2; Y_POS = 3; Z_POS = 4;
X1_POS = 2; Y1_POS = 3; Z1_POS = 4; X2_POS = 5; Y2_POS = 6; Z2_POS = 7;

listHandle = findobj(figHandle,'Tag',listTag);
listString = get(listHandle,'String');

if nargin == 2
   loopRange = 1:size(listString,1);
else
   loopRange = specificItems;
end

% determine the plot type based on the number of delimiting separators
delimAt = findstr(listString{1},DELIM);
plotMode = size(delimAt,2);

pts = [];
tagNames = {};

switch plotMode
case TWO_D
   for i = loopRange
      tagNames = [tagNames; {ParseDelimited(listString{i},TAG_POS)}];
      Coords = ParseDelimited(listString{i},[X_POS Y_POS],'NUM');
      pts = [pts; Coords'];
   end
case THREE_D
   for i = loopRange
      tagNames = [tagNames; {ParseDelimited(listString{i},TAG_POS)}];
      Coords = ParseDelimited(listString{i},[X_POS Y_POS Z_POS],'NUM');
      pts = [pts; Coords'];
   end
case LINES
   for i = loopRange
      tagNames = [tagNames; {ParseDelimited(listString{i},TAG_POS)}];
      Coords = ParseDelimited(listString{i}, ...
         [X1_POS Y1_POS Z1_POS X2_POS Y2_POS Z2_POS],'NUM');
      pts = [pts; Coords'];
   end
end

% end GetPtsFromList
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% GetTMatrix
% ------------------------------------------------------------------------
% initialize the matricies
function [Tcn] = GetTMatrix(TcnPrefix, curFig)
Tcn = [];
for i = 1:4
   % initialize temporary row variables
   TcnRow = [];
   for j = 1:4
      elementTag = [TcnPrefix num2str(i) num2str(j)];
      elementHandle = findobj(curFig,'Tag',elementTag);
      elementString = get(elementHandle,'String');
      TcnRow = [TcnRow str2num(elementString)];
   end
   Tcn = [Tcn; TcnRow];
end

% GetTMatrix
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% GetString
% ------------------------------------------------------------------------
function [s] = GetString(curFig, tagName)

if iscell(tagName)
   s = {};
   for i = 1:size(tagName,1)
      handle = findobj(curFig,'Tag',tagName{i});
      getString = get(handle,'String');
      s = [s; {getString}];
   end
else
   handle = findobj(curFig,'Tag',tagName);
   s = get(handle,'String');
```

```
end

% GetString
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% MakePatch
% ------------------------------------------------------------------------
function [corners] = MakePatch (xlength, ywidth, transformation)

if nargin == 2
   transformation = [];
end
corners = [ 0,0,0;
      xlength,0,0;
      xlength,ywidth,0;
      0,ywidth,0];

if not(isempty(transformation))
   corners = MultHomogeneous(corners,transformation);
end
% MakePatch
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% AddUserData
% ------------------------------------------------------------------------
% newUD = [{string id} {value}]
% ------------------------------------------------------------------------
function AddUserData (figHandle, newUD)
UD = get(figHandle,'UserData');
if isempty(UD)
   set(figHandle,'UserData',newUD);
else
   UD = [UD; newUD];
   set(figHandle,'UserData',UD);
end

% end AddUserData
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% DeleteUserData
% ------------------------------------------------------------------------
function DeleteUserData (figHandle, deleteUD)

TRUE = 1; FALSE = 0;

UD = get(figHandle,'UserData');
if isempty(UD)
   return;
else
   updatedUD = {};
   for i = 1:size(UD,1)
      deleteFound = FALSE;
      for j = 1:size(deleteUD,1)
         if strcmp(UD{i,1},deleteUD{j})
            deleteFound = TRUE;
         end
      end
      if deleteFound == FALSE
         updatedUD = UD(i,:);
      end
   end
   set(figHandle,'UserData',updatedUD);
end

% end DeleteUserData
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ExtractFromUserData
% ------------------------------------------------------------------------
function [UDitem] = ExtractFromUserData(figHandle,UD_Tag)
   UDitem = [];
   UD = get(figHandle,'UserData');
   for i = 1:size(UD,1)
      if UD{i,1} == UD_Tag
         UDitem = UD{i,2};
      end
   end
% end ExtractFromUserData
```

```
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ProcessSortList
% ------------------------------------------------------------------------
% This funtion processes the SortList callback.
% ------------------------------------------------------------------------
function ProcessSortList()

[curObj, guiFig] = gcbo;
buttonTag = get(curObj,'Tag');

switch buttonTag

case 'fpSort_button'
   listTag = 'FP_list';
case 'cpSort_button'
   listTag = 'CP_list';
case 'cbSort_button'
   listTag = 'CB_list';
case 'opSort_button'
   listTag = 'OP_list';
case 'CalibFilm1Sort_button'
   listTag = 'Calib_Film1_list';
case 'CalibFilm2Sort_button'
   listTag = 'Calib_Film2_list';
case 'ReconFilm1Sort_button'
   listTag = 'Recon_Film1_list';
case 'ReconFilm2Sort_button'
   listTag = 'Recon_Film2_list';
end

SortList(guiFig,listTag);

% end ProcessSortList
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% SortList
% ------------------------------------------------------------------------
% This funtion uses sortrows() to sort the lists.
% ------------------------------------------------------------------------
function SortList (guiFig,listTag)

listHandle = findobj(guiFig,'Tag',listTag);
listString = get(listHandle,'String');
if not(isempty(deblank(listString)))
   sortedString = cellstr(sortrows(char(listString)));
end
set(listHandle,'String',sortedString);
% end SortList
% ------------------------------------------------------------------------



% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% Simulation routines
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% RunSim
% ------------------------------------------------------------------------
function RunSim()

TRUE=1;FALSE=0;

[curObj guiFig] = gcbo;

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% gather information for header and simulation
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
editHandle = findobj(guiFig,'Tag','Cage_file');
cageFile = get(editHandle,'String');
editHandle = findobj(guiFig,'Tag','Calib_FilmPts_file');
calibFile = get(editHandle,'String');
editHandle = findobj(guiFig,'Tag','Recon_FilmPts_file');
reconFile = get(editHandle,'String');
editHandle = findobj(guiFig,'Tag','OP_file');
objectFile = get(editHandle,'String');
```

```
HeaderParams = [{'Cage File'} {cageFile}; ...
      {'Calib Film File'} {calibFile}; ...
      {'Recon Film File'} {reconFile}; ...
      {'Object Film File'} {objectFile}];

% determine analysis modes
radioHandle = findobj(guiFig,'Tag','Analyse_DLT');
Analyse_DLT = get(radioHandle,'Value');
radioHandle = findobj(guiFig,'Tag','Analyse_Selvik');
Analyse_Selvik = get(radioHandle,'Value');

if Analyse_DLT == 1
   HeaderParams = [HeaderParams; {'Analyse_DLT'} {'yes'}];
else
   HeaderParams = [HeaderParams; {'Analyse_DLT'} {'no'}];
end
if Analyse_Selvik == 1
   HeaderParams = [HeaderParams; {'Analyse_Selvik'} {'yes'}];
else
   HeaderParams = [HeaderParams; {'Analyse_Selvik'} {'no'}];
end

% read noise
% Note: filmNoise is the measurement error in the film not the x-ray noise
%      of the film
editHandle = findobj(guiFig,'Tag','Film Points Noise');
filmNoise = str2num(get(editHandle,'String'));
editHandle = findobj(guiFig,'Tag','Fiducial Points Noise');
fpNoise = str2num(get(editHandle,'String'));
editHandle = findobj(guiFig,'Tag','Control Points Noise');
cpNoise = str2num(get(editHandle,'String'));

% read number of trails
editHandle = findobj(guiFig,'Tag','Trials');
if isempty(editHandle)
   errordlg('Could not read number of trials.','Rum Simulation Error');
   return;
else
   nTrials = str2num(get(editHandle,'String'));
end

% read comment string
commentHandle = findobj(guiFig,'Tag','Simulation Comments');
simComment = get(commentHandle,'String');

HeaderParams = [HeaderParams; ...
      {'Film Points Noise'} {num2str(filmNoise)}; ...
      {'Fiducial Points Noise'} {num2str(fpNoise)}; ...
      {'Control Points Noise'} {num2str(cpNoise)}; ...
      {'Number of Trials'} {num2str(nTrials)}; ...
      {'Comments'} {simComment}];

%---------------------------------------
% gather the points
%---------------------------------------
[fpTags FP] = GetPtsFromList(guiFig,'FP_list');
[cpTags CP] = GetPtsFromList(guiFig,'CP_list');
[opTags OP] = GetPtsFromList(guiFig,'OP_list');

[calibF1Tags calibF1] = GetPtsFromList(guiFig,'Calib_Film1_list');
[calibF2Tags calibF2] = GetPtsFromList(guiFig,'Calib_Film2_list');
[reconF1Tags reconF1] = GetPtsFromList(guiFig,'Recon_Film1_list');
[reconF2Tags reconF2] = GetPtsFromList(guiFig,'Recon_Film2_list');

%calibF1MatchedPts = MatchFilmAndCage
(calibF1Tags,calibF1,[fpTags;cpTags],[FP;CP]);
%calibF2MatchedPts = MatchFilmAndCage
(calibF2Tags,calibF2,[fpTags;cpTags],[FP;CP]);

% for Selvik's method, need to split control and fiducial points
[FP_F1_Tags FP_F1 CP_F1_Tags CP_F1] = Split_FP_CP(calibF1Tags, calibF1);
[FP_F2_Tags FP_F2 CP_F2_Tags CP_F2] = Split_FP_CP(calibF2Tags, calibF2);

% generate lookup vector of lab coordinates to each film point
FP_F1_LV = GenerateLookupVector(FP_F1_Tags,FP_F1,fpTags,FP);
FP_F2_LV = GenerateLookupVector(FP_F2_Tags,FP_F2,fpTags,FP);
CP_F1_LV = GenerateLookupVector(CP_F1_Tags,CP_F1,cpTags,CP);
CP_F2_LV = GenerateLookupVector(CP_F2_Tags,CP_F2,cpTags,CP);
% moved below begin
%% for 3D DLT, combine control and fiducial points as calibration points
%calibF1_cage = [FP_F1_cage; CP_F1_cage];
%calibF2_cage = [FP_F2_cage; CP_F2_cage];
```

```
% move below end

%----------------------------------------
% open the file to save header and results
%----------------------------------------
fileNameHandle = findobj(guiFig,'Tag','Output_file');
fileName = get(fileNameHandle,'String');

[saveFile savePath] = uiputfile(fileName, ['Save Simulation Results As']);
if saveFile == 0
   return;
else
   set(fileNameHandle,'String',saveFile);
end

% open file
FID = fopen([savePath saveFile],'w');

WriteSimHeader(FID,HeaderParams,opTags,Analyse_DLT,Analyse_Selvik);

% for 3DDLT, Control Points = [FP CP]
ProgressFigure('Create',nTrials);
StartTime = clock;
for i = 1:nTrials
   %----------------------------------------
   % Add noise
   %----------------------------------------
   FP_F1_measured = FP_F1+(filmNoise*randn(size(FP_F1)));
   FP_F2_measured = FP_F2+(filmNoise*randn(size(FP_F2)));
   CP_F1_measured = CP_F1+(filmNoise*randn(size(CP_F1)));
   CP_F2_measured = CP_F2+(filmNoise*randn(size(CP_F2)));
   % Object
   OP_F1_measured = reconF1+(filmNoise*randn(size(reconF1)));
   OP_F2_measured = reconF2+(filmNoise*randn(size(reconF2)));
   %
   % Fiducial Points and Control Points
   FP_uncertain = FP+(fpNoise*randn(size(FP)));
   CP_uncertain = CP+(cpNoise*randn(size(CP)));

   % Note:
   %    This is here instead of outside the loop so that the uncertainty
   %    is added to the FPs and CPs BEFORE matching with film.
   %    If the match with film is done first, then points occuring
   %    in both F1 and F2 will have 2 different uncertainties added to them
   %    when they should be the same.
   %
   % generate the list of matching lab coordinates to each film point
   FP_F1_cage = FP_uncertain(FP_F1_LV,:);
   FP_F2_cage = FP_uncertain(FP_F2_LV,:);
   CP_F1_cage = CP_uncertain(CP_F1_LV,:);
   CP_F2_cage = CP_uncertain(CP_F2_LV,:);

   %----------------------------------------
   % Call Scripts to perform calibration, reconstruction and error analysis
   %----------------------------------------
   if Analyse_DLT == 1
      % for 3D DLT, combine control and fiducial points as calibration points
      calibF1_cage = [FP_F1_cage; CP_F1_cage];
      calibF2_cage = [FP_F2_cage; CP_F2_cage];
      [reconOP,magPtE,xyzRMSE,RMSE] = Script3DDLT( ...
               [FP_F1_measured; CP_F1_measured],calibF1_cage, ...
               [FP_F2_measured; CP_F2_measured],calibF2_cage, ...
               OP_F1_measured, OP_F2_measured, ...
               OP);

      % Write: Trial Number RMSE xRMSE yRMSE zRMSE magPtE

fprintf(FID,'%i\t%f\t%f\t%f\t%f\tspacer',i,RMSE(1),xyzRMSE(1),xyzRMSE(2),xyzRMSE(
3));
      for k = 1:size(magPtE,1)
         fprintf(FID,'\t%f',magPtE(k));
      end
   end

   if Analyse_Selvik == 1
      [reconOP,magPtE,xyzRMSE,RMSE] = ScriptSelvik( ...
               FP_F1_measured,FP_F1_cage, ...
               FP_F2_measured,FP_F2_cage, ...
               CP_F1_measured,CP_F1_cage, ...
               CP_F2_measured,CP_F2_cage, ...
               OP_F1_measured,OP_F2_measured, ...
               OP);
```

```
    % Write: Trial Number RMSE xRMSE yRMSE zRMSE magPtE

fprintf(FID,'%i\t%f\t%f\t%f\t%f\tspacer',i,RMSE(1),xyzRMSE(1),xyzRMSE(2),xyzRMSE(
3));
    for k = 1:size(magPtE,1)
      fprintf(FID,'\t%f',magPtE(k));
    end
  end
  fprintf(FID,'\n');

ProgressFigure('SetNumTrialsComplete',i);
end
fclose(FID);
ProgressFigure('SimComplete',nTrials,etime(clock,StartTime));

% end RunSim
% ------------------------------------------------------------------------
%
% ////////////////////////////////////////////////////////////////////////
% Begin: Routines called from RunSim
% \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

% ------------------------------------------------------------------------
% MatchFilmAndCage
%   Obsolete - see GenerateMatchVector and GenerateMatchedPoints
%   Results in poor performance (O(N^2 /2)) when moved inside the simulation
%   loop in RunSim in order to handle noise/measurment error to cage points.
% ------------------------------------------------------------------------
% Given a list of film coordinates, this function uses their tags to generate
% the corresponding list of cage calibration points.
% ------------------------------------------------------------------------
function matchedPts = MatchFilmAndCage(filmTags,filmPts,cageTags,cagePts)
US = '_';
matchedPts = [];
for i = 1:size(filmTags,1)
  filmPtTag = ParseDelimited(filmTags{i},1,'STRING',US);
  for j = 1:size(cageTags)
    cagePtTag = ParseDelimited(cageTags{j},1,'STRING',US);
    if strcmp(filmPtTag,cagePtTag)
      matchedPts = [matchedPts; cagePts(j,:)];
    end
  end
end
% end MatchFilmAndCage
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% GenerateLookupVector
% ------------------------------------------------------------------------
% Given a list of film coordinates and cage points, this function uses their
% tags to generate a vector of size filmPts indicating the index in cagePts of
% the match. (i.e. a Nx1 lookup table)
% This routine is O(N^2 / 2) and should be called outside the simulation loop
% in RunSim and should be used with:
%   matchedPts = cagePts(lookupVector,:);
% within the loop.
% ------------------------------------------------------------------------
function lookupVector = GenerateLookupVector(filmTags,filmPts,cageTags,cagePts)
US = '_';
lookupVector = zeros(size(filmPts,1),1);
for i = 1:size(filmTags,1)
  filmPtTag = ParseDelimited(filmTags{i},1,'STRING',US);
  for j = 1:size(cageTags)
    cagePtTag = ParseDelimited(cageTags{j},1,'STRING',US);
    if strcmp(filmPtTag,cagePtTag)
      lookupVector(i) = j;
      break;
    end
  end
end
% end GenerateLookupVector
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% GenerateMatchedPoints
% ------------------------------------------------------------------------
% This function extracts a set of points using the indicies of a lookup vector.
% The function also optionally adds the input measurement error to the resultant
% matchedPts.
% This function is used in combination with GenerateLookupVector to replace
% MatchFilmAndCage which resulted in poor performance when it was necessary
```

```
% to add uncertainty to the fiducial and control points
% ------------------------------------------------------------------------
function matchedPts = GenerateMatchedPoints(lookupVector, cagePts, cageErr)
if nargin == 2
   matchedPts = cagePts(lookupVector,:);
else % nargin == 3
   cagePts_noisy = cagePts + cageErr;
   matchedPts = cagePts_noisy(lookupVector,:);
end
% end GenerateMatchedPoints
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% Split_FP_CP
% ------------------------------------------------------------------------
% This function splits the calibration points into control points and fiducial
% points based on the tag name.
% ------------------------------------------------------------------------
function [FP_Tags,FP,CP_Tags,CP] = Split_FP_CP(calibTags, calibPts)

% String token
US = '_';

% Indentifiers
CP_IDer = RSAC('CP_Identifier');
FP_IDer = RSAC('FP_Identifier');

% initialize output
FP_Tags = {};
CP_Tags = {};
FP = [];
CP = [];

for i = 1:size(calibTags,1)
   tagName = ParseDelimited(calibTags{i},1,'STRING',US);
   if isempty(findstr(tagName,CP_IDer))
      % Fiducial Point
      FP_Tags = [FP_Tags; {tagName}];
      FP = [FP; calibPts(i,:)];
   else
      % Control Point
      CP_Tags = [CP_Tags; {tagName}];
      CP = [CP; calibPts(i,:)];
   end
end

% end Split_FP_CP
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% obsolete Get2DCoords
% ------------------------------------------------------------------------
% Parses a list and returns a matrix of the 2D coordinates.
% Assumes the list is delimited with DELIM from RSAC() and the first token is
% the tag name.
function [ptTags, Coords2D] = Get2DCoords(guiFig,listTag)

U = 2; V = 3;

[ptTags listPts] =  GetPtsFromList(guiFig,listTag);
Coords2D = [];
for i = 1:size(listPts,1)
   newPts = [ParseDelimited(listPts{i},[U V],'NUM')]';
   Coords2D = [Coords2D; newPts];
end
% end Get2DCoords
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% obsolete Get3DCoords
% ------------------------------------------------------------------------
% Parses a list and returns a matrix of the 3D coordinates.
% Assumes the list is delimited with DELIM from RSAC() and the first token is
% the tag name.
% ------------------------------------------------------------------------
function [ptTags, Coords3D] = Get3DCoords(guiFig,listTag)

X = 2; Y = 3; Z = 4;

[ptTags listPts] =  GetPtsFromList(guiFig,listTag);
Coords3D = [];
for i = 1:size(listPts,1)
```

```
   newPts = [ParseDelimited(listPts{i},[X Y Z],'NUM')]';
   Coords3D = [Coords3D; newPts];
end
% end Get3DCoords
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% WriteSimHeader
% ------------------------------------------------------------------------
% This function writes the header to the simulation results file.
function WriteSimHeader(FID,HeaderParams,ptTags,DLT,Selvik)

CMT = RSAC('CMT');
TRUE = RSAC('TRUE');
FALSE = RSAC('FALSE');

fprintf(FID,'%sRSA Simulation Ouput File\n',CMT);
fprintf(FID,'%sCreated %s\n\n',CMT,datestr(now));
fprintf(FID,'%sSimulation Parameters:\n',CMT);

FormatHeaderParams = char(HeaderParams{:,1});
for i=1:size(HeaderParams)
   fprintf(FID,'%s%s:\t%s\n',CMT,FormatHeaderParams(i,:),HeaderParams{i,2});
end
fprintf(FID,'\n\n');

% Write: Trial Number RMSE xRMSE yRMSE zRMSE ptRMSE
fprintf(FID,'TrialNumber');
if DLT == TRUE
   fprintf(FID,'\tRMSE_DLT\txRMSE_DLT\tyRMSE_DLT\tzRMSE_DLT\tmagPtE');
   % write magPtE
   for i = 1:size(ptTags,1)
      fprintf(FID,'\t%s',ptTags{i});
   end
end
if Selvik == TRUE

fprintf(FID,'\tRMSE_Selvik\txRMSE_Selvik\tyRMSE_Selvik\tzRMSE_Selvik\tmagPtE');
   % write magPtE
   for i = 1:size(ptTags,1)
      fprintf(FID,'\t%s',ptTags{i});
   end
end
fprintf(FID,'\n');
% end WriteSimHeader
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% ProgressFigure
% ------------------------------------------------------------------------
% This function displays the progress meter for the simulation.
% Create - creates the figure
%       - varargin{1} indicates the number of trials
%
% SetNumTrialsComplete - updates the number of trials completed
%                - checks RSAC('SimProgressInterval') to determine whether
%                  to show update
%
% SimComplete - displays indicator that the test is complete
%          - varargin{1} indicates the number of trials
%          - varargin{2} indicates the processing time
% ------------------------------------------------------------------------
function progressResult = ProgressFigure(Operation,varargin)

switch Operation
case 'Create'
   figHandle = figure;
   set(figHandle,'Tag','ProgressFig', ...
      'MenuBar','none', ...
      'Units','normalized', ...
      'Position',[0.4 0.4 0.2 0.15]);

   progressTitle = uicontrol(figHandle, 'Style', 'text', ...
         'Units','normalized', ...
         'Position',[0 0.8 1 0.2], ...
         'Tag', 'progressTitle', ...
         'String',['Simulation Progress']);
   numTrialsLabel = uicontrol(figHandle, 'Style', 'text', ...
         'Units','normalized', ...
         'Position',[0 0.5 0.6 0.2], ...
         'Tag', 'numTrialsLabel', ...
         'String',['Total Number of Trials: ']);
```

```
   numTrials = uicontrol(figHandle, 'Style', 'text', ...
        'Units','normalized', ...
        'Position',[0.7 0.5 0.5 0.2], ...
        'Tag', 'numTrials', ...
        'String',num2str(varargin{1}));
   trialsCompleteLabel = uicontrol(figHandle, 'Style', 'text', ...
        'Units','normalized', ...
        'Position',[0 0.3 0.6 0.2], ...
        'Tag', 'trialsCompleteLabel', ...
        'String',['Trials Completed: ']);
   trialsComplete = uicontrol(figHandle, 'Style', 'text', ...
        'Units','normalized', ...
        'Position',[0.7 0.3 0.5 0.2], ...
        'Tag', 'trialsComplete', ...
        'String','0');


case 'SetNumTrialsComplete'

   if mod(varargin{1},RSAC('SimProgressInterval')) == 0
      figHandle = findobj('Tag','ProgressFig');
      if not(isempty(figHandle))
         numTrials = findobj(figHandle,'Tag','trialsComplete');
         set(numTrials,'String',num2str(varargin{1}));
         drawnow;
      end
   end

case 'SimComplete'
   figHandle = findobj('Tag','ProgressFig');
   if not(isempty(figHandle))
      numTrials = findobj(figHandle,'Tag','trialsComplete');
      set(numTrials,'String',num2str(varargin{1}));
      testCompleteLabel = uicontrol(figHandle, 'Style', 'text', ...
         'Units','normalized', ...
         'Position',[0 0 0.7 0.2], ...
         'Tag', 'trialsComplete', ...
         'ForegroundColor', 'r', ...
         'String',['Sim Completed in ' num2str(varargin{2}) ' s']);
      okButton = uicontrol(figHandle, 'Style', 'push', ...
         'Units','normalized', ...
         'Position',[0.7 0 0.3 0.2], ...
         'Tag', 'okButton', ...
         'CallBack','RSAGUI_CB("Close_Progress") ', ...
         'String',['Close']);
   end
end

% end ProgressFigure
% ---------------------------------------------------------------------------

% ////////////////////////////////////////////////////////////////////////////
% End: Routines called from RunSim
% \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

% ---------------------------------------------------------------------------
% ReconOPs
% ---------------------------------------------------------------------------
% Reconstruct object points.
%   Uses 3D-DLT
function ReconOPs

TRUE=1;FALSE=0;
X=1;Y=2;Z=3;
DELIM = ',';

[curObj guiFig] = gcbo;

%   % determine analysis modes
%   radioHandle = findobj(guiFig,'Tag','Analyse_DLT');
%   Analyse_DLT = get(radioHandle,'Value');
%   radioHandle = findobj(guiFig,'Tag','Analyse_Selvik');
%   Analyse_Selvik = get(radioHandle,'Value');

% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% gather the points
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[fpTags FP] = GetPtsFromList(guiFig,'FP_list');
[cpTags CP] = GetPtsFromList(guiFig,'CP_list');
%   [opTags OP] = GetPtsFromList(guiFig,'OP_list');

[calibF1Tags calibF1] = GetPtsFromList(guiFig,'Calib_Film1_list');
```

```
[calibF2Tags calibF2] = GetPtsFromList(guiFig,'Calib_Film2_list');
[reconF1Tags reconF1] = GetPtsFromList(guiFig,'Recon_Film1_list');
[reconF2Tags reconF2] = GetPtsFromList(guiFig,'Recon_Film2_list');

% The purpose of this section is to account for points which do not show up
% in both films and to ensure that the ordering is the same.
% This code does not work properly because Split_FP_CP is restricted to
% the naming conventions used originally.
% It is up to the user to ensure that film1 and film2 points are properly
% matched.
%
%  % for Selvik's method, need to split control and fiducial points
%  [FP_F1_Tags FP_F1 CP_F1_Tags CP_F1] = Split_FP_CP(calibF1Tags, calibF1);
%  [FP_F2_Tags FP_F2 CP_F2_Tags CP_F2] = Split_FP_CP(calibF2Tags, calibF2);
%
%  % generate lookup vector of lab coordinates to each film point
%  FP_F1_LV = GenerateLookupVector(FP_F1_Tags,FP_F1,fpTags,FP);
%  FP_F2_LV = GenerateLookupVector(FP_F2_Tags,FP_F2,fpTags,FP);
%  CP_F1_LV = GenerateLookupVector(CP_F1_Tags,CP_F1,cpTags,CP);
%  CP_F2_LV = GenerateLookupVector(CP_F2_Tags,CP_F2,cpTags,CP);
%
%  % generate the list of matching lab coordinates to each film point
%  FP_F1_cage = FP(FP_F1_LV,:);
%  FP_F2_cage = FP(FP_F2_LV,:);
%  CP_F1_cage = CP(CP_F1_LV,:);
%  CP_F2_cage = CP(CP_F2_LV,:);
%
%  % for 3D DLT, combine control and fiducial points as calibration points
%  % measured points
%  calibF1_measured = [FP_F1;CP_F1];
%  calibF2_measured = [FP_F2;CP_F2];
%  % matched corresponding true cage points
%  calibF1_cage = [FP_F1_cage; CP_F1_cage];
%  calibF2_cage = [FP_F2_cage; CP_F2_cage];
%
%  % ---------------------------------------------------------------------------
%  % Reconstruction - 3D DLT
%  % ---------------------------------------------------------------------------
%  % 3D DLT transformation to image volume
%  [dlt_f1, res_f1] = dltfu(calibF1_cage,calibF1_measured);
%  [dlt_f2, res_f2] = dltfu(calibF2_cage,calibF2_measured);
%  DLT_Coeff = [dlt_f1, dlt_f2];

% ---------------------------------------------------------------------------
% Reconstruction - 3D DLT
% ---------------------------------------------------------------------------
% 3D DLT transformation to image volume
[dlt_f1, res_f1] = dltfu([FP;CP],calibF1);
[dlt_f2, res_f2] = dltfu([FP;CP],calibF2);
DLT_Coeff = [dlt_f1, dlt_f2];

objPts_Films = [reconF1, reconF2];
H = reconfu(DLT_Coeff,objPts_Films);
reconOP = H(:,1:3);

echo on;
% ** Warning! Potential Bug **
% This code lists out the OPs in order that they are solved
% They have NOT been checked for ordering after matching with calibration points.
% i.e. sequential ordering is assumed.
% Will be accounted for in next revision.
%  - Ask Anthony Choo
%
echo off;

% generate OP list
OPList = {};
for i = 1:size(reconOP,1)
   rowString = ['ROP' num2str(i) DELIM num2str(reconOP(i,X)) ...
        DELIM num2str(reconOP(i,Y)) DELIM num2str(reconOP(i,Z)) DELIM '0'
DELIM];
   OPList = [OPList; {rowString}];
end

% display the reconstructed object points
listHandle = findobj(guiFig,'Tag','OP_list');
set(listHandle,'String',OPList);

% end ReconOPs
% ---------------------------------------------------------------------------

function ReconOPs2
```

```
TRUE=1;FALSE=0;
X=1;Y=2;Z=3;
DELIM = ',';

[curObj guiFig] = gcbo;

%  % determine analysis modes
%  radioHandle = findobj(guiFig,'Tag','Analyse_DLT');
%  Analyse_DLT = get(radioHandle,'Value');
%  radioHandle = findobj(guiFig,'Tag','Analyse_Selvik');
%  Analyse_Selvik = get(radioHandle,'Value');

%----------------------------------------
% gather the points
%----------------------------------------
[fpTags FP] = GetPtsFromList(guiFig,'FP_list');
[cpTags CP] = GetPtsFromList(guiFig,'CP_list');
%  [opTags OP] = GetPtsFromList(guiFig,'OP_list');
cagePts = [FP;CP];

[calibF1Tags calibF1] = GetPtsFromList(guiFig,'Calib_Film1_list');
[calibF2Tags calibF2] = GetPtsFromList(guiFig,'Calib_Film2_list');
[reconF1Tags reconF1] = GetPtsFromList(guiFig,'Recon_Film1_list');
[reconF2Tags reconF2] = GetPtsFromList(guiFig,'Recon_Film2_list');

%-------------------------------------------------------------------------------
% Reconstruction - 3D DLT
%-------------------------------------------------------------------------------
% 3D DLT transformation to image volume
[dlt_f1, res_f1] = dltfu(cagePts, calibF1);
[dlt_f2, res_f2] = dltfu(cagePts, calibF2);
DLT_Coeff = [dlt_f1, dlt_f2];

objPts_Films = [reconF1, reconF2];
H = reconfu(DLT_Coeff,objPts_Films);
reconOP = H(:,1:3);

echo on;
% ** Warning! Potential Bug **
% This code lists out the OPs in order that they are solved
% They have NOT been checked for ordering after matching with calibration points.
% i.e. sequential ordering is assumed.
% Will be accounted for in next revision.
%  - Ask Anthony Choo
%
echo off;

% generate OP list
OPList = {};
for i = 1:size(reconOP,1)
   rowString = ['ROP' num2str(i) DELIM num2str(reconOP(i,X)) ...
        DELIM num2str(reconOP(i,Y)) DELIM num2str(reconOP(i,Z)) DELIM '0'
DELIM];
   OPList = [OPList; {rowString}];
end

% display the reconstructed object points
listHandle = findobj(guiFig,'Tag','OP_list');
set(listHandle,'String',OPList);

% end ReconOPs
%-------------------------------------------------------------------------------
```

B.2 Sim3DDLT

```
% Sim3DDLT  - simulates a 3D DLT operation
%
% [reconOP,magPtE,xyzRMSE,RMSE] = Sim3DDLT( ...
%                       CP_F1,CP_F1_err,CP_F1_cage, ...
%                       CP_F2,CP_F2_err,CP_F2_cage, ...
%                       OP_F1,OP_F1_err,OP_F2,OP_F2_err, ...
%                       OP)
%
% This function requires the KINMAT toolbox.
%
% Inputs:
%       CP_F1 - coordinates of the control points in film1 [u v; ...]
%       CP_F1_err - standard deviation of measurment error to add to CP_F1
%       CP_F1_cage - corresponding cage coords of CP_F1
%       CP_F2 - coordinates of the control points in film1 [u v; ...]
%       CP_F2_err - standard deviation of measurment error to add to CP_F2
%       CP_F2_cage - corresponding cage coords of CP_F2
%
%       OP_F1 - coordinates of the object points in film1 [u v; ...]
%       OP_F1_err - standard deviation of measurment error to add to OP_F1
%       OP_F2 - coordinates of the object points in film2 [u v; ...]
%       OP_F2_err - standard deviation of measurment error to add to OP_F2
%
%       OP - lab coordinates of object points [x y z; ...]
%
% Outputs:
%       reconOP - reconstructed object points after addition of noise to
%               calibration points (CPFilm1, CPFilm2) and
%               reconstruction points (OPFilm1, OPFilm2)
%       magPtE - magnitude of error vector for each point
%            - is an indicator of proper pairing of film points
%       xyzRSME - RMS error in x,y,z directions
%            - is an indicator calibration anisotropy
%       RMSE - RMS error of fitting = sqrt( (1/N)sum_i_1toN(xei^2+yei^2+zei^2) )
%          - comparison of one RSA setup versus another
%
%
% Notes:
%       Yuan X, Ryd L. J Biomech 2000 Apr;33(4):493-8
%          - uses zero mean, 0.02mm SD of measurement noise
%       Selvik G, Alberius P, Aronson AS. Acta Radiol [Diagn] (Stockh) 193;24(4):343-52
%          - measurement error better than 0.0065 mm
%       Karrholm J. Acta Orthop Scand 1989 Aug;60(4):491-503
%          - measurement error of Wild A8 measuring table is < 0.025 mm
%
% Revision History:
%       July 25, 2000 - Anthony Choo - initial version
%
function [reconOP,magPtE,xyzRMSE,RMSE] = Sim3DDLT( ...
                        CP_F1,CP_F1_err,CP_F1_cage, ...
                        CP_F2,CP_F2_err,CP_F2_cage, ...
                        OP_F1,OP_F1_err,OP_F2,OP_F2_err, ...
                        OP)

