Rotation Points From

Motion Capture Data Using a Closed Form Solution

by

JONATHAN KIPLING KNIGHT

M.A. Applied Mathematics, Cal. State Fullerton, 1995

B.S. Physics, Cal Poly, San Luis Obispo, 1987

A thesis submitted to the Graduate Faculty of the

University of Colorado at Colorado Springs

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2008

Acknowledgements

First and foremost I would like to acknowledge my wife Kiki for her patience during this research. I would also like to thank Gene Johnson for his support during the ups and downs of my life during this major event in my life. He kept telling me to continue. I would also like to thank my advisor Dr. Semwal for allowing me the freedom and guidance to pursue the ideas laid down in this paper.

This thesis for the Philosophical Doctor degree by

Jonathan Kipling Knight

has been approved for the

Department of Computer Science

by

_____
Sudhanshu Kumar Semwal, Chair

_____
Robert Carlson

_____
C. Edward Chow

_____
Jugal Kalita

_____
Charles M. Shub

_____
Date

Knight, Jonathan Kipling (Ph.D., Computer Science)

Rotation Points from Motion Capture Data Using a Closed Form Solution

Thesis directed by Professor Sudhanshu Kumar Semwal

Four new closed-form methods are present to find rotation points of a skeleton from motion capture data. A generic skeleton can be directly extracted from noisy data with no previous knowledge of skeleton measurements. The new methods are ten times faster than the next fastest and a hundred times faster than the most widely accepted method. Two phases are used to produce an accurate skeleton of the captured data. The *first phase*, fitting the skeleton, is robust even with noisy motion capture data. The formulae use an asymptotically unbiased version of the Generalized Delogne-Kása (GDKE) Hyperspherical Estimation (first estimator: UGDK). The second estimator takes advantage of multiple markers located at different distances from the rotation point (MGDK) thereby increasing accuracy. The third estimator removes singularities to allow for cylindrical joint motion (SGDK). The fourth estimator incrementally improves an answer and has advantages of constant memory requirements suitable for firmware applications (IGDK). The UGDK produces the answer faster than any previous algorithm and with the same efficiency with respect to the Cramér-Rao Lower Bound for fitting spheres and circles. The UGDK method significantly reduces the amount of work needed for calculating rotation points by only requiring 26N flops for each joint. The next fastest method, Linear Least-Squares requires 236N flops. In-depth statistical analysis shows the UGDK method converges to the actual rotation point with an error of $O(\sigma/\sqrt{N})$ improving on the GDKE's biased answer of $O(\sigma)$. The *second phase* is a real-time algorithm to draw the

skeleton at each time frame with as little as one point on a segment. This speedy method, on the order of the number of segments, aids the realism of motion data animation by allowing for the subtle nuances of each time frame to be displayed. Flexibility of motion is displayed in detail as the figure follows the captured motion more closely. With the reduced time complexity, multiple figures, even crowds can be animated. In addition, calculations can be reused for the same actor and marker-set allowing different data sets to be blended. The main contributions in this dissertation are the new unbiased center formulae; the full statistical analysis of this new formula; and the analysis of when the best measurement conditions are to initiate the formula. The dissertation further establishes the application of these new formulae to motion capture to produce a real-time method of drawing skeletons of arbitrary articulated figures.

CONTENTS

PREFACE **xiii**

Chapter 1 INTRODUCTION **1**

1.1 Articulated Figure Animation **4**

1.2 Motion Capture Systems **7**

1.3 Symbols and Conventions **9**

Chapter 2 STATEMENT OF THE PROBLEM **11**

2.1 Why are skeleton calculations required? **11**

2.2 Problems encountered **12**

2.2.1 Inaccuracies 12

2.2.2 Non-standard Files 13

2.2.3 Missing Data 13

Chapter 3 SURVEY **14**

3.1 Skeleton Extraction **14**

3.2 Sphere Estimates **14**

3.3 Inverse Kinematics **17**

3.4 Kinetics **18**

Chapter 4 PREVIOUS SOLUTIONS **19**

4.1 Spherical Curve-fitting Approaches **19**

4.1.1 Monte-Carlo Experiment 19

4.1.2 Cramér-Rao Lower Bound 23

4.1.3 Non-linear Maximum-Likelihood Estimator 34

4.1.4 Linear Least-Squares Solution 37

4.1.5 Generalized Delogne-Kása Estimator 40

4.2 Skeleton Approaches 62

Chapter 5 PROPOSED SOLUTION 64

5.1 Unbiased Generalized Delogne-Kása Estimator 64

5.1.1 Derivation 64

5.1.2 Statistical Properties 66

5.2 Cylindrical Joint Solution 75

5.3 Multiple Marker Solution 76

5.4 Incrementally Improved Solution 78

5.5 Hierarchical Skeleton Solution 81

5.5.1 Arbitrary Figure 81

5.5.2 Predefined Marker Association 86

Chapter 6 RESULTS 90

6.1 Case Study of CMU Data 60-08 90

6.2 Case Study of Eric Camper Data 94

6.3 Comparison 95

6.4 Speed 96

6.5 Conclusion 98

6.6 Important Contributions 99

6.7 Further Research 101

Chapter 7 APPENDIX 102

7.1 Mathematical Proofs 102

7.1.1 Moments of Multivariate Normal 102

7.1.2 Positive-Semidefinite Sample Covariance 110

7.2 Inertial Properties of a Tetrahedron 111

7.3 File Formats 114

7.3.1 Marker Association Format 114

7.3.2 Articulated Tetrahedral Model Format 115

7.3.3 MESH Format 117

7.3.4 PLY Format 121

7.3.5 C3D Format 121

7.4 User's Guide to Program 121

7.4.1 Menu 122

7.4.2 Options Pane 126

7.4.3 Animation Pane 127

7.4.4 Graphs Pane 129

7.5 Programmer's Reference 130

7.5.1 Class Diagrams 130

7.6 C++ Implementations 132

7.6.1 Unbiased Generalized Delogne-Kása Method 132

7.6.2 Incrementally Improved Generalized Delogne-Kása 133

7.6.3 Collecting the Raw Data for Rotation Point 134

7.6.4 Rotation Point Calculation of Segment 136

7.6.5 Constants Calculation of Hierarchical Articulated Data 137

7.6.6 Calculation of fixed axes of data 139

7.6.7 Drawing Rotation Points with Constants of Motion 140

Chapter 8 BIBLIOGRAPHY 143

Chapter 9 INDEX 151

TABLES

Table 1 Marker Associations 86

Table 2 Table of Means of Rotation Points 90

Table 3 Table of Standard Deviations of Rotation Points 91

Table 4 Comparison of Center Estimators 95

Table 5 Primitives in MESH Format 117

Table 6 Preservation Adjectives in MESH Format 118

Table 7 Optional Adjectives of Primitives in MESH Format 119

FIGURES

Figure 1 Display of Motion-Capture Data xv

Figure 2 Markers on Actor 2

Figure 3 Human Articulated Shoulder 5

Figure 4 Human Elbow 6

Figure 5 Video Capture Analysis Software (SIMI°MotionCapture 3D)[1] 8

Figure 6 Vicon BodyBuilder Software 9

Figure 7 Constrained Measurements on Circle 20

Figure 8 Relative Error Comparison 22

Figure 9 Eigenvectors of CRLB for Circle 28

Figure 10 Eigenvalues of CRLB for Circle 29

Figure 11 Eigenvectors of CRLB for Sphere 32

Figure 12 Eigenvalues of CRLB for Sphere 33

Figure 13 MLE Compared to CRLB 36

Figure 14 MLE Error Versus Sphere Coverage 36

Figure 15 LLS Compared to CRLB 39

Figure 16 GDKE Compared to CRLB 43

Figure 17 GDKE Error Ellipse 48

Figure 18 UGDK Error Ellipse 68

Figure 19 Sample Size Dependency of Deviation 69

Figure 20 One hundred samples comparison of MLE (green), UGDK (red), GDKE (blue) 70

Figure 21 UGDK Compared to CRLB 70

Figure 22 Circle with Constrained Data 72

Figure 23 MGDK example 78

Figure 24 Inverse Power Law for Rotation Point Calculation 94

Figure 25 Eric Camper Skeleton 94

Figure 26 Timing of Algorithms 97

Figure 27 Timing Comparison to GDKE 98

Figure 28 File Menu GUI 123

Figure 29 Options Pane GUI 127

Figure 30 Animation Pane GUI 128

Figure 31 Graphs Pane GUI 129

Figure 32 UML Diagram of Articulated Figure 131

PREFACE

The research that will be presented in this paper is the culmination of a story. The five-year journey involves the typical parts of a novel with protagonist, antagonists, heartaches, and triumphs. The protagonist is, of course, me – the author. The antagonists in this story were the many hurdles that I stumbled over to finally realize what my actual goal was for the research. The story of this research is a good example of what it means for the research topic itself to tell you are going in the wrong direction. Perseverance has declared a victory in the research only by finally exposing a vital and missing piece of the puzzle that builds up the topic at hand.

So, what is the topic? The research started out five years ago as an idea that current technology for articulated motion analysis and display was much too slow and unrealistic. I started the research, albeit naively, by thinking that a combination of physics and genetic algorithms could infuse a sense of realism and speed into analyzing the motion. So I sat down and coded up the infrastructure needed to build up this combination.

First, I needed a computer representation of an articulated figure. To keep physics in the equation, I decided to make each segment of the figure solid. I found that, for physics related simulations, this is the only way to go since calculations of inertial moments are fairly simple (cf. Chapter 7.2). A few other authors use this approach for high-end physical simulations but, for the most part, most use triangulated surfaces for their "solid" figures. It is usually difficult to calculate moments of inertia from only the surface data. A tetrahedral solid mesh was chosen to simplify building up an arbitrary shape. With a solid shape, and its mass, the rotational and translational physics are fully calculable from the Newton-Euler differential equation. As I progressed further in the

research, I found it increasingly time-intensive to calculate projected motions based on solving the second-order differential equation on an arbitrary articulated figure.

This suggested that I pursue an alternative approach. So, what if I relied more on motion-capture data and infuse only a little bit of physics? I then looked at reading in and using raw motion-capture data. Little did I know that motion-capture data was ex-cruciatingly hard to acquire both on your own and on the Internet. The people and companies that did any kind of work with capturing motion data were usually quite willing to *sell* it. Only a few free examples were found that were of any use. A file reader was built from scratch for the most popular format since no one had open source code. I struggled for a year with reading the quirky data format with the small amount of data I found until a goldmine was found. Carnegie-Melon University Graphics Laboratory had just started placing a large cache of raw and analyzed motion-capture data on the Internet through a government grant. I subsequently downloaded gigabytes of raw binary data for a plethora of motions captured by the students at this lab. One problem solved – gathering data to analyze.

Now I set about trying to analyze the raw data so that I may use it to better analyze the underlying articulated motion. Starting with the simplest issue, I started trying to display the motion on a 3-D scene. As it turns out, the raw data was very inconsistent in storing which measurement units it was using. Sometimes it was in meters, sometimes it was in feet, sometimes it was stored in feet and the format said it was meters. Once you figure out what scale the data is in, you can display the points on the screen.

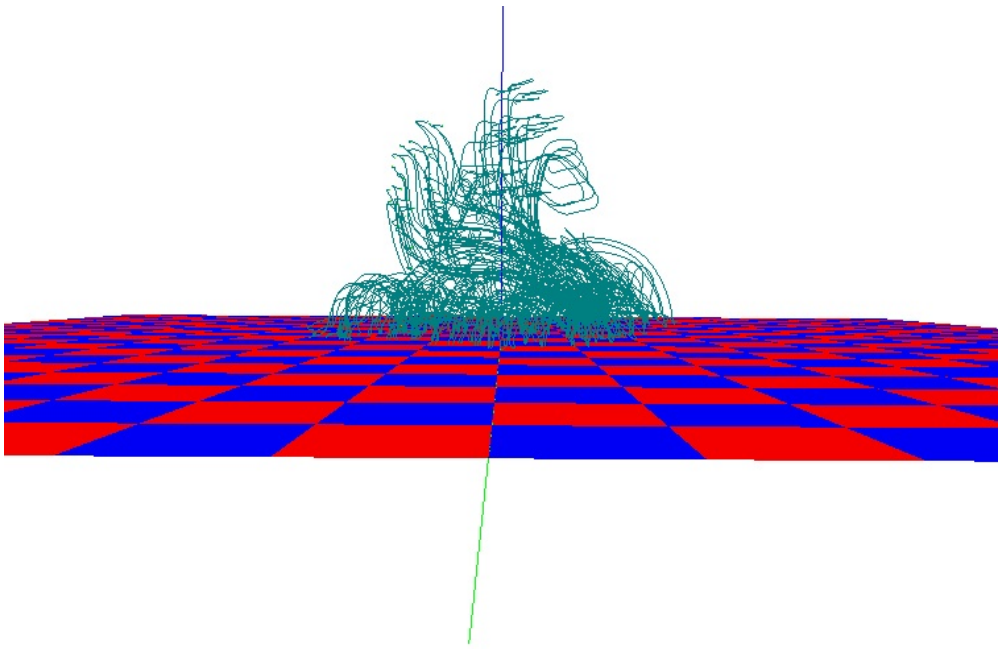

Figure 1 Display of Motion-Capture Data

Now, all I had was a bunch of squirrelly lines on the screen (cf. Figure 1). Clearly, the lines had to be associated and grouped. Each line was attached (i.e. correlated) to a particular segment on the articulated figure. This can be done in one of two ways – autocorrelation and precognition. I chose the latter since autocorrelation is extremely time intensive. Precognition may sound like cheating but nine times out of ten, the user will know ahead of time which bits of data are associated with which segment on the articulated figure.

What do I have so far? I have a bunch of lines that each segment can follow in time. As data shows marker location at each time frame, each segment has some surface positions pin-pointed in space. In addition, exact marker location on segments usually were not known or given. That leaves an unsavory taste in your mouth when the segment can be placed almost anywhere relative to these points. A systematic approach must be

thought of that allows the conclusive attachment of the segments to the data. This implies that the data must contain both relative position and orientation of the segment so the model can be sized and oriented into position. This satisfies an individual segment, but not the whole articulated figure. For that I needed the rotation points in between the segments. What this boils down to is that I needed to draw a stick figure based solely on the motion-capture data. I could then attach whatever shape model I wanted onto the stick figure. I did a search for existing methods and found a few examples in the literature (cf. Chapter 8). The most popular was to perform a least-squares fit and a few still did non-linear fitting of the data. They all had one thing in common – they all assumed the marker moved on a sphere around the rotation point of the joint. This means that, to draw a stick figure, I had to solve for the center of a sphere at every joint. A little research showed there were only three major algorithms to solve for the center of a sphere by data points sitting on the sphere. The most popular was the Maximum-Likelihood Estimator (MLE) that solved the problem in a non-linear fashion. This had the undesirable effect of being excruciatingly slow and sometimes not producing an answer at all. The next most popular type was linear least-squares solutions that involve fairly slow (but faster than MLE) pseudo-inverse matrix solutions. This was the preferred method by most authors concerned with speed. The third type of solution was a small set of approximations that were closed-form solutions to the best-fit sphere. These formulae were unpopular and usually used only in the case to start off a non-linear search for the real answer. They were unpopular because of biases introduced.

I was unsatisfied with any of these solutions found in the existing literature. They would get me to the answer, but they were too slow or quirky. I decided to sit back and

analyze a case study of a particular joint movement. To proceed with analyzing, the raw (x,y,z) data was loaded into a spreadsheet for the step-by-step analysis to see how the rotation point could be retrieved from the data. A column of distances from an unknown point (the rotation point) was made for each data point. Then, thinking about it – what would happen if the unknown point were truly at the center of rotation? Then, the distances to the data points would all be about the same. What this translates into is that the standard deviation from the mean of the distances would be minimized. This can be easily done in Excel with the Solve tool. Thinking about it further, that this is exactly what the Maximum Likelihood Estimator (MLE) solution is – minimizing the standard deviation of the distance to the center of rotation. I looked carefully at the mathematics to see if a closed form solution can be found. The MLE does not solve to a closed form. I then altered the column to be the square of the distance. The Solve tool still got nearly the same answer as the MLE. I looked at the math for this minimizing and lo and behold, a closed form solution dropped out on the floor. So the minimum of the standard deviation of the *square* of the distance to the center of rotation can be solved with a fairly simple formula.

I found this new formula easy to use and very fast compared to the other two techniques. I was all ready to name my new discovery but then I thought it was too good to be true. I tried desperately to find an author in the existing literature that used this formula. I could not find a single author that even implied the existence until a month later. I found an author [119] who had published three months earlier in some obscure electronics conference (yes 3 months!). His formula was a different arrangement of the same solution I had developed. So, at least I knew what to call the formula – Generalized

Delogne-Kása Estimator (GDKE). Apparently this formula had been used off and on in many industries since 1972 [56][57] in only its two-dimensional form (i.e. circle). This new formula can handle any dimensional sphere. The formula had not risen in popularity because of a certain flaw. In a particular arrangement of data, the estimator would be expected NOT to be the answer. The flaw would not show itself if the data points were distributed evenly over the entire sphere. Unfortunately, this ideal case is not too common in the real world. Therefore, the formula ended up only being useful when the measurement system was extremely accurate – another not so common real world case.

Fine, someone beat me to a fast and flawed formula that solves my problem. Not good enough for me, so I set out to remove the flaw and retain the speed. The first thing to do to remove the flaw is to exactly identify it. The original paper from 2004 [119] did not explain in detail what the flaw was, just that the bias was about the size of the measurement error. As it turns out, the multiplication factor can amplify the bias far beyond the simple measurement error. It took over two years to work out the exact relationship of the bias. It involves the probability analysis of multidimensional random variables multiplied together up to six times. It was slow going since the equation grew rather large. Keeping track of all of the variables became unwieldy, as even Mathematica couldn't handle the math. Finally, the flaw was identified and a systematic equation was produced that effectively removed the flaw. The results were a simple modification to the GDKE that made the formula guarantee to converge to the true center and radius as more raw data was thrown at it. The gap in technology for building a skeleton from motion capture data was identified and filled. This paper goes into excruciating detail to prove the capabilities of this new method for skeletal extraction.

Chapter 1 INTRODUCTION

First of all, an explanation is needed for "articulated motion". There are many examples of this kind of motion in everyday life. Just typing this paper involves articulated motion. Riding a bicycle involves articulated motion. These examples themselves are topics of research in their own right. An articulated figure can be formally defined as a collection of segments that are connected by joints. The type of joint defines its motion. If a joint rotates then the two segments can rotate around a point in up to three directions (i.e. degrees of freedom). If a joint can translate in space then the two segments can separate from each other in up to three directions adding more degrees of freedom. These motions for a joint, up to six degrees of freedom (DOF), are considered to be articulated motion – the topic at hand. The complications arise when more than one joint is in the figure. This creates multiple possibilities and there is no straightforward solution to the problem of analyzing the motion.