% ---------------------------------------------------------------------
% Add noise
% ---------------------------------------------------------------------
CP_F1_measured = CP_F1+(CP_F1_err*randn(size(CP_F1)));
CP_F2_measured = CP_F2+(CP_F2_err*randn(size(CP_F2)));
%
OP_F1_measured = OP_F1+(OP_F1_err*randn(size(OP_F1)));
OP_F2_measured = OP_F2+(OP_F2_err*randn(size(OP_F2)));

% ---------------------------------------------------------------------
% Reconstruction - 3D DLT
% ---------------------------------------------------------------------
% 3D DLT transformation to image volume
[dlt_f1, res_f1] = dltfu(CP_F1_cage,CP_F1_measured);
[dlt_f2, res_f2] = dltfu(CP_F2_cage,CP_F2_measured);
DLT_Coeff = [dlt_f1, dlt_f2];

objPts_Films = [OP_F1_measured, OP_F2_measured];
H = reconfu(DLT_Coeff,objPts_Films);
reconOP = H(:,1:3);

% Debug code begin %
% invCalibF1 = invdlt(dlt_film1,tempCalibPts)
% invCalibF2 = invdlt(dlt_film2,tempCalibPts)
% disp('calibration film1 film1 vs invdltfilm1')
```

```
% temp = [CPFilm1; zeros(size(invCalibF1,1)-size(CPFilm1,1),2)];
% [(1:size(temp,1))' temp invCalibF1]
% disp('calibration film2 film2 vs invdltfilm2')
% temp = [CPFilm2; zeros(size(invCalibF2,1)-size(CPFilm2,1),2)];
% [(1:size(temp,1))' temp invCalibF2]
%
% invOPF1 = invdlt(dlt_film1,OP);
% invOPF2 = invdlt(dlt_film2,OP);
% disp('object film1 vs invOPF1');
% [(1:size(OPFilm1,1))' OPFilm1 invOPF1]
% disp('object film2 vs invOPF2');
% [(1:size(OPFilm2,1))' OPFilm2 invOPF2]
% Debug end %

% ------------------------------------------------------------------------
% Errors
% ------------------------------------------------------------------------
nominalError = reconOP - OP;
squareError = nominalError.*nominalError;

% A/
% xRMSE = sqrt( (1/N) sum(xe^2) ), likewise for yRMSE and zRMSE
one_overN = 1/size(squareError,1);
xyzRMSE = sqrt(one_overN*sum(squareError,1));
% magnitude of RMSE = sqrt( xRMSE^2 + yRMSE^2 + zRMSE^2)
magRMSE = sqrt(sum(xyzRMSE.*xyzRMSE));

% B/
% magnitude of pt error = sqrt( xe^2 + ye^2 + ze^2 )
magPtE = sqrt(sum(squareError,2));
% RMS_magnitude_ptE = sqrt( (1/N) sum(magnitude_ptE^2) )
one_overN = 1/size(magPtE,1);
RMS_magPtE = sqrt(one_overN*sum(magPtE.*magPtE));

if abs(magRMSE - RMS_magPtE) < RSAC('magRMSE_RMSmagPtE_Tolerance')
   RMSE = magRMSE;
else
   disp('RMSE inconsistent');
   RMSE = [magRMSE RMS_magPtE];
end
```

B.3 SimSelvik

```
% SimSelvik - simulates Selvik's 3D reconstruction algorithm
%
% [reconOP,magPtE,xyzRMSE,RMSE] = SimSelvik( ...
%                   FP_F1, FP_F1_err, FP_F1_cage, ...
%                   FP_F2, FP_F2_err, FP_F2_cage, ...
%                   CP_F1, CP_F1_err, CP_F1_cage, ...
%                   CP_F2, CP_F2_err, CP_F2_cage, ...
%                   OP_F1, OP_F1_err, ...
%                   OP_F2, OP_F2_err, ...
%                   OP)
%
% This function requires the KINMAT toolbox and the Selvik library.
%
% Inputs:
%     FP_F1 - fiducial points in film 1 [u v;...]
%     FP_F1_err - standard deviation of measurement error to add to FP_F1
%     FP_F1_cage - corresponding lab coordinates of the points FP_F1 [x y 0; ... ]
%     FP_F2 - fiducial points in film 2 [u v;...]
%     FP_F2_err - standard deviation of measurement error to add to FP_F2
%     FP_F2_cage - corresponding lab coordinates of the points FP_F2 [x y 0; ... ]
%
%     CP_F1 - control points in film 1 [u v;...]
%     CP_F1_err - standard deviation of measurement error to add to CP_F1
%     CP_F1_cage - corresponding lab coordinates of the points CP_F1 [x y 0; ... ]
%     CP_F2 - control points in film 2 [u v;...]
%     CP_F2_err - standard deviation of measurement error to add to CP_F2
%     CP_F2_cage - corresponding lab coordinates of the points CP_F2 [x y 0; ... ]
%
%     OP_F1 - coordinates of the object points in film1 [u v;...]
%     OP_F1_err - standard deviation of measurement error to add to OP_F1
%     OP_F2 - coordinates of the object points in film2 [u v;...]
%     OP_F2_err - standard deviation of measurement error to add to OP_F2
%
%     OP - lab coordinates of object points [x y z; ...]
%
% Outputs:
%     reconOP - reconstructed object points after addition of noise to
```

```
%            fiducial pts, control pts and image of object pts
%     magPtE - magnitude of error vector for each point
%         - is an indicator of proper pairing of film points
%     xyzRSME - RMS error in x,y,z directions
%         - is an indicator of calibration anisotropy
%     RMSE - RMS error of fitting = sqrt( (1/N)sum_i_1toN(xei^2+yei^2+zei^2) )
%       - comparison of one RSA setup versus another
%
%
% Notes:
%     Typically,
%       FP_F1_err = FP_F2_err = CP_F1_err = CP_F2_err = OP_F1_err = OP_F2_err
%       since the same imaging and measurement system would normally be used for all.
%
%     Yuan X, Ryd L. J Biomech 2000 Apr;33(4):493-8
%       - uses zero mean, 0.02mm SD of measurement noise
%     Selvik G, Alberius P, Aronson AS. Acta Radiol [Diagn] (Stockh) 193;24(4):343-52
%       - measurement error better than 0.0065 mm
%     Karrholm J. Acta Orthop Scand 1989 Aug;60(4):491-503
%       - measurement error of Wild A8 measuring table is < 0.025 mm
%
% Revision History:
%     July 25, 2000 - Anthony Choo - initial version
%
function [reconOP,magPtE,xyzRMSE,RMSE] = SimSelvik( ...
                    FP_F1, FP_F1_err, FP_F1_cage, ...
                    FP_F2, FP_F2_err, FP_F2_cage, ...
                    CP_F1, CP_F1_err, CP_F1_cage, ...
                    CP_F2, CP_F2_err, CP_F2_cage, ...
                    OP_F1, OP_F1_err, ...
                    OP_F2, OP_F2_err, ...
                    OP)

x=1;y=2;z=3;

% ---------------------------------------------------------------------------
% Add noise
% ---------------------------------------------------------------------------
FP_F1_measured = FP_F1+(FP_F1_err*randn(size(FP_F1)));
FP_F2_measured = FP_F2+(FP_F2_err*randn(size(FP_F1)));
CP_F1_measured = CP_F1+(CP_F1_err*randn(size(CP_F1)));
CP_F2_measured = CP_F2+(CP_F2_err*randn(size(CP_F1)));
% Object
OP_F1_measured = OP_F1+(OP_F1_err*randn(size(OP_F1)));
OP_F2_measured = OP_F2+(OP_F2_err*randn(size(OP_F2)));

% ---------------------------------------------------------------------------
% Reconstruction - Selvik's Algorithm
% ---------------------------------------------------------------------------

% 2D DLT transformation to fiducial plane
[dlt2d_f1, res_f1] = dltfu2d(FP_F1_cage,FP_F1_measured);
[dlt2d_f2, res_f2] = dltfu2d(FP_F2_cage,FP_F2_measured);
DLT_Coeff = [dlt2d_f1, dlt2d_f2];

% Transform image of control points to fiducial plane
CP_F1_FPlane = reconfu2(dlt2d_f1,CP_F1_measured);
CP_F2_FPlane = reconfu2(dlt2d_f2,CP_F2_measured);
% Transform image of object points to fiducial plane
OP_F1_FPlane = reconfu2(dlt2d_f1,OP_F1_measured);
OP_F2_FPlane = reconfu2(dlt2d_f2,OP_F2_measured);

% solve for x-ray sources
%   - each ray has 2 pts (control pt on cage and control pt in fiducial plane)
%   - append column of zeros to make control points in Fiducial Plane 3D
source1Rays = [CP_F1_cage CP_F1_FPlane zeros(size(CP_F1_FPlane,1),1)];
source2Rays = [CP_F2_cage CP_F2_FPlane zeros(size(CP_F1_FPlane,1),1)];
source1_lsSolution = linesIntercept(source1Rays);
source2_lsSolution = linesIntercept(source2Rays);
source1 = source1_lsSolution(1:3)';
source2 = source2_lsSolution(1:3)';

% solve object points as intersection of rays from each film
reconOP = [];
for i = 1:size(OP_F1_FPlane)
  % add zero column to make Plane coordinates 3D
  OP_rays = [OP_F1_FPlane(i,:) 0 source1;
    OP_F2_FPlane(i,:) 0 source2];
  OP_lsSolution = linesIntercept(OP_rays);
  reconOP = [reconOP; OP_lsSolution(1:3)'];
end
```

```
% ------------------------------------------------------------------------
% Errors
% ------------------------------------------------------------------------

nominalError = reconOP - OP;
squareError = nominalError.*nominalError;

% A/
% xRMSE = sqrt( (1/N) sum(xe^2) ), likewise for yRMSE and zRMSE
one_overN = 1/size(squareError,1);
xyzRMSE = sqrt(one_overN*sum(squareError,1));
% magnitude of RMSE = sqrt( xRMSE^2 + yRMSE^2 + zRMSE^2)
magRMSE = sqrt(sum(xyzRMSE.*xyzRMSE));

% B/
% magnitude of pt error = sqrt( xe^2 + ye^2 + ze^2 )
magPtE = sqrt(sum(squareError,2));
% RMS_magnitude_ptE = sqrt( (1/N) sum(magnitude_ptE^2) )
one_overN = 1/size(magPtE,1);
RMS_magPtE = sqrt(one_overN*sum(magPtE.*magPtE));

if abs(magRMSE - RMS_magPtE) < RSAC('magRMSE_RMSmagPtE_Tolerance')
   RMSE = magRMSE;
else
   disp('RMSE inconsistent');
   RMSE = [magRMSE RMS_magPtE];
end
```

B.4 Script3DDLT

```
% Script3DDLT  - scipt processing for 3D DLT calibration, reconstruction,
%           and error analysis
%
% [reconOP,magPtE,xyzRMSE,RMSE] = Script3DDLT( ...
%                     CP_F1_measured,CP_F1_cage, ...
%                     CP_F2_measured,CP_F2_cage, ...
%                     OP_F1_measured,OP_F2_measured, ...
%                     OP)
%
% This function requires the KINMAT toolbox.
%
% Inputs:
%     CP_F1_measured - measured coordinates of the control points in film1 [u v; ...]
%     CP_F1_cage - corresponding cage coords of CP_F1
%     CP_F2_measured - measured coordinates of the control points in film1 [u v; ...]
%     CP_F2_cage - corresponding cage coords of CP_F2
%
%     OP_F1_measured - measured coordinates of the object points in film1 [u v; ...]
%     OP_F2_measured - measured coordinates of the object points in film2 [u v; ...]
%
%     OP - ideal lab coordinates of object points [x y z; ...]
%       - used in error calculations
%
% Outputs:
%     reconOP - reconstructed object points
%     magPtE - magnitude of error vector for each point
%           - is an indicator of proper pairing of film points
%     xyzRSME - RMS error in x,y,z directions
%           - is an indicator calibration anisotropy
%     RMSE - RMS error of fitting = sqrt( (1/N)sum_i_1toN(xei^2+yei^2+zei^2) )
%       - comparison of one RSA setup versus another
%
%
% Notes:
%     Yuan X, Ryd L. J Biomech 2000 Apr;33(4):493-8
%       - uses zero mean, 0.02mm SD of measurement noise
%     Selvik G, Alberius P, Aronson AS. Acta Radiol [Diagn] (Stockh) 193;24(4):343-52
%       - measurement error better than 0.0065 mm
%     Karrholm J. Acta Orthop Scand 1989 Aug;60(4):491-503
%       - measurement error of Wild A8 measuring table is < 0.025 mm
%
% Revision History:
%     July 25, 2000 - Anthony Choo - initial version
%
function [reconOP,magPtE,xyzRMSE,RMSE] = Script3DDLT( ...
                    CP_F1_measured,CP_F1_cage, ...
                    CP_F2_measured,CP_F2_cage, ...
                    OP_F1_measured,OP_F2_measured, ...
                    OP)
```

```
% ------------------------------------------------------------------------
% Reconstruction - 3D DLT
% ------------------------------------------------------------------------
% 3D DLT transformation to image volume
[dlt_f1, res_f1] = dltfu(CP_F1_cage,CP_F1_measured);
[dlt_f2, res_f2] = dltfu(CP_F2_cage,CP_F2_measured);
DLT_Coeff = [dlt_f1, dlt_f2];

objPts_Films = [OP_F1_measured, OP_F2_measured];
H = reconfu(DLT_Coeff,objPts_Films);
reconOP = H(:,1:3);

% ------------------------------------------------------------------------
% Errors
% ------------------------------------------------------------------------
nominalError = reconOP - OP;
squareError = nominalError.*nominalError;

% A/
% xRMSE = sqrt( (1/N) sum(xe^2) ), likewise for yRMSE and zRMSE
one_overN = 1/size(squareError,1);
xyzRMSE = sqrt(one_overN*sum(squareError,1));
% magnitude of RMSE = sqrt( xRMSE^2 + yRMSE^2 + zRMSE^2)
magRMSE = sqrt(sum(xyzRMSE.*xyzRMSE));

% B/
% magnitude of pt error = sqrt( xe^2 + ye^2 + ze^2 )
magPtE = sqrt(sum(squareError,2));
% RMS_magnitude_ptE = sqrt( (1/N) sum(magnitude_ptE^2) )
one_overN = 1/size(magPtE,1);
RMS_magPtE = sqrt(one_overN*sum(magPtE.*magPtE));

if abs(magRMSE - RMS_magPtE) < RSAC('magRMSE_RMSmagPtE_Tolerance')
   RMSE = magRMSE;
else
   disp('RMSE inconsistent');
   RMSE = [magRMSE RMS_magPtE];
end
```

B.5 ScriptSelvik

```
% ScriptSelvik  - script to perform calibration, reconstruction
%          and error analysis using Selvik's 3D reconstruction algorithm
%
% [reconOP,magPtE,xyzRMSE,RMSE] = ScriptSelvik( ...
%                  FP_F1_measured, FP_F1_cage, ...
%                  FP_F2_measured, FP_F2_cage, ...
%                  CP_F1_measured, CP_F1_cage, ...
%                  CP_F2_measured, CP_F2_cage, ...
%                  OP_F1_measured, OP_F2_measured, ...
%                  OP)
%
% This function requires the KINMAT toolbox and the Selvik library.
%
% Inputs:
%     FP_F1_measured - measured fiducial points in film 1 [u v;...]
%     FP_F1_cage - corresponding lab coordinates of the points FP_F1 [x y 0; ... ]
%     FP_F2_measured - measured fiducial points in film 2 [u v;...]
%     FP_F2_cage - corresponding lab coordinates of the points FP_F2 [x y 0; ... ]
%
%     CP_F1_measured - measured control points in film 1 [u v;...]
%     CP_F1_cage - corresponding lab coordinates of the points CP_F1 [x y 0; ... ]
%     CP_F2_measured - measured control points in film 2 [u v;...]
%     CP_F2_cage - corresponding lab coordinates of the points CP_F2 [x y 0; ... ]
%
%     OP_F1_measured - measured coordinates of the object points in film1 [u v;...]
%     OP_F2_measured - measured coordinates of the object points in film2 [u v;...]
%
%     OP - ideal lab coordinates of object points [x y z; ...]
%       - used for error analysis
%
% Outputs:
%     reconOP - reconstructed object points
%     magPtE - magnitude of error vector for each point
%          - is an indicator of proper pairing of film points
%     xyzRSME - RMS error in x,y,z directions
%          - is an indicator of calibration anisotropy
%     RMSE - RMS error of fitting = sqrt( (1/N)sum_i_1toN(xei^2+yei^2+zei^2) )
%          - comparison of one RSA setup versus another
```

```
%
%
% Notes:
%     Typically,
%        FP_F1_err = FP_F2_err = CP_F1_err = CP_F2_err = OP_F1_err = OP_F2_err
%        since the same imaging and measurement system would normally be used for all.
%
%     Yuan X, Ryd L. J Biomech 2000 Apr;33(4):493-8
%        - uses zero mean, 0.02mm SD of measurement noise
%     Selvik G, Alberius P, Aronson AS. Acta Radiol [Diagn] (Stockh) 193;24(4):343-52
%        - measurement error better than 0.0065 mm
%     Karrholm J. Acta Orthop Scand 1989 Aug;60(4):491-503
%        - measurement error of Wild A8 measuring table is < 0.025 mm
%
% Revision History:
%     July 25, 2000 - Anthony Choo - initial version
%
function [reconOP,magPtE,xyzRMSE,RMSE] = ScriptSelvik( ...
                    FP_F1_measured, FP_F1_cage, ...
                    FP_F2_measured, FP_F2_cage, ...
                    CP_F1_measured, CP_F1_cage, ...
                    CP_F2_measured, CP_F2_cage, ...
                    OP_F1_measured, OP_F2_measured, ...
                    OP)

% ------------------------------------------------------------------------
% Reconstruction - Selvik's Algorithm
% ------------------------------------------------------------------------
% planarize the fiducial coordinates
[FP_F1_cage_plane FP_F1_dim] = RemoveThirdDimension(FP_F1_cage);
[FP_F2_cage_plane FP_F2_dim] = RemoveThirdDimension(FP_F2_cage);

% 2D DLT transformation to fiducial plane
[dlt2d_f1, res_f1] = dltfu2d(FP_F1_cage_plane,FP_F1_measured);
[dlt2d_f2, res_f2] = dltfu2d(FP_F2_cage_plane,FP_F2_measured);
DLT_Coeff = [dlt2d_f1, dlt2d_f2];

% Transform image of control points to fiducial plane
CP_F1_FPlane = reconfu2(dlt2d_f1,CP_F1_measured);
CP_F2_FPlane = reconfu2(dlt2d_f2,CP_F2_measured);
% Transform image of object points to fiducial plane
OP_F1_FPlane = reconfu2(dlt2d_f1,OP_F1_measured);
OP_F2_FPlane = reconfu2(dlt2d_f2,OP_F2_measured);

% add third dimension to coordinates in the fiducial planes
CP_F1_FPlane3D = AddThirdDimension(CP_F1_FPlane,FP_F1_dim);
CP_F2_FPlane3D = AddThirdDimension(CP_F2_FPlane,FP_F2_dim);
OP_F1_FPlane3D = AddThirdDimension(OP_F1_FPlane,FP_F1_dim);
OP_F2_FPlane3D = AddThirdDimension(OP_F2_FPlane,FP_F2_dim);

% solve for x-ray sources
%   - each ray has 2 pts (control pt on cage and control pt in fiducial plane)
%   - append column of zeros to make control points in Fiducial Plane 3D
source1Rays = [CP_F1_cage CP_F1_FPlane3D];
source2Rays = [CP_F2_cage CP_F2_FPlane3D];
source1_lsSolution = linesIntercept(source1Rays);
source2_lsSolution = linesIntercept(source2Rays);
source1 = source1_lsSolution(1:3)';
source2 = source2_lsSolution(1:3)';

% solve object points as intersection of rays from each film
reconOP = [];
for i = 1:size(OP_F1_FPlane3D,1)
   % add zero column to make Plane coordinates 3D
   OP_rays = [OP_F1_FPlane3D(i,:) source1;
      OP_F2_FPlane3D(i,:) source2];
   OP_lsSolution = linesIntercept(OP_rays);
   reconOP = [reconOP; OP_lsSolution(1:3)'];
end

% ------------------------------------------------------------------------
% Errors
% ------------------------------------------------------------------------

nominalError = reconOP - OP;
squareError = nominalError.*nominalError;

% A/
% xRMSE = sqrt( (1/N) sum(xe^2) ), likewise for yRMSE and zRMSE
one_overN = 1/size(squareError,1);
xyzRMSE = sqrt(one_overN*sum(squareError,1));
% magnitude of RMSE = sqrt( xRMSE^2 + yRMSE^2 + zRMSE^2)
```

```
magRMSE = sqrt(sum(xyzRMSE.*xyzRMSE));

% B/
% magnitude of pt error = sqrt( xe^2 + ye^2 + ze^2 )
magPtE = sqrt(sum(squareError,2));
% RMS_magnitude_ptE = sqrt( (1/N) sum(magnitude_ptE^2) )
one_overN = 1/size(magPtE,1);
RMS_magPtE = sqrt(one_overN*sum(magPtE.*magPtE));

if abs(magRMSE - RMS_magPtE) < RSAC('magRMSE_RMSmagPtE_Tolerance')
   RMSE = magRMSE;
else
   disp('RMSE inconsistent');
   RMSE = [magRMSE RMS_magPtE];
end

% ------------------------------------------------------------------------
% RemoveThirdDimension
% ------------------------------------------------------------------------
% This function takes points that lie in the XY,XZ or YZ plane and removes the
% approprate coordinate (Z,Y or X).
%
% Function will always return a result - when confused, will print an
% error message and remove the z column.
%
% Input:
%      Coords3D - [x,y,z;...]
%
% Ouput:
%      Coords2D - Coords3D with appropriate column removed
%   ThirdD - [dimension_removed mean_value_of_coordinate_removed]
%
% ------------------------------------------------------------------------
function [Coords2D, ThirdD] = RemoveThirdDimension(Coords3D)

TRUE = RSAC('TRUE'); FALSE = RSAC('FALSE');

xNormToPlane = FALSE;
yNormToPlane = FALSE;
zNormToPlane = FALSE;

xyzSTD = std(Coords3D,0,1);
xSTD = xyzSTD(1);
ySTD = xyzSTD(2);
zSTD = xyzSTD(3);

meanCoords3D = mean(Coords3D,1);

if xSTD < ySTD & xSTD < zSTD
   xNormToPlane = TRUE;
elseif ySTD < xSTD & ySTD < zSTD
   yNormToPlane = TRUE;
elseif zSTD < xSTD & zSTD < ySTD
   zNormToPlane = TRUE;
else
   disp('Error: Selvik method requires fiducial and control points to lie in a plane.')
   disp('     : Unable to determine plane of fiducial/control points - assuming xy');
   zNormToPlane = TRUE;
end

if xNormToPlane == TRUE
   Coords2D = [Coords3D(:,2) Coords3D(:,3)];
   ThirdD = [1 meanCoords3D(1)];
elseif yNormToPlane == TRUE
   Coords2D = [Coords3D(:,1) Coords3D(:,3)];
   ThirdD = [2 meanCoords3D(2)];
elseif zNormToPlane == TRUE
   Coords2D = [Coords3D(:,1) Coords3D(:,2)];
   ThirdD = [3 meanCoords3D(3)];
end

% end RemoveThirdDimension
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% AddThirdDimension
% ------------------------------------------------------------------------
% This function adds a third dimension column to a set of planar points.
% Input:
%      Coords2D - [u,v;...] coordinates
%   ThirdD - [dimension_to_add value_of_coordinate_to_add]
%
```

```
% Ouput:
%     Coords3D - Coords2D with column added of value_of_coordiante_to_add
%             in dimension_to_add position
%
% --------------------------------------------------------------------
function Coords3D = AddThirdDimension(Coords2D,ThirdD)

colVector = [];
colVector(1:size(Coords2D,1),1) = ThirdD(2);

if ThirdD(1) == 1
   Coords3D = [colVector Coords2D];
elseif ThirdD(1) == 2
   Coords3D = [Coords2D(:,1) colVector Coords2D(:,2)];
else
   Coords3D = [Coords2D colVector];
end
% end AddThirdDimension
% --------------------------------------------------------------------
```

B.6 Utilities

B.6.1 RSAC

```
% RSAC     - constants for RSA toolbox
%
% Constant_ID's
%    'TRUE'
%    'FALSE'
%
%    'FP_List'
%    'CP_List'
%    'CB_List'
%    'OP_List'
%    'Calib_Film1_list'
%    'Calib_Film2_list'
%    'Recon_Film1_list'
%    'Recon_Film2_list'
%
%    'cageCOLOUR' % Calibration cage colour
%    'fpCOLOUR'   % Fiducial point colour
%    'fpSYMBOL'   % Fiducial point symbol
%    'cpCOLOUR'   % Control point colour
%    'cpSYMBOL'   % Control point symbol
%
%    'f1COLOUR'   % frame 1 colour
%    'f1SYMBOL'   % frame 1 symbol
%    'f1CALIBCOLOUR'  % frame 1 calib pts colour
%    'f1CALIBSYMBOL'  % frame 1 calib pts symbol
%    'f1RECONCOLOUR'  % frame 1 recon pts colour
%    'f1RECONSYMBOL'  % frame 1 recon pts symbol
%
%    'f2COLOUR'   % frame 2 colour
%    'f2SYMBOL'   % frame 2 symbol
%    'f2CALIBCOLOUR'  % frame 2 calib pts colour
%    'f2CALIBSYMBOL'  % frame 2 calib pts symbol
%    'f2RECONCOLOUR'  % frame 2 recon pts colour
%    'f2RECONSYMBOL'  % frame 2 recon pts symbol
%
%    'opCOLOUR'   % object point colour
%    'opSYMBOL'   % object symbols
%
%    'CP_Identifier'  % string token which indicates a control point
%    'FP_Identifier'  % string token which indicates a fiducial point
%
%    'DELIM'      % string delimit symbol
%    'CMT'    % comment prefix
%    'ONE'
%    'ALL'
%
%    '2D'
%    '3D'
%
%    'CHECKED'
%    'UNCHECKED'
%
%    'SimProgressInterval' % trial interval to update the simulation progress meter
%
%    'SLIDER_STEP'
```

```
%    'TROUGH_STEP'
%
%    'magRMSE_RMSmagPtE_Tolerance' - tolerance for detecting a difference
%                                    in RMSE calculation
%
%    UserData format is [{string tag1} {data1}; {string tag2} {data2}; ...]
%    'UD_RSAFIG'  % RSA setup figure identifier in user data
%    'UD_RSAAXES'     % RSA setup figure axes identifier in user data
%    'UD_GUIFIG'     % GUI figure identifier in user data
%
function [Output_Constant] = RSAC(Constant_ID)

switch Constant_ID

case 'TRUE'
   OC = 1;
case 'FALSE'
   OC = 0;

case {'FP_List', ...
   'CP_List', ...
   'CB_List', ...
   'OP_List', ...
   'Calib_Film1_list', ...
   'Calib_Film2_list', ...
   'Recon_Film1_list', ...
   'Recon_Film2_list'}
   OC = Constant_ID

case 'cageCOLOUR'
   OC = 'g';
case 'fpCOLOUR'
   OC = 'y';
case 'fpSYMBOL'
   OC = 'o';
case 'cpCOLOUR'
   OC = 'y';
case 'cpSYMBOL'
   OC = '*';
case 'f1COLOUR'
   OC = 'r';

case 'f1SYMBOL'
   OC = '*';
case 'f1CALIBCOLOUR'
   OC = 'r';
case 'f1CALIBSYMBOL'
   OC = '*';
case 'f1RECONCOLOUR'
   OC = 'r';
case 'f1RECONSYMBOL'
   OC = 'o';

case 'f2COLOUR'
   OC = 'b';
case 'f2SYMBOL'
   OC = '*';
case 'f2CALIBCOLOUR'
   OC = 'b';
case 'f2CALIBSYMBOL'
   OC = '*';
case 'f2RECONCOLOUR'
   OC = 'b';
case 'f2RECONSYMBOL'
   OC = 'o';

case 'opCOLOUR'
   OC = 'w';
case 'opSYMBOL'
   OC = 'o';

case 'CP_Identifier'
   OC = 'CP';
case 'FP_Identifier'
   OC = 'FP';

case 'DELIM'
   OC = ',';
case 'CMT'
   OC = '% ';
case {'ONE'}
   OC = 1;
case {'ALL'}
```

```
   OC = 2;
case 'LABEL_OS'
   OC = 10;

case '2D'
   OC = 2;
case '3D'
   OC = 3;

case 'CHECKED'
   OC = 1;
case 'UNCHECKED'
   OC = 0;

case 'SimProgressInterval'
   OC = 10;

case 'SLIDER_STEP'
   OC = 1/360;
case 'TROUGH_STEP'
   OC = 10/360;

case 'magRMSE_RMSmagPtE_Tolerance'
   OC = 0.0001;

case 'UD_RSAFIG'
   OC = 'RSAFIG';
case 'UD_RSAXES'
   OC = 'RSAXES';

case 'UD_GUIFIG'
   OC = 'GUIFIG';

case otherwise
   OC = [];
end
Output_Constant = OC;
```

B.6.2 Az_cb

```
% AZ_cb- callback routine for azimuth slide control
%- similar to AzCallBack with additional input parameter axesTag
%
% AZ_cb(slideTag,curTag,axesTag)
%
% This function is a callback for the azimuth slider uicontrol
%
% Input:
%     slideTag - string indicating the slider tag name
%     curTag - string indicating the current value indicator tag name
%axesTag - tag to axes for which slider refers to
%
% Revision History:
%     August 11, 2000 - Anthony Choo - initial version
%
function AZ_cb(slideTag,curTag,axesTag)

vw = get(gca,'View');

[curObj curFig] = gcbo;

sliderHandle = findobj(curFig,'Tag',slideTag);
currentHandle = findobj(curFig,'Tag',curTag);
axesHandle = findobj(curFig,'Tag',axesTag);

str = num2str(get(sliderHandle,'Value'));
newview = [get(sliderHandle,'Value') vw(2)];
set(currentHandle,'String',str)
set(axesHandle,'View',newview)
```

B.6.3 El_cb

```
% EL_cb - callback routine for elevation slide control
%  - similar to ElCallBack with additional input parameter axesTag
%
% EL_cb(slideTag,curTag)
%
% This function is a callback for the elevation slider uicontrol
```

```
%
% Input:
%     slideTag - string indicating the slider tag name
%     curTag - string indicating the current value indicator tag name
%  axesTag - tag to axes for which slider refers to
%
% Revision History:
%     August 11, 2000 - Anthony Choo - initial version
%
function EL_cb(slideTag,curTag,axesTag)

vw = get(gca,'View');

[curObj curFig] = gcbo;
sliderHandle = findobj(curFig,'Tag',slideTag);
currentHandle = findobj(curFig,'Tag',curTag);
axesHandle = findobj(curFig,'Tag',axesTag);

str = num2str(get(sliderHandle,'Value'));
newview = [vw(1) get(sliderHandle,'Value')];
set(currentHandle,'String',str)
set(axesHandle,'View',newview)
```

B.6.4 AddPts2D

```
% AddPoints2D  - add a set of points and labels to a 2D graph
%
% [ptsTags, labelsTags] = AddPoints2D (axesHandle, pts, colour, symbol, labelOS,
tagName)
%
% Inputs:
%     axesHandle - figure to add points to
%     pts - [x1,y1;x2,y2;...] points to add
%     colour - colour of points and text labels
%     symbol - symbol for points
%     labelOS - offset for label to be placed from point
%     tagName - an nx1 cell array of strings
%
% Outputs:
%     ptsTags - cell array of string tag names for each point
%     labelsTags - cell array of tag names for each point's label
%           - same as ptsTags with 'Txt' suffix
%
% Revision History:
%     June 28, 2000 - Anthony Choo - initial version
%
function [ptsTags, labelsTags] = AddPoints2D (axesHandle, pts, colour, symbol, ...
          labelOS, tagName)


x=1;y=2;
TRUE = 1;
FALSE = 0;

axes(axesHandle);
hold on;

ptsTags = {};
labelsTags = {};
for i = 1:size(pts,1)
  % Plot point
  ptsHandle = plot(pts(i,x),pts(i,y),[colour symbol]);

  % determine tags and plot labels
  ptTagName = tagName{i};
  % Plot label
  txtHandle = text(pts(i,x)+labelOS,pts(i,y)+labelOS, ...
    ptTagName,'Color',colour);
  labelTagName = [ptTagName 'Txt'];

  % set tags
  set(ptsHandle,'Tag',ptTagName);
  ptsTags = [ptsTags; {ptTagName}];

  set(txtHandle,'Tag',labelTagName);
  labelsTags = [labelsTags; {labelTagName}];

  hold on;
end
```

B.6.5 AddPts3D

```
% AddPts3D  - add a set of points and labels to a 3D graph
%
% [ptsTags, labelsTags] = AddPts3D (axesHandle, pts, colour, symbol, labelOS, tagName)
%
% Requires ParseDelimited function in Utils library.
%
% Inputs:
%       axesHandle - axes to add points to
%       pts - [x1,y1,z1;x2,y2,z2;...] points to add
%       colour - single value or column vector of colour of points and text labels
%       symbol - single value or column vector of symbol for points
%       labelOS - offset for label to be placed from point
%       tagName - cell array of string tags attached to each point
%
% Outputs:
%       ptsTags - cell array of string tag names for each point
%       labelsTags - cell array of string of tag names for point labels
%               - same as ptsTags with "Txt' appended
%
% Revision History:
%       June 29, 2000 - Anthony Choo - initial verison
%
function [ptsTags, labelsTags] = AddPts3D (axesHandle, pts, colour, symbol, labelOS,
tagName)

x=1;y=2;z=3;
TRUE = 1;
FALSE = 0;

if size(colour,2) == 1
   colour(1:size(pts,1)) = colour;
end
if size(symbol,2) == 1
   symbol(1:size(pts,1)) = symbol;
end

axes(axesHandle);

ptsTags = {};
labelsTags = {};
for i = 1: size(pts,1)
   % Plot point
   ptsHandle = plot3(pts(i,x),pts(i,y),pts(i,z),[colour(i) symbol(i)]);

   % determine tags and plot labels
   ptTagName = tagName{i};
   % Plot label
   txtHandle = text(pts(i,x)+labelOS,pts(i,y)+labelOS,pts(i,z)+labelOS, ...
      ptTagName,'Color',colour(i));
   labelTagName = [ptTagName 'Txt'];

   % set tags
   set(ptsHandle,'Tag',ptTagName);
   ptsTags = [ptsTags {ptTagName}];

   set(txtHandle,'Tag',labelTagName);
   labelsTags = [labelsTags {labelTagName}];

   hold on;
end
```

B.6.6 AddLns3D

```
% AddLns3D  - add a set of lines and optional labels to a 3D graph
%
% [linesTags, labelsTags] = AddLns3D (axesHandle, lineStart, lineEnd, colour, tagName,
labelOS)
%
% Usage:
%
% [linesTags, labelsTags] = AddLns3D (axesHandle, lineStart, lineEnd, colour, tagName,
labelOS)
%   - adds lines with labels
% [linesTags] = AddLns3D (axesHandle, lineStart, lineEnd, colour, tagName)
%   - adds lines without labels
%
% Inputs:
```

```
%       axesHandle - figure to add points to
%       lineStart - start point for the lines [x,y,z] or [x1,y1,z1;x2,y2,z2;...]
%               - if only one point is given, all lineEnd will be drawn from this point
%               - else one line will be drawn for each row from [lineStart; lineEnd]
%       lineEnd - end point for the lines [x1,y1,z1;x2,y2,z2;...]
%       colour - colour of lines and text labels
%       tagName - cell array of tag names for each line
%       labelOS - offset for label to be placed from line destination (x2,y2,z2)
%
% Outputs:
%       linesTags - equal to tagName
%       labelsTags - equal to tag name with 'Txt' appended
%
% Revision History:
%       July 4, 2000 - Anthony Choo - initial version
%
function [linesTags, labelsTags] = AddLns3D (axesHandle, lineStart, lineEnd, colour,
tagName, labelOS)

% Constants
TRUE = 1;
FALSE = 0;
x=1;y=2;z=3;

% Check if Labelling is ON or OFF
% set default DELIM if needed
if nargout == 1
   LabelsOn = FALSE;
else
   LabelsOn = TRUE;
end

axes(axesHandle);

linesTags = {};
labelsTags = {};
for i = 1:size(lineEnd,1)
   % Plot line
   if size(lineStart,1) == 1
      % all lines from this origin
      lx = [lineStart(1,x),lineEnd(i,x)];
      ly = [lineStart(1,y),lineEnd(i,y)];
      lz = [lineStart(1,z),lineEnd(i,z)];
   else
      % one line per row
      lx = [lineStart(i,x),lineEnd(i,x)];
      ly = [lineStart(i,y),lineEnd(i,y)];
      lz = [lineStart(i,z),lineEnd(i,z)];
   end
   lineHandle = line(lx,ly,lz);
   set(lineHandle,'Color',colour);

   % Set line tag
   lineTagName = tagName{i};
   set(lineHandle,'Tag',lineTagName);
   linesTags = [linesTags; {lineTagName}];

   % Plot labels and Set labels tags
   if LabelsOn == TRUE
      % Plot label
      % lineTagName already found above
      txtHandle = text(lineEnd(i,x)+labelOS,lineEnd(i,y)+labelOS,lineEnd(i,z)+labelOS, ...
         lineTagName,'Color',colour);
      labelTagName = [lineTagName 'Txt'];
      set(txtHandle,'Tag',labelTagName);
      labelsTags = [labelsTags; {labelTagName}];
   end

   hold on;
end
```

B.6.7 AddPatch3D

```
% AddPatch3D    - add one or more patches to the figure
%
% [patchTags, labelTags] = AddPatch3D (axesHandle, corners, colour, labelOS, tagNames)
%
% Inputs:
%       axesHandle - axes to add patch to
%       corners - ({patch1};{patch2} ..) where,
```

```
%            patch1 = [x1,y1,z1;x2,y2,z2;...] for 4 corners of the patch
%      colour - cell array of colour of patch edge and text labels
%      labelOS - offset for label to be placed from patch origin
%      tagName - string tags attached to each patch origin
%
% Outputs:
%      patchTag - cell array of string tag names for each patch
%      labelTag - cell array of string of tag names for patch labels
%               - same as patchTags with "Txt' appended
%
% Revision History:
%      June 30, 2000 - Anthony Choo - initial verison
%      July 5, 2000 - Anthony Choo - change figureHandle to axesHandle
%
function [patchTags, labelTags] = AddPatch3D (axesHandle, corners, colour, labelOS,
tagNames)

x=1;y=2;z=3;

axes(axesHandle);

patchTags = tagNames;
labelTags = {};
for i = 1:size(tagNames,1)
   patchHandle = patch(corners{i}(:,x), corners{i}(:,y), corners{i}(:,z), 'k');
   set(patchHandle, 'EdgeColor', colour{i});
   set(patchHandle, 'Tag' , tagNames{i});

   txtHandle = text(corners{i}(1,x)+labelOS, corners{i}(1,y)+labelOS, ...
         corners{i}(1,z)+labelOS, tagNames{i});
   set(txtHandle,'Color',colour{i});
   set(txtHandle, 'Tag', [tagNames{i} 'Txt']);
   labelTags = [labelTags {[tagNames{i} 'Txt']}];
end
```

B.6.8 Delimit

```
% Delimit   - concatenate a column of numbers with a delimiter
%
% [delimitedString] = Delimit (nums, DELIM)
%
% Input:
%      nums - column of numbers to concatenate
%      DELIM - optional delimiter in concatenation
%            note that delimitedString will be terminated with
%            the delimiter as well
%
% Output:
%      delimitedString - numbers concatenated into a single row
%
% Revision History:
%      June 14, 2000 - Anthony Choo - initial version
%
function [delimitedString] = Delimit (nums, DELIM)

delimitedString = [];
if nargin == 1
   DELIM = [];
end
for i = 1:size(nums,1)
   delimitedString = [delimitedString num2str(nums(i)) DELIM];
end
```

B.6.9 ParseDelimited

```
% ParseDelimited    - returns a set of components in a delimited string
%
% [iComps] = ParseDelimited(delimitedString, i, iType, DELIM)
%
% Inputs:
%      delimitedString - string delimited by DELIM
%      i - positions of strings to extract
%      iType - optional type: 'STRING' or 'NUM', 'STRING' is default
%      DELIM - optional delimiter, [','] is default
%
% Outputs:
%      iComps - column of strings or numbers returned
%
```

```
% Example:
%       s = ['this,is,a,test,'];
%       ParseDelimited(s,[4,2]) returns ['test';'is  ']
%
% Revision History:
%       June 14, 2000 - Anthony Choo - initial version
%       June 27, 2000 - modified to return multiple strings and numbers
%
function [iComps] = ParseDelimited(delimitedString, i, iType, DELIM)

if nargin == 2
   iType = 'STRING';
   DELIM = ',';
elseif nargin == 3
   DELIM = ',';
end

% check that string is terminated with DELIM
LAST = size(delimitedString,2);
if strcmp(delimitedString(LAST:LAST),DELIM) ~= 1
   delimitedString = [delimitedString DELIM];
end

delimAt = findstr(delimitedString,DELIM);

iComps = [];
for j = 1:size(i,2)
   if i(j) == 1
      ithComp = delimitedString(1:delimAt(i(j))-1);
   else
      ithComp = delimitedString(delimAt(i(j)-1)+1:delimAt(i(j))-1);
   end
   if strcmp(iType,'STRING')
      iComps = strvcat(iComps,ithComp);
   else
      iComps = [iComps; str2num(ithComp(1,:))];
   end
end
```

B.6.10 MultHomogeneous

```
% MultHomogeneous   - Multiply series of points by a homogeneous transformation
%
% Pts_1 = MultHomogeneous (Pts_2, T12)
%
% Inputs:
%       Pts_2 - [x,y,z;...] points in frame 2
%             - {[x,y,z];...] points in frame 2
%
%       T12 - homogeneous transformation from frame 2 to frame 1
%
% Outputs:
%       Pts_1 - Pts_2 transformed into frame 1
%
% Revision History:
%       June 9, 2000 - Anthony Choo - Initial Revision
%       June 30, 2000 - Anthony Choo - cell handling added
%
function Pts_1 = MultHomogeneous (Pts_2, T12)

if iscell(Pts_2)
   Pts_1 = {};
   for i = 1:size(Pts_2,1)
      newPts = T12*[Pts_2{i} 1]';
      Pts_1 = [Pts_1; {newPts(1:3)'}];
   end
else
   Pts_1 = [];
   for i = 1:size(Pts_2,1)
      newPts = T12*[Pts_2(i,:)' 1];
      Pts_1 = [Pts_1; newPts(1:3)'];
   end
end
```

B.6.11 Xpose

```
% Xpose     - projects a set of 3D points from a source onto a 2D film
%
```

```
% [filmPts, filmTags] = Xpose(pts3D, sourcePt, filmDim, ptTags, filmSuffix)
%
% Warning:
%       The function assumes correct usage and does not check that the
%       points are between the source and the xy-plane.
%       If the source is between the point and the xy-plane, the function
%       will project the point onto the xy-plane along the line joining
%       the point and the source
%
% Inputs:
%       pts3D - [x1,y1,z1; x2,y2,z2; ...] of points
%               - points are in the coordinate frame of the film
%                 with the film parallel to the xy-plane
%       sourcePt - [x,y,z] coordinate of the x-ray source in the film frame
%       filmDim - optional dimensions of the film (u,v)
%               - if specified, points exposed outside the boundaries of
%                 the film will be removed from the filmPts returned
%               - used with ptsTags
%       ptsTags - optional string tags for the points
%               - allows identification of points removed
%               - used with filmDim
%       filmSuffix - string suffix that is appened to ptsTags to form filmTags
%               - used with filmDim
%
% Outputs:
%       filmPts - (u,v) coordinates of each of the points in 3Dpts projected
%                 onto the xy-plane
%       filmTags - string tags of original points with filmSuffix appended
%               - empty if filmDim and ptsTags is not specified
%
% Revision History:
%       May 31, 2000 - Anthony Choo - initial revision
%       June 27, 2000 - Anthony Choo - filmDim,ptTags,filmSuffix and filmTags added
%
function [filmPts, filmTags] = Xpose(pts3D, sourcePt, filmDim, ptTags, filmSuffix)

TRUE = 1;
FALSE = 0;
if nargin == 5
  CHECK_BOUNDARIES = TRUE;
else
  CHECK_BOUNDARIES = FALSE;
end

filmPts = [];
for i = 1:size(pts3D,1)
  dirVec = sourcePt - pts3D(i,:);
  [u, v] = PlaneIntercept(pts3D(i,:), dirVec, 'xy');
  filmPts = [filmPts; u,v];
end

filmTags = [];
if CHECK_BOUNDARIES == TRUE
  u = 1; v = 2;
  checkPts = filmPts;
  filmPts = [];
  for i = 1:size(checkPts,1)
    if (checkPts(i,u) >= 0) & (checkPts(i,u) <= filmDim(u)) & ...
       (checkPts(i,v) >= 0) & (checkPts(i,v) <= filmDim(v))
       filmPts = [filmPts; checkPts(i,:)];
       filmTags = strvcat(filmTags,[deblank(ptTags(i,:)) filmSuffix]);
    end
  end
end
```

B.6.12 PlaneIntercept

```
% PlaneIntercept   - returns the intercept of the line with the plane
%
% [a1 a2] = PlaneIntercept(linePoint, dirVector, plane)
%
% Inputs:
%       linePoint - 3D point on the line
%       dirVector - direction vector of the point/line
%       plane - string ('xy','xz','yz') indicating intersection plane
%
% Outputs:
%       [a1 a2] - coordinate of the intersection of the line
%                 with the plane
%                 'xy': a1 = x, a2 = y
```

```
%              'xz': a1 = x, a2 = z
%              'yz': a1 = y, a2 = z
%
% Last Modified:
%      June 2, 2000 - Anthony Choo
%
function [a1, a2] = PlaneIntercept(linePoint, dirVector, plane)
X=1;
Y=2;
Z=3;

if plane == 'xy'

   t = -linePoint(:,Z)./dirVector(:,Z);
   a1 = linePoint(:,X) + t.*dirVector(:,X);
   a2 = linePoint(:,Y) + t.*dirVector(:,Y);
elseif plane == 'xz'
   t = -linePoint(:,Y)./dirVector(:,Y);
   a1 = linePoint(:,X) + t.*dirVector(:,X);
   a2 = linePoint(:,Z) + t.*dirVector(:,Z);
elseif plane == 'yz'
   t = -linePoint(:,X)./dirVector(:,X);
   a1 = linePoint(:,Y) + t.*dirVector(:,Y);
   a2 = linePoint(:,Z) + t.*dirVector(:,Z);
end
```

B.6.13 LinesIntercept

```
% linesIntercept   - returns the least squares intercept of a set of lines
%          where each line is defined by 2 points
%
% If P1 and P2 are 2 points on a line, then all points on the line are given by,
%    r = P1 + k(P1-P2) where k is some scalar.
%
% For multiple rays r_i = P1_i + k_i*(P1_i-P2_i), where i = 1..n
%    the least squares intercept is the solution of:
%    Ax = B, or in MATLAB, x = A\B where
%             A = [I  -(P1_1-P2_1)   0..0      0..0;
%        I  0     -(P1_i-P2_i)   0..0;
%        I  0     0..0      -(P1_n-P2_n)]
%             B = [P1_i; ... P1_n]
%             x = [x; y; (z;) k_1; .. k_n]
%             I = identity matrix of dimension 2D or 3D
% Inputs:
%      linePts - [x1,y1,z1,x2,y2,z2;...] (3D usage)
%              - [x1,y1,x2,y2;...] (2D usage)
%              - each row specifies two points on the line
%
% Outputs:
%      ls_solution - the least squares solution of Ax = B
%                  - the coordinates of the interception are the first N
%                    values (where N is the dimension of space - i.e. 2 or 3)
%
% Revision History:
%      July 24, 2000 - Anthony Choo - initial version
%
function [ls_solution] = linesIntercept(linePts)

% number of lines
numLines = size(linePts,1);
% dimensions of space
dimSpace = size(linePts,2)/2;

% setup system of equations: Ax = B
%  where x = [Fx Fy {Fz} k1 ... kn]
%
% Construct A
numRowsA = dimSpace*numLines;
numColsA = dimSpace+numLines;
% initialize A
A = zeros(numRowsA,numColsA);

% construct P1-P2
P1_P2 = linePts(:,1:dimSpace) - linePts(:,dimSpace+1:dimSpace+dimSpace);

% fill in A
for i = 1:numLines
   rowStart = (i-1)*dimSpace+1;
   rowEnd = rowStart + dimSpace - 1;
   A(rowStart:rowEnd,1:dimSpace) = eye(dimSpace);
```

```
   colFill = dimSpace+i;
   A(rowStart:rowEnd,colFill) = -P1_P2(i,:)';
end
%
% Construct B
B = linePts(:,1:dimSpace);
B = reshape(B',numRowsA,1);

ls_solution = A\B;
```

Appendix C

RSA ACCURACY – MATLAB CODE

C.1 Digital Measurement Accuracy

C.1.1 AccuracyTst

```
% accuracyTst.m - script file to generate the images for the
%                measurement accuracy tests
%
% calls mkXray.m function
% calls GenCentres.m function
%
% Revision History:
%  Oct. 24, 2000 - Anthony Choo - rand test run
%  Oct. 25, 2000 - Anthony Choo - randn test run
%  Nov.  1, 2000 - Anthony Choo - randn test with smaller r's
%
clear all;
close all;

% conversion factors
inch2mm = 25.4;

% fixed parameters for this setup
DPI = 300;
filmLength = 5;
filmWidth = 5;
filmUnits = 'in';
rUnits = 'mm';
centreUnits = 'in';
filePrefix = 'test';
showFig = 0;

% variable parameters
r = [0.5 1 1.5 2 2.5]; % diameters 1,2,3,4,5 mm
T = [0.1 0.3 0.5 0.7 0.9];
Tst = [10 30 50 70 90]; % for string output purposes
numCentres = 10;

% for testing
%  r = [5];
%  T = [0.5];
%  Tst = [10 50 90];   % for string output purposes

% for testing
%  filmLength = 1;
%  filmWidth = 1;
%  r = [0.5];
%  T = [0.2];
%  showFig = 1;

% Generate and save images with randn centre distributions
% initialize ExactCoords matrix
ExactCoords = [];
for i = 1:size(r,2)
   for j = 1:size(T,2)
      disp(['... Generating Image: R=' num2str(r(i)) ' T=' int2str(Tst(j)) '%']);
      drawnow;
      % get centre coordinates
      [rcCentre, gridRow, gridCol] =
GenCentres(numCentres*2,r(i)/inch2mm,filmLength,filmWidth,'randn');
```

```
      % remove extra centre coordinate
      rcCentre = unifDecimate(rcCentre,numCentres);
      fileName = [filePrefix 'r' num2str(r(i)) 't' int2str(Tst(j))];
      [f, exactC] = mkXray(DPI, filmLength, filmWidth, filmUnits, r(i), rUnits,
rcCentre(1:numCentres,:), gridRow, gridCol, centreUnits, T(j), fileName, showFig);
      % ExactCoords is in (x,y) so must reverse the (row,col) coords returned by
      % mkXray
      ExactCoords = [ExactCoords exactC(:,2) exactC(:,1)];
   end
end

% Generate and save an "exact" centre distributions (for guidance purposes)
disp('... Generating Centred Image');
drawnow;
rex = 1;
Tex = 0.001;
% get centre coordinates
[rcCentre, gridRow, gridCol] =
GenCentres(numCentres*2,rex/inch2mm,filmLength,filmWidth,'exact');
% remove extra centre coordinate
rcCentre = unifDecimate(rcCentre,numCentres);
fileName = [filePrefix 'r' int2str(rex) 't' int2str(Tex) '_centred'];
f = mkXray(DPI, filmLength, filmWidth, filmUnits, rex, rUnits, rcCentre, gridRow,
gridCol, centreUnits, Tex, fileName, showFig);

% write ExactCoords to datafile
disp('... saving exact coordinates to data file');
drawnow;
% open file
savePath = [pwd '\'];
saveFile = 'ExactCoords.txt';
saveMode = 'w';
FID = fopen([savePath saveFile],saveMode);
fprintf(FID,'Marker#,');   % column spacer for number ID
% write headers
for i = 1:size(r,2)
   for j = 1:size(T,2)
      fprintf(FID,'x-r%it%i,y-r%ir%i,',r(i),Tst(j),r(i),Tst(j));
   end
end
fprintf(FID,'\n');
% write coordinates
for i = 1:size(ExactCoords,1)
   fprintf(FID,'%i,',i);
   for j = 1:size(ExactCoords,2)
      fprintf(FID,'%i,',ExactCoords(i,j));
   end
   fprintf(FID,'\n');
end
fclose(FID);
disp('accuracytst completed generating images');
% test code: remove
%  filmLength = 100;
%  filmWidth = 100;
%  filmUnits = 'pix';
%  r = 10;
%  T = 0.5;
%  rUnits = 'pix';
%  rcCentre = [50 50];
%  centreUnits = 'pix';
%
%  fileName = 'test1';
%  f1 = mkXray(DPI, filmLength, filmWidth, filmUnits, r, rUnits, rcCentre, centreUnits, T,
fileName);
```

C.1.2 MkXray

```
% mkXray - generates an X-ray image and saves it as a TIFF file
%
% Inputs:
%  res - resolution in dpi
%  filmLength - length (image rows)
%  filmWidth - width (image columns)
%  filmUnits - 'in' for inches
%            - 'mm' for milimeters
%            - 'pix' for pixels
%  r - sphere radius
%     rUnits - 'in' for inches
%           - 'mm' for milimeters
%           - 'pix' for pixels
%  rcCentre - [row,col;...] coordinates if the sphere centres
%  gridRow - optional row-axes values to draw a grid
```

```
%  gridCol - optional col-axes values to draw a grid
%     rcUnits - units for coordinates of rcCentre, gridRow, gridCol
%           - 'in' for inches
%           - 'mm' for milimeters
%           - 'pix' for pixels
%  T - transmission factor at centre of sphere [0,1]
%  writeFile - string for output file name
%  ShowFig - boolean, 1 = render a figure
%                0 = do not render a figure, figHandle = []
%
% Outputs:
%  figHandle - handle to figure showing images
%  coords - (row,col) coordinates of points in pixels
%  writeFile.tif - tiff file of image
%  writeFile_ideal.tiff - tiff file of image without noise
%
% Revision History:
%  Oct. 25, 2000 - Anthony Choo - initial version
%
%   (Based on makexray.m from BME595F 1997 Lab problem 1
%    Instructor: Professor Michael Joy, Univeristy of Toronto)
%
function [figHandle, coords] = mkXray (res, filmLength, filmWidth, filmUnits, R, rUnits,
rcCentre, gridRow, gridCol, rcUnits, T, writeFile, showFig)

% constants
SHOW = 1;
NOSHOW = 0;

% conversion factors
inch2mm = 25.4;
mm2pixels = res/inch2mm;

background = 100; % Photon fluence without any patient

% convert sphere radius to pixels - must be integer
% using floor helps ensure that the points do not go
% out of bounds when randn centre distribution is used
if strcmp(rUnits,'in')
   r = floor(R*inch2mm*mm2pixels);
elseif strcmp(rUnits,'mm')
   r = floor(R * mm2pixels);
end

% compute image size in pixels - must be integer
if strcmp(filmUnits,'in')
   imRows = ceil(filmLength*res);
   imCols = ceil(filmWidth*res);
elseif strcmp(filmUnits,'mm')
   imRows = ceil(filmLength*mm2pixels);
   imCols = ceil(filmWidth*mm2pixels);
else
   imRows = ceil(filmLength);
   imCols = ceil(filmWidth);
end

% convert rcCentre and gridRow, gridCol to pixel units - must be integer
if strcmp(rcUnits,'in')
   rcCentre = ceil(rcCentre * inch2mm * mm2pixels);
   gridRow = ceil(gridRow * inch2mm * mm2pixels);
   gridCol = ceil(gridCol * inch2mm * mm2pixels);
elseif strcmp(xyUnits,'mm')
   rcCentre = ceil(rcCentre * mm2pixels);
   gridRow = ceil(gridRow * mm2pixels);
   gridCol = ceil(gridCol * mm2pixels);
end
coords = rcCentre;

[Ri, Ci] = meshgrid(-r:r, -r:r); % Grid the sphere's bounding box
thick = psphere(Ri, Ci); % The thickness, relative to 2R, of the sphere over this grid

sphereImage = ones(imRows,imCols);

exponent  = zeros(size(sphereImage)); % The initial value of the exponent in Beer's Law is
0 all over
transmis  = zeros(size(sphereImage)); % This will contain the transmission factor
idealImage = ones(size(sphereImage)) ; %  Will contain the noiseless image

% add number of spheres
tLogT = thick*log(T);   % optimization for some speed
for i = 1:size(rcCentre,1)
   Rpos = rcCentre(i,1);
```

```
    Cpos = rcCentre(i,2);
    % Add the exponent of Beer's Law to the affected points
    % i.e. points outside of the sphere have a Beer's law of e^0 = 1
    %      points within the sphere have an exponent propotional to their thickness
    % the expression is in terms of a square matrix - sizes should match
    exponent(Rpos-r:Rpos+r, Cpos-r:Cpos+r) ...
      = exponent(Rpos-r:Rpos+r, Cpos-r:Cpos+r) + tLogT;
end

% Compute the transmission = e^exponant
transmis = exp(exponent);

% Compute the noiseless image
idealImage = background * transmis ; % Compute the actual fluence = BG * transmission
if any(idealImage < 0 ) error('Impossible'); end % Just checking for a logical error

% if necessary add grid to idealImage
if not(isempty(gridRow))
    % set idealImage (rows, all columns) = black
    idealImage(gridRow,:) = 0;
end
if not(isempty(gridCol))
    % set idealImage (all rows, cols) = black
    idealImage(:,gridCol) = 0;
end

% Add Poisson noise
sphereImage = idealImage + sqrt(idealImage).*randn(size(idealImage));

% ? should this be absolute value instead??? -
% Get rid of annoying negative values by replacing them with zero.
negative = sphereImage < 0 ; % A "logical' type value
zero = zeros(size(sphereImage));
sphereImage(negative) = zero(negative); % Selective assignment of negative locations with
0.

% The next line is one way to ensure each image is scaled about the same
% Anthony: I think this line sets the minimum and maximum values - henced used for
%          the colormap(gray) scale
sphereImage(1, 1) = 0 ; sphereImage(1,2) = background + 5*sqrt(background);

% for exporting - NEED TO RESCALE IMAGE
% imwrite will use: uint8(round(255*image_matrix)) to scale double
%                   the lines below will scale values to [0,1]
sphereImage = sphereImage./max(max(sphereImage));
idealImage = idealImage./max(max(idealImage));

if showFig == SHOW
    figHandle = figure;
r
T
    set(gcf,'name',['r ' num2str(R) ' T ' num2str(T)]);
    % Make the display pretty
    colormap(gray);
    figure(figHandle);
    subplot(1,2,1);
    imagesc(idealImage);
    axis('off');
    title(['Ideal Image']);
    subplot(1,2,2);
    imagesc(sphereImage);
    axis('off');
    title(['Image with Noise']);
else
    figHandle = [];
end
% Export image to file
% imwrite(A,filename,fmt)
%
% Choice = TIFF uncompressed
% Notes:
%    - did not use bmp because no grayscale (255 color, or 255 black & white)
%    - did not use jpg because compression loses infomation - want a 1:1 mapping
%      between matlab generated image and coordinates measured
imwrite(sphereImage,[writeFile '.tif'],'tif','Compression','none');
imwrite(idealImage,[writeFile '_ideal.tif'],'tif','Compression','none');
```

C.1.3 GenCentre

```
% GenCentres - function generates a set of distributed centres in an area
%
```

```
% Input:
%  n = number of centres
%  r = radius of centres
%  l = length of area (rows)
%  w = width of area (columns)
%  d = 'exact' for the exact centres
%       'randn' for normal random distribution
%       'rand' for uniform random distribution
%
% Output:
%  Centres - [l,w;...] centres
%  gridRow - row-axes values for creating a grid bounding the centres
%  gridCol - col-axes values for creating a grid bounding the centres
%
% Revision History:
%  Oct. 25, 2000 - Anthony Choo - initial version
%
function [Centres, gridRow, gridCol] = GenCentres(n, r, l, w, d)

% compute number of cells along length
nL = ceil(sqrt(n*l/w));
% compute number of cells along width
nW = ceil(nL*w/l);

% convention: rows=l, cols=w
% row-col increments
rowIncr = l/nL;
colIncr = w/nW;

% error checking
if rowIncr/2-r < 0 | colIncr/2-r < 0
   error('Radius is too large for even distribution!');
   Centres = [];
   return;
end

% determine the vectors representing centre coordinates on each axis
gridRow = [0:nL-1]*rowIncr;
gridCol = [0:nW-1]*colIncr;
% Offset row and col to the centre of the cells
rows = gridRow+rowIncr/2;
cols = gridCol+colIncr/2;
% strip off first zero from gridX and gridY
gridRow = gridRow(2:nL);
gridCol = gridCol(2:nW);

Centres = [];
% generate centres and add variation if necessary
% some redundant code used for faster execution
rowBdry = rowIncr/2-r;
colBdry = colIncr/2-r;
if strcmp(d,'exact')
   for i = 1:size(rows,2)
      for j = 1:size(cols,2)
         Centres = [Centres; rows(i) cols(j)];
      end
   end
elseif strcmp(d,'randn')
   for i = 1:size(rows,2)
      for j = 1:size(cols,2)
         % ensure that randn is always less than 1 to guarentee
         % the circles remain within the boundary of the image
         varRow = randn*rowBdry;
         while abs(varRow) >= rowBdry
            varRow = randn*rowBdry;
         end
         varCol = randn*colBdry;
         while abs(varCol) >= colBdry
            varCol = randn*colBdry;
         end
         %% Delete the algorithm below
         %  varRow = randn*rowBdry;
         %  varCol = randn*colBdry;
         %  % ensure normal rand variable is within bounds
         %  while rows(i)+varRow-r < 0 | cols(j)+varCol-r < 0 | ...
         %  rows(i)+varRow+r > l | cols(j)+varCol+r > w
         %        varRow = randn*rowBdry;
         %        varCol = randn*colBdry;
         %     end
         Centres = [Centres; rows(i)+varRow cols(j)+varCol];
      end
   end
```

```
elseif strcmp(d,'rand')
   for i = 1:size(rows,2)
      for j = 1:size(cols,2)
         varRow = rand*rowBdry;
         varCol = rand*colBdry;
         Centres = [Centres; rows(i)+varRow cols(j)+varCol];
      end
   end
end
```

C.1.4 Psphere

```
% From BME 595F 1998 University of Toronto – BioMedical Imaging
%
% Projection through a sphere
function thickness = psphere(Xi, Yi)
% Projection through a sphere
% thickness is an image of a projection through a sphere
% of density one and which just fills the square
% matrices Xi and Yi. These matrices contain the
% x and y coordinates of image pixels. They are square and
% have an odd number of rows and columns.