First, a computer representation of an articulated figure is needed. To keep physics in the equation, each segment of the figure must be solid. For physics related simulations, this is the only way to go since calculations of moments of inertia are fairly simple (cf. Chapter 7.2). A few other authors [97][113][121][122] use this approach for high-end physical simulations but most use triangulated surfaces for their "solid" figures. It is usually difficult to calculate moments of inertia from only the surface data. A tetrahedral solid mesh was chosen to simplify building up an arbitrary shape. With a solid shape, and its mass, the rotational physics and translational physics are fully calculable from solving the Newton-Euler differential equation. This unfortunately proves time-intensive.

More reliance on motion data can increase the speed. The motion data is usually made of raw position measurements of markers that have been placed on the surface of the body as in Figure 2.

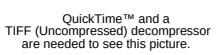

Figure 2 Markers on Actor

Each marker in the data must be assigned (i.e. correlated) to a particular segment on the articulated figure. This can be done in one of two ways – autocorrelation and pre-

cognition. The latter is better for this research since autocorrelation is extremely time intensive. Precognition may sound like cheating but nine times out of ten, the user will know ahead of time which bits of data are associated with which segment on the articulated figure. A systematic approach must be thought of that allows the conclusive attachment of the segments to the data. This implies that the data must contain both relative position and orientation of the segment so the model can be sized and oriented into position. This satisfies an individual segment, but not the whole articulated figure. The rotation points in between the segments are needed. What this boils down to is that a stick figure needs to be drawn based solely on the motion-capture data. Whatever shape model can then be attached onto the stick figure. Existing methods were found in a few examples in the literature (cf. Chapter 8). The most popular was to perform a least-squares fit and a few still did non-linear fitting of the data. They all had one thing in common – they all assumed the marker moved on a sphere around the rotation point of the joint. This means that, to draw a stick figure, the center of a sphere must be solved at every joint. A little research shows there were only three major algorithms that solve the center of a sphere from data points sitting on the sphere. The most popular was the Maximum-Likelihood Estimator (MLE) that solved the problem in a non-linear fashion. This had the undesirable effect of being excruciatingly slow and sometimes not producing an answer at all. The next most popular type was linear least-squares solutions that involve fairly slow (but faster than MLE) pseudo-inverse matrix solutions. This was the preferred method by most authors concerned with speed. The third type of solution was a small set of approximations that were closed-form solutions to the best-fit sphere. These

formulae were unpopular because of biases and usually used only in the case to start off a non-linear search for the real answer.

1.1 Articulated Figure Animation

An articulated figure has a simple definition. It is defined as a set of jointed segments. The joint in between two segments can be free to rotate in one, two, or three directions and translate in one, two, or three directions. These types of movements are a rating system called the degrees of freedom (DOF). One DOF means the joint is free to rotate around only one axis. Two DOF and three DOF are similar. Four DOF joints are free to rotate around all three orthogonal axes and translate along one of them. Five and six add more translational freedom. A full six DOF joint has no restrictions in movement.

An articulated figure is made of a set of linked segments. A human figure and many other animal forms have a tree-like set of connected segments. Realistically, each segment is not some independent object attached at a specific point in space. Examining the human shoulder (see Figure 3) will uncover a complex joint system involving the scapula (shoulder blade), clavicle, and the humerus (upper arm bone).

Figure 3 Human Articulated Shoulder

Not only does each of these bones rotate around sockets but also have small amounts of translational freedom. Even with this complexity, certain levels of approximation can make this joint look like it is capable of simple three DOF motion. It is also customary to approximate the elbow joint (see Figure 4) with a one DOF model even though there are three bones involved (humerus, radius, ulna).

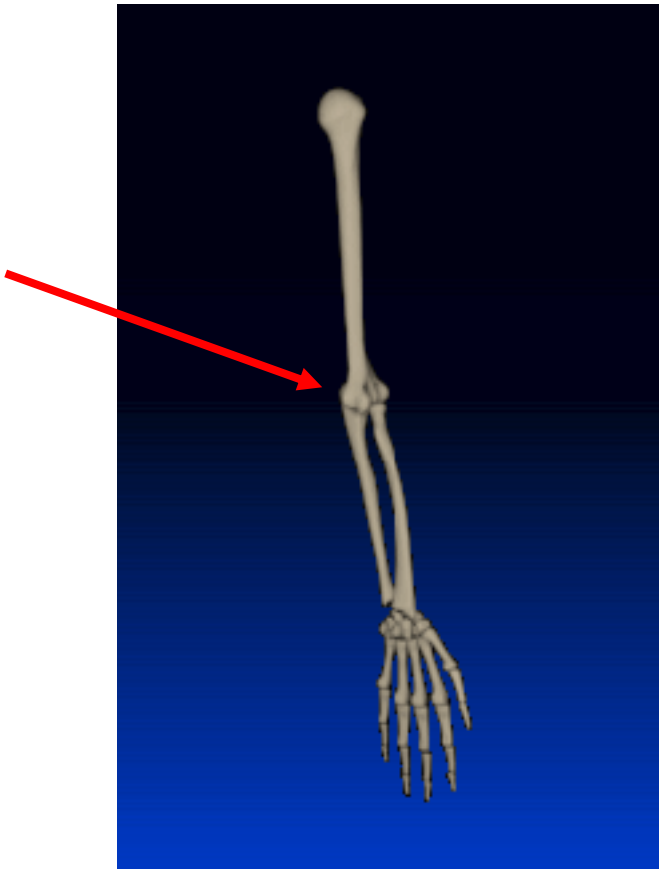

Figure 4 Human Elbow

Full physics based animation must present a suitable musculoskeletal model that produces the level of approximation desired for the figure. The attachments of muscles to the skeleton must be well defined. Forward kinetics is the study of motion by applying forces on a skeleton to produce the joint motion. This method of animation is usually used in simulators. Inverse kinetics is backwards to this approach and calculates forces that are necessary to make a joint move from one position to another. The ability for this method to get from one pose to the next is what makes it ideal for cartoon or game animation when key framing is used. Kinematics is different from kinetics in that it uses angular motion instead of forces. Forward kinematics produces skeletal motion by apply-

ing joint rotations from a specific motion model. Inverse kinematics produces the desired pose by calculating the necessary angular movements.

1.2 Motion Capture Systems

Studying articulated motion is not possible without capturing positions of real motion during the activities. There have been many methods devised to capture positional information. The most common information that is captured is the three-dimensional position of specific points on the articulated figure. There are various commercial systems for 3D positions that involve many technologies. zFlo, Inc. (http://www.zflomotion.com/) has a video based system. The Basler A 600 series of digital cameras can take 105 frames per second.

Three or more of these cameras are placed around the actor in such a way to be able to see the markers on the body throughout the motion under study. After the videos are taken, the markers can be triangulated assuming that the camera positions are previously measured or calibrated. zFlo provides software for this analysis (Figure 5).

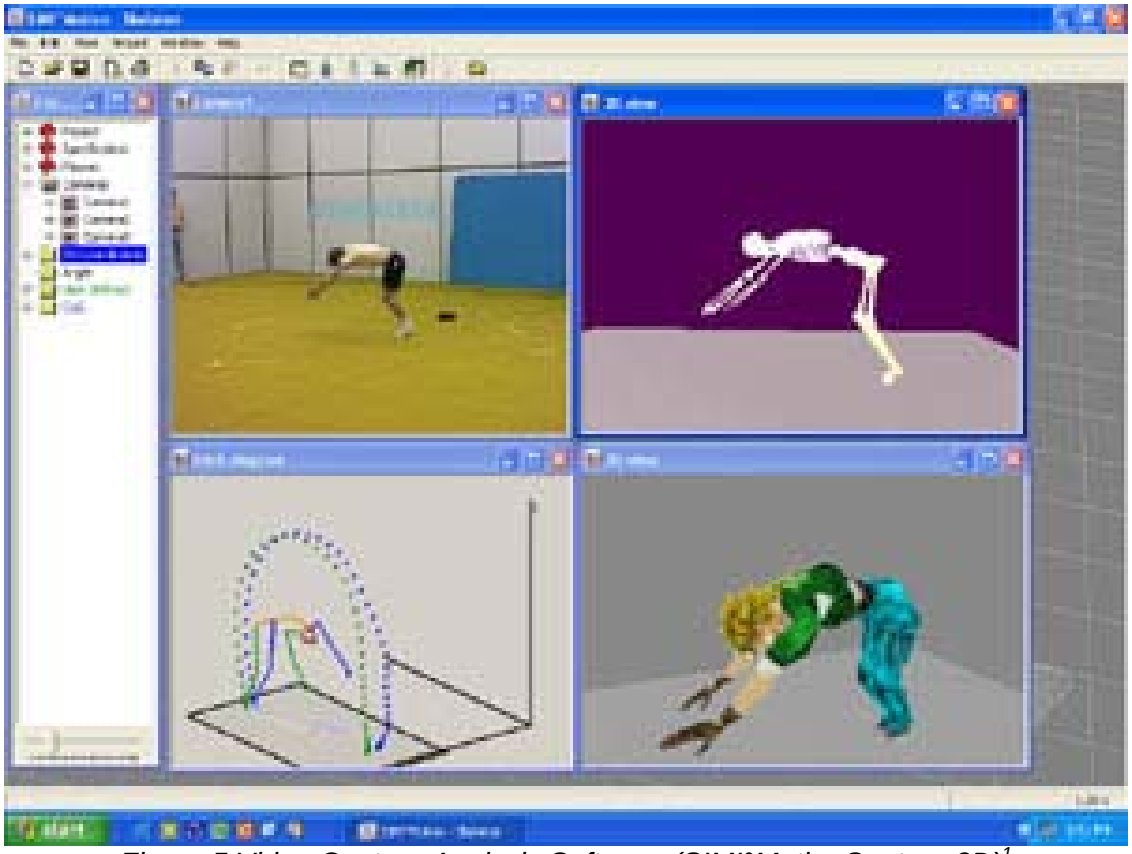

Figure 5 Video Capture Analysis Software (SIMI°MotionCapture 3D)[1]

Another more popular video capture system is BodyBuilder (Figure 6) from Vicon Peak (http://www.vicon.com). Their system has been around for more than ten years and has been used in movies by Industrial Light and Magic. Their software is considered robust with many man-years of software development.

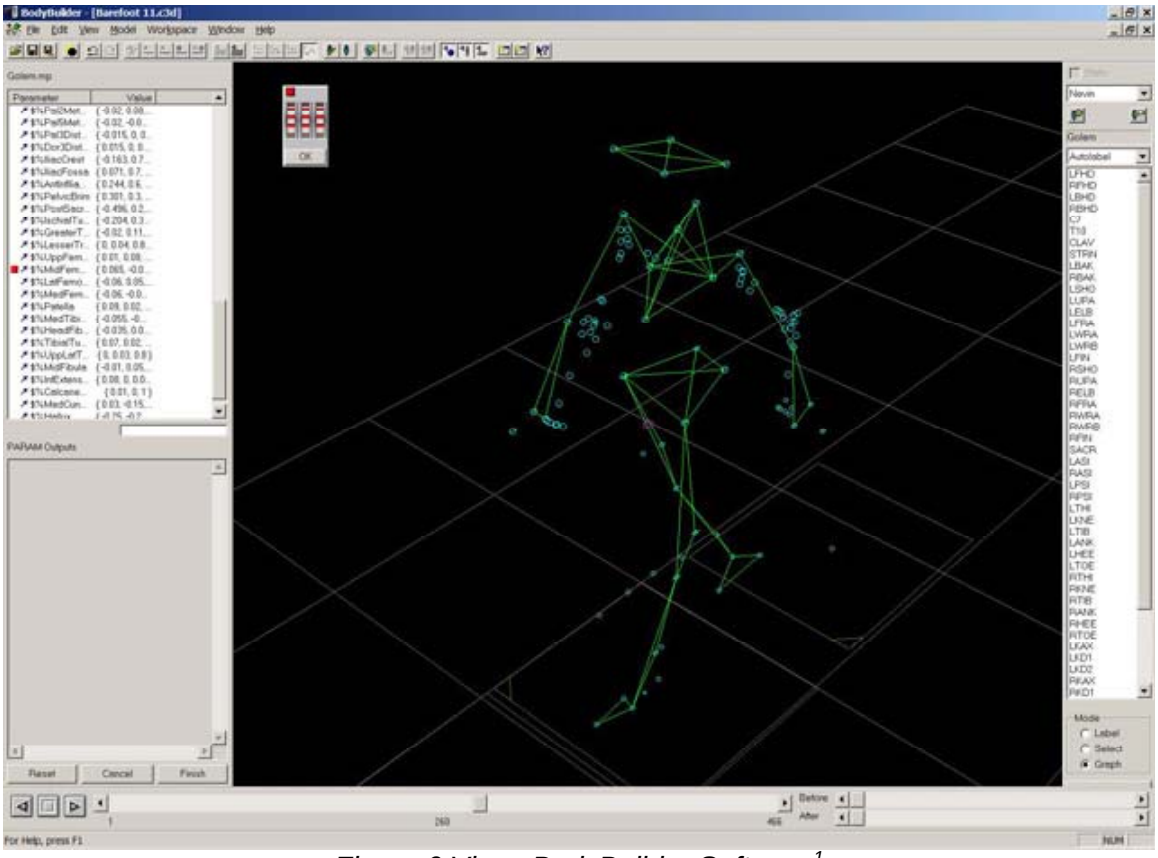

Figure 6 Vicon BodyBuilder Software[1]

1.3 Symbols and Conventions

The mathematical symbols and acronyms that are used consistently through this paper are presented in the following table. The definitions are presented later when they are first used.

$O(y)$	on the order of (i.e. size of) y
N	number of measurements
$CRLB$	Cramér-Rao Lower Bound covariance of estimator
D	dimension of hypersphere
$FLOP$	floating point operation
x_i	i^{th} measurement of position
$\bar{x}$	average of all measurements of positions

[1] Copyright Vicon Peak used with permission

μ_i ..... expectation of the i^{th} position
$\overline{\mu}$ ..... average of expectations of all positions
Σ ..... measurement covariance of each position
$\hat{\Sigma}$ ..... measurement covariance estimator
σ ..... measurement standard deviation
$\mathbf{C}$ ..... sample covariance
$\mathbf{C}_0$ ..... true covariance of expected positions
$\mathbf{S}$ ..... sample third central vector moment
$\mathbf{S}_0$ ..... true third central moment of expected positions
$\mathbf{F}_0$ ..... true fourth central moment of expected positions
$\hat{c}$ ..... center estimator
c_0 ..... true center of hypersphere
$\hat{r}$ ..... radius estimator
r_0 ..... true radius of hypersphere
λ ..... eigenvalue of matrix
v ..... eigenvector of matrix
x^T ..... transpose of column vector into row vector
A^{-1} ..... multiplicative inverse of matrix
A^T ..... transpose of matrix
∇ ..... vector gradient operator
$|x|$ ..... magnitude of vector
$|A|$ ..... determinant of matrix
$N_3(\mu,\Sigma)$ ..... three dimensional vector Normal Distribution
$\rho(A)$ ..... spectral radius of matrix
$Tr(A)$ ..... trace (sum of diagonals) of matrix
$E(A)$ ..... expectation of random variate
$Var(A)$ ..... variance of random variate
$Cov(A,B)$ ..... covariance of two variates (matrix or scalar)

Chapter 2 STATEMENT OF THE PROBLEM

Skeletons from motion capture can be produced by two broad categories: predictive and direct. Predictive methods start with an existing pose and, through constraints or rules, projects forward to find the next pose or fills in between data. These methods are good for filling in missing data or solving for new poses for a skeleton where no data existed before. Examples of predictive methods are inverse kinematics and kinetics. These methods are basically simulations of what the skeleton should look like based on some rules. Direct methods, on the other hand, are raw calculations that produce a skeleton from existing motion data. Examples of direct methods are sphere-fit joints and best-fit whole skeleton squishing. The problem addressed in this thesis lies in direct skeleton formation from motion capture data. The solutions presented do not in any way predict where a skeleton should be but instead calculates directly from raw motion data where the skeleton is currently. The motion capture systems in use today produce real-time data, but producing a skeleton usually involves many off-line processes. Iterative procedures, both linear and non-linear, are currently used to produce a skeleton that fits the data. Although some of these are considered fast with today's computers, they still take much computer labor. The goal of this thesis is to produce a faster direct measurement of a skeleton that produces a solid basis for further uses.

2.1 Why are skeleton calculations required?

Joint locations are one of the key steps in producing animation from motion capture data. Just like the human skeleton, a skeleton from motion capture data is a solid framework on which shapes can be attached. The movie industry has been using motion

capture for the purposes of animating a figure (e.g. Gollum of "The Lord Of The Rings") for many years. The status quo for this science seems to satisfy many industries. The research presented here will show that producing a skeleton can be improved ten-fold in speed.

2.2 Problems encountered

Many animation systems require painstaking detail to determine point of rotation. A-priori knowledge of the skeleton is a major obstacle in finding joint locations. The human body does not come in standard sizes and the point of rotation is below the skin making measurements difficult. The a-priori measurements taken are usually segment lengths and marker positions on the body. Unfortunately, most readily available motion capture data do not come with this detail. There are ways around the measurements of the actor. They usually involve a non-linear iteration of squishing a skeleton until it fits the motion capture data as in Kirk:2005 [60].

2.2.1 Inaccuracies

The inaccuracies in determining joint location can be divided into two types - positional and systematic inaccuracies. Positional errors are those that simply explain the error ellipsoid around the position coordinates. These are usually referred to as measurement error and are symbolized as σ for the standard deviation and Σ for the full variance-covariance matrix in this paper. Systematic errors involve gradual drifts, hysteresis, or other kinds of "slippages". One simple example on a marker system is when a marker is placed on loose clothing so that the marker position does not represent a stationary position on the segment. Even if the marker is glued flush with the skin, the fact muscles

flex and contract will introduce systematic errors. It would be better if the markers were drilled under the skin and attached to the bone, but volunteers are hard to find.

2.2.2 Non-standard Files

Another issue that comes with motion capture data is that files incompletely follow a standard format. This may result in integers being stored in signed instead of unsigned format or vice versa. This produces messed up data that may or may not be visible after the file is read. Some of the fields in files may be filled incorrectly. This would lead to unknown behavior or worse.

2.2.3 Missing Data

There are many situations that occur during a motion capture session when data becomes unavailable to grab. Markers can be occluded during even normal motion depending on angles and positions of cameras. This shows up in the data file as various frames of data with missing pieces. The reader of the file must carefully consider what to do when data is missing. Frame-based storage of the data has inherent holes whereas linked data can skip over these holes as long as the non-linear time increments are considered.