% Check the arguments
sx = size(Xi) ; sy = size(Yi) ;
if any( sx ~= sy | sx(1) ~= sx(2) )
   error( 'The two arguments must be square and of equal size');
end

global Radius thick2 inside;
thickness = zeros(sx);
Radius = (sx(1) - 1) / 2 * ones(sx);
thick2 = Radius.^2 - Xi.^2 - Yi.^2;
inside = thick2 >= 0 ;
thick  = sqrt(thick2) ./ Radius;
thickness(inside) = thick(inside);
```

C.2 Benchtop Accuracy Test

C.2.1 DOERCage10

```
% Cage 10 data from Tilly Medical - MODIFIED
%  Tag names renamed with leading zeros removed
%  I and II used instead of suffix 1 and 2
%  FP and CP used instead of leading 0 and 1
%  regrouped so all FPI's together and all FPII's together etc..
% FP_list
% FP_Tag,FP_x,FP_y,FP_z,
FPI_1,-60.0,0.0,0.0,
FPI_2,0.0,0.0,0.0,
FPI_3,60.0,0.0,0.0,
FPI_4,-60.0,80.0,0.0,
FPI_5,0.0,80.0,0.0,
FPI_6,60.0,80.0,0.0,
FPI_7,-60.0,160.0,0.0,
FPI_8,0.0,160.0,0.0,
FPI_9,60.0,160.0,0.0,
FPII_1,-124.5,0.0,169.8,
FPII_2,-124.5,0.0,109.8,
FPII_3,-124.5,0.0,49.8,
FPII_4,-124.5,80.0,169.8,
FPII_5,-124.5,80.0,109.8,
FPII_6,-124.5,80.0,49.8,
FPII_7,-124.5,160.0,169.8,
FPII_8,-124.5,160.0,109.8,
FPII_9,-124.5,160.0,49.8,
% END_LIST

% CP_list
% CP_Tag,CP_x,CP_y,CP_z,
CPI_1,-45.0,20.0,249.0,
CPI_2,0.0,20.0,249.0,
CPI_3,45.0,20.0,249.0,
CPI_4,-45.0,80.0,249.0,
CPI_5,0.0,80.0,249.0,
CPI_6,45.0,80.0,249.0,
CPI_7,-45.0,140.0,249.0,
CPI_8,0.0,140.0,249.0,
CPI_9,45.0,140.0,249.0,
CPII_1,124.5,20.0,154.6,
CPII_2,124.5,20.0,109.6,
CPII_3,124.5,20.0,64.6,
CPII_4,124.5,80.0,154.6,
CPII_5,124.5,80.0,109.6,
CPII_6,124.5,80.0,64.6,
CPII_7,124.5,140.0,154.6,
CPII_8,124.5,140.0,109.6,
CPII_9,124.5,140.0,64.6,
% END_LIST
```

C.2.2 Tlocal

```
% TLocal - routine which transforms the global coordinates to a local coordinate system
%
% Inputs:
%  sourceFile - string of text file with coordinates
%        - lines beginning with '%' are comments and are ignored
%        - first line is origin (x,y,z)
%        - second line is x-axis point (fixed)
%        - third line is xy-plane point
%        - all subsequent lines are coordinates
%
%  outputFile - string of text file that will be generated with transformed pts
%
%  sourcePath - path to source file, current path is default
%
%  outputPath - path to save outputFile, current path is default
%
% Outputs:
%
%  outputFile - text file with points (4th line and on) converted to local coordinates
%
% Revision History:
%  Nov. 8, 2000 - Anthony Choo - initial version
%
function TLocal(sourceFile,outputFile,sourcePath,outputPath)
```

```
CMT = '%';

% Load File
FID = fopen([sourcePath sourceFile],'rt');

% read axes data
P_Origin = sscanf(getNextLine(FID,CMT),'%f,%f,%f');
P_X = sscanf(getNextLine(FID,CMT),'%f,%f,%f');
P_Y = sscanf(getNextLine(FID,CMT),'%f,%f,%f');

% construct local orthonormal basis
%  x-axis: normalized and fixed
x_dir = P_X - P_Origin;
x_axis = x_dir./norm(x_dir);

%  y-axis will be solved after z-axis, used P_Y for x-y plane
y_dir_pseudo = P_Y - P_Origin;

%  z-axis: normalized
z_dir = cross(x_axis,y_dir_pseudo);
z_axis = z_dir./norm(z_dir);

%  y-axis: normalized
y_axis = cross(z_axis,x_axis);

if abs(norm(y_axis) - 1) > 0.001
   error ('Error detected, y-axis is not normalized');
end

% Create T_gl: local frame with respect to global frame
R_gl = [x_axis y_axis z_axis];
T_gl = [R_gl P_Origin; 0 0 0 1];
T_lg = inv(T_gl);

% Read remaining points
Pts_g = [];
while not(feof(FID))
   Pts_g = [Pts_g; sscanf(getNextLine(FID,CMT),'%f,%f,%f')'];
end
fclose(FID);

% Multiply points by T_lg
Pts_l = MultHomogeneous(Pts_g,T_lg);

% Save New Points
OID = fopen([outputPath outputFile],'w');
fprintf(OID,'%f,%f,%f\n',Pts_l');
fclose(OID);

% ------------------------------------------------------------------------
% Sub-functions
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% getNextLine - reads line from file ignoring lines beginning with the commentChr
% ------------------------------------------------------------------------
function [nextLine] = getNextLine (FID, commentChr)
while not(feof(FID))
   nextLine = fgetl(FID);
   if strncmp(nextLine,commentChr,1)==0
      break;
   end
end

% MultHomogeneous is copied from Utils lib created from RSALib
%
% MultHomogeneous   - Multiply series of points by a homogeneous transformation
%
% Pts_1 = MultHomogeneous (Pts_2, T12)
%
% Inputs:
%       Pts_2 - [x,y,z;...] points in frame 2
%             - {[x,y,z];...] points in frame 2
%
%       T12 - homogeneous transformation from frame 2 to frame 1
%
% Outputs:
%       Pts_1 - Pts_2 transformed into frame 1
%
% Revision History:
%       June 9, 2000 - Anthony Choo - Initial Revision
```

```
%       June 30, 2000 - Anthony Choo - cell handling added
%
function Pts_1 = MultHomogeneous (Pts_2, T12)

if iscell(Pts_2)
   Pts_1 = {};
   for i = 1:size(Pts_2,1)
      newPts = T12*[Pts_2{i} 1]';
      Pts_1 = [Pts_1; {newPts(1:3)'}];
   end
else
   Pts_1 = [];
   for i = 1:size(Pts_2,1)
      newPts = T12*[Pts_2(i,:)' 1];
      Pts_1 = [Pts_1; newPts(1:3)'];
   end
end
```

C.2.3 csv2xry

```
% csv2xry - script that calls TLocal.m to process files (hardcoded into script)
%
% Revision History:
%  Nov. 9, 2000 - Anthony Choo - initial version
%

% by default the path is set to the current path so...csv2xry must be in
% the same folder as the files to be processed
sourcePath = [pwd '/'];
outputPath = [pwd '/'];

% add a list of files to be processed - do not put the extension - see below
filePrefixes = { ...
      'Tx0_F1', ...
      'Tx0_F2', ...
      'Tx1_F1', ...
      'Tx1_F2', ...
      'RCal_F1', ...
      'RCal_F2', ...
      'R0_F1', ...
      'R0_F2', ...
      'R90_F1', ...
      'R90_F2'
   };

% The file extensions are assumed below
inFileExt = '.csv'; % input files
outFileExt = '.xry';   % output files

numFiles = size(filePrefixes,2);
for i = 1:numFiles
   disp(['Processing ' int2str(i) ' of ' int2str(numFiles) ' :' ...
      filePrefixes{i} inFileExt]);
   TLocal([filePrefixes{i} inFileExt],[filePrefixes{i} outFileExt],sourcePath,outputPath);
end
```

Appendix D

CADAVER KINEMATICS – SUPPLEMENTARY RESULTS

D.1 Humeral Elevation Results

D.1.1 Rz – Scapular Upward Rotation

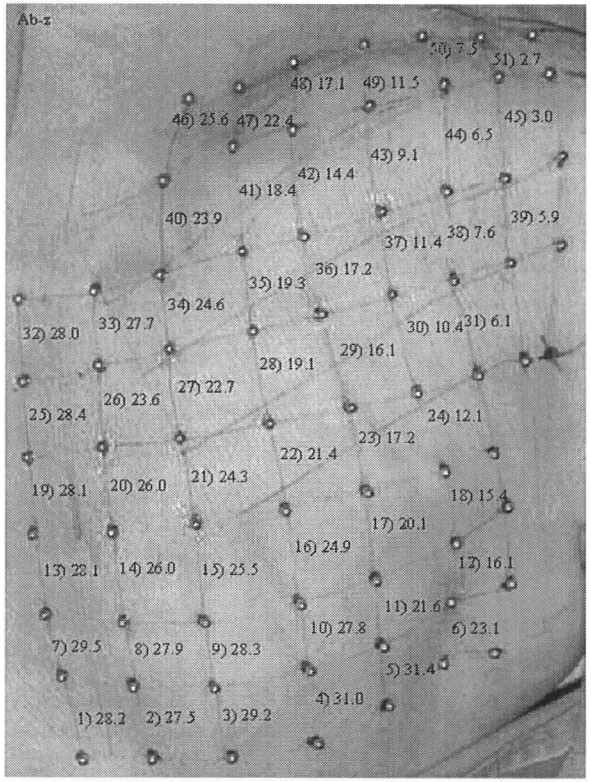

Figure D-1: Scapular upward rotation RMSE during humeral elevation

RMSE (°) computed for 45°, 90°, 135°, 180° of targeted humeral elevation using rotations measured with embedded bone RSA markers as the gold-standard.

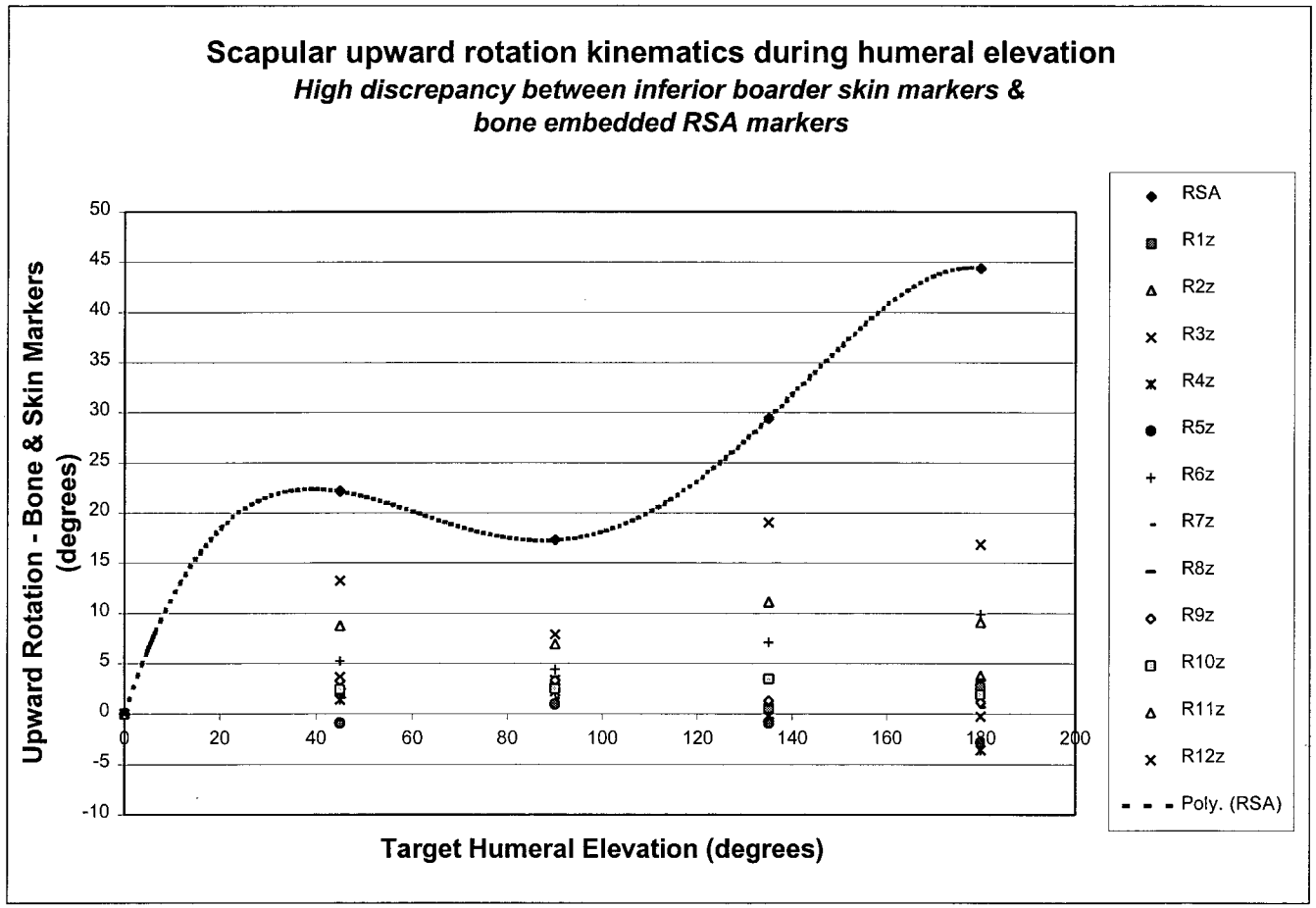

Figure D-2: Scapular upward rotation - inferior boarder patches

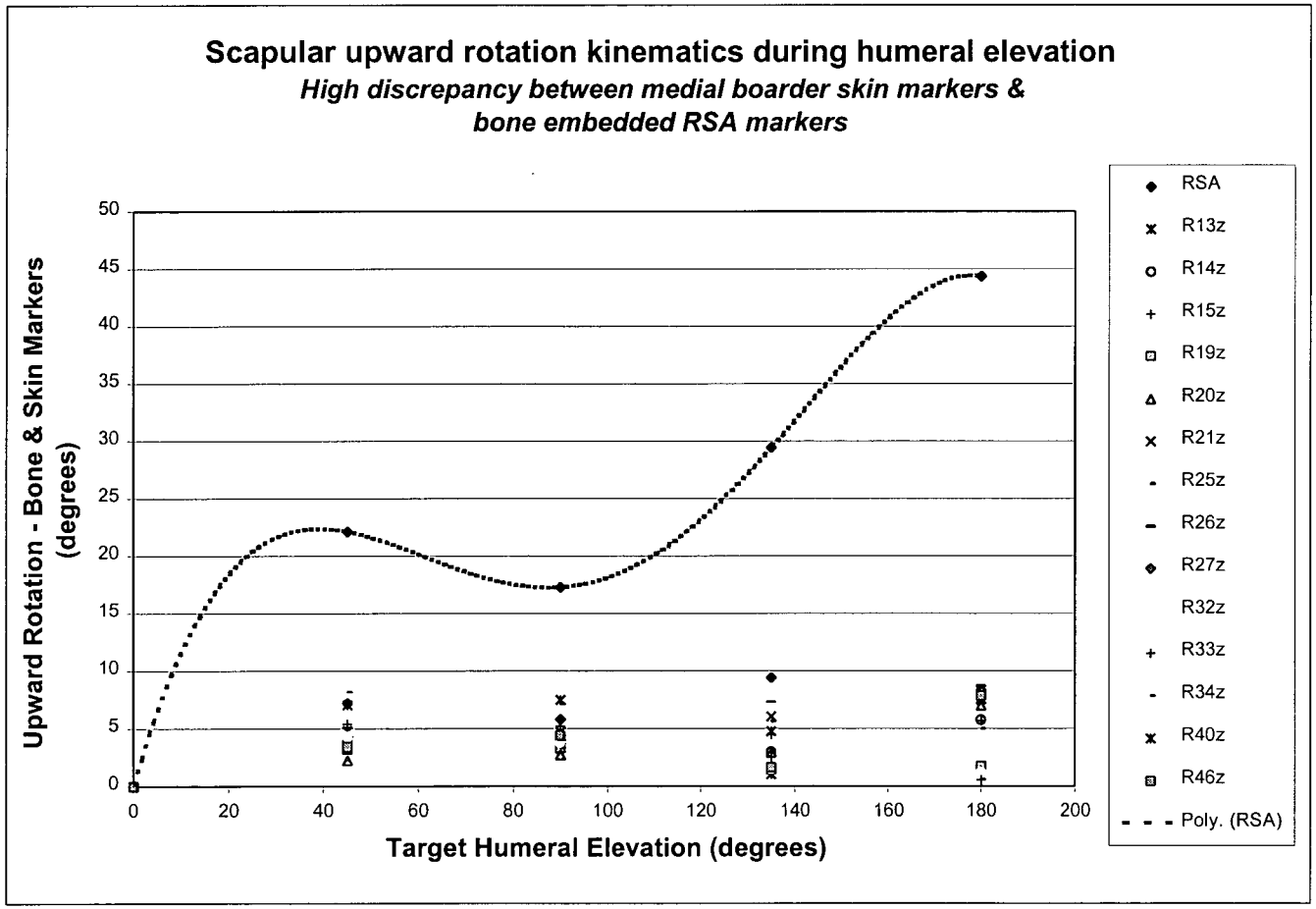

Figure D-3: Scapular upward rotation - medial boarder patches

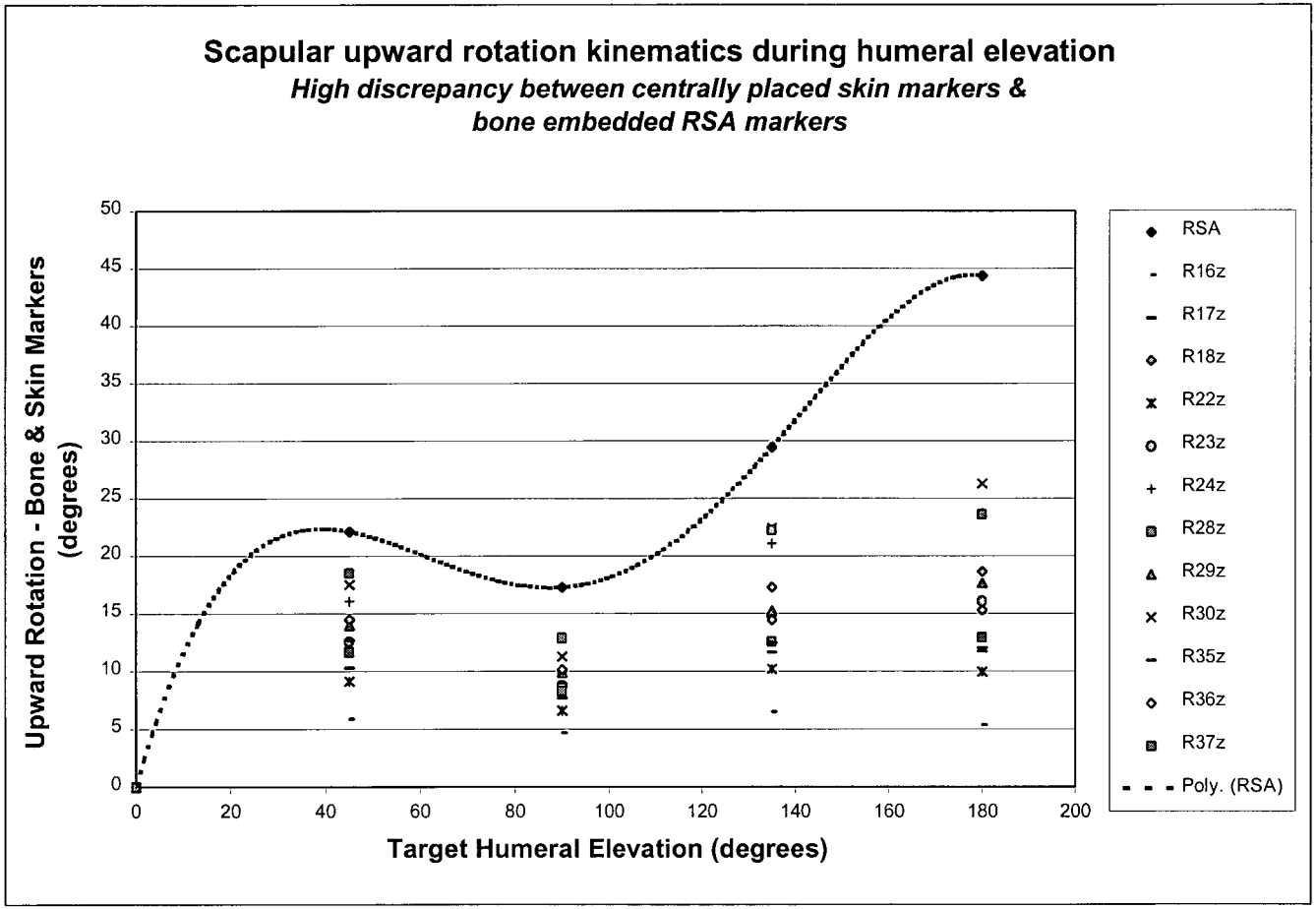

Figure D-4: Scapular upward rotation – central patches

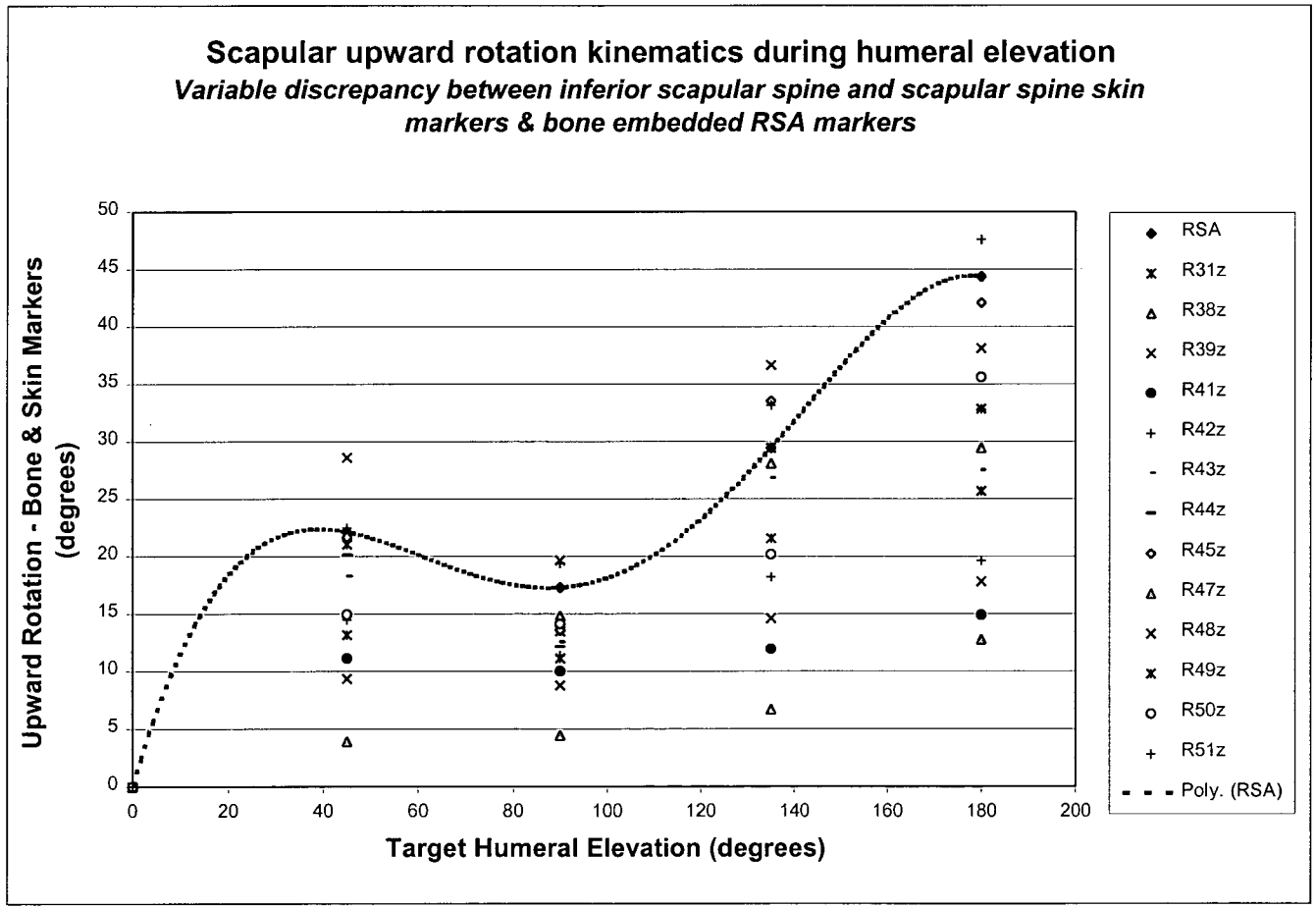

Figure D-5: Scapular upward rotation - patches in region of scapular spine

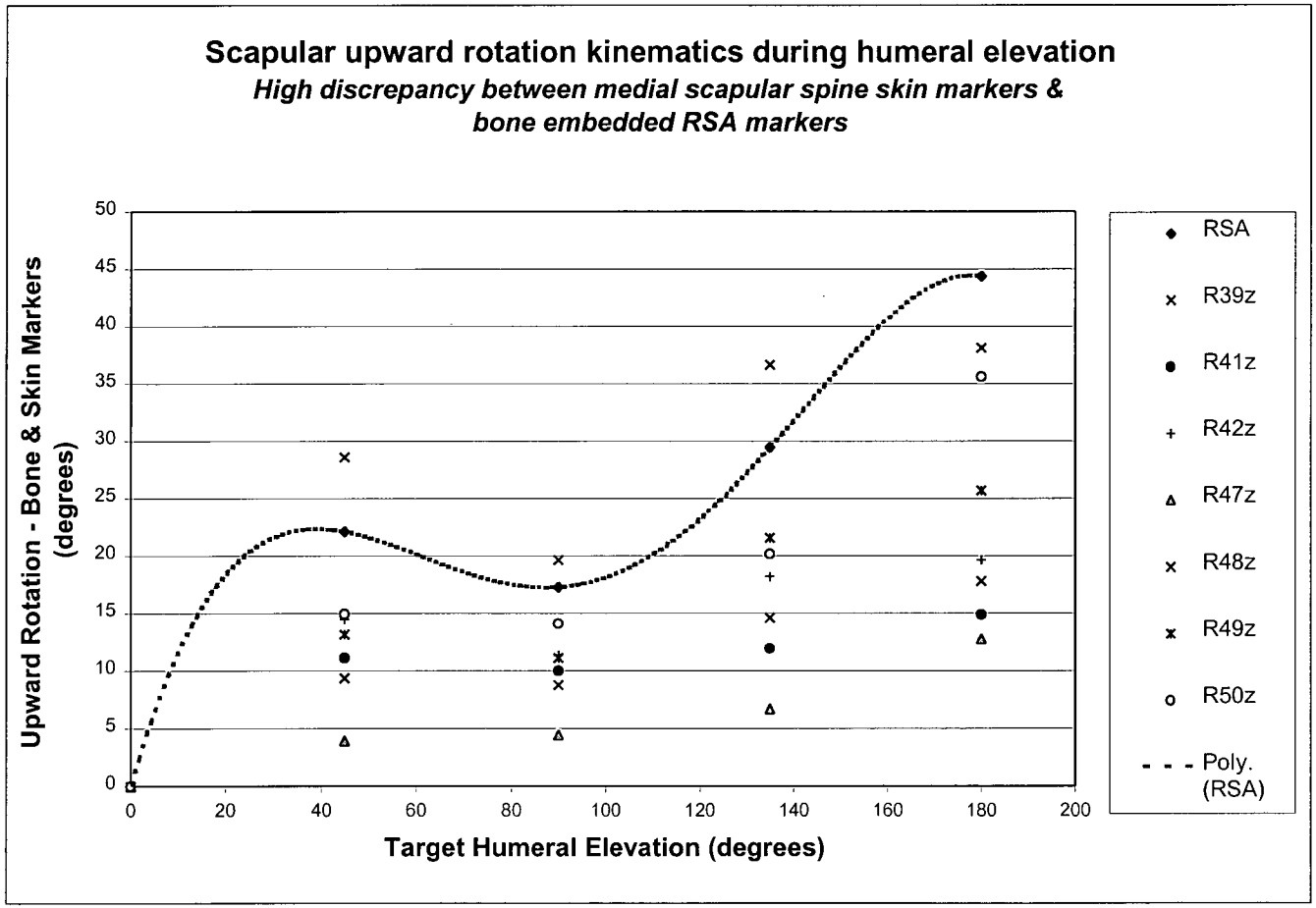

Figure D-6: Scapular upward rotation - medial scapular spine patches

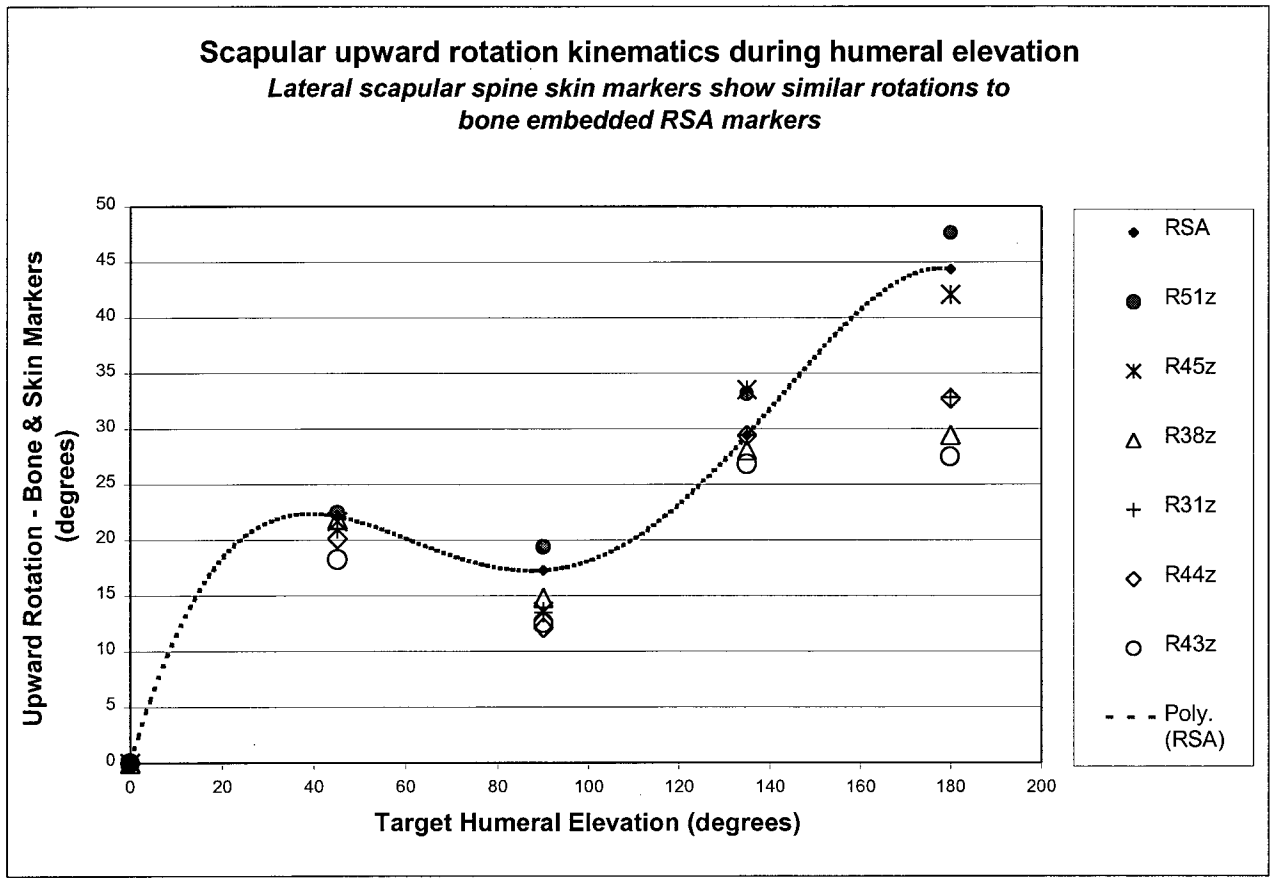

Figure D-7: Scapular upward rotation - lateral scapular spine patches

Below 135° of elevation, the markers on the lateral region of the scapular spine and the lateral inferior region of the scapular spine showed very similar rotations compared to the bone markers. Deviation between skin and bone markers increased at maximum elevation.

D.1.2 Ry – Scapular Internal/External Rotation

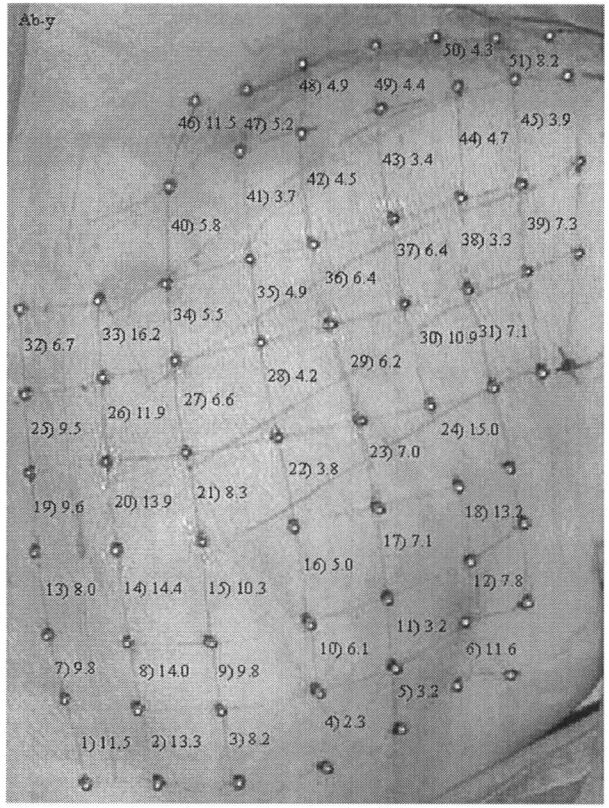

Figure D-8: Scapular intenal/external rotation RMSE during humeral elevation

RMSE (°) computed for 45°, 90°, 135°, 180° of targeted humeral elevation using rotations measured with embedded bone RSA markers as the gold-standard.

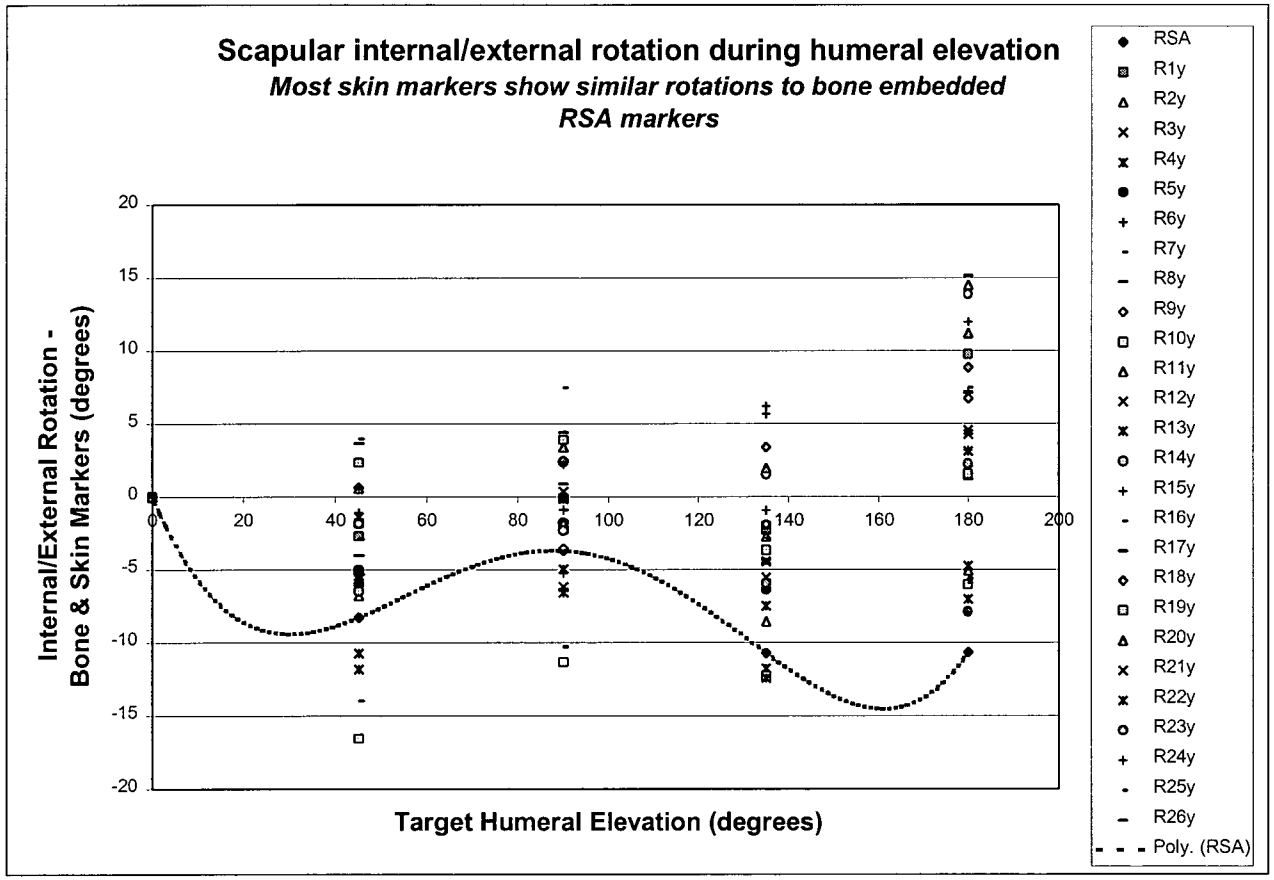

Figure D-9: Scapular internal/external rotation – patches 1 to 26

No general trend was observed indicating a particular region of skin markers was the best candidate for measuring internal/external scapular rotation. All regions demonstrated an RMSE < 15°

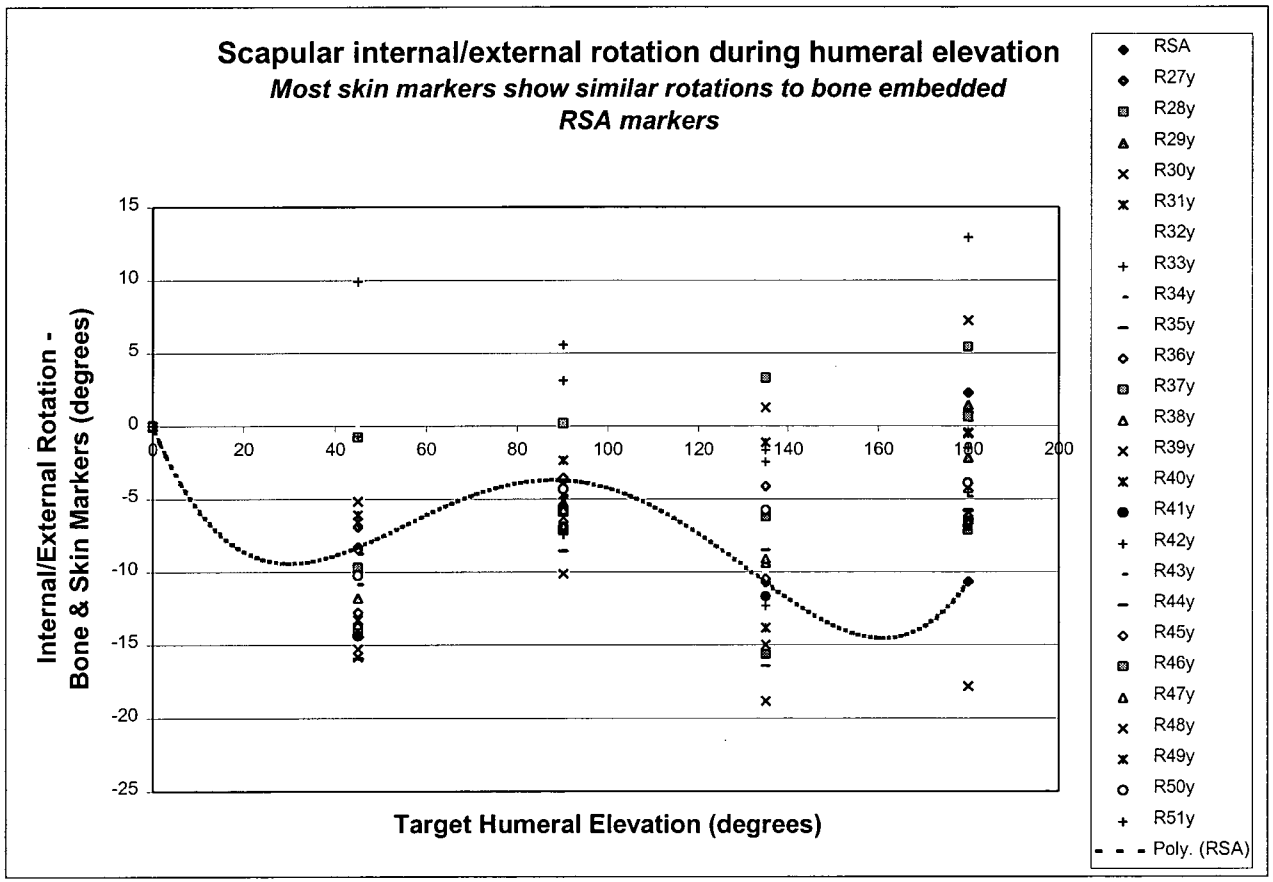

Figure D-10: Scapular internal/external rotation – patches 27 to 51

No general trend was observed indicating a particular region of skin markers was the best candidate for measuring internal/external scapular rotation. All regions demonstrated an RMSE < 15°

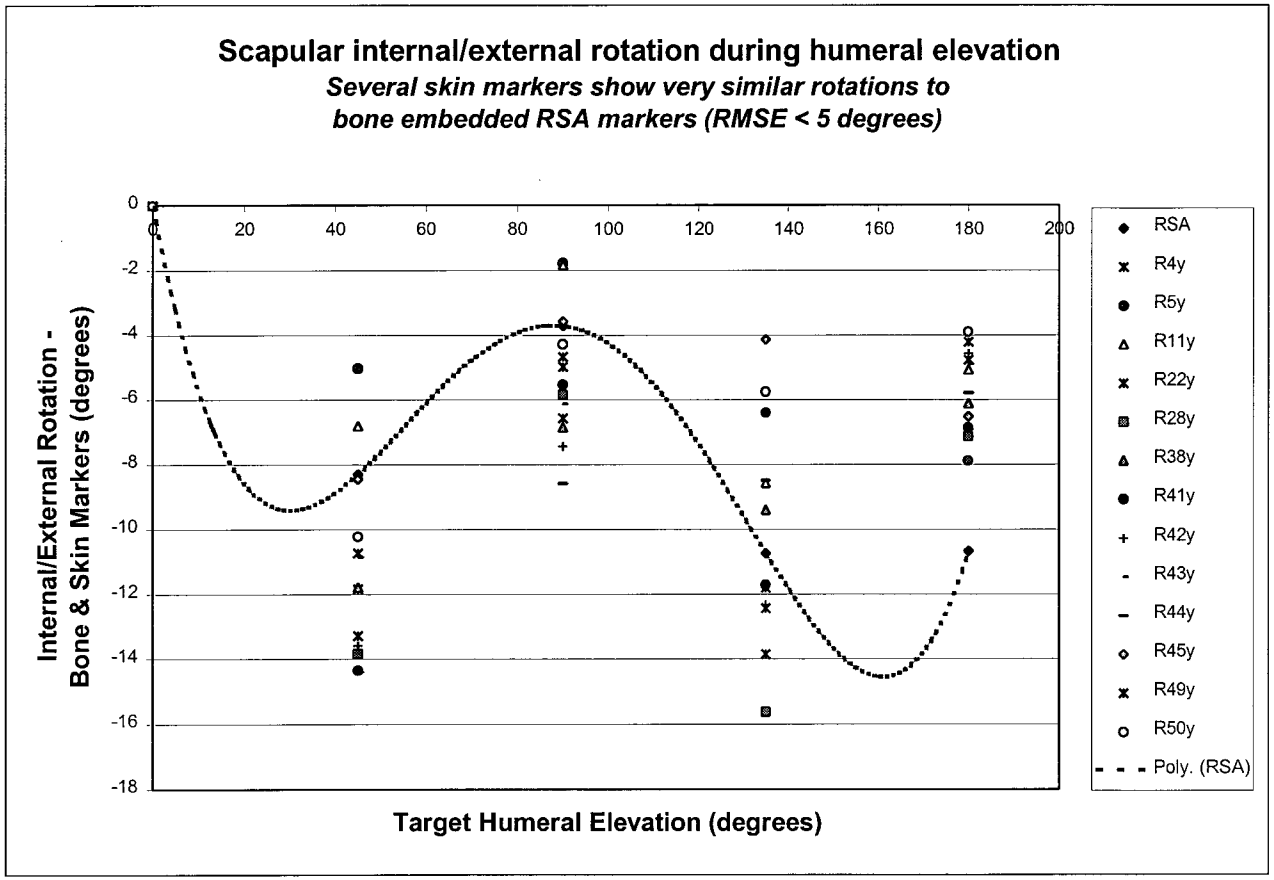

Figure D-11: Scapular internal/external rotation patches with RMSE < 5°

No general trend was observed indicating a particular region of skin markers was the best candidate for measuring internal/external scapular rotation. This graph shows the best skin patches for which the RMSE was below 5°.

D.1.3 Rx – Scapular Tipping

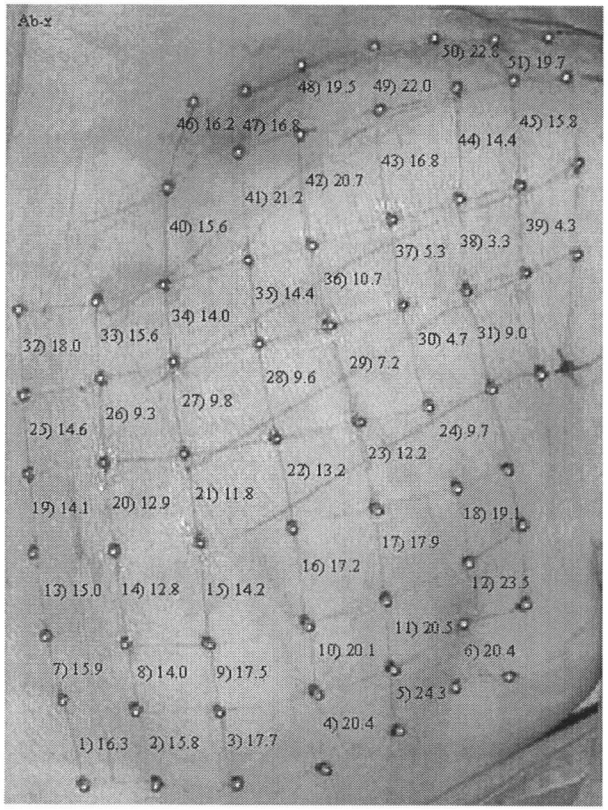

Figure D-12: Scapular tipping RMSE during humeral elevation

RMSE (°) computed for 45°, 90°, 135°, 180° of targeted humeral elevation using rotations measured with embedded bone RSA markers as the gold-standard.

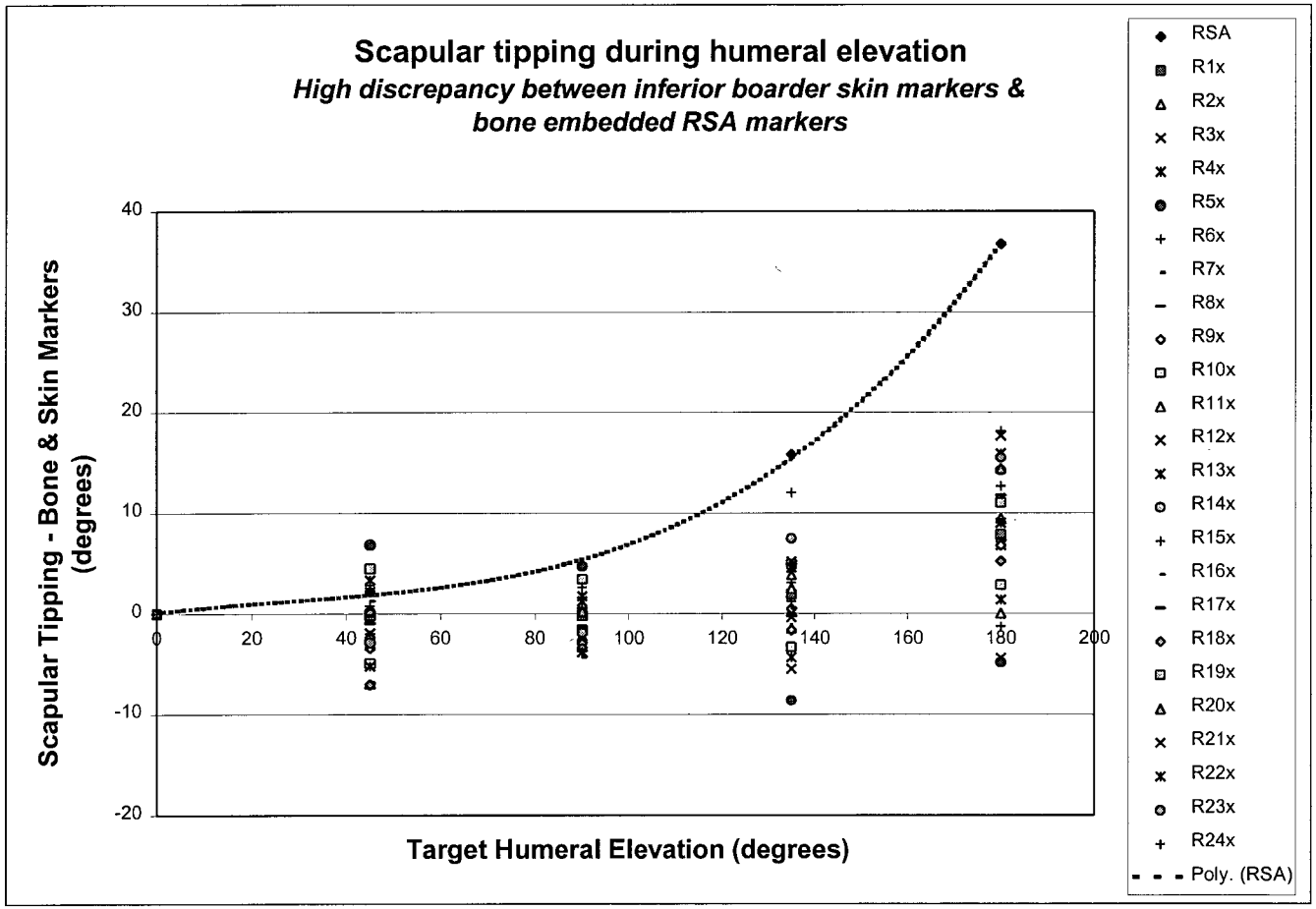

Figure D-13: Scapular tipping - inferior boarder patches

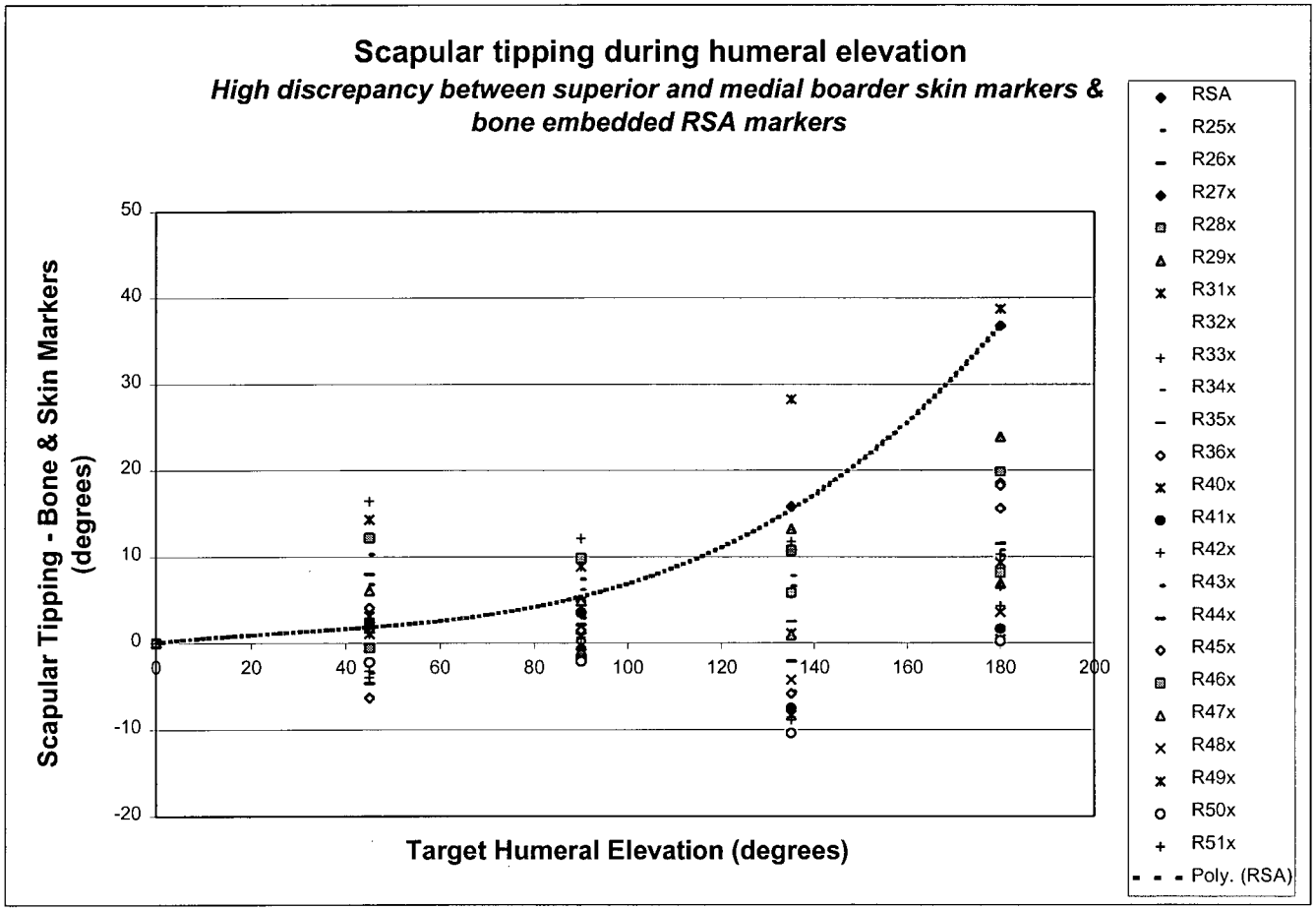

Figure D-14: Scapular tipping - superior and medial boarder patches

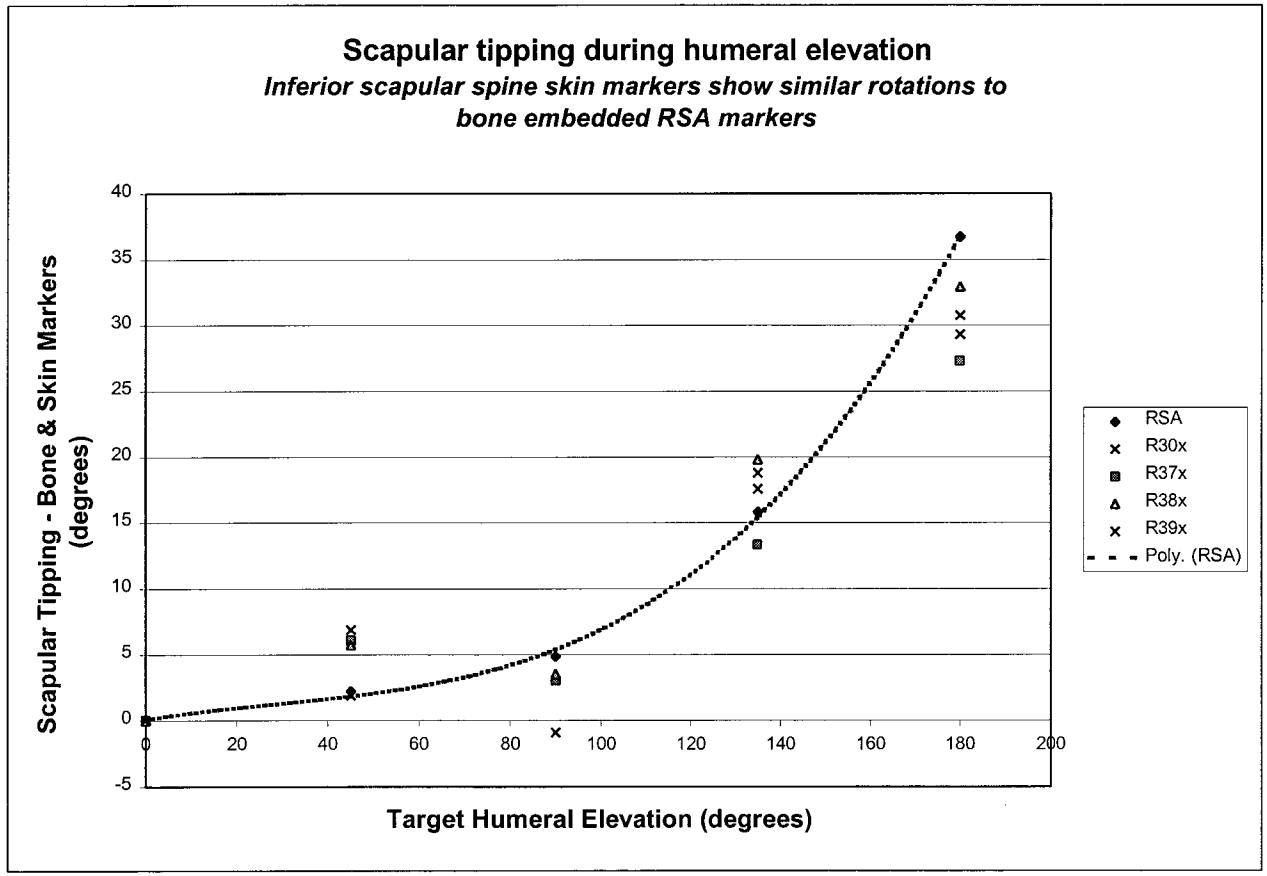

Figure D-15: Scapular tipping - inferior scapular spine patches

Patches 30, 37, 38 and 39 lie in the region just inferior to the scapular spine. These patches showed the most similar rotations about the x-axis (axis along the scapular spine). The patches directly on the scapular spine show poorer results for rotation about this (scapular spine) axis. This intuitively makes sense since the scapular spine patches are measuring the rotation of a cylindrical-like structure beneath the skin.