Chapter 3 SURVEY

3.1 Skeleton Extraction

O'Brien, et al., in 2000 [84] and with Kirk in 2005 [60] have produced a fast method for skeleton formation using a linear least squares method assuming there is a relatively stationary point between two segments, and then solving for that point, which is the rotation point.

3.2 Sphere Estimates

Leendert de Witte [116], in 1960, found a solution for a circle in 3-D space. He used spherical trigonometry to solve the minimized distance from the best great circle path. He used approximations that are convenient for large radii and had to choose from three solutions to get the correct one.

The Maximum Likelihood Estimator (MLE) was the first solution for finding the sphere parameters. In 1961 Stephen Robinson [95] presented the iterative method of solving the sphere by minimizing

(1) $$\varepsilon^2 = \sum_{i=1}^{N} \left(\sqrt{(x_i - \hat{c})^T (x_i - \hat{c})} - \hat{r} \right)^2$$

There is a closed form solution for the radius estimator but not for the center estimator. Robinson showed the radius estimator to be

(2) $$\hat{r} = \tfrac{1}{N} \sum_{i=1}^{N} \sqrt{(x_i - \hat{c})^T (x_i - \hat{c})}$$

A truly closed form solution wasn't found until another function to minimize was recognized. The new function to minimize was

(3) $$\varepsilon^2 = \sum_{i=1}^{N} \left(\left(x_i - \hat{c} \right)^T \left(x_i - \hat{c} \right) - \hat{r}^2 \right)^2$$

Paul Delogne pioneered what would eventually become the Generalized Delogne-Kása estimator. In 1972, Delogne presented [41] a method for solving a circle for the purposes of determining reflection measurements on transmission lines. Delogne's solution to the circle involves the inverse of a 3x3 matrix. He presented the matrix solution and a rough error analysis as

(4) $$\begin{pmatrix} \hat{c} \\ \hat{c}^T \hat{c} - \hat{r}^2 \end{pmatrix} = \begin{pmatrix} 8\sum_{i=1}^{N} x_i x_i^T & -4N\bar{x} \\ -4N\bar{x}^T & 2N \end{pmatrix}^{-1} \begin{pmatrix} 4\sum_{i=1}^{N} x_i x_i^T x_i \\ -2\sum_{i=1}^{N} x_i^T x_i \end{pmatrix}$$

(5) $$\sigma_i^2 = 2\hat{r} \left| \sqrt{\left(x_i - \hat{c} \right)^T \left(x_i - \hat{c} \right)} - \hat{r} \right|.$$

István Kása did more analysis in 1976 [57]. He was the first to recognize the bias in the answer and produced better error analysis.

(6) $$\begin{pmatrix} \hat{c} \\ \hat{r}^2 - \hat{c}^T \hat{c} \end{pmatrix} = \begin{pmatrix} 2N\bar{x}^T & N \\ 2\sum_{i=1}^{N} x_i x_i^T & N\bar{x} \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{N} x_i^T x_i \\ \sum_{i=1}^{N} x_i x_i^T x_i \end{pmatrix}$$

(7) $$\sigma^2 = \frac{1}{4\hat{r}^2 N} \sum_{i=1}^{N} \left(\left(x_i - \hat{c} \right)^T \left(x_i - \hat{c} \right) - \hat{r}^2 \right)^2$$

Vaughan Pratt [93] in 1987 produced a very generic linear least squares method for algebraic surfaces. His solution was slower for spheres due to the need to extract the eigenvectors.

Gander, et. al. in 1994 [47] produced the linear least-squares method for circle fitting with the equation

$$0 = \begin{pmatrix} x_1^T x_1 & x_1^T & 1 \\ \vdots & \vdots & \vdots \\ x_N^T x_N & x_N^T & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2\hat{c} \\ \hat{c}^T \hat{c} - \hat{r}^2 \end{pmatrix} \tag{8}$$

They solved this through finding the null vector of the left matrix that is in essence the singular value decomposition.

Samuel Thomas and Y. Chan in 1995 [108] created a formula for the Cramér-Rao Lower Bound for the circle estimation.

In 1997, Lukács, et. al. [73], produced some improvements on non-linear minimization for spheres.

Corral, et. al. [40] in 1998 analyzed Kása's formula in more detail and a way to reject the answer if the confinement angle got too small.

A paper in nuclear physics describes a method in 2000 to produce the circular arc of a particular traveling in a cloud chamber. Strandlie, et. al. [106] transformed a Riemann sphere into a plane and fit the plane using standard methods involving the eigenvalues of the sample covariance matrix.

Zelniker in 2003 [120] reformulated the circle equation to solve directly for the center using the pseudo-inverse ($^{\#}$) of a 2xN matrix.

(9)
$$\hat{c} = \frac{1}{2}\begin{pmatrix} x_1^T - \bar{x}^T \\ x_2^T - \bar{x}^T \\ \vdots \\ x_N^T - \bar{x}^T \end{pmatrix}^{\#} \begin{pmatrix} x_1^T x_1 - \frac{1}{N}\sum_{i=1}^{N} x_i^T x_i \\ x_2^T x_2 - \frac{1}{N}\sum_{i=1}^{N} x_i^T x_i \\ \vdots \\ x_N^T x_N - \frac{1}{N}\sum_{i=1}^{N} x_i^T x_i \end{pmatrix}$$

Zelniker also came up with a better evaluation of the Cramér-Rao Lower Bound for the center estimator.

(10)
$$Cov\left(\hat{c},\hat{c}^T\right) = r^2\sigma^2 \left(\begin{pmatrix} \mu_1^T - \bar{\mu}^T \\ \mu_2^T - \bar{\mu}^T \\ \vdots \\ \mu_N^T - \bar{\mu}^T \end{pmatrix}^T \begin{pmatrix} \mu_1^T - \bar{\mu}^T \\ \mu_2^T - \bar{\mu}^T \\ \vdots \\ \mu_N^T - \bar{\mu}^T \end{pmatrix} \right)^{-1}$$

Michael Burr, et. al. [31] in 2004 created a geometric inversion technique which far surpasses the complexity needed to solve for a hypersphere. It was a non-linear approach and they failed to recognize there were simpler linear solutions to what they were solving.

In 2007, Knight et al. [63], published the initial results from the research for this dissertation in which a skeleton was formed from a closed-form solution of generic motion capture data.

3.3 Inverse Kinematics

Badler [12] in 1986 produced an interactive 6 DOF controlled technique to produce a skeleton using inverse kinematics and a joystick. Wiley et al. [115] in 1997, came up with a way to splice various regimes of motion capture and skeleton formation based on inverse kinematics. Bodenheim et al. [21], in 1997, produced an articulated skeleton

by painstaking hand measurements of the markers and inverse kinematics to optimize the joint angles.

3.4 Kinetics

In 1998 Znamenáček [118] created an efficient recursive algorithm for multibody forward kinetics.

Chapter 4 PREVIOUS SOLUTIONS

The following chapters provide an exposé on the methods that have gone before this. The theory in this thesis is presented in two phases and each phase can be analyzed individually. The two phases are the determination of the rotation points of each joint and the generation of the whole skeleton. The first phase boils down to a single mathematical problem: that of finding the center of a sphere from noisy surface data. The second phase, drawing the skeleton, is described in Chapter 4.1.5.3.

4.1 Spherical Curve-fitting Approaches

There are three techniques in use today: iterative; least squares; and algebraic best fits. They each have their advantages and disadvantages. Iterative techniques are good for accuracy, least squares are faster than iterative but slower than algebraic, and algebraic techniques are good for speed.

4.1.1 Monte-Carlo Experiment

In order to compare the three, a Monte-Carlo experiment was run using 1000 trials, each of which had anywhere from 4 to 1,000,000 samples. The runs took over a week of computational effort to collect. Each trial had a fixed standard deviation of the samples from the sphere with values ranging from 10^{-13} to 10^{14}. Each trial was further varied by a limiting angle from some random point on a sphere. The limiting angle varied from 0 to 180°, where 180° means full sphere coverage. It is important to have a limiting angle in this experiment because all sphere-fit algorithms are error-prone when this angle gets smaller. Each trial also had a radius r_0 and center c_0 randomly chosen. The

radius was varied from 0.06 to 38.5 and the center moved as much as 3.6. The individual samples were multivariate normal random variates thus

(11) $$x_i \sim N_3\left(\mu_i, \sigma^2\right)$$

where the covariance is a diagonal matrix with each diagonal equal to the square of the standard deviation. The expected value of each sample was a point that lies on a sphere thus

(12) $$\left|\mu_i - c_0\right| = r_0.$$

Each sample was confined to cover only a partial part of the sphere by the following relationship (cf. Figure 7).

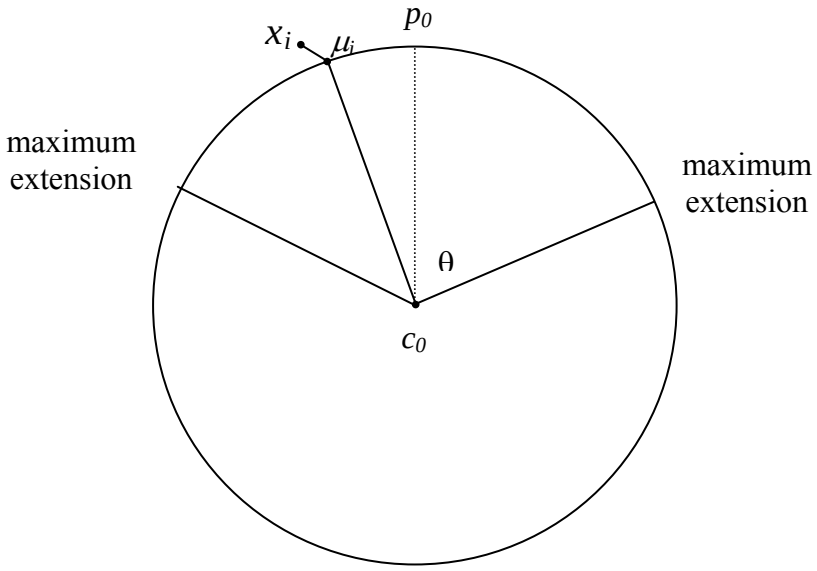

Figure 7 Constrained Measurements on Circle

The formula for checking if a point is acceptably within the limits becomes

(13) $$\left(x_i - c_0\right)^T \left(p_0 - c_0\right) > \left|x_i - c_0\right| r_0 \cos(\theta).$$

Said in another way, the sample must be confined to be within an angle from a fixed point on the sphere. This experiment allows for the in-depth analysis of the error in the answer from four different estimators. The four techniques are Maximum-Likelihood Estimator (MLE), Linear Least-Squares (LLS), Generalized Delogne-Kása Estimator (GDKE), and the new Unbiased Generalized Delogne-Kása estimator (UGDK) explained in the Chapter 5.1. The first three are established formulae and has been used for two hundred years (MLE [100]) to as young as three years (GDKE [119]).

The following graph (Figure 8) shows the errors in the estimators compared to the standard deviation of the samples. The graph shows that the relative error versus relative standard deviation is a line for each of the four methods. The error is thus proportional to the standard deviation.

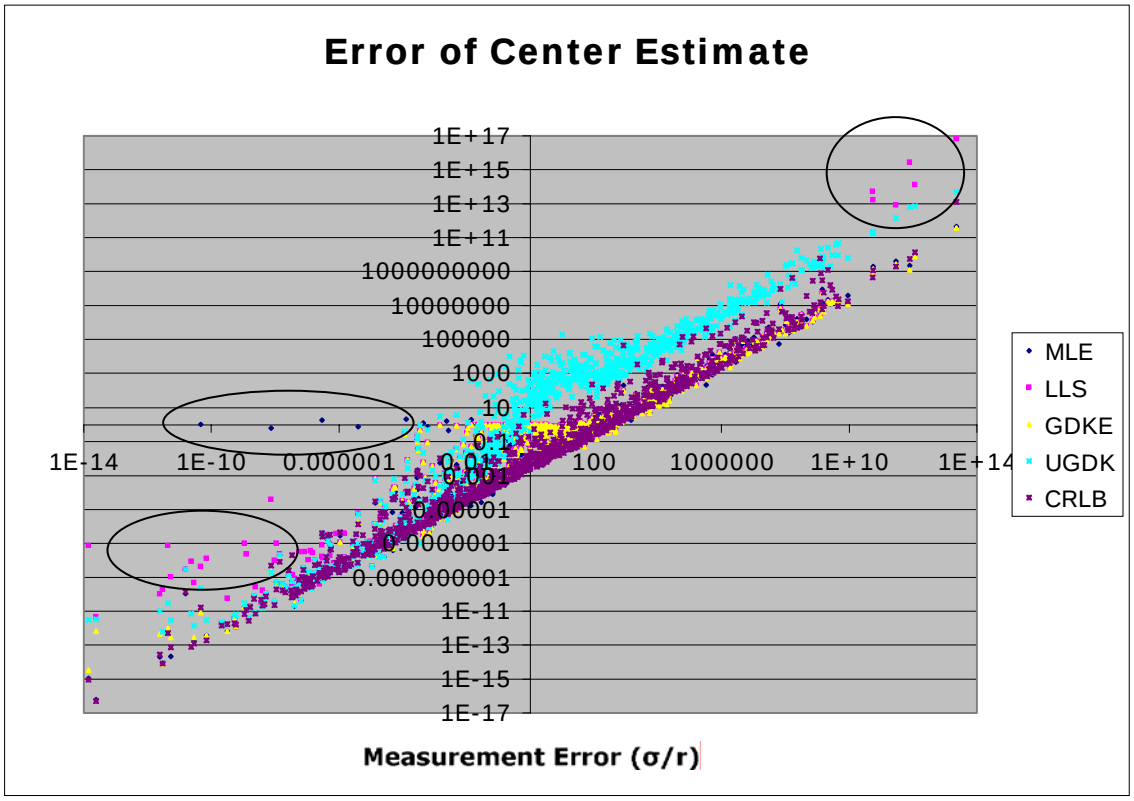

Figure 8 Relative Error Comparison

What is also clear from the graph is the comparison. The outliers on the graph have been circled. The obvious differences between the estimators show up in the graph by deviations from the straight line when error equals standard deviation. From the graph, it appears that there is a common limitation to the error in the estimator. This common limitation is named the Cramér-Rao Lower Bound to the covariance of an estimator. Section 4.1.2 discusses the limit to all estimators for this particular problem. The MLE has outliers when the error equals the radius. This is due to multiple solutions when the error equals the radius. The LLS has outliers when the standard deviation is below about 10^{-6} times the radius and greater than 10^{10} times the radius. These are due to numerical instability during the extremes of using finite representation of decimal numbers. The GDKE seems to be on par with the MLE except for the MLE outliers. The

UGDK has consistently larger error when the standard deviation is greater than about 0.1 times the radius. Each one of these shortcomings will be further discussed in sections 4.1.3, 4.1.4, 4.1.5, and 5.1.

4.1.2 Cramér-Rao Lower Bound

The Cramér-Rao Lower Bound (CRLB) is the proven lower bound for any estimator's covariance. It is equal to the inverse of the Fisher Information. It is an important measure when dealing with any estimator because it is the best error that an estimator can achieve. All estimators will have at best an error of the CRLB. The CRLB for the center and radius estimators are given for this particular problem [55] as

$$CRLB^{-1} = \sum_{i=1}^{N} \begin{pmatrix} (\mu_i - c_0)(\mu_i - c_0)^T & (\mu_i - c_0) r_0 \\ (\mu_i - c_0)^T r_0 & r_0^2 \end{pmatrix} \frac{1}{Tr\left((\mu_i - c_0)(\mu_i - c_0)^T \Sigma\right)} \tag{14}$$

where

μ_i is the expected value of the i^{th} measurement on the surface

c_0 is the true center

r_0 is the true radius

Σ is the measurement covariance.

This is a positive-definite matrix of size (D+1)x(D+1) where D is the number of dimensions of the hypersphere. The *CRLB* matrix is the smallest value of the covariance of a estimator for a particular measurement system. This can be reduced using the isotropic covariance assumption ($\Sigma = I\sigma^2$). The result of this substitution is

(15) $$CRLB = \sigma^2 \begin{pmatrix} \sum_{i=1}^{N} \frac{(\mu_i - c_0)(\mu_i - c_0)^T}{r_0^2} & \sum_{i=1}^{N} \frac{(\mu_i - c_0) r_0}{r_0^2} \\ \sum_{i=1}^{N} \frac{(\mu_i - c_0)^T r_0}{r_0^2} & \sum_{i=1}^{N} \frac{r_0^2}{r_0^2} \end{pmatrix}^{-1}$$

because of the identity

(16) $$(\mu_i - c_0)^T (\mu_i - c_0) = r_0^2$$

This summation can be expanded and simplified. The equivalent sum is

(17) $$CRLB = \sigma^2 \begin{pmatrix} \frac{NB}{r_0^2} & N \frac{(\bar{\mu} - c_0)}{r_0} \\ N \frac{(\bar{\mu} - c_0)^T}{r_0} & N \end{pmatrix}^{-1}$$

where

(18) $$\mathbf{B} = \frac{1}{N} \sum_{i=1}^{N} (\mu_i - c_0)(\mu_i - c_0)^T$$

or equivalently

(19) $$CRLB = \frac{\sigma^2}{N} \begin{pmatrix} \frac{\mathbf{B}}{r_0^2} & \frac{(\bar{\mu} - c_0)}{r_0} \\ \frac{(\bar{\mu} - c_0)^T}{r_0} & 1 \end{pmatrix}^{-1}$$

This partitioned matrix is inverted to

(20) $$CRLB = \tfrac{1}{N} \sigma^2 \begin{pmatrix} r_0^2 \mathbf{B}^{-1} \left(\mathbf{I} + \alpha (\bar{\mu} - c_0)(\bar{\mu} - c_0)^T \mathbf{B}^{-1} \right) & -\alpha r_0 \mathbf{B}^{-1} (\bar{\mu} - c_0) \\ -\alpha r_0 (\bar{\mu} - c_0)^T \mathbf{B}^{-1} & \alpha \end{pmatrix}$$

where

(21) $$\alpha = \left(1 - Tr\left(\left(\bar{\mu} - c_0\right)\left(\bar{\mu} - c_0\right)^T \mathbf{B}^{-1}\right)\right)^{-1}$$

The determinant of this matrix is

(22) $$\left|CRLB\right| = \left(\tfrac{1}{N}\sigma^2 r_0^2\right)^{D+1} \frac{\alpha}{\left|\mathbf{B}\right|}$$

For large amounts of sampling, the Cramér-Rao Lower Bound has an even more refined formula for the constrained samples on the surface of the hypersphere. Like in Figure 7, the constrained sampling is confined to an angle θ from a particular point on the surface. The two-dimensional case (i.e. the circle) can be setup so that the calculus is made simpler. First, use a coordinate system that has its center where the center of the circle is. The y-axis is projected towards the particular point that lies on the surface and is the center of the confinement. A point on the circumference is then defined as

(23) $$v_2 = \begin{pmatrix} r_0 \sin a \\ r_0 \cos a \end{pmatrix}$$

where *a* is the angle from the y-axis towards the x-axis. The center of the circle is then a zero vector in this coordinate frame.