D.2 Retraction Results

D.2.1 Rz – Scapular Upward Rotation

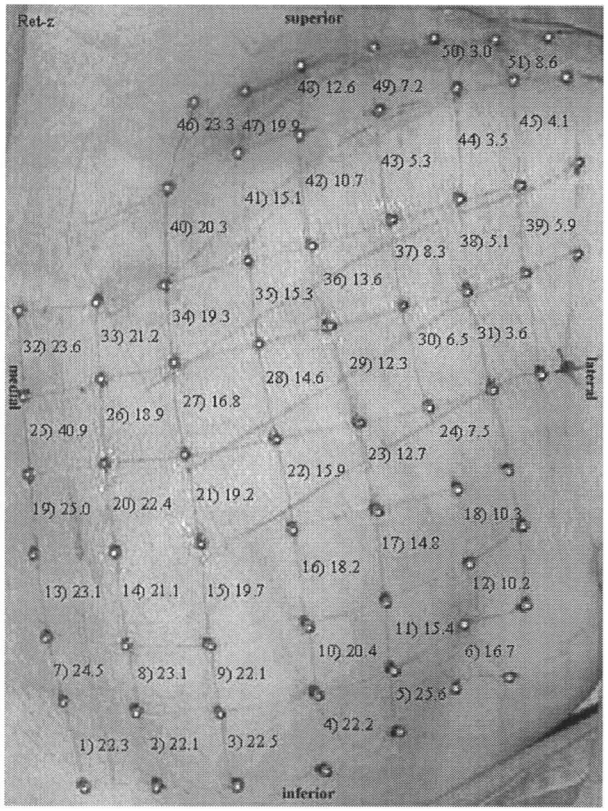

Figure D-16: Scapular upward rotation RMSE during humeral retraction

RMSE (°) computed for 45°, 75°, 90°, of targeted humeral retraction using rotations measured with embedded bone RSA markers as the gold-standard.

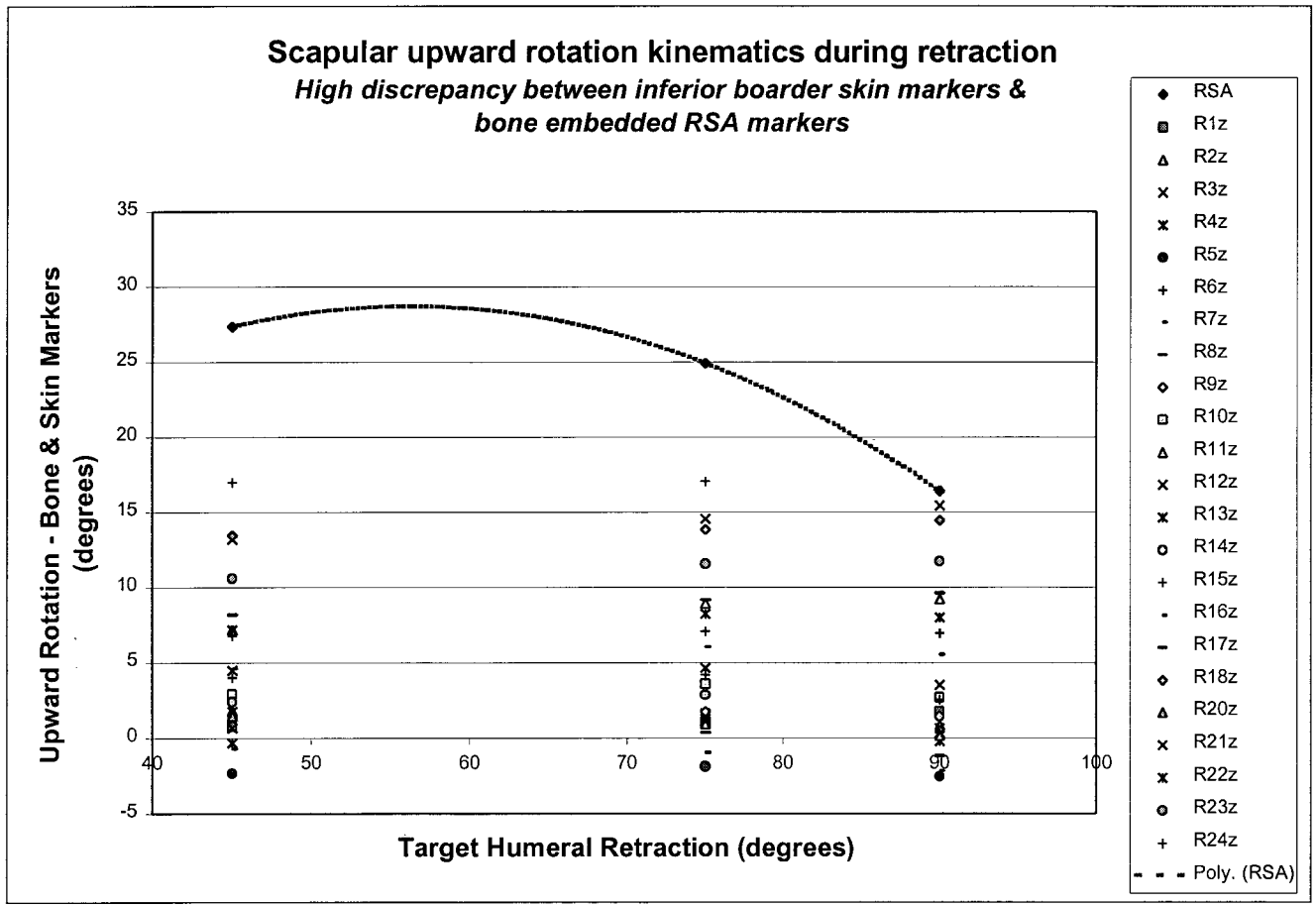

Figure D-17: Scapular upward rotation - inferior boarder patches

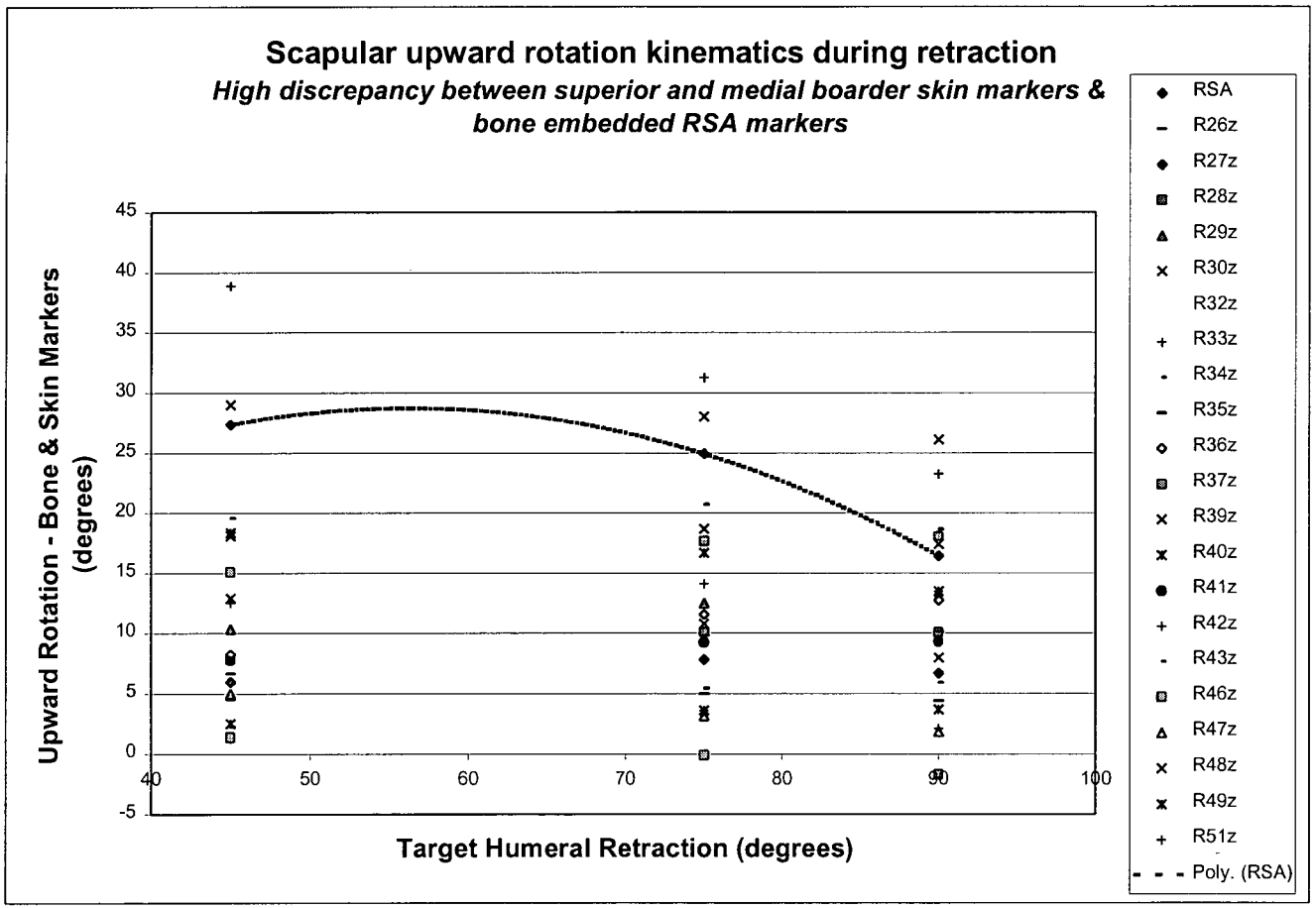

Figure D-18: Scapular upward rotation - superior and medial boarder patches

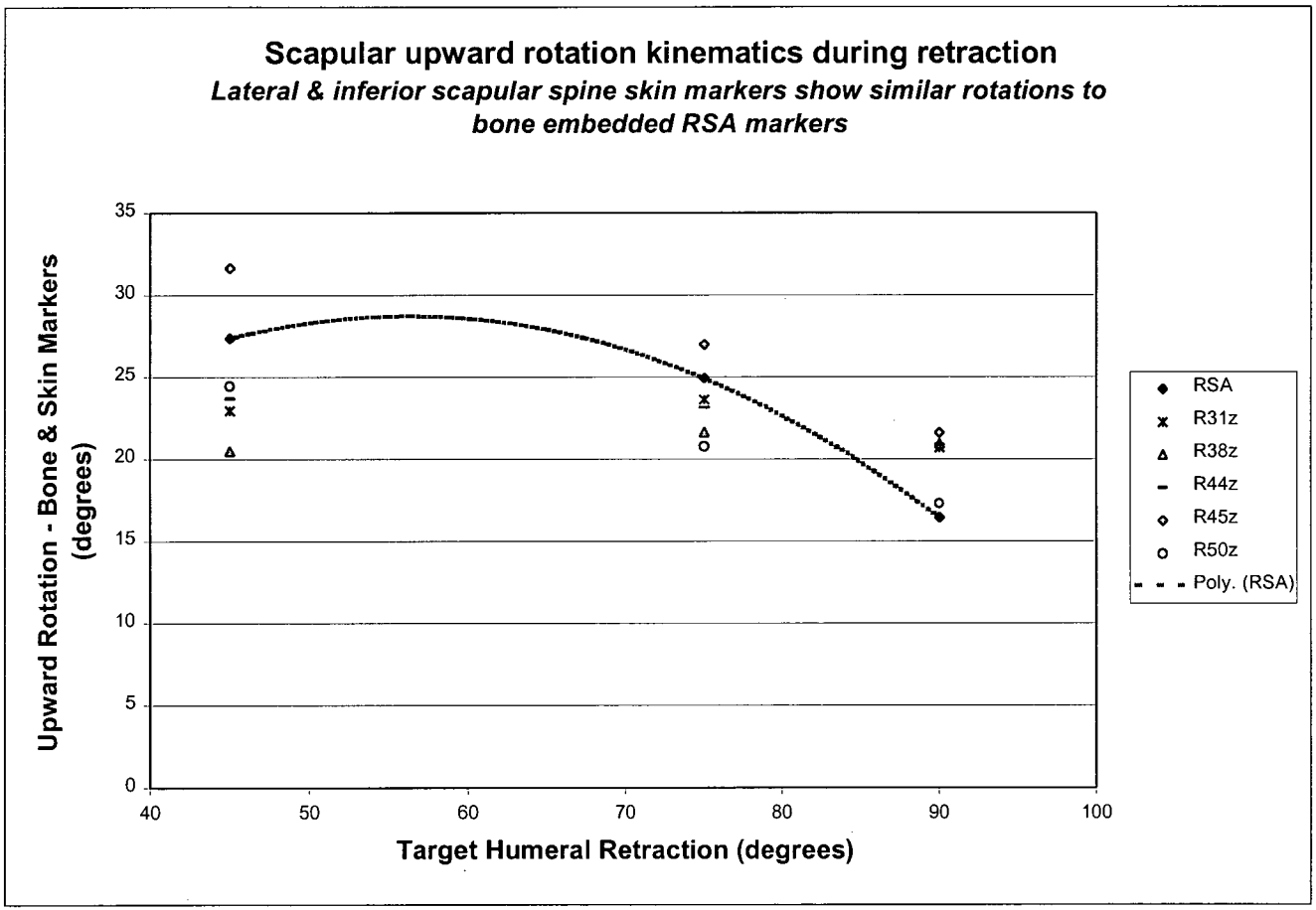

Figure D-19: Scapular upward rotation - lateral and inferior scapular spine patches

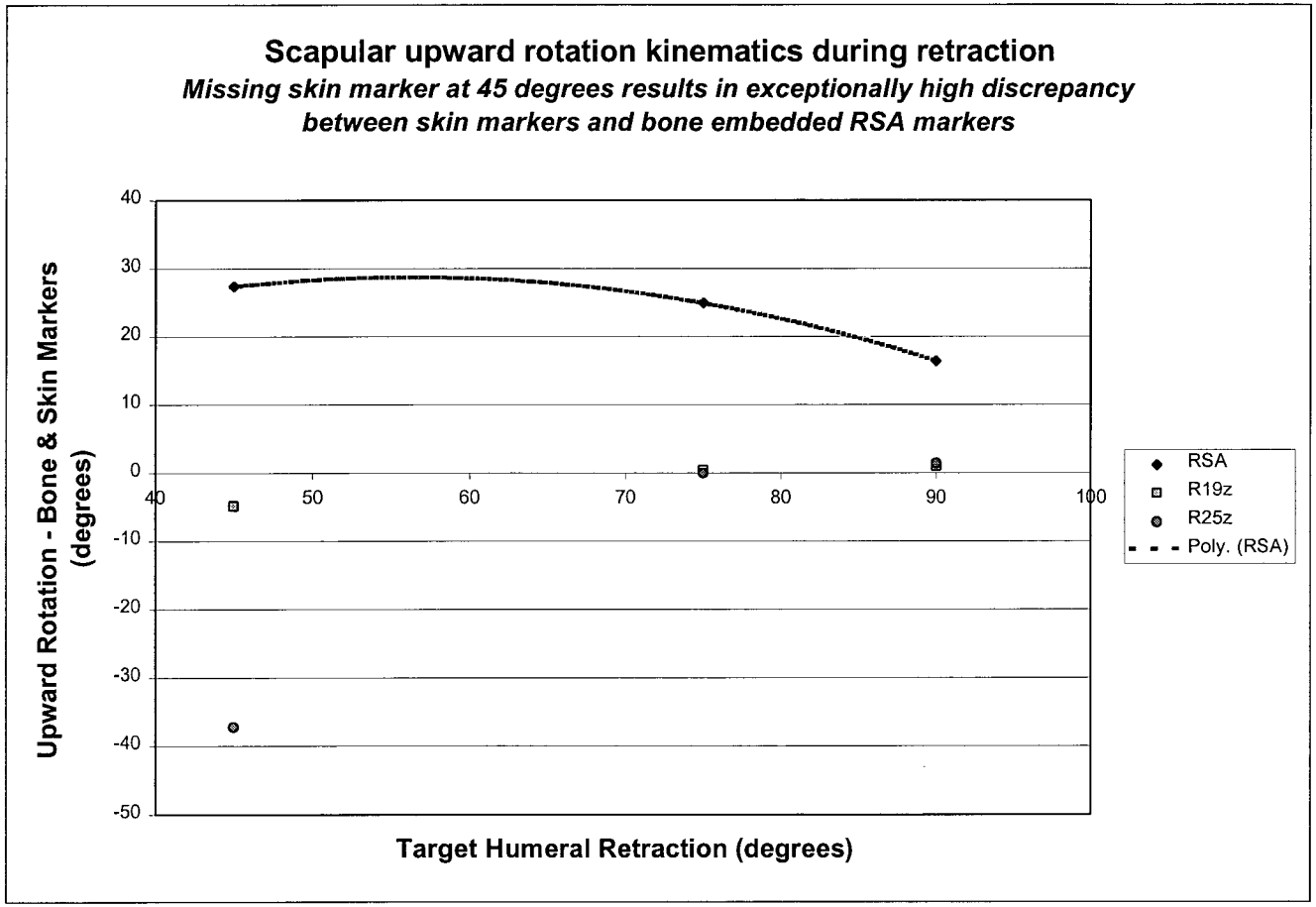

Figure D-20: Scapular upward rotation error due to missing skin markers

A single missing skin marker on the medial boundary of the skin grid caused the two affected patches (19 and 25) to have significantly higher errors.

D.2.2 Ry – Scapular internal/external rotation

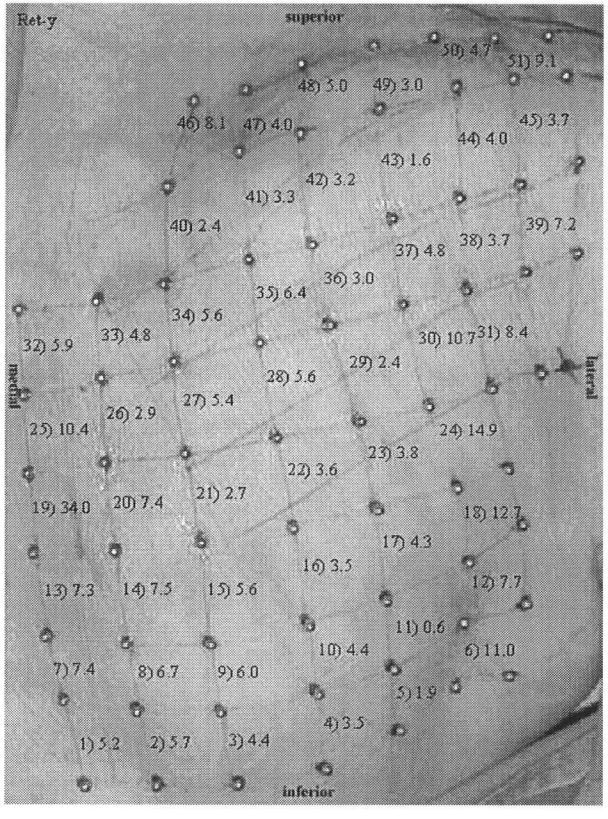

Figure D-21: Scapular internal/external rotation RMSE during humeral retraction

RMSE (°) computed for 45°, 75°, 90°, of targeted humeral retraction using rotations measured with embedded bone RSA markers as the gold-standard.

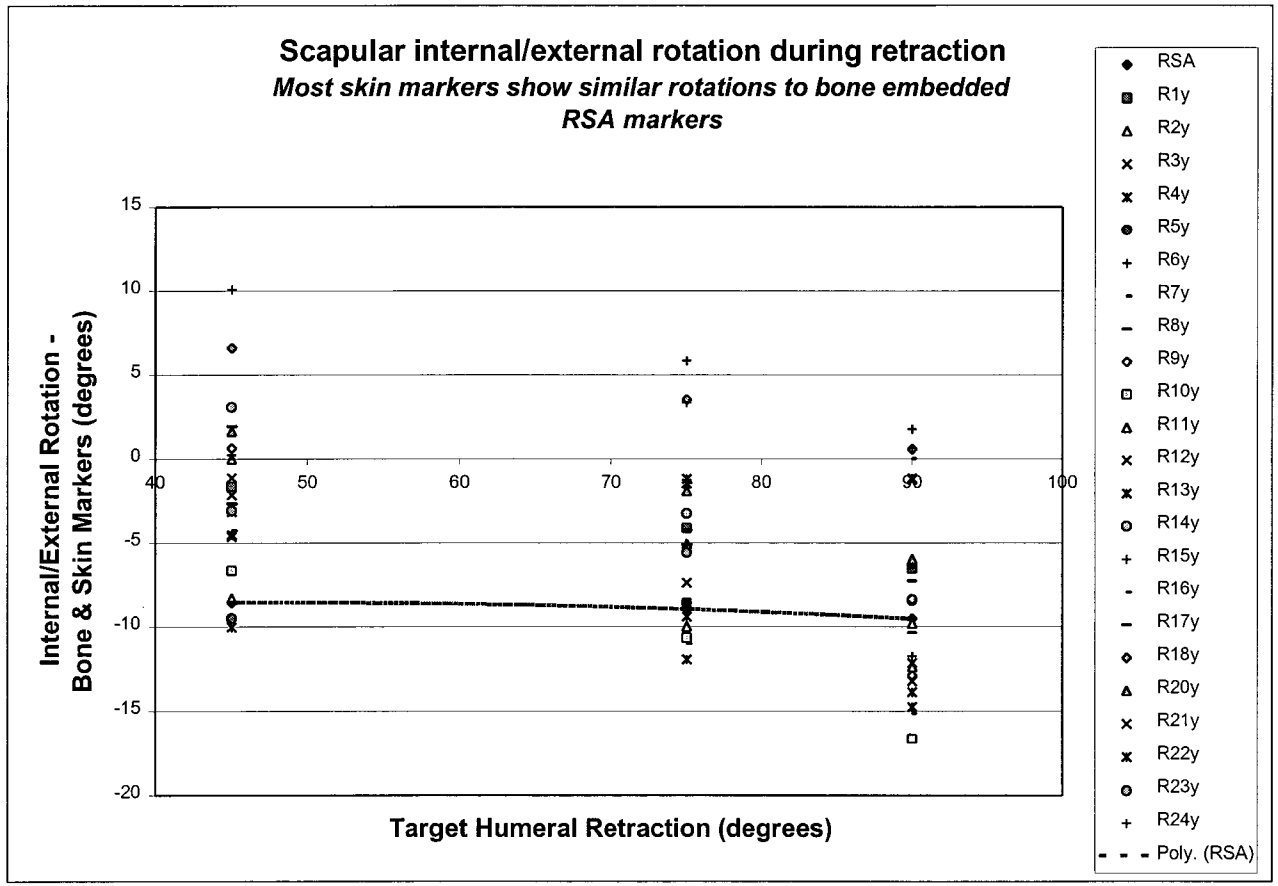

Figure D-22: Scapular internal/external rotation – patches 1 to 24

No general trend was observed indicating a particular region of skin markers was the best candidate for measuring internal/external scapular rotation. Almost all regions were found to have an RMSE<10°.

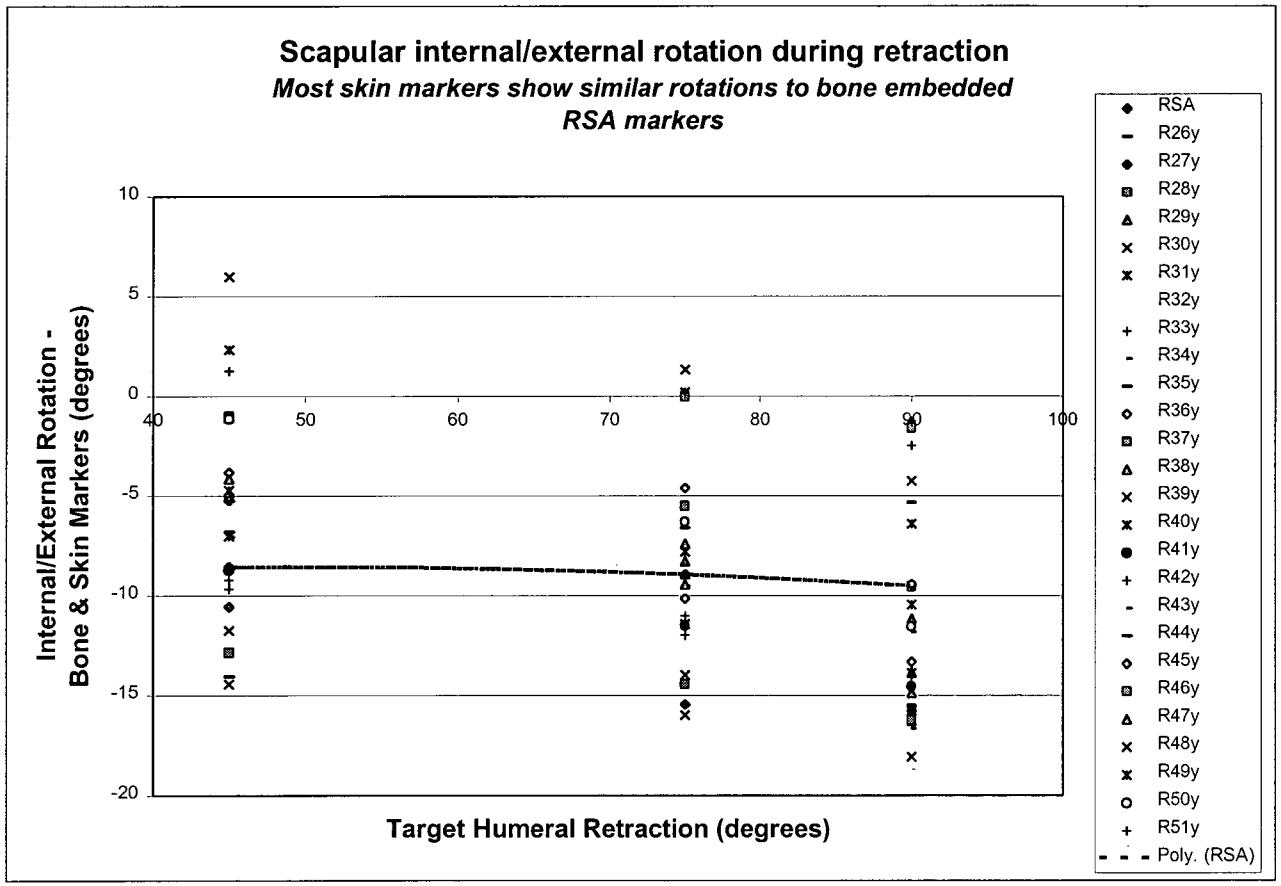

Figure D-23: Scapular internal/external rotation – patches 26 to 51

No general trend was observed indicating a particular region of skin markers was the best candidate for measuring internal/external scapular rotation. Almost all regions were found to have an RMSE<10°.

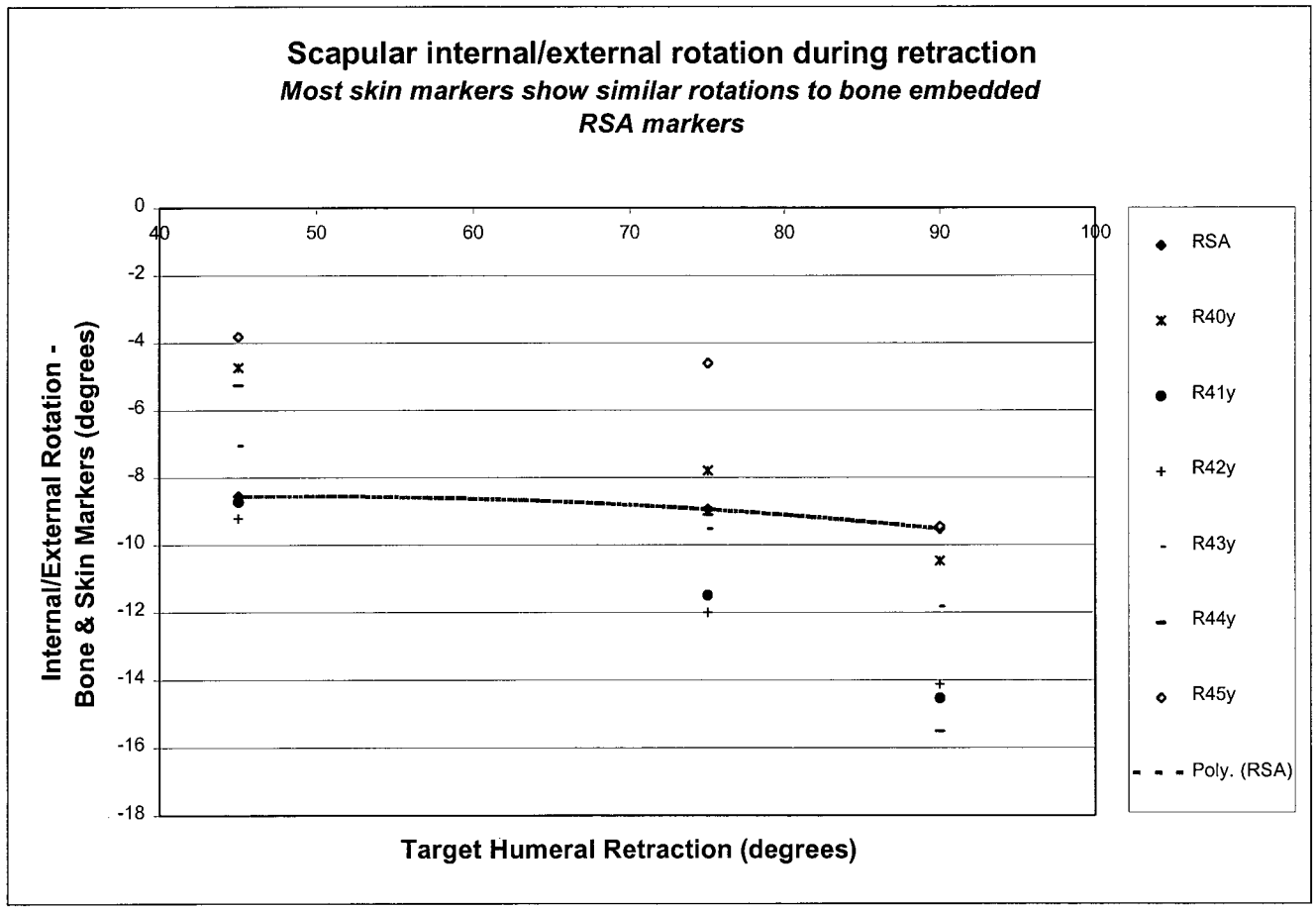

Figure D-24: Scapular internal/external rotation – patches on scapular spine
Most skin patches showed similar results with RSA. As with humeral elevation, the patches on the scapular spine showed good results.

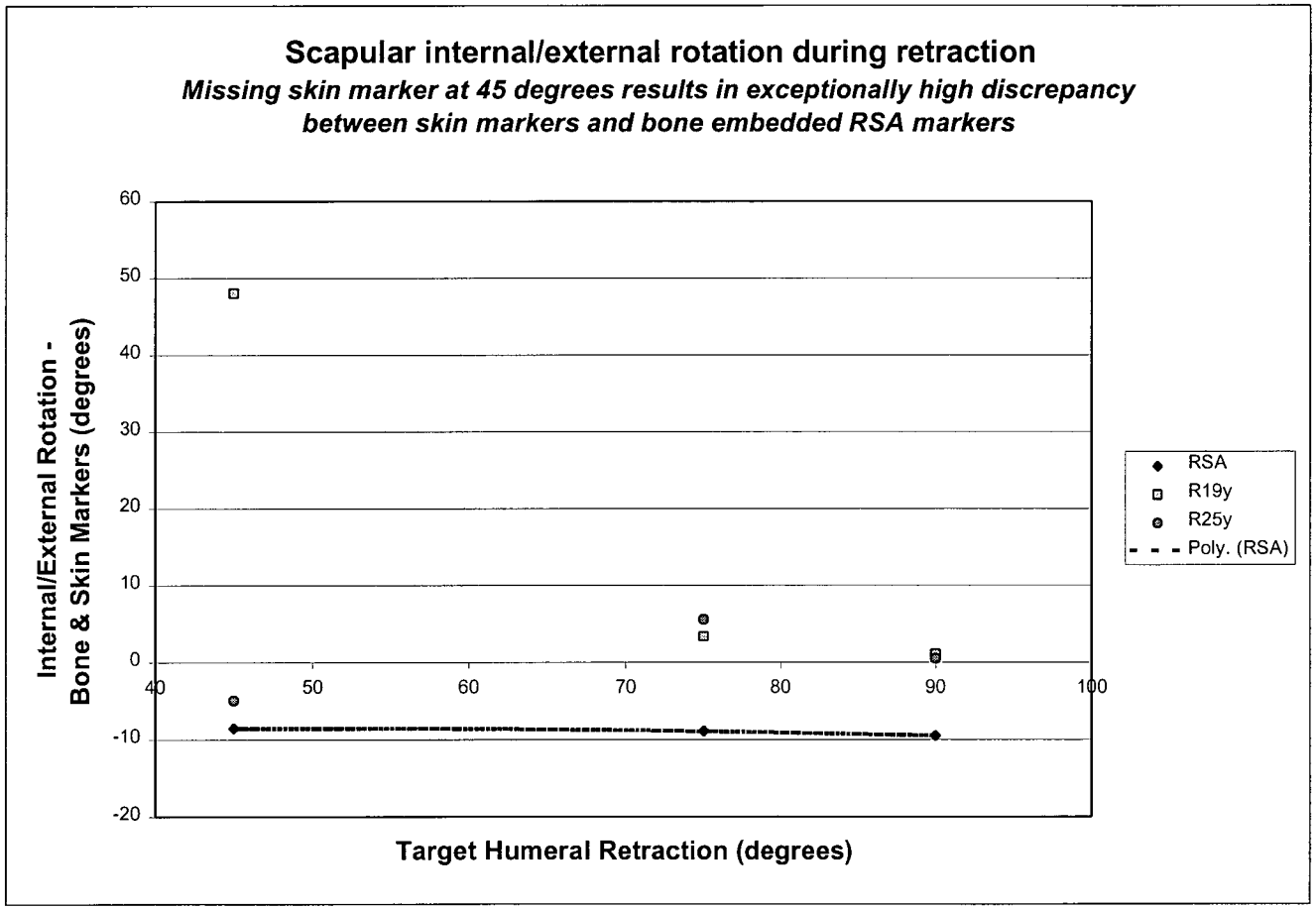

Figure D-25: Scapular internal/external rotation error due to missing skin markers
A single missing skin marker on the medial boundary of the skin grid caused the two affected patches (19 and 25) to have significantly higher errors.

D.2.3 Rx – Scapular Tipping

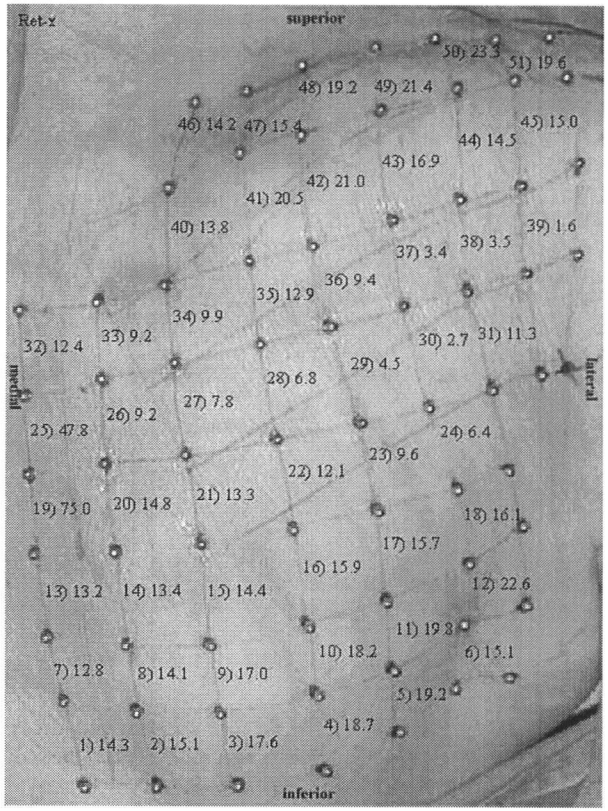

Figure D-26: Scapular tipping RMSE during humeral retraction

RMSE (°) computed for 45°, 75°, 90°, of targeted humeral retraction using rotations measured with embedded bone RSA markers as the gold-standard.

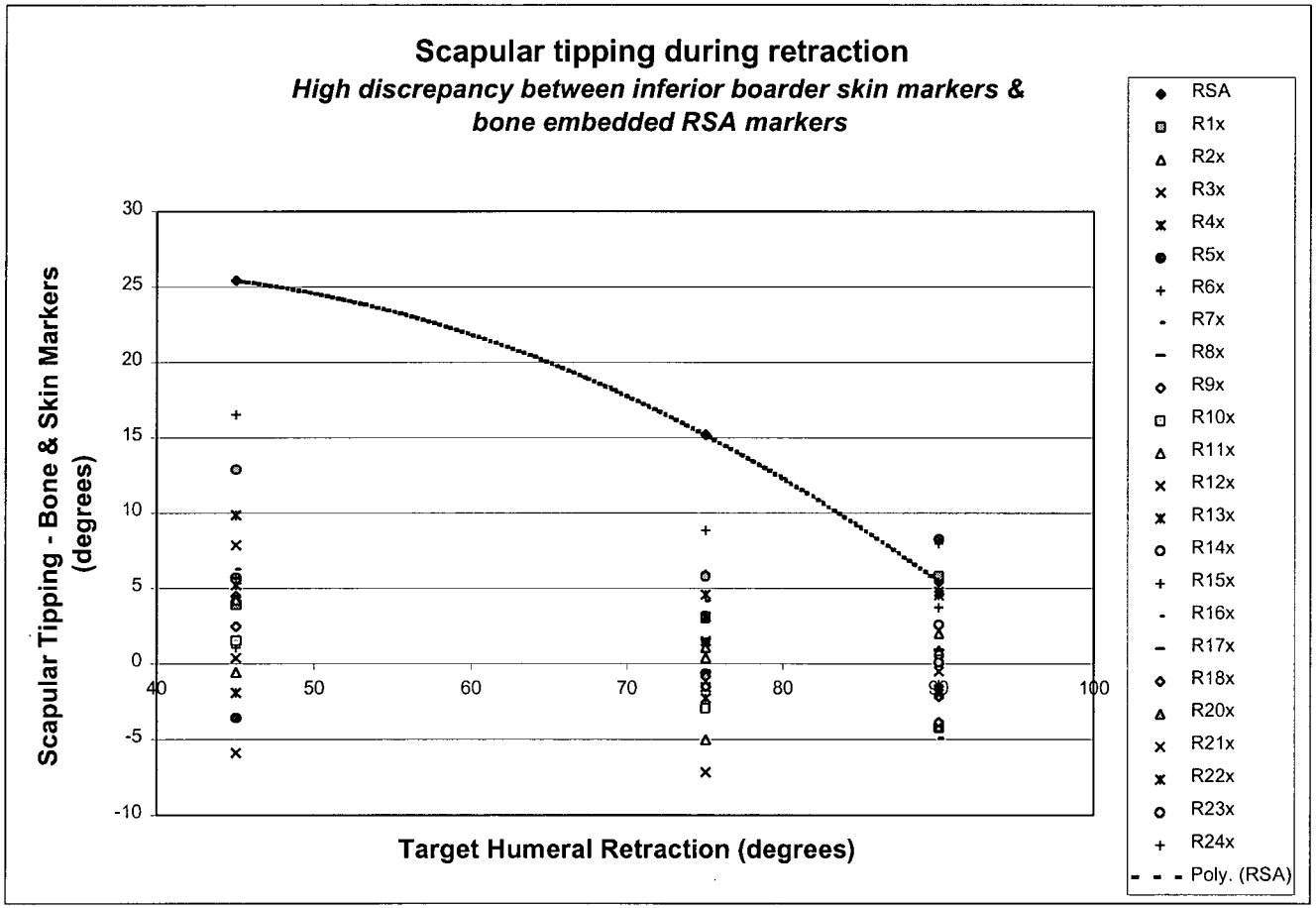

Figure D-27: Scapular tipping - inferior boarder patches

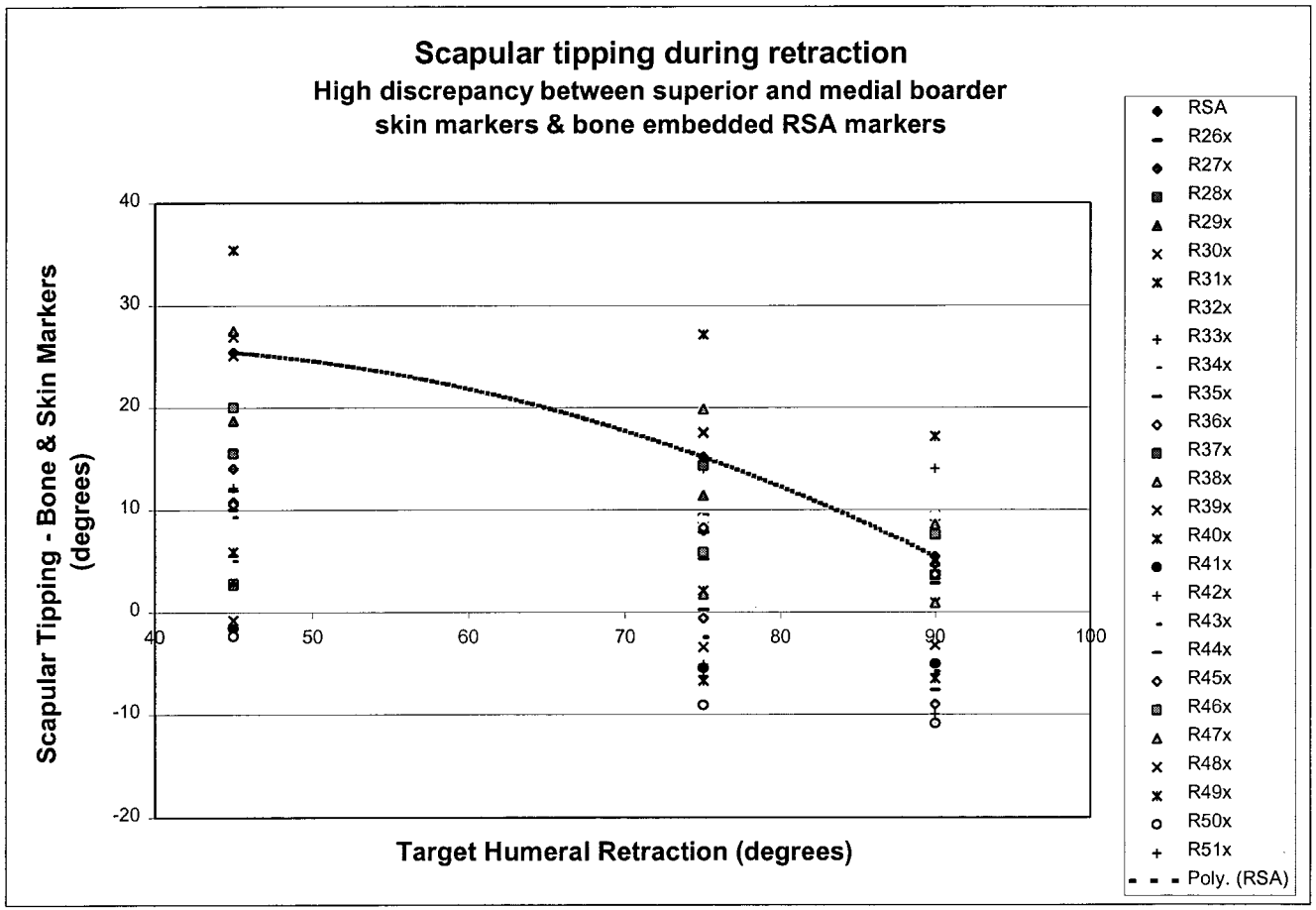

Figure D-28: Scapular tipping - superior and medial boarder patches

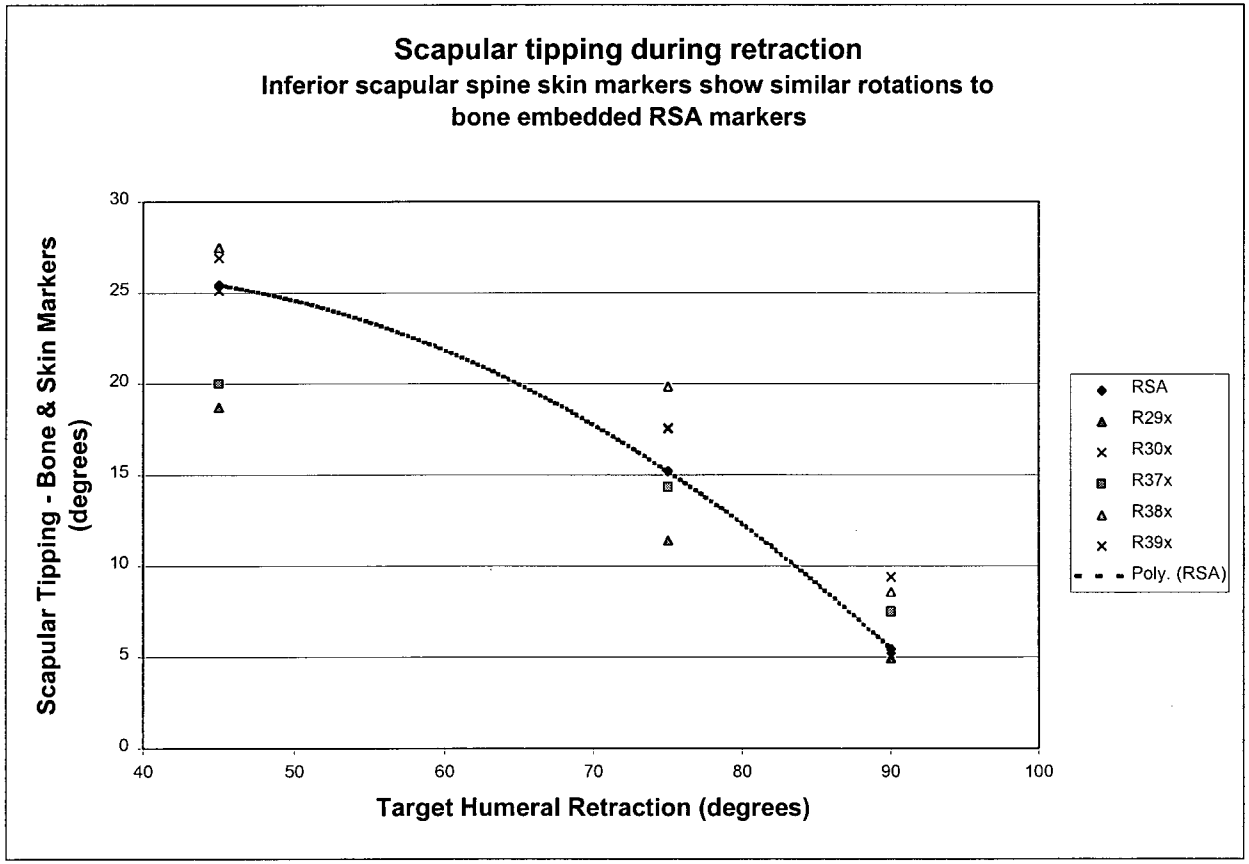

Figure D-29: Scapular tipping - inferior scapular spine patches

Patches 29, 30, 37, 38 and 39 lie in the region just inferior to the scapular spine. These patches showed the most similar rotations about the x-axis (axis along the scapular spine). The patches directly on the scapular spine show poorer results for rotation about this (scapular spine) axis. This intuitively makes sense since the scapular spine patches are measuring the rotation of a cylindrical-like structure beneath the skin.

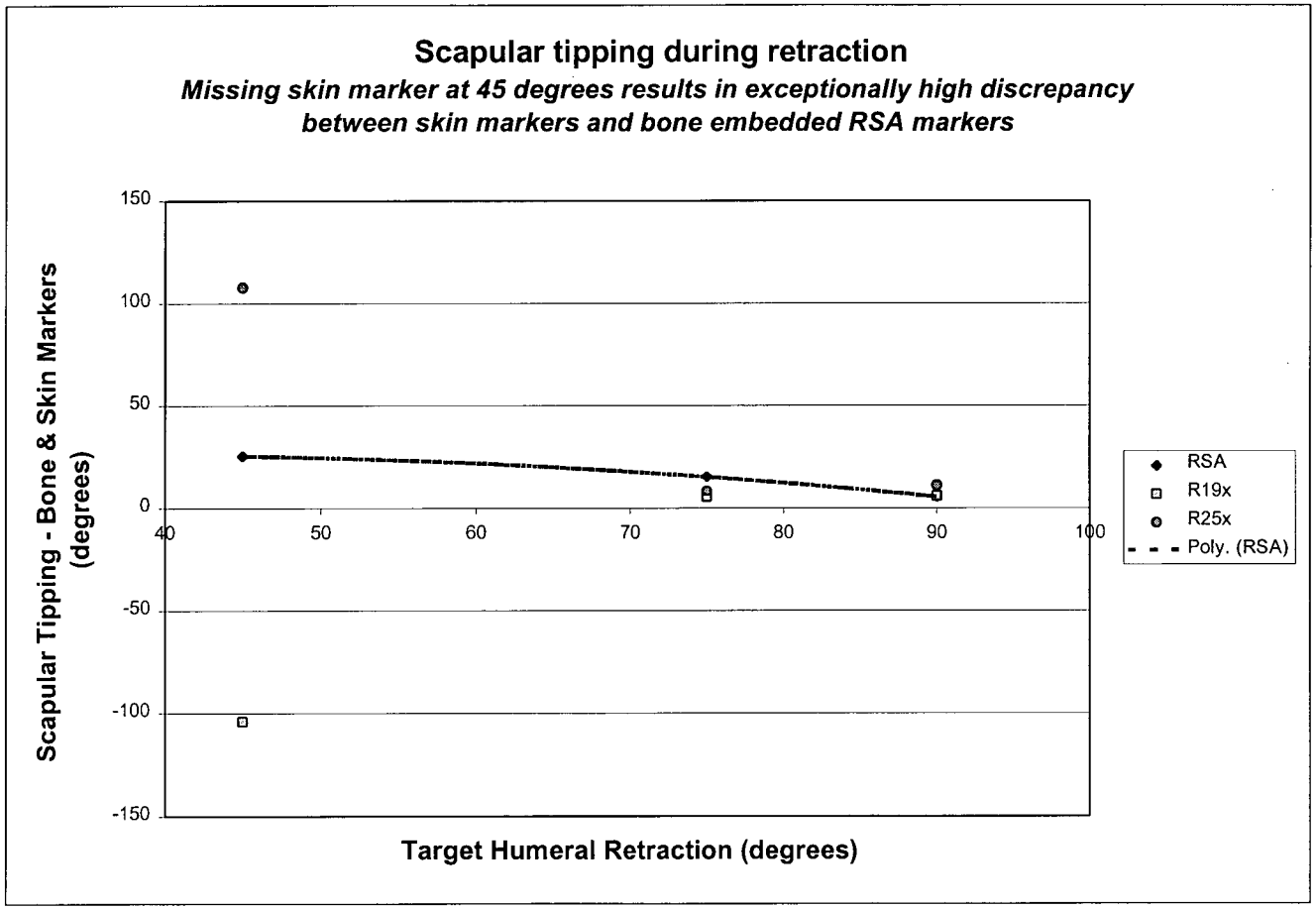

Figure D-30: Scapular tipping error due to missing skin markers

A single missing skin marker on the medial boundary of the skin grid caused the two affected patches (19 and 25) to have significantly higher errors.

Appendix E

CADAVER KINEMATICS - MATLAB CODE

E.1 matchStereoPts

```
% matchStereoPts   - script outputs the error vector for each combination
%                   of points from the input
%                 - ideally, with zero measurement error, the size
%                   of the error vector is zero
%
% Revision History:
%     Feb. 13, 2001 - Anthony Choo - initial version
%
clear all;close all;

TRUE = 1; FALSE = 0;

% call scripts to initialize the known coordinates of the calibration cage
DefineDOERCage10ForRetReCal;   % initializes DOERCage10
% call script to initialize measured calibration coordinates from each film
RetReCal;  % initialize calibF1 and calibF2

% 3D DLT transformation to image volume
[dlt_f1, res_f1] = dltfu(DOERCage10, calibF1);
[dlt_f2, res_f2] = dltfu(DOERCage10, calibF2);
DLT_Coeff = [dlt_f1, dlt_f2];
DLT_Coeff
res_f1
res_f2

for exposure = 1:8
  % initialize object points from Film1 and Film2
  % initialize F1pts and F2pts
switch exposure
case 1
  Ab0;
  saveFile = 'Ab0pairs.csv';
case 2
  Ab45;
  saveFile = 'Ab45pairs.csv';
case 3
  Ab90;
  saveFile = 'Ab90pairs.csv';
case 4
  Ab135;
  saveFile = 'Ab135pairs.csv';
case 5
  AbMax;
  saveFile = 'AbMaxpairs.csv';
case 6
  Ret45;
  saveFile = 'Ret45pairs.csv';
case 7
  Ret75;
  saveFile = 'Ret75pairs.csv';
case 8
  Ret90;
  saveFile = 'Ret90pairs.csv';
otherwise
  disp('Error: Undefined case');
end

% convert to local coordinates
```

```
  %[F1pts F2pts]
  F1pts = TLocal(F1pts);
  F2pts = TLocal(F2pts);
  %[F1pts F2pts]

  % open file
  savePath = [pwd '\'];
  saveMode = 'w';
  FID = fopen([savePath saveFile],saveMode);

  % compute 3D coords along with residual error to determine
  % if pairings are correct
  fprintf(FID,'F1-F2,x,y,z,res\n');
  for i = 1:size(F1pts,1)
  for j = 1:size(F2pts,1)
    skipFound = FALSE;
    for k = 1:size(skipF1,2)
    if skipF1(k) == i
      skipFound = TRUE;
      break;
    end
    end
    for k = 1:size(skipF2,2)
    if skipF2(k) == j
      skipFound = TRUE;
      break;
    end
    end
    if skipFound == FALSE
    objPts_Films = [F1pts(i,:) F2pts(j,:)];
    H = reconfu(DLT_Coeff,objPts_Films);
    fprintf(FID,'p%i-%i,%f,%f,%f,%f\n', i,j,H(1),H(2),H(3),H(4));
    else
    fprintf(FID,'p%i-%i,,,,\n', i,j);
    end
  end
  end
  fclose(FID);
end
```

E.2 assignPts

```
% assignPts - script outputs the distance between DH and CR points to all
%          all other pts.
%        - assumes DH is first and CR is 8th point in pts matrix
%
% Revision History:
%     Feb. 13, 2001 - Anthony Choo - initial version
%
clear all;close all;

TRUE = 1; FALSE = 0;

for exposure = 1:8
  % initialize pts
  switch exposure
  case 1
    Ab03D;
    saveFile = 'Ab0Dists.csv';
  case 2
    Ab453D;
    saveFile = 'Ab45Dists.csv';
  case 3
    Ab903D;
    saveFile = 'Ab90Dists.csv';
  case 4
    Ab1353D;
    saveFile = 'Ab135Dists.csv';
  case 5
    AbMax3D;
    saveFile = 'AbMaxDists.csv';
  case 6
    Ret453D;
    saveFile = 'Ret45Dists.csv';
  case 7
    Ret753D;
    saveFile = 'Ret75Dists.csv';
  case 8
    Ret903D;
    saveFile = 'Ret90Dists.csv';
```

```
  otherwise
    disp('Error: Undefined case');
  end

  % initialize DH and CR
  DH = pts(1,:)';
  CR = pts(8,:)';

  % open file
  savePath = [pwd '\'];
  saveMode = 'w';
  FID = fopen([savePath saveFile],saveMode);

  % compute DH and CR distances to other points
  fprintf(FID,'i,DHD-i,DCR-i\n');
  for i = 2:7
  skipFound = FALSE;
  for k = 1:size(skipPts,2)
    if i == skipPts
    skipFound = TRUE;
    fprintf(FID,'%i,%i-skip,%i-skip\n',i,i,i);
    break;
    end
  end
  if skipFound == FALSE
    pt = pts(i,:)';
    tmp=[DH-pt].^2;
    DDH = sqrt(sum(tmp));
    tmp=[CR-pt].^2;
    DCR = sqrt(sum(tmp));
    %[DDH DCR]
    fprintf(FID,'%i,%f,%f\n',i,DDH,DCR);
  end
  end
  fclose(FID);
end
```

E.3 solveKin

```
% solvekin - script that processes the 3D data from Cadaver experiment
%        and outputs the intersegmental rotations between
%        thorax and scapula - bone markers
%        scapula and humerus - bone markers
%        thorax and scapula - via skin markers
%
% Revision History:
%     Feb. 23, 2001 - Anthony Choo - initial version
%     Mar. 6, 2001 - Anthony Choo - T2 in Ab45 excluded
%
clear all;close all;

TRUE = 1; FALSE = 0;

% Digitization to setup local coordinate frames
Ab0obj; % data in format: x,y,z,residual where each row is a point
numPts = size(pts,1);
pts0 = [pts(:,1:3)]';  % strip off residuals and make each pt a column
HumerusAb0obj;  % digitized humerus points
Hpts = [Hpts(:,1:3)]';  % strip off residuals and make each pt a column

defMarkers; % define which points in pts correspond to which bodies

% define relationship between marker carrier and local anatomical frames
% i.e. T_Anatomical_Marker
%
% BY DEFINITION of SVD, the orientation of the marker carriers
% will be initially coincident with that of the fixed coordinate system.
% (refer to tests, testrxyz1.m and testrxyz2.m)
% Hence, T_Anatomical_Marker = T_Anatomical_Cage denoted T_AC
%
% using:
% T = mkT(origin, lockedAxisPts, floatAxisPts, lockedAxis, floatAxis)
%
% Thoracic frame (T) and thoracic marker carriers (Tm)
% (T_CT = T_TmT - see above)
T_TmT = mkT(pts0(:,Torigin)',pts0(:,Txaxis)',pts0(:,Tyaxis)','x','y');
T_TTm = inv(T_TmT);
```

```
% Scapular frame (S) and scapular marker carriers (Sm)
% (bone markers or skin markers: markers aligned with fixed)
% (T_CS = T_SmS - see above)
T_SmS = mkT(pts0(:,Sorigin)',pts0(:,Sxaxis)',pts0(:,Syaxis)','x','y');

% Humeral frame and humeral (bone) marker carriers
% (T_CH = T_HmH - see above)
T_HmH = mkT(Hpts(:,Horigin)',Hpts(:,Hyaxis)',Hpts(:,Hzaxis)','y','z');

for exposure = 1:8
  % reinitialize markers due to exclusions
  defMarkers; % define which points in pts correspond to which bodies

  % initialize object points
  switch exposure
  case 1
    Ab0obj;
    saveFile = 'Ab0.kin';
    % exclude A3 (row 8) - always excluded because
    % this is reference image
  case 2
    Ab45obj;
    saveFile = 'Ab45.kin';
    % exclude H2 (row 11)
    % * new * found necessary to exclude T2 (row 2)
    % refer to "T2 problem.txt" in Recon
    % original Ab45.kin renamed "old T2 Ab45.kin"
    Hm = [9 10 12];
    Tm = [1 3 4];
  case 3
    Ab90obj;
    saveFile = 'Ab90.kin';
    % exclude T2 (row 2), H2 (row 11)
    Tm = [1 3 4];
    Hm = [9 10 12];
  case 4
    Ab135obj;
    saveFile = 'Ab135.kin';
    % exclude T2 (row 2)
    Tm = [1 3 4];
  case 5
    AbMaxobj;
    saveFile = 'AbMax.kin';
    % exclude T2 (row 2)
    Tm = [1 3 4];
  case 6
    Ret45obj;
    saveFile = 'Ret45.kin';
    % exclude H3 (row 12), skin 3 (row 15)
    Hm = [9 10 11];
    PATCH19 = [4 10 11];
    PATCH25 = [2 9 10];
  case 7
    Ret75obj;
    saveFile = 'Ret75.kin';
    % exclude H3 (row 12)
    Hm = [9 10 11];
  case 8
    Ret90obj;
    saveFile = 'Ret90.kin';
  otherwise
    disp('Error: Undefined case');
  end

  pts = [pts(:,1:3)]';   % strip off residuals and make each pt a column
  % Algorithm for each marker carrier
  %   1) All relations between marker and anatomical
  %      frames already computed
  %   2) Compute motion of thoracic marker carrier
  %   3) compute intersegmental transformation
  %      between thoracic and segment marker carriers (see note)
  %   4) convert transformation matrix to anatomical transformations
  %   5) solve for cardan angles
  %
  % Note:
  %   compute intersegmental transformation matrix between
  %   each marker carrier and the thoracic marker carrier
  %   soder(B1,B2) computes T_B1B2 but since
  %   SVD assumes orientation of B1 coincident with
  %   fixed (cage) frame, soder(B1,B2) gives T_CB2
  %   soder(thoracic_initial,thoracic_move) gives T_CTm
```

```
%    soder(part_initial,part_move) gives T_CPm
%    want: T_TmC*T_CPm
%
% 2) Thoracic
numPts = size(Tm,2);   % a row vector
[T_CTm TmRes] =
soder([reshape(pts0(:,Tm),1,3*numPts);reshape(pts(:,Tm),1,3*numPts)]);
T_TmC = inv(T_CTm);

% for debugging
disp(['T_CTm ' saveFile]);
rxyzsolv(T_CTm)

% Scapular Bone
numPts = size(Sm,2);   % a row vector
[T_CSm SmRes] =
soder([reshape(pts0(:,Sm),1,3*numPts);reshape(pts(:,Sm),1,3*numPts)]);
T_TmSm = T_TmC*T_CSm;  % 3) markers
T_TS = T_TTm*T_TmSm*T_SmS;  % 4) anatomical
cardanS = rxyzsolv(T_TS);  % 5) cardan angles
% Humeral Bone
numPts = size(Hm,2);   % a row vector
[T_CHm HmRes] =
soder([reshape(pts0(:,Hm),1,3*numPts);reshape(pts(:,Hm),1,3*numPts)]);
T_TmHm = T_TmC*T_CHm;  % 3) markers
%[n,point,phi,t] = screw(T_TmHm);
T_TH = T_TTm*T_TmHm*T_HmH;  % 4) anatomical
cardanH = rxyzsolv(T_TH);  % 5) cardan angles
%
% Scapular Skin Patches
for i = 1:51
   eval(['numPts = size(PATCH' num2str(i) ',2);']);   % a row vector
eval(['[T_CPatch Patch' num2str(i) ...
   'Res] = soder([reshape(pts0(:,PATCH' num2str(i) '),1,' num2str(3*numPts) ...
   ');reshape(pts(:,PATCH' num2str(i) '),1,' num2str(3*numPts) ')]);']);
T_TmPATCH = T_TmC*T_CPatch; % 3) markers
T_TSpatch = T_TTm*T_TmPATCH*T_SmS;  % 4) anatomical
eval(['cardanPatch' num2str(i) ' = rxyzsolv(T_TSpatch);']); % 5) cardan angles
end

% write results to file
% open file
savePath = [pwd '\'];
saveMode = 'w';
FID = fopen([savePath saveFile],saveMode);

fprintf(FID,'RzRyRx sequence\n');
fprintf(FID,'Tag,theta_x,theta_y,theta_z,Hx,Hy,Hz,res\n');
fprintf(FID,'T,,,,,,,%f\n',TmRes);
fprintf(FID,'TS,%f,%f,%f,%f,%f,%f,%f\n',cardanS,SmRes);
fprintf(FID,'TH,%f,%f,%f,%f,%f,%f,%f\n',cardanH,HmRes);
for i = 1:51
fprintf(FID,'TPatch%i,%f,%f,%f,%f,%f,%f,%f\n', ...
   i,eval(['cardanPatch' num2str(i)]), ...
   eval(['Patch' num2str(i) 'Res']));
end
fclose(FID);
end
```

E.4 mkT

```
% mkT  - make Transformation matrix
%
% T = mkT(origin, lockedAxisPts, floatAxisPts, lockedAxis, floatAxis)
%
% Inputs:
%  origin - (x,y,z) of T2 origin in T1 frame
%  lockedAxisPts - (x1,y1,z2; x2,y2,z2) of axis to be fixed
%              direction is from [x1,y1,z1] to [x2,y2,z2]
%  floatAxisPts - (x1,y1,z2; x2,y2,z2) of axis to be crossed with
%                 lockedAxis to get perpendicular third axis
%              direction is from [x1,y1,z1] to [x2,y2,z2]
%  lockedAxis - 'x','y','z'
%  floatAxis - 'x','y','z'
%
% Outputs:
%  T - transformation matrix T12 (T2 with respect to T1)
%
% Revision History:
%      Feb. 18 2001 - Anthony Choo - initial version
```

```
%
function T = mkT(origin, lockedAxisPts, floatAxisPts, lockedAxis, floatAxis)

locked = lockedAxisPts(2,:) - lockedAxisPts(1,:);
locked = [locked./norm(locked)]';
float = floatAxisPts(2,:) - floatAxisPts(1,:);
float = [float./norm(float)]';

lf = [lockedAxis floatAxis];
switch lf
   case 'xy'
      X = locked;
      Z = cross(X,float);
   Y = cross(Z,X);
   case 'xz'
      X = locked;
      Y = cross(float,X);
   Z = cross(X,Y);
   case 'yx'
      Y = locked;
      Z = cross(float,Y);
   X = cross(Y,Z);
   case 'yz'
      Y = locked;
      X = cross(Y,float);
   Z = cross(X,Y);
   case 'zx'
      Z = locked;
      Y = cross(Z,float);
   X = cross(Y,Z);
   case 'zy'
      Z = locked;
      X = cross(float,Z);
   Y = cross(Z,X);
   case otherwise
      disp('Error: Locked and float axes specified incorrectly.');
   T = [NaN NaN NaN NaN; NaN NaN NaN NaN; NaN NaN NaN NaN; NaN NaN NaN
NaN];
   return;
end

T = [X Y Z origin'; 0 0 0 1];
```

E.5 defMarkers

```
% Define Markers - define which points correspond to which bodies
%
% Digitized Points
xlsOffset = 12;
% Thoracic Frame
Txaxis = [16 52]+xlsOffset;% skin points 16 and 52
Tyaxis = [32 16]+xlsOffset;% skin points 32 and 16
Torigin = [1];     % Thoracic carrier 1
% Scapular Frame
Sxaxis = [16 52]+xlsOffset;% skin points 16 and 52
Syaxis = [32 16]+xlsOffset;% skin points 32 and 16
Sorigin = [16]+xlsOffset;     % skin point 16
% Humeral Frame
% poor digitization data, use yaxis as locked axis
Hzaxis = [2 1];% lesser tuberosity to greater tuberosity
Hyaxis = [1 3];% greater tuberosity to deltoid tuberosity
Horigin = [1];% greater tuberosity

% Thoracic Frame Markers
Tm = [1 2 3 4];

% Scapular Frame Markers
% Sm = [5 6 7 8]; A3 (8) is missing from Ab0
Sm = [5 6 7];

% Humeral Frame Markers
Hm = [9 10 11 12];

% Patches
PATCH1 = [6 7 13 14] + xlsOffset;
PATCH2 = [13 14 22 23] + xlsOffset;
PATCH3 = [22 23 31 32] + xlsOffset;
PATCH4 = [31 32 40 41] + xlsOffset;
PATCH5 = [40 41 49 50] + xlsOffset;
PATCH6 = [49 50 58 59] + xlsOffset;
```

```
PATCH7 = [5 6 12 13] + xlsOffset;
PATCH8 = [12 13 21 22] + xlsOffset;
PATCH9 = [21 22 30 31] + xlsOffset;
PATCH10 = [30 31 39 40] + xlsOffset;
PATCH11 = [39 40 48 49] + xlsOffset;
PATCH12 = [48 49 57 58] + xlsOffset;
PATCH13 = [4 5 11 12] + xlsOffset;
PATCH14 = [11 12 20 21] + xlsOffset;
PATCH15 = [20 21 29 30] + xlsOffset;
PATCH16 = [29 30 38 39] + xlsOffset;
PATCH17 = [38 39 47 48] + xlsOffset;
PATCH18 = [47 48 56 57] + xlsOffset;
PATCH19 = [3 4 10 11] + xlsOffset;
PATCH20 = [10 11 19 20] + xlsOffset;
PATCH21 = [19 20 28 29] + xlsOffset;
PATCH22 = [28 29 37 38] + xlsOffset;
PATCH23 = [37 38 46 47] + xlsOffset;
PATCH24 = [46 47 55 56] + xlsOffset;
PATCH25 = [2 3 9 10] + xlsOffset;
PATCH26 = [9 10 18 19] + xlsOffset;
PATCH27 = [18 19 27 28] + xlsOffset;
PATCH28 = [27 28 36 37] + xlsOffset;
PATCH29 = [36 37 45 46] + xlsOffset;
PATCH30 = [45 46 54 55] + xlsOffset;
PATCH31 = [54 55 63 64] + xlsOffset;
PATCH32 = [1 2 8 9] + xlsOffset;
PATCH33 = [8 9 17 18] + xlsOffset;
PATCH34 = [17 18 26 27] + xlsOffset;
PATCH35 = [26 27 35 36] + xlsOffset;
PATCH36 = [35 36 44 45] + xlsOffset;
PATCH37 = [44 45 53 54] + xlsOffset;
PATCH38 = [53 54 62 63] + xlsOffset;
PATCH39 = [62 63 67 68] + xlsOffset;
PATCH40 = [16 17 25 26] + xlsOffset;
PATCH41 = [25 26 34 35] + xlsOffset;
PATCH42 = [34 35 43 44] + xlsOffset;
PATCH43 = [43 44 52 53] + xlsOffset;
PATCH44 = [52 53 61 62] + xlsOffset;
PATCH45 = [61 62 66 67] + xlsOffset;
PATCH46 = [15 16 24 25] + xlsOffset;
PATCH47 = [24 25 33 34] + xlsOffset;
PATCH48 = [33 34 42 43] + xlsOffset;
PATCH49 = [42 43 51 52] + xlsOffset;
PATCH50 = [51 52 60 61] + xlsOffset;
PATCH51 = [60 61 65 66] + xlsOffset;
```

Appendix F

SLIP ESTIMATION WITH FOURIER ANALYSIS - MATLAB CODE

F.1 solveFFT

```
% solveFFT  - script which uses 2D FFT to solve for skin slippage rotation
%
% Revision History:
%      March 6, 2001 - Anthony Choo - initial version
%
clear all;close all;
TRUE = 1; FALSE = 0;

% Output Options
PLOT_WINDOWS = FALSE;
PLOT_SKIN = FALSE;
PLOT_SKIN_WINDOWED = FALSE;
PLOT_FFT = FALSE;
PLOT_FFT_REFLECTED = FALSE;

PLOT_DELTAK = FALSE;
PLOT_LOCUS = TRUE;
PLOT_HIST = TRUE;

SAVE_HIST = FALSE;

% Analysis Options
RUN_EXPOSURES = [1 2 3 4 5 6 7 8];
% Select
RUN_EXPOSURES = [1 2];
RUN_EXPOSURES = [2 3];
RUN_EXPOSURES = [3 4];
RUN_EXPOSURES = [4 5];
RUN_EXPOSURES = [1 6];
RUN_EXPOSURES = [6 7];
RUN_EXPOSURES = [7 8];

WINDOW_IMAGE = FALSE;
R_WINDOW = 0.9;

THETA_BINS = 360;
%SEPARATE_AXES = TRUE;   % consider Q1&Q3 separate from Q2&Q4
SEPARATE_AXES = FALSE;   % consider Q1&Q3 separate from Q2&Q4
% consider theta in vicinity of origin
RHO_MIN = 1;
RHO_MAX = 10;

INTERP_RES = 128;   % interpolation of skin data size
PADDED_SIZE = 512;
AXES_SIZE = 50;

USE_PADDED = TRUE;
%USE_PADDED = FALSE;

% Display options
FCLR = 'white';
ACLR = 'black';

if USE_PADDED == TRUE
```

```
   fftDim = PADDED_SIZE;
   midDim = PADDED_SIZE/2;
else
   fftDim = INTERP_RES;
   midDim = INTERP_RES/2;
end
AX = [midDim-AXES_SIZE:midDim+AXES_SIZE];

% source scripts for data points
dataPts = {'Ab0sp' 'Ab45sp' 'Ab90sp' 'Ab135sp' ...
        'AbMaxsp' 'Ret45sp' 'Ret75sp' 'Ret90sp'};
saveFiles = {'Ab0hist' 'Ab45hist' 'Ab90hist' 'Ab135hist' ...
        'AbMaxhist' 'Ret45hist' 'Ret75hist' 'Ret90hist'};

% optional
% DC offset to images
% scale images to min value

% ------------------------------------------------------------------------
% search for a square that fits X and Y for all images
% ------------------------------------------------------------------------
eval(dataPts{1});
minX = min(pts(:,1)); maxX = max(pts(:,1));
minY = min(pts(:,2)); maxY = max(pts(:,2));
for i = 2:8
   eval(dataPts{i});
   minX = min(minX,min(pts(:,1))); maxX = max(maxX,max(pts(:,1)));
   minY = min(minY,min(pts(:,2))); maxY = max(maxY,max(pts(:,2)));
end
maxX = ceil(maxX); minX = floor(minX);
maxY = ceil(maxY); minY = floor(minY);
if abs(maxX - minX) > abs(maxY - minY)
   filler = abs(maxX - minX) - abs(maxY - minY);
   minY = minY - floor(filler/2);
   maxY = maxY + filler - floor(filler/2);
else
   filler = abs(maxY - minY) - abs(maxX - minX);
   minX = minX - floor(filler/2);
   maxX = maxX + filler - floor(filler/2);
end

% define image window to round boundary and reduce DFT leakage
if WINDOW_IMAGE == TRUE
   % ideal cylinder
   [f1,f2] = freqspace(INTERP_RES,'meshgrid');
   Hd = zeros(INTERP_RES,INTERP_RES); d = sqrt(f1.^2 + f2.^2) < R_WINDOW;
   Hd(d) = 1;
   % use hamming to window cylinder's response
   h = fwind1(Hd,hamming(INTERP_RES));
   [H,f1,f2] = freqz2(h,[INTERP_RES INTERP_RES]);
   %
   if PLOT_WINDOWS == TRUE
      [fH aH] = Figure3D;
      mesh(f1,f2,Hd)
      title(['ideal cylinder'],'Color','k');
      %
      [fH aH] = Figure3D;
      mesh(f1,f2,abs(H));
      axis([-1 1 -1 1 0 1.2]);
      title(['cylinder with hamming window'],'Color',ACLR);
      axis('tight');
      set(fH,'Color',FCLR); set(aH,'Color',FCLR);
      set(aH,'XColor',ACLR,'YColor',ACLR,'ZColor',ACLR);
      set(aH,'GridLineStyle',':');
   end
end

% ------------------------------------------------------------------------
% interpolate data points, compute fft and generate plots
% ------------------------------------------------------------------------
fHs = []; aHs = [];
firstFlag = TRUE;
for exposure = RUN_EXPOSURES
   % interpolation
   % use power of 2 image size for faster fft
   [X Y] = meshgrid(linspace(minX,maxX,INTERP_RES), ...
            linspace(minY,maxY,INTERP_RES));
   eval(dataPts{exposure});
   pts = pts(13:80,:);    % strip of marker carriers
                  % leave only skin markers

   Z = griddata(pts(:,1),pts(:,2),pts(:,3),X,Y,'cubic');
```

```
if PLOT_SKIN == TRUE
   [fH aH] = Figure3D;
   fHs = [fHs fH]; aHs = [aHs aH];

   title(['Skin' dataPts{exposure}],'Color',ACLR);
   %[dZdx, dZdy]=gradient(Z);
   %dZ=sqrt(dZdx.^2 + dZdy.^2);
   %surf(X,Y,Z,dZ);
   surf(X,Y,Z);
   colormap('copper');
   hold on;
   plot3(pts(:,1),pts(:,2),pts(:,3),'o');
   hold on;
   plot3(pts(1,1),pts(1,2),pts(1,3),'g*');
   axis('equal');
   set(fH,'Color',FCLR); set(aH,'Color',FCLR);
   set(aH,'XColor',ACLR,'YColor',ACLR,'ZColor',ACLR);
   set(aH,'GridLineStyle',':');
   axis([-100 100 -160 50 -20 50]);
end