(24) $$c_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

For large amounts of sampling of the two-dimensional case, Equation (14) becomes an integral that averages the integrand over the region on the circumference of the circle thus

$$(25)\quad \lim_{N\to\infty} CRLB \equiv CRLB_2 = \tfrac{1}{N}\phi_2 r_0^2 \begin{pmatrix} \int_{-\theta}^{\theta} \frac{(v_2-c_0)(v_2-c_0)^T}{(v_2-c_0)^T\Sigma(v_2-c_0)}da & \int_{-\theta}^{\theta} \frac{(v_2-c_0)r_0}{(v_2-c_0)^T\Sigma(v_2-c_0)}da \\ \int_{-\theta}^{\theta} \frac{(v_2-c_0)^T r_0}{(v_2-c_0)^T\Sigma(v_2-c_0)}da & \int_{-\theta}^{\theta} \frac{r_0^2}{(v_2-c_0)^T\Sigma(v_2-c_0)}da \end{pmatrix}^{-1}$$

where

$$(26)\qquad \phi_2 = \int_{-\theta}^{\theta} da = 2\theta$$

The $CRLB_2$ simplifies greatly with an isotropic covariance ($\Sigma=I\sigma^2$) as well as a zero center:

$$(27)\qquad CRLB_2 = \tfrac{1}{N}\sigma^2\phi_2 r_0^2 \begin{pmatrix} \int_{-\theta}^{\theta} v_2 v_2^T da & r_0\int_{-\theta}^{\theta} v_2 da \\ r_0\int_{-\theta}^{\theta} v_2 da & r_0^2\int_{-\theta}^{\theta} da \end{pmatrix}^{-1}$$

Taking the integral produces the answer in this coordinate system.

$$(28)\qquad CRLB_2 = \tfrac{1}{N}\sigma^2 2\theta \begin{pmatrix} \theta-\cos\theta\sin\theta & 0 & 0 \\ 0 & \theta+\cos\theta\sin\theta & 2\sin\theta \\ 0 & 2\sin\theta & 2\theta \end{pmatrix}^{-1}$$

The inverse produces

$$(29)\qquad CRLB_2 = \tfrac{1}{N}\sigma^2 \frac{2\theta}{\det_2} \begin{pmatrix} \frac{\det_2}{\theta-\cos\theta\sin\theta} & 0 & 0 \\ 0 & 2\theta & -2\sin\theta \\ 0 & -2\sin\theta & \theta+\cos\theta\sin\theta \end{pmatrix}$$

where

$$(30)\qquad \det_2 = 2\theta(\theta+\cos\theta\sin\theta) - 4\sin^2\theta$$

This has three distinct eigenvalues. The one for the x-direction (perpendicular to the line of symmetry) is

(31) $$\lambda_{m2} = \tfrac{1}{N}\sigma^2 \frac{2\theta}{\theta - \cos\theta \sin\theta}$$

corresponding to the direction

(32) $$v_{x2} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

The largest eigenvalue corresponds, for the most part, to the y-direction and is

(33) $$\lambda_{M2} = \tfrac{1}{N}\sigma^2 \left(\frac{4\theta}{3\theta + \cos\theta\sin\theta - \sqrt{(\theta - \cos\theta\sin\theta)^2 + 16\sin^2\theta}} \right)$$

corresponding to the direction

(34) $$v_{y2} = \begin{pmatrix} 0 \\ 1 \\ -\alpha \end{pmatrix}$$

where

(35) $$\alpha = \frac{-\theta + \cos\theta\sin\theta + \sqrt{(\theta - \cos\theta\sin\theta)^2 + 16\sin^2\theta}}{4\sin\theta}$$

The variable α is a function going from $\alpha=1$ at $\theta=0$ to $\alpha=0$ at $\theta=\pi$. It is graphed in the next figure:

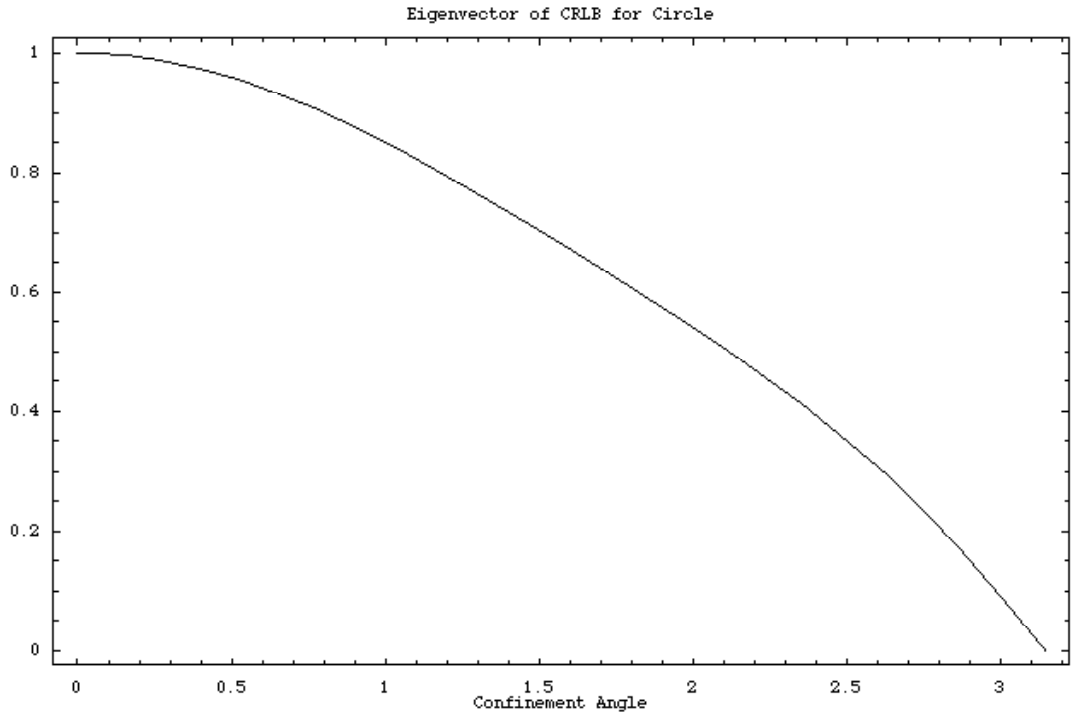

Figure 9 Eigenvectors of CRLB for Circle

The smallest eigenvalue corresponds, for the most part, to the r-direction (radius error) and is

$$\lambda_{r2} = \tfrac{1}{N}\sigma^2\left(\frac{4\theta}{3\theta+\cos\theta\sin\theta+\sqrt{(\theta-\cos\theta\sin\theta)^2+16\sin^2\theta}}\right) \tag{36}$$

corresponding to the direction

$$v_{r2} = \begin{pmatrix} 0 \\ \alpha \\ 1 \end{pmatrix} \tag{37}$$

The eigenvalues follow the relationship

$$\lambda_{r2} \le \lambda_{m2} \le \lambda_{M2} \tag{38}$$

for all values of the confinement angle. They are graphed in the following figure.

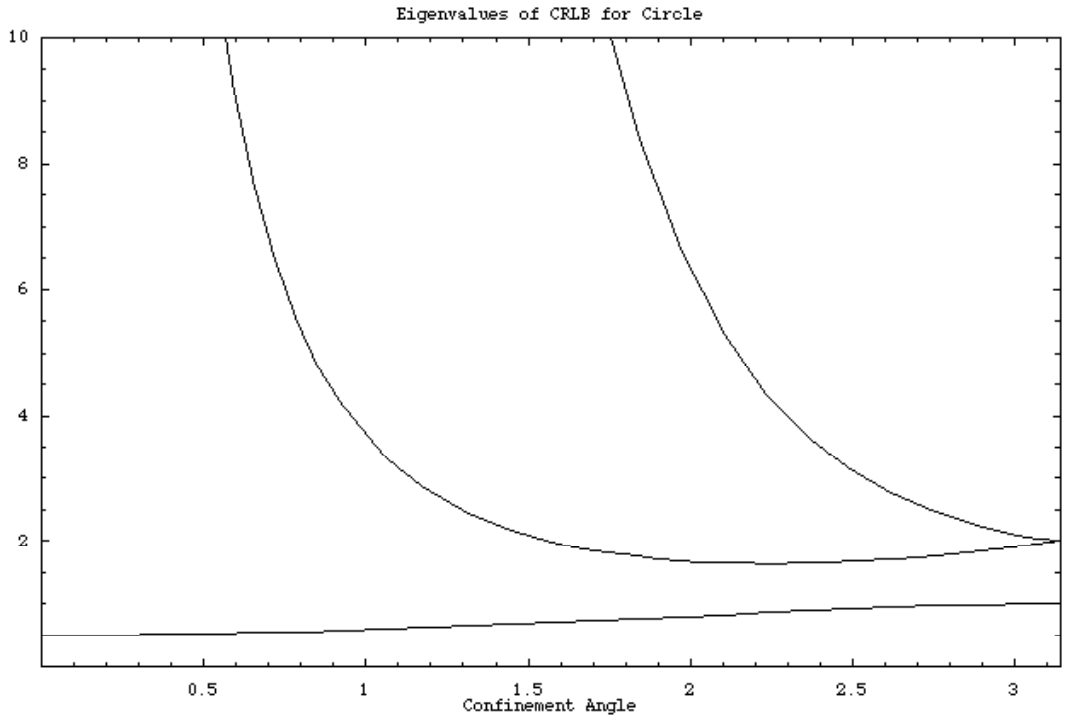

Figure 10 Eigenvalues of CRLB for Circle

Figure 10 shows that the variance for the center estimation increases to infinite as the confinement angle approaches zero. These eigenvalues determine the size of the error ellipse, which bulges towards the points on the surface of the circle.

The three-dimensional equivalent (i.e. sphere) is a bit different. The idea is still the same, which is to confine the measurements to be within an angle from a specific point on the surface. The coordinate system is set up so that the z-axis is pointing towards the confinement point on the surface. A point on the surface is then defined by two angles *a* and *b* thus

$$v_3 = \begin{pmatrix} r_0 \sin a \cos b \\ r_0 \sin a \sin b \\ r_0 \cos a \end{pmatrix} \tag{39}$$

The center of the sphere is at the center of the coordinate system thus

(40) $$c_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

The CRLB in three dimensions has the limit for large sampling as

(41) $$\lim_{N\to\infty} CRLB \equiv CRLB_3 = \tfrac{1}{N}\sigma^2 r_0^2 \phi_3 \begin{pmatrix} \int_0^{\theta}\int_0^{2\pi} v_3 v_3^T \sin a \, db \, da & r_0 \int_0^{\theta}\int_0^{2\pi} v_3 \sin a \, db \, da \\ r_0 \int_0^{\theta}\int_0^{2\pi} v_3^T \sin a \, db \, da & r_0^2 \int_0^{\theta}\int_0^{2\pi} \sin a \, db \, da \end{pmatrix}^{-1}$$

where

(42) $$\phi_3 = \int_0^{\theta}\int_0^{2\pi} \sin a \, db \, da = 4\pi \sin^2\left(\tfrac{\theta}{2}\right)$$

Taking the double integral produces

(43) $$CRLB_3 = \tfrac{1}{N}\sigma^2 \begin{pmatrix} \frac{(2+\cos\theta)\sin^2\left(\frac{\theta}{2}\right)}{3} & 0 & 0 & 0 \\ 0 & \frac{(2+\cos\theta)\sin^2\left(\frac{\theta}{2}\right)}{3} & 0 & 0 \\ 0 & 0 & \frac{3\cos^4\left(\frac{\theta}{2}\right)+\sin^4\left(\frac{\theta}{2}\right)}{3} & \cos^2\left(\frac{\theta}{2}\right) \\ 0 & 0 & \cos^2\left(\frac{\theta}{2}\right) & 1 \end{pmatrix}^{-1}$$

(44) $$CRLB_3 = \tfrac{1}{N}\sigma^2 \frac{3}{\sin^4\left(\frac{\theta}{2}\right)} \begin{pmatrix} \frac{\sin^2\left(\frac{\theta}{2}\right)}{(2+\cos\theta)} & 0 & 0 & 0 \\ 0 & \frac{\sin^2\left(\frac{\theta}{2}\right)}{(2+\cos\theta)} & 0 & 0 \\ 0 & 0 & 1 & -\cos^2\left(\frac{\theta}{2}\right) \\ 0 & 0 & -\cos^2\left(\frac{\theta}{2}\right) & \frac{3\cos^4\left(\frac{\theta}{2}\right)+\sin^4\left(\frac{\theta}{2}\right)}{3} \end{pmatrix}$$

The middle sized eigenvalue is

(45) $$\lambda_{m3} = \tfrac{1}{N}\sigma^2 \frac{3}{(2+\cos\theta)\sin^2\left(\frac{\theta}{2}\right)}$$

corresponding to the directions

(46) $$v_{x3} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \text{ and } v_{y3} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

The largest eigenvalue is

(47)

$$\lambda_{M3} = \tfrac{1}{N}\sigma^2\left(\frac{24}{18 + 4\cos\theta + 2\cos(2\theta) - \sqrt{2}\sqrt{131 + 124\cos\theta + 28\cos(2\theta) + 4\cos(3\theta) + \cos(4\theta)}}\right)$$

corresponding to the directions

(48) $$v_{z3} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \alpha \end{pmatrix}$$

where

(49) $$\alpha = -\frac{1}{3}(2 + \cos\theta)\tan^2\left(\tfrac{\theta}{2}\right) + \frac{\sqrt{2}\sqrt{131 + 124\cos\theta + 28\cos(2\theta) + 4\cos(3\theta) + \cos(4\theta)}}{12(1 - \cos\theta)\cos^2\left(\tfrac{\theta}{2}\right)}$$

The third and final eigenvalue corresponds to the *r*-direction and is

(50) $$\lambda_{r3} = \tfrac{1}{N}\sigma^2\left(2 + \frac{3}{\tan^2\left(\tfrac{\theta}{2}\right)\sin^2\left(\tfrac{\theta}{2}\right)} - \frac{\sqrt{131 + 124\cos\theta + 28\cos(2\theta) + 4\cos(3\theta) + \cos(4\theta)}}{4\sqrt{2}\sin^4\left(\tfrac{\theta}{2}\right)}\right)$$

corresponding to the directions

(51)
$$v_{r3} = \begin{pmatrix} 0 \\ 0 \\ -\alpha \\ 1 \end{pmatrix}$$

The last two eigenvectors are determined by the one function α of the confinement angle. This function is graphed in Figure 11.

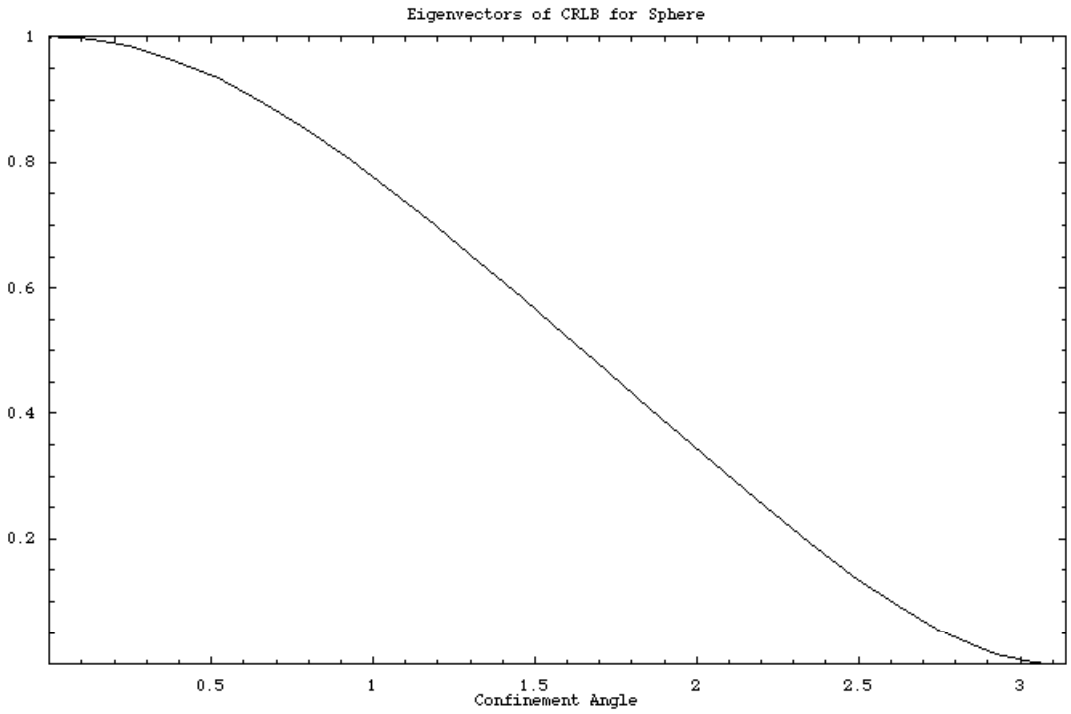

Figure 11 Eigenvectors of CRLB for Sphere

All three eigenvalues for the confined points on a sphere are graphed in Figure 12.