% window image to round boundaries, reduce DFT leakage
if WINDOW_IMAGE == TRUE
   Z = Z.*abs(H);
end

if PLOT_SKIN_WINDOWED == TRUE
   [fH aH] = Figure3D;
   fHs = [fHs fH]; aHs = [aHs aH];

   title(['Windowed Skin' dataPts{exposure}],'Color',ACLR);
   %[dZdx, dZdy]=gradient(Z);
   %dZ=sqrt(dZdx.^2 + dZdy.^2);
   %surf(X,Y,Z,dZ);
   surf(X,Y,Z);
   colormap('copper');
   shading interp;
   hold on;
   plot3(pts(:,1),pts(:,2),pts(:,3),'o');
   hold on;
   plot3(pts(1,1),pts(1,2),pts(1,3),'g*');
   axis('equal');
   set(fH,'Color',FCLR); set(aH,'Color',FCLR);
   set(aH,'XColor',ACLR,'YColor',ACLR,'ZColor',ACLR);
   set(aH,'GridLineStyle',':');
end

% set NaN to zero
Z(isnan(Z)) = 0;

Tsample = (maxX - minX)/INTERP_RES; % sample period in mm
fsample = 1/Tsample;
if USE_PADDED == TRUE
   fftZ = fft2(Z,PADDED_SIZE,PADDED_SIZE); % use to pad fft
   % in terms of frequency
   %[fftX fftY] = meshgrid(linspace(-
fsample*PADDED_SIZE/2,fsample*PADDED_SIZE/2,PADDED_SIZE), ...
   %          linspace(-
fsample*PADDED_SIZE/2,fsample*PADDED_SIZE/2,PADDED_SIZE));
   % in terms of samples
   [fftX fftY] = meshgrid(linspace(-
PADDED_SIZE/2,PADDED_SIZE/2,PADDED_SIZE), ...
            linspace(-PADDED_SIZE/2,PADDED_SIZE/2,PADDED_SIZE));
else
   fftZ = fft2(Z);
   % in terms of frequency
   %[fftX fftY] = meshgrid(linspace(-
fsample*INTERP_RES/2,fsample*INTERP_RES/2,INTERP_RES), ...
   %          linspace(-
fsample*INTERP_RES/2,fsample*INTERP_RES/2,INTERP_RES));
   % in terms of samples
   [fftX fftY] = meshgrid(linspace(-INTERP_RES/2,INTERP_RES/2,INTERP_RES), ...
            linspace(-INTERP_RES/2,INTERP_RES/2,INTERP_RES));
end

% fftZ_0 = fftZ(1,1);  % this doesn't get the maximum value
            % perhaps add low pass filter
fftZ_0 = max(max(fftZ));
fftZ = fftshift(fftZ);

if firstFlag == FALSE   % subsequent images
   % unshift, reflect image about rows, reapply shift, shift reflected also
```

```
%  - can do after shifting but then difficult to find zero axis
numRows = size(fftZ,1);
fftZ = fftshift(fftZ);  % unshift
for i = 1:numRows
   fftZafter(i,:) = fftZ(numRows-i+1,:);
end
fftZ = fftshift(fftZ);  % reshift
fftZafter = fftshift(fftZafter);    % shift reflected
fftZafter_0 = fftZ_0;

% calculate deltaK
deltaK = (abs(fftZbefore).*abs(fftZbefore))./(abs(fftZbefore_0)^2) ...
        - (abs(fftZafter).*abs(fftZafter))./(abs(fftZafter_0)^2);

% use contourc to compute locus of deltaK = 0
cLocus = contourc(fftX(1,AX),fftY(AX,1),deltaK(AX,AX),[0 0]);
% extract out Locus points
numLoci = 0;
zeroLocusX = [];
zeroLocusY = [];
for i = 1:size(cLocus,2)   % number of cols
   if numLoci == 0
      numLoci = cLocus(2,i);  % number of points
   else
      zeroLocusX = [zeroLocusX; cLocus(1,i)];
      zeroLocusY = [zeroLocusY; cLocus(2,i)];
      numLoci = numLoci - 1;
   end
end

theta = atan2(zeroLocusY,zeroLocusX);
%histF = figure;
%hist(theta.*180/pi, 360);
%title(['Original Histogram of theta'],'Color','k');
%axis([-180 180 -inf inf]);

% find rotations relative to each of 4 axes, use cascading sequence
% theta ranges from -pi to pi
if SEPARATE_AXES == TRUE
   % separate Quadrant 1&3 from 2&4
   % leave Q2
      % map Q3 to Q1
      targetAngles = theta < -pi/2;
      theta(targetAngles) = theta(targetAngles) + pi;
      % map Q4 to Q2
      targetAngles = theta < 0;
      theta(targetAngles) = theta(targetAngles) + pi;
   else
      % map all to Q1
      % map Q2 to Q1
      targetAngles = theta > pi/2;
      theta(targetAngles) = theta(targetAngles) - pi/2;
      % map Q3 to Q1
      targetAngles = theta < -pi/2;
      theta(targetAngles) = theta(targetAngles) + pi;
      % map Q4 to Q1
      targetAngles = theta < 0;
      theta(targetAngles) = theta(targetAngles) + pi/2;
   end
   theta = theta.*2;

   rho = sqrt(zeroLocusX.*zeroLocusX + zeroLocusY.*zeroLocusY);
   considerTheta = (rho > RHO_MIN) & (rho <= RHO_MAX);

% else first image, will be stored in fftZbefore
end
fftZbefore = fftZ;
fftZbefore_0 = fftZ_0;

% plot
if PLOT_FFT == TRUE
   [fH aH] = Figure3D;
   fHs = [fHs fH]; aHs = [aHs aH];

   title(['DFT ' dataPts{exposure}],'Color',ACLR);
   surf(fftX(AX,AX),fftY(AX,AX),abs(fftZ(AX,AX)));
   colormap('jet');
   axis('tight');
   set(fH,'Color',FCLR); set(aH,'Color',FCLR);
   set(aH,'XColor',ACLR,'YColor',ACLR,'ZColor',ACLR);
   set(aH,'GridLineStyle',':');
end
```

```
if firstFlag == FALSE   % subsequent images

  if PLOT_FFT_REFLECTED == TRUE
    [fH aH] = Figure3D;
    fHs = [fHs fH]; aHs = [aHs aH];

    title(['DFT reflected ' dataPts{exposure}],'Color',ACLR);
    surf(fftX(AX,AX),fftY(AX,AX),abs(fftZafter(AX,AX)));
    colormap('jet');
    axis('tight');
    set(fH,'Color',FCLR); set(aH,'Color',FCLR);
    set(aH,'XColor',ACLR,'YColor',ACLR,'ZColor',ACLR);
    set(aH,'GridLineStyle',':');
  end

  if PLOT_DELTAK == TRUE
    [fH aH] = Figure3D;
    fHs = [fHs fH]; aHs = [aHs aH];

    title(['deltaK ' dataPts{exposure}],'Color',ACLR);
    surf(fftX(AX,AX),fftY(AX,AX),deltaK(AX,AX));
    colormap('jet');
    axis('tight');
    set(fH,'Color',FCLR); set(aH,'Color',FCLR);
    set(aH,'XColor',ACLR,'YColor',ACLR,'ZColor',ACLR);
    set(aH,'GridLineStyle',':');
  end

  if PLOT_LOCUS == TRUE
    lF = figure;
    contour(fftX(AX,AX),fftY(AX,AX),deltaK(AX,AX),[0 0]);
    title(['Locus deltaK = 0 ' dataPts{exposure}],'Color','k');
    set(lF,'Color',FCLR);
    grid off;

    lF = figure;
    plot(zeroLocusX,zeroLocusY,'k.');
    title(['Locus deltaK = 0, ' dataPts{exposure}], ...
        'Color','k');
    set(lF,'Color',FCLR);

    lF = figure;
    plot(zeroLocusX(considerTheta),zeroLocusY(considerTheta),'k.');
    title(['Locus deltaK = 0, ' dataPts{exposure}], ...
        'Color','k');
    set(lF,'Color',FCLR);
  end
  if PLOT_HIST == TRUE
    histF = figure;
    hist(theta.*180/pi, THETA_BINS);
    set(histF,'Color',FCLR);
    title(['Histogram of theta']);

    histF = figure;
    hist(theta(considerTheta).*180/pi, THETA_BINS);
    set(histF,'Color',FCLR);
    [N, angles] = hist(theta(considerTheta).*180/pi,THETA_BINS);
    anglePeaks = sortrows([N' angles'],[1]);
    twoPeaks = [num2str(anglePeaks(THETA_BINS,2)) ',' ...
          num2str(anglePeaks(THETA_BINS-1,2))];
    title(['Histogram of theta: rotation = ' twoPeaks ...
        ' for ' dataPts{exposure} ', ' ...
        'rho in [' num2str(RHO_MIN) ',' num2str(RHO_MAX) ']' ...
        ],'Color','k');
    if SEPARATE_AXES == TRUE
      axis([0 THETA_BINS -inf inf]);
    else
      axis([0 THETA_BINS/2 -inf inf]);
    end
  end
  if SAVE_HIST == TRUE
    % open file
    savePath = [pwd '\'];
    saveMode = 'w';
    FID = fopen([savePath saveFiles{exposure} '.csv'],saveMode);
    fprintf(FID,'Number Counted,Angle\n');
    [N, angles] = hist(theta(considerTheta).*180/pi,THETA_BINS);
    fprintf(FID,'%i,%f\n',[N; angles]);
    fclose(FID);
  end
end
```

```
  firstFlag = FALSE;
end
```

F.2 convertSP

```
% convertSP - script to convert skin surface markers into scapular plane
%
% Revision History:
%     March 6, 2001 - Anthony Choo - initial version
%
clear all;close all;
TRUE = 1; FALSE = 0;

% Digitization to setup local coordinate frames
Ab0obj; % data in format: x,y,z,residual where each row is a point
numPts = size(pts,1);
pts0 = [pts(:,1:3)]';  % strip off residuals and make each pt a column

defSP;  % define which markers define the scapular plane
     % and which markers define the thoracic frame

% Note:
%  The thoracic frame and scapular plane are fixed with
%  respect to each other.
%  The specimen may move within each exposure,
%  therefore must 1) convert to thoracic frame, then 2) to scapular plane
%  Cannot directly convert to scapular plane because we don't
%  have a scapular plane marker carrier.

% Thoracic frame (T) and thoracic marker carriers (Tm)
% (T_CT = T_TmT - see above)
T_TmT = mkT(pts0(:,Torigin)',pts0(:,Txaxis)',pts0(:,Tyaxis)','x','y');
T_TTm = inv(T_TmT);

% Scapular frame (S) and scapular marker carriers (Sm)
% (bone markers or skin markers: markers aligned with fixed)
% (T_CS = T_SmS - see above)
T_CS = mkT(pts0(:,Sorigin)',pts0(:,Sxaxis)',pts0(:,Syaxis)','x','y');
T_SC = inv(T_CS);

% relationship between scapular plane and thoracic carrier frames
% initially, Tm = C so
T_TmS = T_CS;
T_STm = T_SC;

dataPts = {'Ab0obj' 'Ab45obj' 'Ab90obj' 'Ab135obj' ...
        'AbMaxobj' 'Ret45obj' 'Ret75obj' 'Ret90obj'};
saveFiles = {'Ab0sp.m' 'Ab45sp.m' 'Ab90sp.m' 'Ab135sp.m' ...
        'AbMaxsp.m' 'Ret45sp.m' 'Ret75sp.m' 'Ret90sp.m'};

for exposure = 1:8

  % exclude missing Tm
  if exposure == 2 | exposure == 3 | exposure == 4 | exposure == 5
    Tm = [1 3 4];
  else
    Tm = [1 2 3 4];
  end

  eval(dataPts{exposure});   % initialize object points
  pts = [pts(:,1:3)]';   % strip off residuals and make each pt a column

  % 2) Thoracic
  numPts = size(Tm,2);   % a row vector
  [T_CTm TmRes] =
soder([reshape(pts0(:,Tm),1,3*numPts);reshape(pts(:,Tm),1,3*numPts)]);
  T_TmC = inv(T_CTm);

  %  disp('debug T_CTm');
  %  T_CTm
  % T_CTm is a fixed frame solution from soder, change it to current frame
  % i.e. adjust location of origin
  TempO = T_CTm(1:3,1:3)*pts0(:,Torigin);  % rotate old origin
  newO = TempO - pts0(:,Torigin) + T_CTm(1:3,4);   % solve for new origin
                                % relative to old origin
  T_CTm_cfrm = [T_CTm(1:3,1:3) newO; 0 0 0 1];   % cfrm is current frame
  T_TmC_cfrm = inv(T_CTm_cfrm);

  % for debugging
  disp(dataPts{exposure});
```

```
  R = rxyzsolv(T_TmC_cfrm);
  R(1:3)

  %  disp('debug T_CTm_cfrm');
  %  T_CTm_cfrm

  %  disp('debug old origin')
  %  pts0(:,Torigin)

  T_SCexposure = T_STm*T_TmC_cfrm;
  ptsSP = MultHomogeneous(pts',T_SCexposure); % convert to scapular plane

  % write results to file
  % open file
  savePath = [pwd '\'];
  saveMode = 'w';
  FID = fopen([savePath saveFiles{exposure}],saveMode);

  fprintf(FID,'%% All pts in scapular plane coordinates\n');
  fprintf(FID,'%% T1;T2;T3;T4;CR;A1;A2;A3;H0;H1;H2;H3;1;..;68\n');
  fprintf(FID,'%% x,y,z\n');
  fprintf(FID,'pts = [ ...\n');
  for i = 1:size(ptsSP,1)
    fprintf(FID,'%f,%f,%f\n',ptsSP(i,:));
  end
  fprintf(FID,'];\n');
  fclose(FID);
end
```

F.3 defSP

```
% Define Scapular Plane - define which points in Ab0 exposure defines
%                  the scapular plane
%
% Digitized Points
xlsOffset = 12;
% Thoracic Frame
Txaxis = [16 52]+xlsOffset; % skin points 16 and 52
Tyaxis = [32 16]+xlsOffset; % skin points 32 and 16
Torigin = [1];    % Thoracic carrier 1

% Thoracic Frame Markers
Tm = [1 2 3 4];

% Scapular Frame
Sxaxis = [16 52]+xlsOffset; % skin points 16 and 52
Syaxis = [32 16]+xlsOffset; % skin points 32 and 16
Sorigin = [16]+xlsOffset;     % skin point 1
```

REFERENCE LIST

1. **Alberius P**. Bone reactions to tantalum markers. A scanning electron microscopic study. *Acta Anat.(Basel.)* 115:310-318, 1983.

2. **Almen A, Nilsson M**. Simple methods for the estimation of dose distributions, organ doses and energy imparted in paediatric radiology. *Phys.Med.Biol.* 41:1093-1105, 1996.

3. **Altchek DW, Warren RF, Wickiewicz TL, Ortiz G**. Arthroscopic labral debridement. A three-year follow-up study. *American Journal of Sports Medicine* 20:702-706, 1992.

4. **An KN, Browne AO, Korinek S, Tanaka S, Morrey BF**. Three-dimensional kinematics of glenohumeral elevation. *Journal of Orthopaedic Research* 9:143-149, 1991.

5. **Anderson JE**. *Grant's Atlas of Anatomy*. Baltimore, Williams & Wilkins, 1983.

6. **Andrews JR, Broussard TS, Carson WG**. Arthroscopy of the shoulder in the management of partial tears of the rotator cuff: a preliminary report. *Arthroscopy.* 1:117-122, 1985.

7. **Bagg SD, Forrest WJ**. A biomechanical analysis of scapular rotation during arm abduction in the scapular plane. *American Journal of Physical Medicine & Rehabilitation* 67:238-245, 1988.

8. **Bernardini, R., Cortelazzo, G. M., and Mian, G. A.** A constant-geometry multidimensional FFT Cappellini, V. 3, 12-18, 1994. *Time-Varying Image Processing and Moving Object Recognition*. Florence,Italy.

9. **Besl PJ**. *Surfaces in range image understanding*. New York, Springer-Verlag, 1988.

10. **Blevins FT**. Rotator cuff pathology in athletes. *Sports Med.* 24:205-220, 1997.

11. **Borghese NA, Cerveri P, Ferrigno GC**. Statistical comparison of DLT versus ILSSC in the calibration of a photogrammetric stereo-system. *J.Biomech.* 30:409-413, 1997.

12. **Budoff JE, Nirschl RP, Guidi EJ**. Debridement of partial-thickness tears of the rotator cuff without acromioplasty. Long-term follow-up and review

of the literature. [Review] [58 refs]. *Journal of Bone & Joint Surgery - American Volume* 80:733-748, 1998.

13. **Carlsson GA, Carlsson CA**. Relations between effective dose equivalent and mean absorbed dose (energy imparted) to patients in diagnostic radiology. *Phys.Med.Biol.* 31:911-921, 1986.

14. **Carroll QB**. *Fuchs's Principles of Radiographic Exposure, Processing and Quality Control*. Springfield,Illinois, Charles C Thomas, 1990.

15. **Cerveri P, Borghese NA, Pedotti A**. Complete calibration of a stereo photogrammetric system through control points of unknown coordinates. *Journal of Biomechanics* 31:935-940, 1998.

16. **Challis JH, Kerwin DG**. Accuracy assessment and control point configuration when using the DLT for photogrammetry. *J.Biomech.* 25:1053-1058, 1992.

17. **Chen L, Armstrong CW, Raftopoulos DD**. An investigation on the accuracy of three-dimensional space reconstruction using the direct linear transformation technique. *J.Biomech.* 27:493-500, 1994.

18. **de Lange A, Huiskes R, Kauer JM**. Measurement errors in roentgen-stereophotogrammetric joint-motion analysis. *J.Biomech.* 23:259-269, 1990.

19. **Ellman H**. Diagnosis and treatment of incomplete rotator cuff tears. *Clin.Orthop.* 64-74, 1990.

20. **Ellman H**: The treatment of rotator cuff disease: the scalpel or the scope? In *Surgery of the Shoulder*. Edited by M Vastamaki, P Jalovaara, Amsterdam, Elsevier, 1995, pp 85-89.

21. **Freedman L, Munro RR**. Abduction of the arm in the scapular plane: scapular and glenohumeral movements. A roentgenographic study. *Journal of Bone & Joint Surgery - American Volume* 48:1503-1510, 1966.

22. **Gartsman GM**. Massive, irreparable tears of the rotator cuff. Results of operative debridement and subacromial decompression. *Journal of Bone & Joint Surgery - American Volume* 79:715-721, 1997.

23. **Gazzani F**. Performance of a 7-parameter DLT method for the calibration of stereophotogrammetric systems using 1-D transducers. *J.Biomed.Eng.* 14:476-482, 1992.

24. **Gazzani F**. A new algorithm for calibrating stereophotogrammetric systems devoted to motion analysis. *Human Movement Science* 12:403-425, 1993a.

25. **Gazzani F**. Comparative assessment of two algorithms for calibrating stereophotogrammetric systems. *J.Biomech.* 26:1449-1454, 1993b.

26. **Gkanatsios NA, Huda W**. Computation of energy imparted in diagnostic radiology. *Med.Phys.* 24:571-579, 1997.

27. **Glasgow SG, Bruce RA, Yacobucci GN, Torg JS**. Arthroscopic resection of glenoid labral tears in the athlete: a report of 29 cases. *Arthroscopy.* 8:48-54, 1992.

28. **Hatze H**. High-precision three-dimensional photogrammetric calibration and object space reconstruction using a modified DLT-approach. *J.Biomech.* 21:533-538, 1988.

29. **Hawkins RJ, Mohtadi NG**. Controversy in anterior shoulder instability. *Clin.Orthop.* 152-161, 1991.

30. **Hebert LJ, Moffet H, McFadyen BJ, St-Vincent G**. A method of measuring three-dimensional scapular attitudes using the optotrak probing system. *Clin.Biomech (Bristol., Avon.)* 15:1-8, 2000.

31. **Hidajat N, Maurer J, Schroder RJ, Nunnemann A, Wolf M, Pauli K, Felix R**. Relationships between physical dose quantities and patient dose in CT. *Br.J.Radiol.* 72:556-561, 1999.

32. **Hinrichs RN, McLean SP**. NLT and extrapolated DLT:3-D cinematography alternatives for enlarging the volume of calibration. *J.Biomech.* 28:1219-1223, 1995.

33. **Hogfors C, Peterson B, Sigholm G, Herberts P**. Biomechanical model of the human shoulder joint--II. The shoulder rhythm. *Journal of Biomechanics* 24:699-709, 1991.

34. **Holloway AF, Taylor KW, Hobbs BB**. Dose and quality control (DQC) in diagnostic radiology. *J.Can.Assoc.Radiol.* 35:149-153, 1984.

35. **Holzreiter S**. Calculation of the instantaneous centre of rotation for a rigid body. *J.Biomech.* 24:643-647, 1991.

36. **Huda W, Bissessur K**. Effective dose equivalents, HE, in diagnostic radiology. *Med.Phys.* 17:998-1003, 1990.

37. **Huda W, Gkanatsios NA**. Effective dose and energy imparted in diagnostic radiology. *Med.Phys.* 24:1311-1316, 1997.

38. **Huda W, Sandison GA, Palser RF, Savoie D**. Radiation doses and detriment from chest x-ray examinations. *Phys.Med.Biol.* 34:1477-1492, 1989.

39. **Huiskes R, Kremers J, de Lange A, Woltring HJ, Selvik G, van Rens TJ**. Analytical stereophotogrammetric determination of three-dimensional knee-joint geometry. *J.Biomech.* 18:559-570, 1985.

40. **Inman VT, Saunders JB, Abbott LC**. Observations of the function of the shoulder joint. 1944 [classical article]. *Clinical Orthopaedics & Related Research* 3-12, 1996.

41. **Johnsson R, Selvik G, Stromqvist B, Sunden G**. Mobility of the lower lumbar spine after posterolateral fusion determined by roentgen stereophotogrammetric analysis. *Spine* 15:347-350, 1990.

42. **Karduna AR, McClure P, Michener L, Sennett B**. Dynamic Measurement of Three Dimensional Scapular Kinematics: A Validation Study. *Journal of Biomechanical Engineering* accepted for publication:2001.

43. **Karduna AR, Williams GR, Williams JL, Iannotti JP**. Glenohumeral joint translations before and after total shoulder arthroplasty. A study in cadavera. *Journal of Bone & Joint Surgery - American Volume* 79:1166-1174, 1997.

44. **Karrholm J**. Roentgen stereophotogrammetry. Review of orthopedic applications. *Acta Orthop.Scand.* 60:491-503, 1989.

45. **Le Heron JC**. Estimation of effective dose to the patient during medical x-ray examinations from measurements of the dose-area product. *Phys.Med.Biol.* 37:2117-2126, 1992.

46. **Lee S, Harris KG, Goel VK, Clark CR**. Spinal motion after cervical fusion. In vivo assessment with roentgen stereophotogrammetry. *Spine.* 19:2336-2342, 1994.

47. **Lucchese, L. and Cortelazzo, G. M.** On the practical estimability of planar roto-translations with the locus delta(K) 435-438, 1999. *Proc. 5th Int. Symp. Signal Process. Appl.* Brisbane, Australia.

48. **Lucchese L, Cortelazzo GM**. A noise-robust frequency domain technique for estimating planar roto-translations. *Ieee Transactions On Signal Processing* 48:1769-1786, 2000.

49. **Lucchese, L., Cortelazzo, G. M., and Monti, C.** A frequency domain technique for estimating rigid planar rotations 2, 774-777, 1996. *Proc. IEEE Int. Symp. Circuits Syst.* Atlanta, GA.

50. **Lucchese, L., Cortelazzo, G. M., and Monti, C.** High resolution estimation of planar rotations based on fourier transform and radial projections 1181-1184, 1997. Hong Kong.

51. **Lucchese, L., Cortelazzo, G. M., and Rizzato, M.** A phase correlation technique for estimating planar rotations Cappellini, V. 4, 224-249, 1997. *Time-Varying Image Processing and Moving Object Recognition.* Florence,Italy.

52. **Lucchese, L., Doretto, G., and Cortelazzo, G. M.** Frequency domain estimation of 3-D rigid motion based on range and intensity data 107-112, 1997. *Proc. Int. Conf. Recent Adv. 3-D Digital Imag. Modeling.* Ottawa, Ont., Canada.

53. **Ludewig PM, Cook TM, Nawoczenski DA.** Three-dimensional scapular orientation and muscle activity at selected positions of humeral elevation. [Review] [58 refs]. *Journal of Orthopaedic & Sports Physical Therapy* 24:57-65, 1996.

54. **Lukasiewicz AC, McClure P, Michener L, Pratt N, Sennett B.** Comparison of 3-dimensional scapular position and orientation between subjects with and without shoulder impingement. *J.Orthop.Sports Phys.Ther.* 29:574-583, 1999.

55. **Lyons PM, Orwin JF.** Rotator cuff tendinopathy and subacromial impingement syndrome. *Med.Sci.Sports Exerc.* 30:S12-S171998.

56. **Lyons RG.** *Understanding Digital Signal Processing.* Reading, Massachusetts, Addison-Wesley, 1997.

57. **Marks PH, Warner JJ, Irrgang JJ.** Rotator cuff disorders of the shoulder. *J.Hand Ther.* 7:90-98, 1994.

58. **Martin DR, Garth WP, Jr.** Results of arthroscopic debridement of glenoid labral tears. *American Journal of Sports Medicine* 23:447-451, 1995.

59. **McQuade KJ, Smidt GL.** Dynamic scapulohumeral rhythm: the effects of external resistance during elevation of the arm in the scapular plane. *Journal of Orthopaedic & Sports Physical Therapy* 27:125-133, 1998.

60. **Melillo AS, Savoie FH, Field LD**. Massive rotator cuff tears: debridement versus repair. *Orthopedic Clinics of North America* 28:117-124, 1997.

61. **Meskers CG, van der Helm FC, Rozendaal LA, Rozing PM**. In vivo estimation of the glenohumeral joint rotation center from scapular bony landmarks by linear regression. *Journal of Biomechanics* 31:93-96, 1998.

62. **Milner SC, Naylor E**. An estimation of doses received by patients in a diagnostic x-ray department. *Radiogr.Today* 55:16-18, 1989.

63. **Muijtjens AM, Roos JM, Arts T, Hasman A**. Maximum likelihood estimation in calibrating a stereo camera setup. *Med.Phys.* 26:310-318, 1999.

64. **Nilsson KG, Karrholm J, Linder L**. Femoral component migration in total knee arthroplasty: randomized study comparing cemented and uncemented fixation of the Miller-Galante I design. *Journal of Orthopaedic Research* 13:347-356, 1995.

65. **Nystrom L, Soderkvist I, Wedin PA**. A note on some identification problems arising in roentgen stereo photogrammetric analysis. *J.Biomech.* 27:1291-1294, 1994.

66. **O'Neill DB**. Arthroscopic Bankart repair of anterior detachments of the glenoid labrum. A prospective study. *J.Bone Joint Surg.Am.* 81:1357-1366, 1999.

67. **Oppenheim AV, Willsky AS, Hawab SH**. *Signals and Systems*. Upper Saddle River, New Jersey, Prentice Hall, 1997.

68. **Ostgaard SE, Gottlieb L, Toksvig-Larsen S, Lebech A, Talbot A, Lund B**. Roentgen stereophotogrammetric analysis using computer-based image-analysis. *J.Biomech.* 30:993-995, 1997.

69. **Paley KJ, Jobe FW, Pink MM, Kvitne RS, ElAttrache NS**. Arthroscopic findings in the overhand throwing athlete: evidence for posterior internal impingement of the rotator cuff. *Arthroscopy* 16:35-40, 2000.

70. **Panjabi MM, Krag MH, Goel VK**. A technique for measurement and description of three-dimensional six degree-of-freedom motion of a body joint with an application to the human spine. *J.Biomech.* 14:447-460, 1981.

71. **Payne LZ, Jokl P**. The results of arthroscopic debridement of glenoid labral tears based on tear location. *Arthroscopy* 9:560-565, 1993.

72. **Pettrone FA**. *Athletic Injuries of the Shoulder*. New York, McGraw-Hill Inc., 1995.

73. **Poppen NK, Walker PS**. Normal and abnormal motion of the shoulder. *Journal of Bone & Joint Surgery - American Volume* 58:195-201, 1976.

74. **Reddy BS, Chatterji BN**. An FFT-based technique for translation, rotation, and scale- invariant image registration. *Ieee Transactions On Image Processing* 5:1266-1271, 1996.

75. **Rockwood CAJ, Williams GRJ, Burkhead WZ, Jr.** Debridement of degenerative, irreparable lesions of the rotator cuff. *Journal of Bone & Joint Surgery - American Volume* 77:857-866, 1995.

76. **Rune B, Sarnas KV, Selvik G, Jacobsson S**. Roentgen stereometry in the study of craniofacial anomalies--the state of the art in Sweden. *Br.J.Orthod.* 13:151-157, 1986.

77. **Russell JG**. How dangerous are diagnostic X-rays? *Clin.Radiol.* 35:347-351, 1984.

78. **Sciavicco L, Siciliano B**. *Modeling and Control of Robot Manipulators*. New York, The McGraw-Hill Companies, Inc., 1996.

79. **Selvik G**. Roentgen stereophotogrammetry. A method for the study of the kinematics of the skeletal system. *Acta Orthop.Scand.Suppl.* 232:1-51:1-51, 1989.

80. **Selvik G, Alberius P, Aronson AS**. A roentgen stereophotogrammetric system. Construction, calibration and technical accuracy. *Acta Radiol.[Diagn.] (Stockh.)* 24:343-352, 1983.

81. **Soderkvist I, Wedin PA**. Determining the movements of the skeleton using well-configured markers. *J.Biomech.* 26:1473-1477, 1993.

82. **Spielmann AL, Forster BB, Kokan P, Hawkins RH, Janzen DL**. Shoulder after rotator cuff repair: MR imaging findings in asymptomatic individuals--initial experience. *Radiology* 213:705-708, 1999.

83. **Spoor CW, Veldpaus FE**. Rigid body motion calculated from spatial co-ordinates of markers. *J.Biomech.* 13:391-393, 1980.

84. **Statkiewicz-Sherer MA, Visconti PJ, Ritenour ER**. *Radiation Protection in Medical Radiography*. St. Louis, Missouri, Mosby, 1998.

85. **Stearns SD, David RA**. *Signal Processing Algorithms in Matlab*. Upper Saddle River, New Jersey, Prentice Hall P T R, 1996.

86. **Talkhani IS, Kelly CP**. Scapulothoracic rhythm in normal male volunteers. *Biomedical Sciences Instrumentation* 34:327-331, 1997.

87. **Tomlinson RJJ, Glousman RE**. Arthroscopic debridement of glenoid labral tears in athletes. *Arthroscopy*. 11:42-51, 1995.

88. **Veldpaus FE, Woltring HJ, Dortmans LJ**. A least-squares algorithm for the equiform transformation from spatial marker co-ordinates. *J.Biomech*. 21:45-54, 1988.

89. **Vrooman HA, Valstar ER, Brand GJ, Admiraal DR, Rozing PM, Reiber JH**. Fast and accurate automated measurements in digitized stereophotogrammetric radiographs. *J.Biomech*. 31:491-498, 1998.

90. **Warner J, Fu FH**: Inta-articular lesions of the long head of biceps brachii: diagnosis and treatment. In *Surgery of the Shoulder*. Edited by M Vastamaki, P Jalovaara, Amsterdam, Elsevier, 1995, pp 117-121.

91. **Weber SC**. Arthroscopic debridement and acromioplasty versus mini-open repair in the treatment of significant partial-thickness rotator cuff tears. *Arthroscopy*. 15:126-131, 1999.

92. **Wise KN, Sandborg M, Persliden J, Carlsson GA**. Sensitivity of coefficients for converting entrance surface dose and kerma-area product to effective dose and energy imparted to the patient. *Phys.Med.Biol*. 44:1937-1954, 1999.

93. **Woltring HJ**. Calibration and measurement in 3-dimensional monitoring of human motion by optoelectronic means. I. Preliminaries and theoretical aspects. *Biotelemetry*. 2:169-196, 1975.

94. **Woltring HJ**. Representation and calculation of 3-D joint movement. *Human Movement Science* 10:603-616, 1991.

95. **Woltring HJ**. 3-D attitude representation of human joints: a standardization proposal. *J.Biomech*. 27:1399-1414, 1994.

96. **Woltring HJ, Huiskes R, de Lange A, Veldpaus FE**. Finite centroid and helical axis estimation from noisy landmark measurements in the study of human joint kinematics. *J.Biomech.* 18:379-389, 1985.

97. **Woltring HJ, Long K, Osterbauer PJ, Fuhr AW**. Instantaneous helical axis estimation from 3-D video data in neck kinematics for whiplash diagnostics. *J.Biomech.* 27:1415-1432, 1994.

98. **Wood GA, Marshall RN**. The accuracy of DLT extrapolation in three-dimensional film analysis. *J.Biomech.* 19:781-785, 1986.

99. **Young GJ, Chellappa R**. 3-D motion estimation using a sequency of noisy stereo images: models, estimation, and uniqueness results. *IEEE Transactions on Pattern Analysis* 12:735-759, 1990.

100. **Yuan X, Ryd L**. Accuracy analysis for RSA: a computer simulation study on 3D marker reconstruction. *J.Biomech* 33:493-498, 4-2-0.

101. **Yuan X, Ryd L, Blankevoort L**. Error propagation for relative motion determined from marker positions. *J.Biomech.* 30:989-992, 1997.