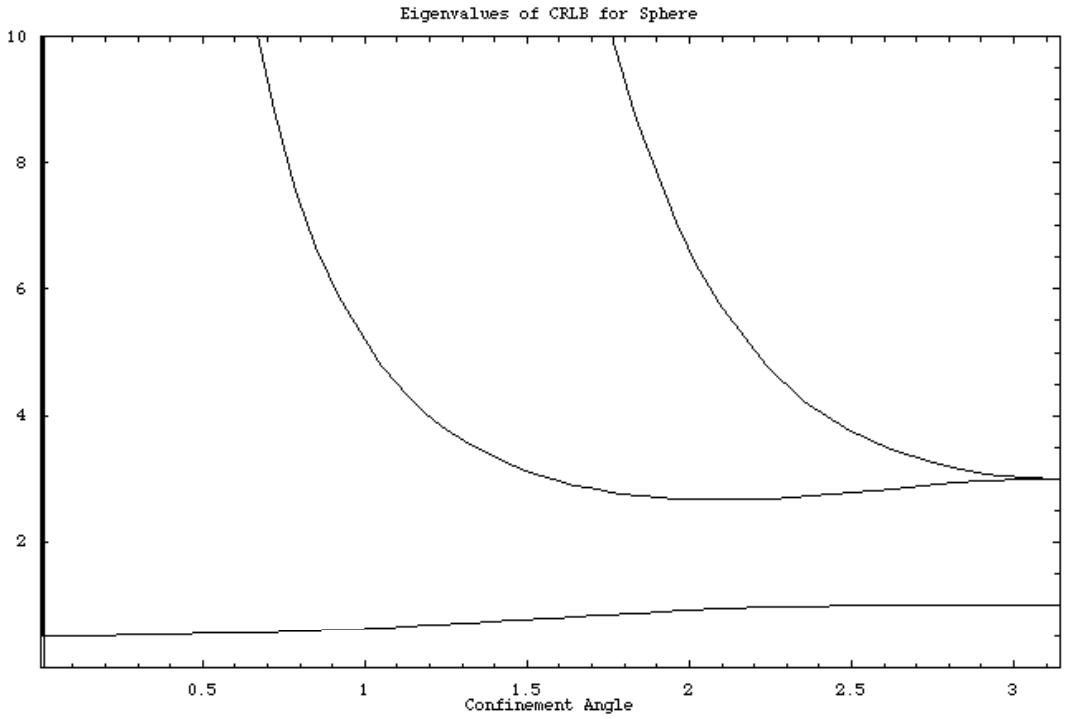

Figure 12 Eigenvalues of CRLB for Sphere

So what does all this gives me? The formulae for the eigenvalues provide the best-case error of problem at hand, i.e. all estimators aspire to these limits. There is a condition where the error in the estimate exceeds the error in the measurement. This condition occurs, for the circle, as

(52) $$CRLB_2 > \sigma^2$$

The comparison here is element by element in the matrix. The worst case for the center estimator is

(53) $$N < \frac{2\theta^2}{\theta(\theta + \cos\theta\sin\theta) - 2\sin^2\theta}$$

The sphere equivalent limit is

(54) $$N < \frac{3}{\sin^4\left(\frac{\theta}{2}\right)}$$

These two lower limits for number of samples is a good start when the circumstances arise for picking the number of samples.

4.1.3 Non-linear Maximum-Likelihood Estimator

According to the National Institute of Standards and Technology (NIST) [100], the best way to find the center of a sphere is through non-linear minimization of the variance of the radius. This is also called the Maximum-Likelihood Estimator (MLE) for the center $\hat{c}$ and radius $\hat{r}$. The variance is written as

(55) $$s_{MLE}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(r_i - \hat{r}\right)^2$$

where N is the number of samples whose positions are x_i on or near the surface of the sphere. The sample radius r_i is expressed as the vector norm (distance) of the sample from the center of the sphere thus

(56) $$r_i \equiv \left|x_i - \hat{c}\right|.$$

The two estimators, i.e. the radius $\hat{r}$ and center $\hat{c}$, are unknown in the equation and can be solved by minimizing the variance. The radius is solved easily through taking the derivative of the variance and setting it equal to zero. The derivative is

(57) $$\frac{\partial s_{MLE}^2}{\partial \hat{r}} = \frac{1}{N-1}\sum_{i=1}^{N} 2\left(\hat{r} - r_i\right) = 0$$

This summation is simplified to

(58) $$\sum_{i=1}^{N}\hat{r} = \sum_{i=1}^{N} r_i$$

which ultimately simplifies to

(59) $$\hat{r} = \frac{1}{N}\sum_{i=1}^{N} r_i$$

This estimator of the radius is still dependent on the center estimator $\hat{c}$ because of the definition of r_i. The minimization is usually carried out by iterative methods like the Levenberg-Marquardt Method [100] and cannot be solved directly. One disadvantage of the iterative technique is the need to produce an initial guess. The following graphs comes from the Monte-Carlo experiment previously mentioned and is another way of showing the error, this time dividing by the square root of the CRLB instead of the radius. The first graph compares the error to the standard deviation.

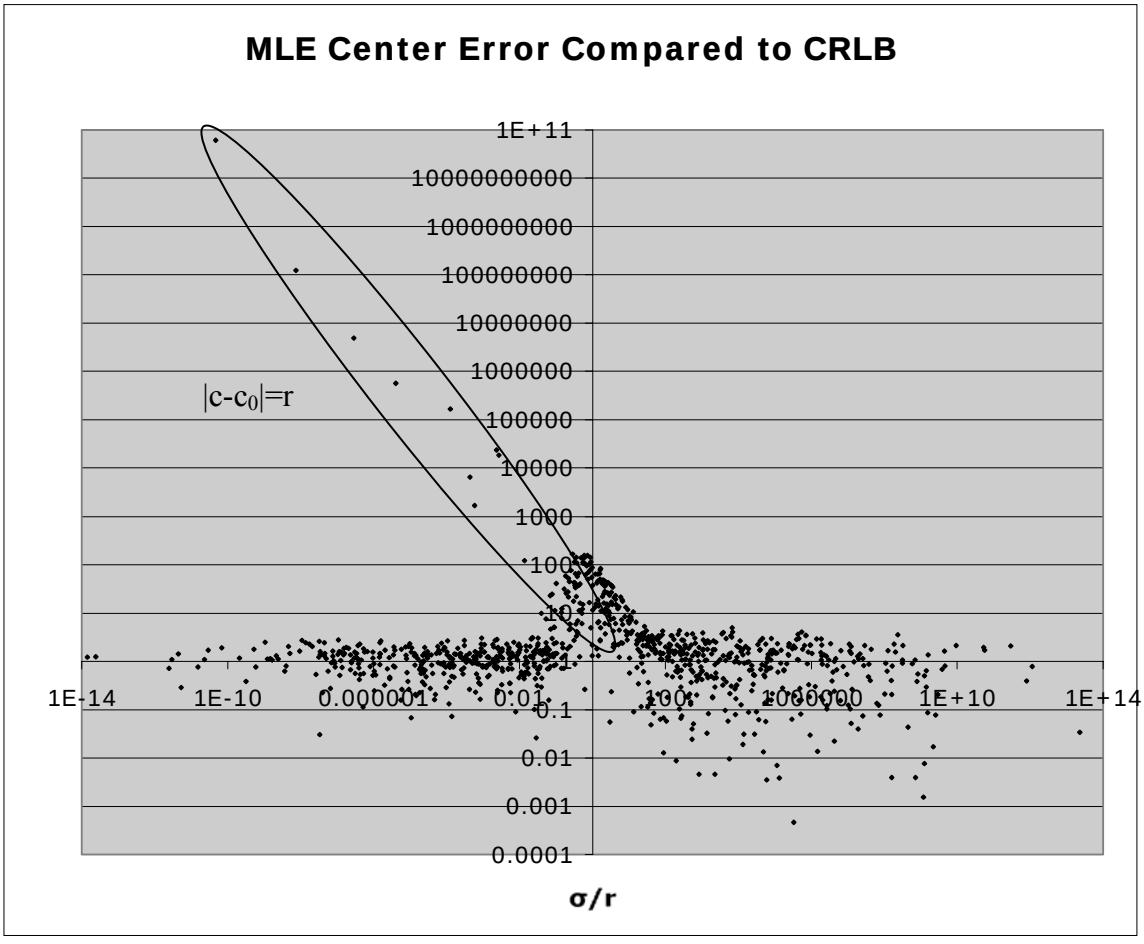

Figure 13 MLE Compared to CRLB

The outliers are clearly visible and correspond to when the error equals the radius. This is the case when the coverage angle is small and the MLE converges to the middle of the points instead of nearer to the center. The next graph explains when the outliers occur in comparison to the coverage angle. An angle of 180° means full coverage of the sphere, whereas an angle of 0° means all points confined to a single spot on the sphere.

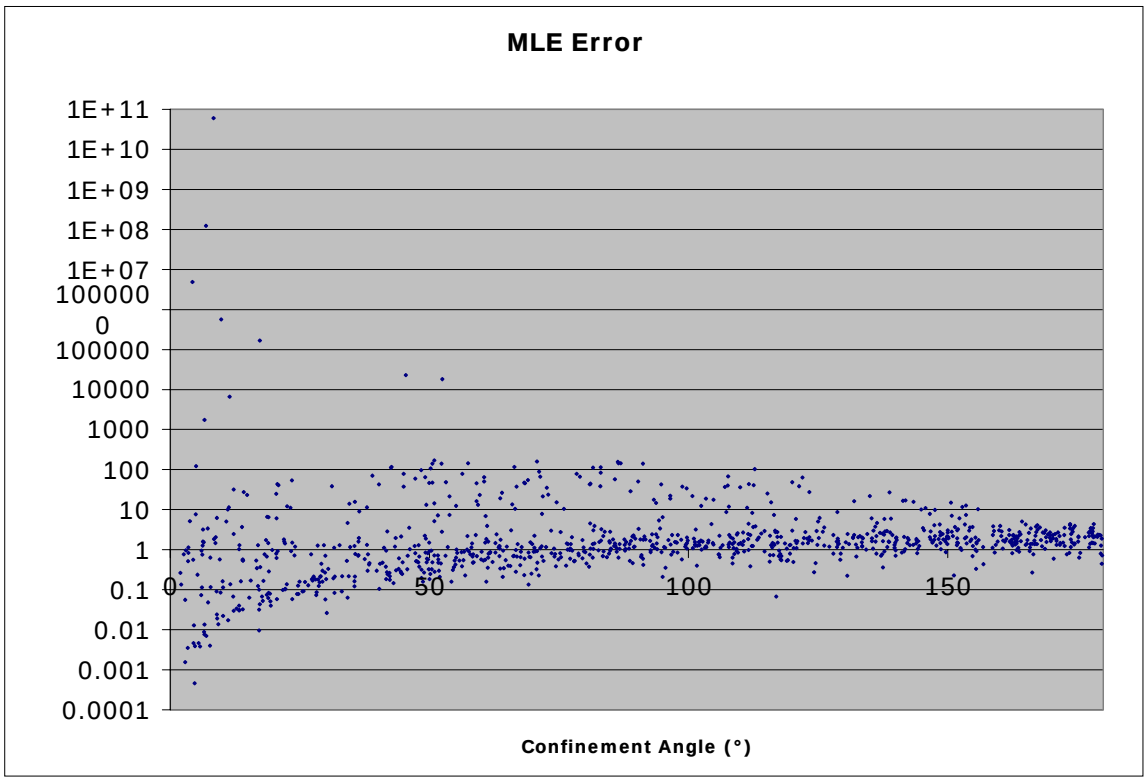

Figure 14 MLE Error Versus Sphere Coverage

This graph shows the outliers occur at low angles and indicate that the Levenberg-Marquardt technique converged on an answer that was embedded in the set of points that reside on the sphere as in the equation

(60) $$\left|\hat{c} - c_0\right| \approx r_0.$$

These outliers occur when the samples during a run are confined to a small area on the sphere. For the most part, the answer error is proportional to the square root of the CRLB as evident in Figure 13 and Figure 14.

(61) $$\left|\hat{c} - c_0\right|^2 \approx CRLB$$

4.1.4 Linear Least-Squares Solution

The next best thing to the very slow MLE method is through linear least-squares solution. Most authors use a similar solution to the one presented here. The method starts out the equation for a hypersphere

(62) $$\hat{r}^2 = \left(x_i - \hat{c}\right)^T \left(x_i - \hat{c}\right).$$

Expanding the vector product produces *N* linear equations for the sphere thus

(63) $$x_i^T x_i = 2x_i^T \hat{c} + \left(\hat{r}^2 - \hat{c}^T \hat{c}\right).$$

This is usually an over-constrained problem since there are N equations and four (i.e. *D*+1) unknowns. The *N* equations can be put into a single matrix equation to solve with standard linear algebra techniques. The linear equation is

(64) $$Y = X\hat{A}$$

where

(65) $$Y = \begin{pmatrix} x_1^T x_1 \\ x_2^T x_2 \\ \vdots \end{pmatrix}$$

(66) $$X = \begin{pmatrix} 2x_{1x} & 2x_{1y} & 2x_{1z} & 1 \\ 2x_{2x} & 2x_{2y} & 2x_{2z} & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

$$\hat{A} = \begin{pmatrix} \hat{c}_x \\ \hat{c}_y \\ \hat{c}_z \\ \hat{r}^2 - \hat{c}^T \hat{c} \end{pmatrix} \tag{67}$$

The Singular Value Decomposition (SVD) method is used to solved for $\hat{A}$ thereby retrieving $\hat{c}$ and $\hat{r}$. SVD has the advantage of being able to solve equations that contain near singular matrices. It has the disadvantage of being a slow algorithm.

Figure 15 shows the error of the center estimator. Once again, the outlier case shows when the answer erroneously lies on the sphere. This example, the LLS method shows possible bad results at the extreme ends of the standard deviation spectrum.

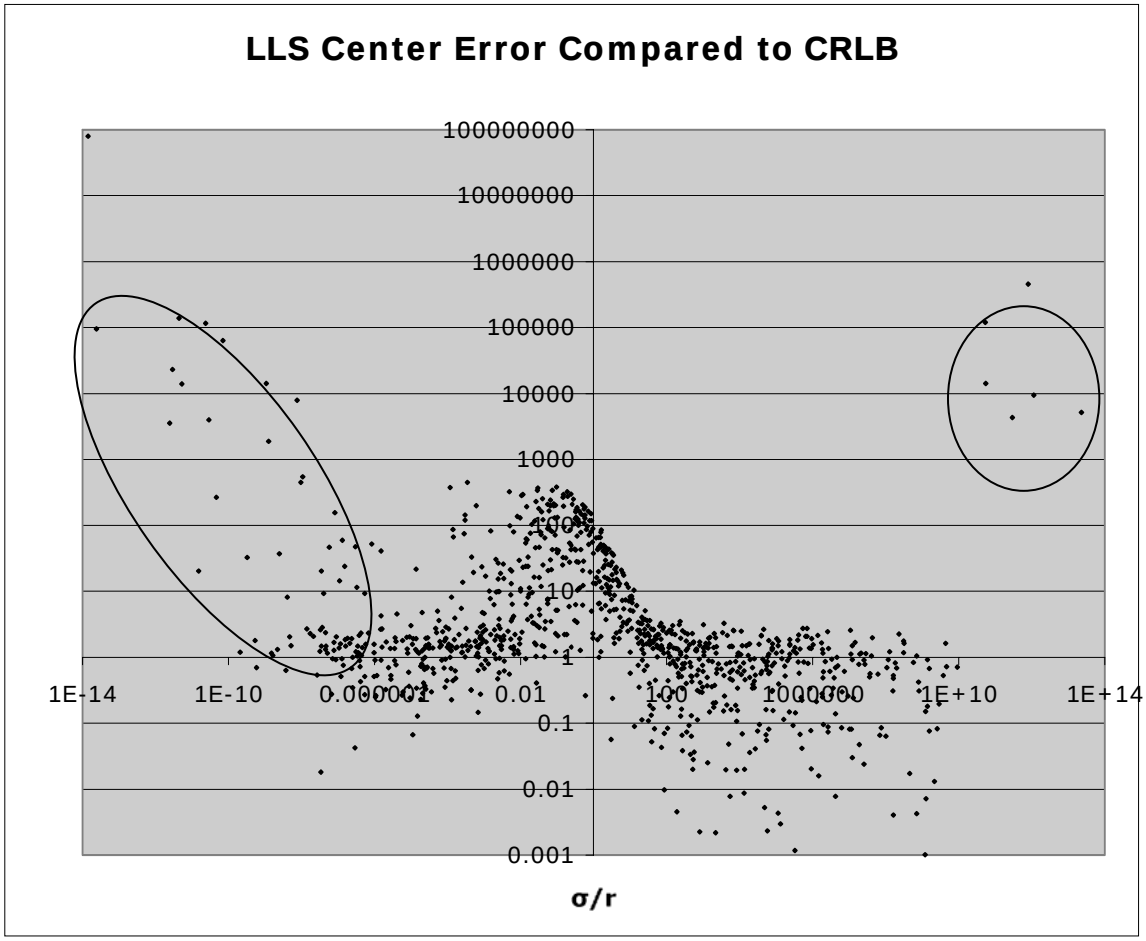

Figure 15 LLS Compared to CRLB

For the most part, once again, the answer error is proportional to the square root of the CRLB.

(68) $$\left|\hat{c} - c_0\right|^2 \approx CRLB.$$

The speed of the algorithm can be measured in flop count. Flops are floating point operations and the operations included in the count are not very well defined. Many authors allow only multiplication, division, addition and subtraction of floating point numbers to be included. Others also include the square-root function. Some exclude addition and subtraction. We adopt the one with the most (i.e. * / + - √). The LLS method here involves the Singular Value Decomposition. The Numerical Recipes [94] implementation is the most popular. Counting the individual + - * / √ operations proves cumbersome and involves an assumption about an iteration cycle in the implementation. The setup and breakdown of Equation (64) contributes

(69) $$FLOPs = N(3D-1) + (2D+1)$$

Each iteration for the algorithm increases the count by

(70) $$FLOPs = N(6D^2 + 12D + 6) + \tfrac{1}{2}(53D^2 + 117D + 70)$$

If the iterations are assumed to be one pass then the total count is

(71) $$FLOPs = N(14D^2 + 32D + 14) + \tfrac{1}{3}(13D^3 + 102D^2 + 245D + 123)$$

where D is the dimension of the sphere (e.g. 2 for circle, 3 for sphere). So for a sphere, the Linear Least-Squares method has

(72) $$FLOPs = N236 + 709$$

4.1.5 Generalized Delogne-Kása Estimator

The GDKE was first introduced in 2004 by Zelniker [119] as an extension of the estimator of the circle center. The circle estimator had been published since 1972 starting with Paul Delogne [41]. István Kása [57] further expanded the 2D theory in 1976. In the process of my investigating this problem, the research for this dissertation took the same path as Zelniker and independently discovered the equation in a more efficient form in 2005. The new but algebraically equivalent equation was presented at the Winter School of Computer Graphics Conference at Plzeň, Česká Republika in 2007 [63]. The estimator has an algebraic solution and is considered fast in relation to the previously mentioned methods.

4.1.5.1 Derivation

The problem starts out similar to the MLE, where a minimum to a variance is needed. The variance takes on the form

$$s_{GDK}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(r_i^2 - \hat{r}^2\right)^2 \tag{73}$$

where

$$r_i^2 = \left(x_i - \hat{c}\right)^T\left(x_i - \hat{c}\right) \tag{74}$$

The solution for the radius estimator falls out similar to the MLE except this time it is the square root of the sum of the squares (RMS) of the sample radii.

$$\hat{r} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} r_i^2} \tag{75}$$

If this estimator is then plugged back into the variance, it becomes evident that this estimator is the absolute minimum of the variance for any center estimator. Choose an estimator r' other than $\hat{r}$ and the difference of the variances becomes

$$s^2(r') - s^2(\hat{r}) = \left(r'^2 - \hat{r}^2\right)^2 \geq 0 \tag{76}$$

Don't forget that the center estimator $\hat{c}$ is in Equation (56) of the sample radii r_i for the radius estimator. To solve for the center estimator, the gradient of the variance with respect to the changing center estimate must be taken and then set equal to zero

$$\nabla s_{GDK}^2 = 0 \tag{77}$$

where the gradient is defined as the vector operator with respect to the center estimate

$$\nabla = \left(\frac{\partial}{\partial \hat{c}_x} \quad \frac{\partial}{\partial \hat{c}_y} \quad \frac{\partial}{\partial \hat{c}_z} \right)^T \tag{78}$$

Applying the derivative produces

$$\nabla s_{GDK}^2 = \tfrac{1}{N-1}\left(\sum_{i=1}^{N} \nabla r_i^4 - 2N\hat{r}^2 \nabla \hat{r}^2 \right) \tag{79}$$

The further derivatives can be shown to be

$$\nabla \hat{r}^2 = 2(\hat{c} - \bar{x}) \tag{80}$$

$$\nabla r_i^4 = 4 r_i^2 (\hat{c} - x_i) \tag{81}$$

These relationships and further manipulation can produce

$$\sum_{i=1}^{N} \nabla r_i^4 = 4(N-1)\left(2\mathbf{C}(\hat{c} - \bar{x}) - \mathbf{S}\right) + 4N\hat{r}^2(\hat{c} - \bar{x}) \tag{82}$$

where the special values **C** and **S** are defined as

(83) $$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$$

(84) $$\mathbf{S} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T (x_i - \bar{x})$$

C is the standard definition of the sample covariance matrix. **S** is the multidimensional equivalent of the third central moment and as such can be called the sample third central vector moment. Combining all of these makes the gradient of the variance end up being

(85) $$\nabla s^2_{GDK} = 4\left(2\mathbf{C}(\hat{c} - \bar{x}) - \mathbf{S}\right)$$

Setting this to zero, the solution for the center estimator becomes

(86) $$\hat{c} = \bar{x} + \tfrac{1}{2}\mathbf{C}^{-1}\mathbf{S}$$

If this estimator is then plugged back into the variance, it becomes evident that this estimator is the absolute minimum of the variance. Choose an estimator c' other than $\hat{c}$ and the difference of the variances becomes

(87) $$s^2(c') - s^2(\hat{c}) = (c' - \hat{c})^T \mathbf{C}(c' - \hat{c}) \geq 0$$

Applying this formula produced statistically similar results to the MLE iterations in the Monte-Carlo experiment. The following graph shows that, similar to the MLE, it has error

(88) $$\left|\hat{c} - c_0\right| = O(\sigma)$$

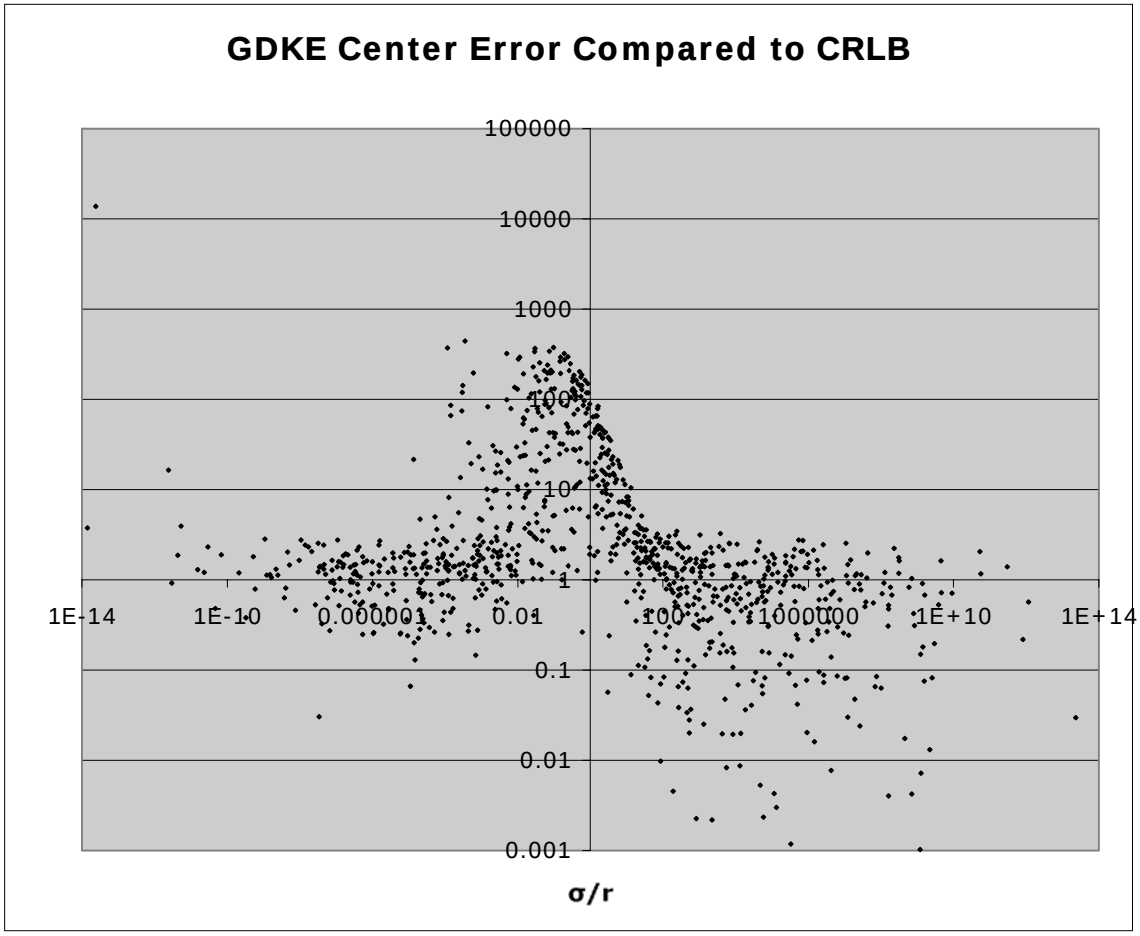

Figure 16 GDKE Compared to CRLB

The floating-point operations count for the GDKE of the center of a hypersphere of dimension *D* is

(89) $$FLOPs = N\left(D^2 + 6D - 1\right) + \left(\tfrac{1}{3}D^3 + 3D^2 + \tfrac{8}{3}D + 1\right)$$

where *N* is the number of samples. The spherical solution (D=3) becomes

(90) $$FLOPs = N26 + 45$$

You might ask at this point - it's faster, produces similar results, what's wrong with it? This fast method has not been fully accepted because it is a biased estimator. An estimator of a parameter is considered biased if it is expected to be a little off of the real

answer. Zelniker [119] has shown that the bias of the GDKE is on the order of the measurement standard deviation. The following statistical analysis will show exactly what the bias is and that there are cases that the bias is quite significant and does not disappear when more samples are taken.

4.1.5.2 Statistical Behavior

To understand completely the statistical behavior of the estimator, one must multiply multivariate Normal dependent variables six times to find the standard deviation of the center estimator. Another obstacle to finding the behavior is the inverse of a matrix. All these make it difficult to find an exact answer to the mean and standard deviation. A limiting approximation can be achieved if the individual components are analyzed.

Let's start off with our samples. Let us assume that each measurement x_i is a D-dimensional multivariate random Normal with constant covariance and differing means that can be expressed as

$$x_i \sim N_D(\mu_i, \Sigma) \tag{91}$$

where all of the means lie on the D-dimensional hypersphere.

$$|\mu_i - c_0| = r_0 \tag{92}$$

If the average of the samples is taken, then the new number is still a multivariate Normal

$$\bar{x} \sim N_D(\bar{\mu}, \tfrac{1}{N}\Sigma) \tag{93}$$

Section 4.1.5.3 proves the following property of the sample covariance matrix:

$$E(\mathbf{C}) = \mathbf{C}_0 + \Sigma \tag{94}$$

(95) $$Cov\left(\mathbf{C},\mathbf{C}^T\right)=\frac{1}{(N-1)}\left(\Sigma\left(\mathbf{C}_0+Tr\left(\mathbf{C}_0\right)\right)+\left(\mathbf{C}_0+\Sigma\right)\left(\Sigma+Tr\left(\Sigma\right)\right)\right)$$

where

(96) $$\mathbf{C}_0=\frac{1}{N-1}\sum_{i=1}^{N}\left(\mu_i-\overline{\mu}\right)\left(\mu_i-\overline{\mu}\right)^T.$$

The expectation for **C** is here shown to be a biased estimator of the true covariance $\mathbf{C_0}$. Most studies around this matrix are trying to estimate the measurement covariance Σ and say that the sample covariance matrix is unbiased, but that is only in the case when all samples have the same mean, thereby setting $\mathbf{C}_0$=0.

Similar analysis (cf. Section 4.1.5.4) will show that the expectation of the sample third central moment is

(97) $$E\left(\mathbf{S}\right)=\mathbf{S}_0$$

where

(98) $$\mathbf{S}_0=\frac{1}{N-1}\sum_{i=1}^{N}\left(\mu_i-\overline{\mu}\right)\left(\mu_i-\overline{\mu}\right)^T\left(\mu_i-\overline{\mu}\right)$$

Now with the properties of the sample mean, sample covariance C and sample third central moment S, the following property holds for the GDKE center estimator. The inverse of the sample covariance can be expanded by Leontief inverse as

(99) $$\mathbf{C}^{-1}=\left(\mathbf{C}_0+\Sigma\right)^{-1}+\sum_{k=1}^{\infty}\left(\mathbf{I}-\left(\mathbf{C}_0+\Sigma\right)^{-1}\mathbf{C}\right)^k\left(\mathbf{C}_0+\Sigma\right)^{-1}$$

Substituting this into the equation of the center produces

(100) $$\hat{c}=\overline{x}+\frac{1}{2}\left(\mathbf{C}_0+\Sigma\right)^{-1}\mathbf{S}+\sum_{k=1}^{\infty}\left(\mathbf{I}-\left(\mathbf{C}_0+\Sigma\right)^{-1}\mathbf{C}\right)^k\left(\mathbf{C}_0+\Sigma\right)^{-1}\mathbf{S}$$

This approximation results in the expectation of the center estimation as

$$E(\hat{c}) = c_0 + \tfrac{1}{2}\left(\left(\mathbf{C}_0 + \Sigma\right)^{-1} - \mathbf{C}_0^{-1}\right)\mathbf{S}_0 + \cdots \tag{101}$$

The covariance of the center estimator can be expanded to be

$$\begin{aligned} Cov\left(\hat{c},\hat{c}^T\right) = {} & \tfrac{1}{N}\Sigma + \tfrac{1}{2}\left(\mathbf{C}_0 + \Sigma\right)^{-1} Cov\left(\mathbf{S},\bar{x}^T\right) + \tfrac{1}{2} Cov\left(\bar{x},\mathbf{S}^T\right)\left(\mathbf{C}_0 + \Sigma\right)^{-1} \\ & + \tfrac{1}{4}\left(\mathbf{C}_0 + \Sigma\right)^{-1} Cov\left(\mathbf{S},\mathbf{S}^T\right)\left(\mathbf{C}_0 + \Sigma\right)^{-1} + \cdots \end{aligned} \tag{102}$$

The intermediate covariances are found to be independent (cf. Chapter 4.1.5.4)

$$Cov\left(\mathbf{S},\bar{x}^T\right) = Cov\left(\bar{x},\mathbf{S}^T\right) = 0 \tag{103}$$

These reduce the covariance to

$$Cov\left(\hat{c},\hat{c}^T\right) = \tfrac{1}{N}\Sigma + \tfrac{1}{4}\left(\mathbf{C}_0 + \Sigma\right)^{-1} Cov\left(\mathbf{S},\mathbf{S}^T\right)\left(\mathbf{C}_0 + \Sigma\right)^{-1} + \cdots \tag{104}$$

The covariance of the sample third central moment is

$$Cov\left(\mathbf{S},\mathbf{S}^T\right) = \tfrac{1}{N}\sigma^2\left(8\mathbf{F}_0 + Tr\left(\mathbf{F}_0\right) - 4\mathbf{C}_0^2 - 4Tr\left(\mathbf{C}_0\right)\mathbf{C}_0 - Tr\left(\mathbf{C}_0\right)^2\right) + O\left(\tfrac{1}{N}\sigma^4\right) \tag{105}$$

which makes the covariance of the center estimator

$$\begin{aligned} Cov\left(\hat{c},\hat{c}^T\right) = {} & \tfrac{1}{N}\sigma^2\left(\mathbf{I} - \left(\mathbf{C}_0 + \sigma^2\right)^{-1}\mathbf{C}_0^2\left(\mathbf{C}_0 + \sigma^2\right)^{-1}\right) \\ & + \tfrac{1}{4N}\sigma^2\left(\mathbf{C}_0 + \sigma^2\right)^{-1}\left(8\mathbf{F}_0 + Tr\left(\mathbf{F}_0\right) - 4Tr\left(\mathbf{C}_0\right)\mathbf{C}_0 - Tr\left(\mathbf{C}_0\right)^2\right)\left(\mathbf{C}_0 + \sigma^2\right)^{-1} \\ & + O\left(\tfrac{1}{N}\sigma^4\right) \end{aligned} \tag{106}$$

The radius estimator is similarly biased. Starting from the square of the radius estimator

$$\hat{r}^2 = \tfrac{N-1}{N} Tr\left(\mathbf{C}\right) + \left(\hat{c} - \bar{x}\right)^T\left(\hat{c} - \bar{x}\right) \tag{107}$$

The expectation of the square is then

(108) $$E\left(\hat{r}^2\right)=\frac{N-1}{N}Tr\left(\mathbf{C}_0+\Sigma\right)+Tr\left(Cov\left(\hat{c},\hat{c}^T\right)+E\left(\hat{c}\right)E\left(\hat{c}^T\right)\right)-2E\left(\bar{x}^T\hat{c}\right)+\bar{\mu}^T\bar{\mu}+\frac{1}{N}Tr\left(\Sigma\right)$$

The intermediate expectation, which is related to the covariance between the mean and the center estimator, is

(109) $$E\left(\hat{c}\bar{x}^T\right)=E\left(\bar{x}\bar{x}^T\right)+\frac{1}{2}E\left(\mathbf{C}^{-1}\mathbf{S}\bar{x}\right)$$

(110) $$E\left(\hat{c}\bar{x}^T\right)=c_0\bar{\mu}^T+\frac{1}{N}\Sigma+\frac{1}{2}\left(\left(\mathbf{C}_0+\Sigma\right)^{-1}-\mathbf{C}_0^{-1}\right)\mathbf{S}_0\bar{\mu}+\cdots$$

Inserting this into the expectation of the square, reduces it to

(111) $$E\left(\hat{r}^2\right)=r_0^2+\frac{N-1}{N}Tr\left(\Sigma\right)+\frac{1}{4}\mathbf{S}_0^T\left(\left(\mathbf{C}_0+\Sigma\right)^{-1}+\mathbf{C}_0^{-1}\right)\left(\left(\mathbf{C}_0+\Sigma\right)^{-1}-\mathbf{C}_0^{-1}\right)\mathbf{S}_0+\cdots$$

This expectation shows the bias in the radius estimator that remains non-zero as long as there is error in the measurement system. The variance of the square of the radius estimator comes out to be

(112) $$Var\left(\hat{r}^2\right)=O\left(\frac{1}{N}\Sigma\right)$$

A typical example of the (mis-)use of these estimators can be displayed using Mathematica as in Figure 17.

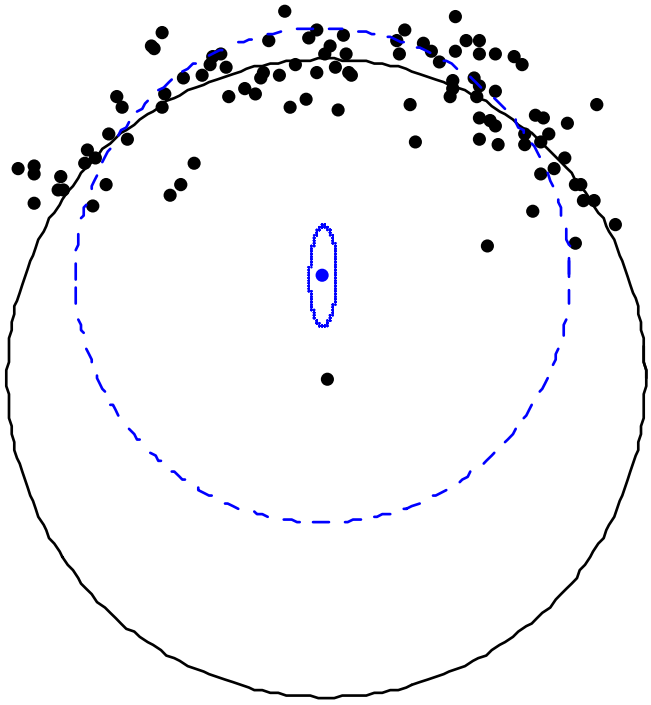

Figure 17 GDKE Error Ellipse

The dashed bars in Figure 17 represent the error involved in the center estimate. As can be seen, the true center is not anywhere within the error ellipsoid of the estimate. This is due solely to the bias in the estimator. Figure 17 is a good example of why the GDKE has not been adopted as much as the others. There is a simple way to remove this non-diminishing bias and it is the method of choice for speed and accuracy as long as a bit of a-priori knowledge is available (cf. Section 5.1).

4.1.5.3 Expectation of Sample Covariance

Theorem:

The sample covariance matrix has biased expectation when the samples are independently measured from the multivariate normal distribution with the same measurement covariance but differing expectations.

Proof:

The sample covariance is defined as

$$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)\left(x_i - \bar{x}\right)^T \tag{113}$$

where the samples are independent of each other and come from the multivariate normal distribution

$$x_i \sim N_D\left(\mu_i, \Sigma\right) \tag{114}$$

The sample covariance can be expanded to

$$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i x_i^T - \bar{x}x_i^T - x_i\bar{x}^T + \bar{x}\bar{x}^T\right) \tag{115}$$

When the summation is expanded, the covariance becomes

$$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N} x_i x_i^T - \frac{N}{N-1}\bar{x}\bar{x}^T \tag{116}$$

and ultimately to

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T - \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{j\neq i} x_i x_j^T\,. \tag{117}$$

The expectation of the sample covariance is

(118) $$E(\mathbf{C}) = \frac{1}{N}\sum_{i=1}^{N} E\left(x_i x_i^T\right) - \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{j\neq i} E\left(x_i x_j^T\right)$$

Since these samples are independent, the expectation breaks down to

(119) $$E(\mathbf{C}) = \frac{1}{N}\sum_{i=1}^{N} E\left(x_i x_i^T\right) - \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{j\neq i} E(x_i)E\left(x_j\right)^T$$

The expectation of the same two multivariate normal vectors multiplied together is

(120) $$E\left(x_i x_i^T\right) = \mu_i \mu_i^T + \Sigma$$

The expectation of the sample covariance then becomes

(121) $$E(\mathbf{C}) = \mathbf{C}_0 + \Sigma$$

where

(122) $$\mathbf{C}_0 = \frac{1}{N-1}\sum_{i=1}^{N}(\mu_i - \bar{\mu})(\mu_i - \bar{\mu})^T .$$

The sample covariance is thus biased whether it is an estimate of Σ or $\mathbf{C_0}$. The GDKE is using the matrix as an estimate of $\mathbf{C_0}$ with Σ interfering. The variance of $\mathbf{C}$ is more complicated involving six multivariate normal vectors multiplied together in a non-independent manner. A solution was achieved by analyzing the individual elements of the matrix. The covariance of each element in the matrix is defined as

(123) $$Cov(\mathbf{C}_{ij}, \mathbf{C}_{mn}) = E(\mathbf{C}_{ij}\mathbf{C}_{mn}) - E(\mathbf{C}_{ij})E(\mathbf{C}_{mn})$$

The square of the expectation of each element of the matrix is then

(124) $$E(\mathbf{C}_{ij})E(\mathbf{C}_{mn}) = \mathbf{C}_{0ij}\mathbf{C}_{0mn} + \mathbf{C}_{0ij}\Sigma_{mn} + \mathbf{C}_{0mn}\Sigma_{ij} + \Sigma_{ij}\Sigma_{mn}$$

where

$$\mathbf{C}_{0ij} = \frac{1}{N-1}\sum_{k=1}^{N}\left(\mu_{ki} - \overline{\mu}_i\right)\left(\mu_{kj} - \overline{\mu}_j\right) = \frac{1}{N-1}\sum_{k=1}^{N}\mu_{ki}\mu_{kj} - \frac{N}{N-1}\overline{\mu}_i\overline{\mu}_j \tag{125}$$

The expectation of the square of each element is much more complicated involving combinations of four random variates multiplied to each other. The square starts as

$$\begin{aligned}\mathbf{C}_{ij}\mathbf{C}_{mn} = &\frac{1}{(N-1)^2}\sum_{k=1}^{N}\sum_{l=1}^{N} x_{ki}x_{kj}x_{lm}x_{nl} \\ &- \frac{N}{(N-1)^2}\sum_{k=1}^{N} x_{ki}x_{kj}\overline{x}_m\overline{x}_n + x_{km}x_{kn}\overline{x}_i\overline{x}_j \\ &+ \frac{N^2}{(N-1)^2}\overline{x}_i\overline{x}_j\overline{x}_m\overline{x}_n\end{aligned} \tag{126}$$

Since the points are independent of each other, the expectation of this is

$$\begin{aligned}E\left(\mathbf{C}_{ij}\mathbf{C}_{mn}\right) = &\frac{1}{(N-1)^2}\sum_{k=1}^{N}\sum_{l=1}^{N} E\left(x_{ki}x_{kj}\right)E\left(x_{lm}x_{nl}\right) \\ &+ \frac{1}{(N-1)^2}\sum_{k=1}^{N} E\left(x_{ki}x_{kj}x_{km}x_{nk}\right) - E\left(x_{ki}x_{kj}\right)E\left(x_{km}x_{nk}\right) \\ &- \frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{N} E\left(x_{ki}x_{kj}\right)E\left(x_{lm}\right)E\left(x_{pn}\right) + E\left(x_{km}x_{kn}\right)E\left(x_{li}\right)E\left(x_{pj}\right) \\ &- \frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l\neq k}^{N} E\left(x_{ki}x_{kj}x_{km}\right)E\left(x_{\ln}\right) + E\left(x_{km}x_{kn}x_{ki}\right)E\left(x_{lj}\right) \\ &\quad - E\left(x_{ki}x_{kj}\right)E\left(x_{km}\right)E\left(x_{\ln}\right) - E\left(x_{km}x_{kn}\right)E\left(x_{ki}\right)E\left(x_{lj}\right) \\ &- \frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l\neq k}^{N} E\left(x_{ki}x_{kj}x_{kn}\right)E\left(x_{lm}\right) + E\left(x_{km}x_{kn}x_{kj}\right)E\left(x_{li}\right) \\ &\quad - E\left(x_{ki}x_{kj}\right)E\left(x_{lm}\right)E\left(x_{kn}\right) - E\left(x_{km}x_{kn}\right)E\left(x_{li}\right)E\left(x_{kj}\right) \\ &- \frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l\neq k}^{N} E\left(x_{ki}x_{kj}\right)E\left(x_{lm}x_{\ln}\right) + E\left(x_{km}x_{kn}\right)E\left(x_{li}x_{lj}\right) \\ &\quad - E\left(x_{ki}x_{kj}\right)E\left(x_{lm}\right)E\left(x_{\ln}\right) - E\left(x_{km}x_{kn}\right)E\left(x_{li}\right)E\left(x_{lj}\right) \\ &- \frac{1}{(N-1)^2 N}\sum_{k=1}^{N} 2E\left(x_{ki}x_{kj}x_{km}x_{kn}\right) - E\left(x_{ki}x_{kj}\right)E\left(x_{km}\right)E\left(x_{kn}\right) \\ &\quad - E\left(x_{km}x_{kn}\right)E\left(x_{ki}\right)E\left(x_{kj}\right) \\ &+ \frac{N^2}{(N-1)^2}E\left(\overline{x}_i\overline{x}_j\overline{x}_m\overline{x}_n\right)\end{aligned} \tag{127}$$

$$
\begin{aligned}
E\left(\mathbf{C}_{ij}\mathbf{C}_{mn}\right) &= \frac{1}{(N-1)^2}\sum_{k=1}^{N}\sum_{l=1}^{N}\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\left(\mu_{lm}\mu_{nl}+\Sigma_{mn}\right) \\
&+\frac{1}{(N-1)^2}\sum_{k=1}^{N}\begin{pmatrix}\mu_{ki}\mu_{kj}\mu_{km}\mu_{kn}+\Sigma_{ij}\Sigma_{mn}+\Sigma_{im}\Sigma_{nj}+\Sigma_{in}\Sigma_{jm}\\ +\Sigma_{ij}\mu_{km}\mu_{kn}+\Sigma_{im}\mu_{kj}\mu_{kn}+\Sigma_{in}\mu_{kj}\mu_{km}+\Sigma_{jm}\mu_{ki}\mu_{kn}\\ +\Sigma_{jn}\mu_{ki}\mu_{km}+\Sigma_{mn}\mu_{ki}\mu_{kj}-\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\left(\mu_{km}\mu_{kn}+\Sigma_{mn}\right)\end{pmatrix} \\
&-\frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{N}\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\mu_{lm}\mu_{pn}+\left(\mu_{km}\mu_{kn}+\Sigma_{mn}\right)\mu_{li}\mu_{pj} \\
&-\frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l\neq k}^{N}\begin{pmatrix}\left(\mu_{ki}\mu_{kj}\mu_{km}+\Sigma_{ij}\mu_{km}+\Sigma_{im}\mu_{kj}+\Sigma_{jm}\mu_{ki}\right)\mu_{nl}\\ +\left(\mu_{km}\mu_{kn}\mu_{ki}+\Sigma_{mn}\mu_{ki}+\Sigma_{mi}\mu_{kn}+\Sigma_{ni}\mu_{km}\right)\mu_{lj}\\ -\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\mu_{km}\mu_{\ln}-\left(\mu_{km}\mu_{kn}+\Sigma_{mn}\right)\mu_{ki}\mu_{lj}\end{pmatrix} \\
&-\frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l\neq k}^{N}\begin{pmatrix}\left(\mu_{ki}\mu_{kj}\mu_{kn}+\Sigma_{ij}\mu_{kn}+\Sigma_{in}\mu_{kj}+\Sigma_{jn}\mu_{ki}\right)\mu_{lm}\\ +\left(\mu_{km}\mu_{kn}\mu_{kj}+\Sigma_{mn}\mu_{kj}+\Sigma_{mj}\mu_{kn}+\Sigma_{nj}\mu_{km}\right)\mu_{li}\\ -\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\mu_{lm}\mu_{kn}-\left(\mu_{km}\mu_{kn}+\Sigma_{mn}\right)\mu_{li}\mu_{kj}\end{pmatrix} \\
&-\frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l\neq k}^{N}\begin{pmatrix}\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\left(\mu_{lm}\mu_{\ln}+\Sigma_{mn}\right)\\ +\left(\mu_{km}\mu_{kn}+\Sigma_{mn}\right)\left(\mu_{li}\mu_{lj}+\Sigma_{ij}\right)\\ -\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\mu_{lm}\mu_{\ln}-\left(\mu_{km}\mu_{kn}+\Sigma_{mn}\right)\mu_{li}\mu_{lj}\end{pmatrix} \\
&-\frac{1}{(N-1)^2 N}\sum_{k=1}^{N}\begin{pmatrix}2\begin{pmatrix}\mu_{ki}\mu_{kj}\mu_{km}\mu_{kn}+\Sigma_{ij}\Sigma_{mn}+\Sigma_{im}\Sigma_{nj}+\Sigma_{in}\Sigma_{jm}\\ +\Sigma_{ij}\mu_{km}\mu_{kn}+\Sigma_{im}\mu_{kj}\mu_{kn}+\Sigma_{in}\mu_{kj}\mu_{km}+\Sigma_{jm}\mu_{ki}\mu_{kn}\\ +\Sigma_{jn}\mu_{ki}\mu_{km}+\Sigma_{mn}\mu_{ki}\mu_{kj}\end{pmatrix}\\ -\left(\mu_{ki}\mu_{kj}+\Sigma_{ij}\right)\mu_{km}\mu_{kn}-\left(\mu_{km}\mu_{kn}+\Sigma_{mn}\right)\mu_{ki}\mu_{kj}\end{pmatrix} \\
&+\frac{N^2}{(N-1)^2}\begin{pmatrix}\overline{\mu}_i\overline{\mu}_j\overline{\mu}_m\overline{\mu}_n+\frac{1}{N^2}\Sigma_{ij}\Sigma_{mn}+\frac{1}{N^2}\Sigma_{im}\Sigma_{nj}+\frac{1}{N^2}\Sigma_{in}\Sigma_{jm}\\ +\frac{1}{N}\Sigma_{ij}\overline{\mu}_m\overline{\mu}_n+\frac{1}{N}\Sigma_{im}\overline{\mu}_j\overline{\mu}_n+\frac{1}{N}\Sigma_{in}\overline{\mu}_j\overline{\mu}_m+\frac{1}{N}\Sigma_{jm}\overline{\mu}_i\overline{\mu}_n\\ +\frac{1}{N}\Sigma_{jn}\overline{\mu}_i\overline{\mu}_m+\frac{1}{N}\Sigma_{mn}\overline{\mu}_i\overline{\mu}_j\end{pmatrix}
\end{aligned}
\tag{128}
$$

Expanding the sums and canceling terms reduces this gigantic equation to

$$
\begin{aligned}
E\left(\mathbf{C}_{ij}\mathbf{C}_{mn}\right) = {} & \tfrac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l=1}^{N}\left(N\mu_{ki}\mu_{kj}\mu_{lm}\mu_{nl} + (N-1)\Sigma_{ij}\mu_{km}\mu_{nk} + (N-1)\Sigma_{mn}\mu_{ki}\mu_{kj}\right) \\
& + \tfrac{1}{(N-1)^2 N}\sum_{k=1}^{N}\begin{pmatrix}(N-2)\Sigma_{im}\Sigma_{nj} + (N-2)\Sigma_{in}\Sigma_{jm} + N(N-2)\Sigma_{ij}\Sigma_{mn} \\ +N\Sigma_{im}\mu_{kj}\mu_{kn} + N\Sigma_{in}\mu_{kj}\mu_{km} + N\Sigma_{jm}\mu_{ki}\mu_{kn} + N\Sigma_{jn}\mu_{ki}\mu_{km}\end{pmatrix} \\
& - \tfrac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{N}\left(\mu_{ki}\mu_{kj}\mu_{lm}\mu_{pn} + \mu_{km}\mu_{kn}\mu_{li}\mu_{pj}\right) \\
& - \tfrac{1}{(N-1)^2 N}\sum_{k=1}^{N}\sum_{l=1}^{N}\begin{pmatrix}2\Sigma_{jm}\mu_{ki}\mu_{nl} + 2\Sigma_{mi}\mu_{kn}\mu_{lj} + 2\Sigma_{ni}\mu_{lm}\mu_{kj} + 2\Sigma_{jn}\mu_{ki}\mu_{lm} \\ +N\Sigma_{ij}\mu_{lm}\mu_{kn} + N\Sigma_{mn}\mu_{li}\mu_{kj}\end{pmatrix} \\
& + \tfrac{N^2}{(N-1)^2}\begin{pmatrix}\overline{\mu}_i\overline{\mu}_j\overline{\mu}_m\overline{\mu}_n + \tfrac{1}{N^2}\Sigma_{ij}\Sigma_{mn} + \tfrac{1}{N^2}\Sigma_{im}\Sigma_{nj} + \tfrac{1}{N^2}\Sigma_{in}\Sigma_{jm} \\ + \tfrac{1}{N}\Sigma_{ij}\overline{\mu}_m\overline{\mu}_n + \tfrac{1}{N}\Sigma_{im}\overline{\mu}_j\overline{\mu}_n + \tfrac{1}{N}\Sigma_{in}\overline{\mu}_j\overline{\mu}_m + \tfrac{1}{N}\Sigma_{jm}\overline{\mu}_i\overline{\mu}_n \\ + \tfrac{1}{N}\Sigma_{jn}\overline{\mu}_i\overline{\mu}_m + \tfrac{1}{N}\Sigma_{mn}\overline{\mu}_i\overline{\mu}_j\end{pmatrix}
\end{aligned} \tag{129}
$$

And finally, more cancellations produce the answer

$$
\begin{aligned}
E\left(\mathbf{C}_{ij}\mathbf{C}_{mn}\right) = {} & \mathbf{C}_{0ij}\mathbf{C}_{0mn} + \mathbf{C}_{0ij}\Sigma_{mn} + \Sigma_{ij}\mathbf{C}_{0mn} + \Sigma_{ij}\Sigma_{mn} \\
& + \tfrac{1}{(N-1)}\left(\Sigma_{im}\Sigma_{jn} + \Sigma_{in}\Sigma_{jm} + \Sigma_{im}\mathbf{C}_{0jn} + \Sigma_{in}\mathbf{C}_{0jm} + \Sigma_{jm}\mathbf{C}_{0in} + \mathbf{C}_{0im}\Sigma_{jn}\right)
\end{aligned} \tag{130}
$$

Subtracting the square of the expectation of each produces the covariance:

$$
Cov\left(\mathbf{C}_{ij},\mathbf{C}_{mn}\right) = \tfrac{1}{(N-1)}\left(\Sigma_{im}\Sigma_{nj} + \Sigma_{in}\Sigma_{jm} + \Sigma_{im}\mathbf{C}_{0jn} + \Sigma_{in}\mathbf{C}_{0jm} + \Sigma_{jm}\mathbf{C}_{0in} + \Sigma_{jn}\mathbf{C}_{0im}\right) \tag{131}
$$

The full matrix covariance then becomes

$$
Cov\left(\mathbf{C},\mathbf{C}^T\right) = \tfrac{1}{(N-1)}\left(\Sigma\left(\mathbf{C}_0 + Tr\left(\mathbf{C}_0\right)\right) + \left(\mathbf{C}_0 + \Sigma\right)\left(\Sigma + Tr\left(\Sigma\right)\right)\right) \tag{132}
$$

If the measurement covariance is $\Sigma = \sigma^2\mathbf{I}$ then the variance reduces to

$$
Cov\left(\mathbf{C},\mathbf{C}^T\right) = \tfrac{1}{(N-1)}\sigma^2\left(\mathbf{C}_0\left(2+D\right) + Tr\left(\mathbf{C}_0\right) + \sigma^2\left(1+D\right)\right) \tag{133}
$$

$$
Tr\left(\mathbf{C}\right)^2 = \sum_{i=1}^{D}\sum_{j=1}^{D}\mathbf{C}_{ii}\mathbf{C}_{jj} \tag{134}
$$

$$
E\left(Tr\left(\mathbf{C}\right)^2\right) = \sum_{i=1}^{D}\sum_{j=1}^{D}\mathbf{C}_{0ii}\mathbf{C}_{0jj} + \mathbf{C}_{0ii}\Sigma_{jj} + \Sigma_{ii}\mathbf{C}_{0jj} + \Sigma_{ii}\Sigma_{jj} + \tfrac{1}{(N-1)}\left(2\Sigma_{ij}\Sigma_{ij} + 4\Sigma_{ij}\mathbf{C}_{0ij}\right) \tag{135}
$$

$$E\left(Tr(\mathbf{C})^2\right) = Tr(\mathbf{C}_0)^2 + 2Tr(\mathbf{C}_0)Tr(\Sigma) + Tr(\Sigma)^2 + \tfrac{1}{(N-1)}2Tr(\Sigma^2) + \tfrac{1}{(N-1)}4Tr(\mathbf{C}_0\Sigma) \tag{136}$$

The inverse of the sample covariance is an important value to find its expectation. The inverse of any matrix can be expanded by the Leontief Inverse to

$$A^{-1} = \sum_{k=0}^{\infty}\left(I - E(A)^{-1}A\right)^k E(A)^{-1} \tag{137}$$

as long as the spectral radius (largest absolute eigenvalue) is

$$\rho\left(\mathbf{I} - E\left(A^{-1}\right)A\right) < 1 \tag{138}$$

Substituting A for the sample covariance produces

$$\mathbf{C}^{-1} = (\mathbf{C}_0 + \Sigma)^{-1} + \sum_{k=1}^{\infty}\left(\mathbf{I} - (\mathbf{C}_0 + \Sigma)^{-1}\mathbf{C}\right)^k (\mathbf{C}_0 + \Sigma)^{-1} \tag{139}$$

An intermediate expectation is needed to get the second order approximation

$$\begin{aligned} E(\mathbf{C}A\mathbf{C}) &= \mathbf{C}_0 A\mathbf{C}_0 + \mathbf{C}_0 A\Sigma + \Sigma A\mathbf{C}_0 + \Sigma A\Sigma \\ &+ \tfrac{1}{(N-1)}\begin{pmatrix} \Sigma A\Sigma + \Sigma Tr(A\Sigma) + \Sigma A\mathbf{C}_0 + \mathbf{C}_0 A\Sigma \\ + \Sigma Tr(A\mathbf{C}_0) + Tr(\Sigma A)\mathbf{C}_0 \end{pmatrix} \end{aligned} \tag{140}$$

Then, using this identity, the expectation of the inverse expands to

$$\begin{aligned} E\left(\mathbf{C}^{-1}\right) &= (\mathbf{C}_0 + \Sigma)^{-1} \\ &+ \tfrac{1}{(N-1)}(\mathbf{C}_0 + \Sigma)^{-1}\left(\Sigma(2+D) + Tr\left(\Sigma(\mathbf{C}_0 + \Sigma)^{-1}\right)\mathbf{C}_0 - \Sigma(\mathbf{C}_0 + \Sigma)^{-1}\Sigma\right)(\mathbf{C}_0 + \Sigma)^{-1} \\ &+ \sum_{k=3}^{\infty} E\left(\left(\mathbf{I} - (\mathbf{C}_0 + \Sigma)^{-1}\mathbf{C}\right)^k\right)(\mathbf{C}_0 + \Sigma)^{-1} \end{aligned} \tag{141}$$

So the second order approximation using the diagonal covariance is

$$E\left(\mathbf{C}^{-1}\right) = \mathbf{C}_0^{-1} + \tfrac{1}{N-1}\sigma^2\mathbf{C}_0^{-1}\left((2+D)\mathbf{C}_0^{-1} + Tr\left(\mathbf{C}_0^{-1}\right) - \mathbf{C}_0^{-2}\sigma^2\right) \tag{142}$$

4.1.5.4 Expectation of Sample Third Central Moment

Theorem:

The Sample Third Central Moment is unbiased when the samples are independently measured from the multivariate Normal distribution with the same covariance but differing expectations.

Proof:

The Sample Third Central Moment for multivariates is defined as

$$\mathbf{S} = \frac{1}{N-1}\sum_{k=1}^{N}(x_k - \bar{x})(x_k - \bar{x})^T(x_k - \bar{x}) \tag{143}$$

where the samples are independent of each other and come from the multivariate Normal distribution

$$x_i \sim N_D(\mu_i, \Sigma) \tag{144}$$

$$x_i - \bar{x} \sim N_D(\mu_i - \bar{\mu}, \tfrac{N-1}{N}\Sigma) \tag{145}$$

Then expectation of the moment is

$$E(\mathbf{S}) = \frac{1}{N-1}\sum_{k=1}^{N}E\left((x_k - \bar{x})(x_k - \bar{x})^T(x_k - \bar{x})\right) \tag{146}$$

Then using the expectation and covariance from (144) gives

$$E(\mathbf{S}_i) = \frac{1}{N-1}\sum_{k=1}^{N}\sum_{j=1}^{D}(\mu_{ki} - \bar{\mu}_i)(\mu_{kj} - \bar{\mu}_j)^2 + (\mu_{ki} - \bar{\mu}_i)\tfrac{N-1}{N}\Sigma_{jj} + 2\tfrac{N-1}{N}\Sigma_{ij}(\mu_{kj} - \bar{\mu}_j) \tag{147}$$

and the last two parts disappear because of the definition of the average, leaving behind

(148) $$E(\mathbf{S}) = \mathbf{S}_0 = \frac{1}{N-1}\sum_{i=1}^{N}(\mu_i - \overline{\mu})(\mu_i - \overline{\mu})^T(\mu_i - \overline{\mu})$$

As can be seen, the sample third central moment is an unbiased estimator in the general case of constant covariance but different mean.

Another important property is the covariance between the sample third central moment and the sample mean. This value starts with expanding the sum as

(149) $$\mathbf{S}\overline{x}^T = \frac{1}{N-1}\sum_{k=1}^{N} x_k x_k^T x_k \overline{x}^T - \frac{2}{N-1}\sum_{k=1}^{N} x_k x_k^T \overline{x}\overline{x}^T - \frac{1}{N-1}\sum_{k=1}^{N} \overline{x} x_k^T x_k \overline{x}^T + \frac{2N}{N-1}\overline{x}\overline{x}^T\overline{x}\overline{x}^T$$

And expanding the means produce

(150) $$\mathbf{S}\overline{x}^T = \frac{1}{(N-1)N}\sum_{j=1}^{N}\sum_{k=1}^{N} x_k x_k^T x_k x_j^T - \frac{1}{(N-1)N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\left(2x_k x_k^T x_i x_j^T + x_i x_k^T x_k x_j^T\right) + \frac{2N}{N-1}\overline{x}\overline{x}^T\overline{x}\overline{x}^T$$

Separating the independent values produce the expanded sums

$$
\begin{aligned}
E\left(\mathbf{S}\overline{x}^T\right) &= \tfrac{1}{(N-1)N}\sum_{j=1}^{N}\sum_{k=1}^{N} E\left(x_k x_k^T x_k\right)E\left(x_j^T\right) \\
&+ \tfrac{1}{(N-1)N}\sum_{k=1}^{N} E\left(x_k x_k^T x_k x_k^T\right) - E\left(x_k x_k^T x_k\right)E\left(x_k^T\right) \\
&- \tfrac{1}{(N-1)N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} 2E\left(x_k x_k^T\right)E\left(x_i\right)E\left(x_j^T\right) + E\left(x_i\right)E\left(x_k^T x_k\right)E\left(x_j^T\right) \\
&- \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N} 3E\left(x_k x_k^T x_k\right)E\left(x_j^T\right) \\
&+ \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N} 2E\left(x_k x_k^T\right)E\left(x_k\right)E\left(x_j^T\right) + E\left(x_k\right)E\left(x_k^T x_k\right)E\left(x_j^T\right) \\
&- \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N} 2E\left(x_k x_k^T \mu_j x_k^T\right) + E\left(x_j\right)E\left(x_k^T x_k x_k^T\right) \\
&+ \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N} 2E\left(x_k x_k^T\right)E\left(x_j\right)E\left(x_k^T\right) + E\left(x_j\right)E\left(x_k^T x_k\right)E\left(x_k^T\right) \\
&- \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N} 2E\left(x_k x_k^T\right)E\left(x_j x_j^T\right) + E\left(x_j x_j^T\right)E\left(x_k^T x_k\right) \\
&+ \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N} 2E\left(x_k x_k^T\right)E\left(x_j\right)E\left(x_j^T\right) + E\left(x_j\right)E\left(x_k^T x_k\right)E\left(x_j^T\right) \\
&- \tfrac{1}{(N-1)N^2}\sum_{k=1}^{N} 3E\left(x_k x_k^T x_k x_k^T\right) - 2E\left(x_k x_k^T\right)E\left(x_k\right)E\left(x_k^T\right) - E\left(x_k\right)E\left(x_k^T x_k\right)E\left(x_k^T\right) \\
&+ \tfrac{2N}{N-1}E\left(\overline{xx}^T\overline{xx}^T\right)
\end{aligned}
\tag{151}
$$

Substituting the expectations produce the sums

$$
\begin{aligned}
E\left(\mathbf{S}\bar{x}^T\right) = & \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k=1}^{N}\left(N\mu_k\mu_k^T\mu_k\mu_j^T + NTr(\Sigma)\mu_k\mu_j^T + 2N\Sigma\mu_k\mu_j^T\right) \\
& + \tfrac{1}{(N-1)N^2}\sum_{k=1}^{N}\left(2N\Sigma^2 + N\Sigma Tr(\Sigma) + 2N\mu_k\mu_k^T\Sigma + N\Sigma\mu_k^T\mu_k\right) \\
& - \tfrac{1}{(N-1)N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\begin{pmatrix}2\mu_k\mu_k^T\mu_i\mu_j^T + 2\Sigma\mu_i\mu_j^T \\ +\mu_i\mu_k^T\mu_k\mu_j^T + Tr(\Sigma)\mu_i\mu_j^T\end{pmatrix} \\
& - \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}\left(2Tr(\Sigma)\mu_k\mu_j^T + 4\Sigma\mu_k\mu_j^T\right) \\
& - \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}\left(2\Sigma\mu_k^T\mu_j + 2\mu_k\mu_j^T\Sigma + 2\mu_j\mu_k^T\Sigma\right) \\
& - \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}\begin{pmatrix}2\mu_k\mu_k^T\Sigma + 2\Sigma\Sigma + \mu_j\mu_j^T\mu_k^T\mu_k \\ +\Sigma\mu_k^T\mu_k + Tr(\Sigma)\Sigma\end{pmatrix} \\
& + \tfrac{1}{(N-1)N^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}\left(\mu_j\mu_k^T\mu_k\mu_j^T\right) \\
& - \tfrac{1}{(N-1)N^2}\sum_{k=1}^{N}\begin{pmatrix}6\Sigma^2 + 3\Sigma Tr(\Sigma) + 2Tr(\Sigma)\mu_k\mu_k^T + 6\mu_k\mu_k^T\Sigma \\ +4\Sigma\mu_k\mu_k^T + 3\Sigma\mu_k^T\mu_k\end{pmatrix} \\
& + \tfrac{2N}{N-1}\begin{pmatrix}\bar{\mu}\bar{\mu}^T\bar{\mu}\bar{\mu}^T + 2\tfrac{1}{N^2}\Sigma^2 + \tfrac{1}{N^2}\Sigma Tr(\Sigma) + \tfrac{1}{N}Tr(\Sigma)\bar{\mu}\bar{\mu}^T \\ +2\tfrac{1}{N}\bar{\mu}\bar{\mu}^T\Sigma + 2\tfrac{1}{N}\Sigma\bar{\mu}\bar{\mu}^T + \tfrac{1}{N}\Sigma\bar{\mu}^T\bar{\mu}\end{pmatrix}
\end{aligned}
\tag{152}
$$

This rather large sum simplifies all the way down to

$$E\left(\mathbf{S}\bar{x}^T\right) = \mathbf{S}_0\bar{\mu}^T \tag{153}$$

thus proving that the sample third central moment is independent of the sample mean. The covariance of the sample third central moment is much more difficult involving the multiplication of six multivariate normals together. To determine the covariance, the following matrix must be analyzed.

$$\mathbf{S}\mathbf{S}^T = \tfrac{1}{(N-1)^2}\sum_{j=1}^{N}\sum_{k=1}^{N}\left(x_j-\bar{x}\right)\left(x_j-\bar{x}\right)^T\left(x_j-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T \tag{154}$$

The expectation of this matrix is

$$E\left(\mathbf{S}\mathbf{S}^T\right)=\frac{1}{(N-1)^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}E\left(\left(x_j-\bar{x}\right)\left(x_j-\bar{x}\right)^T\left(x_j-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\right)$$
$$+\frac{1}{(N-1)^2}\sum_{k=1}^{N}E\left(\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\right) \tag{155}$$

Separating out the covariance produces

$$E\left(\mathbf{S}\mathbf{S}^T\right)=\frac{1}{(N-1)^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}E\left(\left(x_j-\bar{x}\right)\left(x_j-\bar{x}\right)^T\left(x_j-\bar{x}\right)\right)E\left(\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\right)$$
$$+\frac{1}{(N-1)^2}\sum_{k=1}^{N}E\left(\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\right) \tag{156}$$
$$+\frac{1}{(N-1)^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}Cov\left(\left(x_j-\bar{x}\right)\left(x_j-\bar{x}\right)^T\left(x_j-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\right)$$

These expectations are found in Section (7.1.1) and produces the covariance as

$$Cov\left(\mathbf{S},\mathbf{S}^T\right)=\frac{N-1}{N^2}\begin{pmatrix}\Sigma Tr\left(\mathbf{F}_0\right)+2\mathbf{F}_0\Sigma+2\Sigma\mathbf{F}_0\\+4\frac{N}{(N-1)^2}\sum_{k=1}^{N}\left(\mu_k-\bar{\mu}\right)\left(\mu_k-\bar{\mu}\right)^T\Sigma\left(\mu_k-\bar{\mu}\right)\left(\mu_k-\bar{\mu}\right)^T\end{pmatrix}$$
$$+\frac{N-1}{N^2}\begin{pmatrix}4\Sigma\mathbf{C}_0\Sigma+8\Sigma^2\mathbf{C}_0+8\mathbf{C}_0\Sigma^2+2\mathbf{C}_0\Sigma Tr\left(\Sigma\right)+2\Sigma Tr\left(\Sigma\right)\mathbf{C}_0\\+2Tr\left(\Sigma^2\right)\mathbf{C}_0+4\Sigma^2Tr\left(\mathbf{C}_0\right)+2\Sigma Tr\left(\Sigma\right)Tr\left(\mathbf{C}_0\right)\\+4\Sigma\frac{1}{N-1}\sum_{k=1}^{N}\left(\mu_k-\bar{\mu}\right)^T\Sigma\left(\mu_k-\bar{\mu}\right)\end{pmatrix} \tag{157}$$
$$+\frac{N-1}{N^2}\left(\Sigma Tr\left(\Sigma\right)^2+2\Sigma Tr\left(\Sigma^2\right)+8\Sigma^3+4\Sigma^2Tr\left(\Sigma\right)\right)$$
$$+\frac{1}{(N-1)^2}\sum_{j=1}^{N}\sum_{k\neq j}^{N}Cov\left(\left(x_j-\bar{x}\right)\left(x_j-\bar{x}\right)^T\left(x_j-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\right)$$

The j and k vectors are not quite independent with their covariance being

$$Cov\left(\left(x_j-\bar{x}\right)\left(x_k-\bar{x}\right)^T\right)=-\frac{1}{N}\Sigma \tag{158}$$

This means the larger covariance in Equation (157) can be approximated by

(159) $$Cov\left(\left(x_j-\bar{x}\right)\left(x_j-\bar{x}\right)^T\left(x_j-\bar{x}\right),\left(x_k-\bar{x}\right)\left(x_k-\bar{x}\right)^T\left(x_k-\bar{x}\right)\right)=O\left(\tfrac{1}{N}\Sigma\mathbf{C}_0^2\right)$$

Using this and the special case of diagonal covariance

(160) $$\Sigma=\sigma^2\mathbf{I}$$

produces the simplified equation for the covariance of the sample third vector moment as

(161) $$Cov\left(\mathbf{S},\mathbf{S}^T\right)=\tfrac{1}{N}\sigma^2\left(8\mathbf{F}_0+Tr\left(\mathbf{F}_0\right)-4\mathbf{C}_0^2-4Tr\left(\mathbf{C}_0\right)\mathbf{C}_0-Tr\left(\mathbf{C}_0\right)^2+38\sigma^2\mathbf{C}_0+14\sigma^2Tr\left(\mathbf{C}_0\right)\right)+O\left(\tfrac{1}{N}\sigma^6\right)$$

4.1.5.5 Center Equation Theorem

This section proves that the GDKE estimator of the center of the hypersphere is truly the center of the sphere when the points all lie on the sphere.

Theorem:

The Center Equation (86) calculates the true center of the hypersphere if all points are equidistant from the center, i.e. they all lie on the surface of the same hypersphere.

Proof:

The Center Equation can be rearranged to be

(162) $$\mathbf{0}=2\mathbf{C}\left(\bar{x}-c_0\right)+\mathbf{S}.$$

Expanding this into the summation produces

(163) $$\mathbf{0}=2\sum_{i=1}^{N}\left(x_i-\bar{x}\right)\left(x_i-\bar{x}\right)^T\left(\bar{x}-c_0\right)+\sum_{i=1}^{N}\left(x_i-\bar{x}\right)\left(x_i-\bar{x}\right)^T\left(x_i-\bar{x}\right)$$

Combining the two summations gives

(164) $$\mathbf{0} = \sum_{i=1}^{N} \left(x_i - \bar{x}\right)\left(x_i - \bar{x}\right)^T \left(x_i + \bar{x} - 2c_0\right)$$

The difference can be changed to be around the center instead of the mean thus

(165) $$\mathbf{0} = \sum_{i=1}^{N} \left(x_i - \bar{x}\right)\left(x_i - c_0 + c_0 - \bar{x}\right)^T \left(x_i - c_0 + \bar{x} - c_0\right)$$

Then expanding the second and third term under the summation produces

(166) $$\mathbf{0} = \sum_{i=1}^{N} \left(x_i - \bar{x}\right)\left(\left(x_i - c_0\right)^T \left(x_i - c_0\right) - \left(\bar{x} - c_0\right)^T \left(\bar{x} - c_0\right)\right)$$

The square of the distance of point *i* from the center is defined as

(167) $$r_i^2 \equiv \left(x_i - c_0\right)^T \left(x_i - c_0\right)$$

The summation can be then simplified to

(168) $$\mathbf{0} = \sum_{i=1}^{N} \left(x_i - \bar{x}\right) r_i^2 - \left(N\bar{x} - N\bar{x}\right)\left(\bar{x} - c_0\right)^T \left(\bar{x} - c_0\right)$$

The second term is then eliminated, producing the simple summation

(169) $$\mathbf{0} = \sum_{i=1}^{N} \left(x_i - \bar{x}\right) r_i^2$$

It is easy to see that if all points are equidistant (i.e. r_i=r) from the center then this equation becomes true. There are other ways to produce zero with this sum, but that is where the GDKE steps in. The GDKE solves this equation when the distances are not equal.

4.2 Skeleton Approaches

Producing a skeleton involves finding the centers of joint rotations, which was addressed in Section 4.1, the hierarchy of segmentation, and the orientation of each segment. This thesis does not rely on novel ideas for the remaining steps of producing a skeleton. The hierarchy of segmentation is known ahead of time. The orientation of a segment is determined by one of two methods. Either it is given in the raw data, e.g. magnetic trackers, or it is calculated by the fastest technique known. The fastest way to calculate the orientation of a segment from positional information of markers is quite intuitive and has been in use for a long time. One of the earliest uses found in motion capture were published in 2000 by Herda, et al. [50]. Unfortunately, many authors don't realize that this speedy method of orienting the segment can also be used to calculate the entire skeleton. The most common approach to calculating a skeleton from motion capture data is through minimization until the skeleton fits where the rotation points have been approximated. The minimization involves squishing segments and moving joints until all joints are nearest to the calculated rotation points. O'Brien, et al. [84] uses a linear least-squares minimization that produces the rotation points from a collection of time frames. Their method relies on the constraint that two connected segments have a common point between them during rotation. Their solution calculates a point from the knowledge of the orientation of both the parent and the child segments and calculates the best fit point that remains the most still relative to the two segments. The solution involves the Singular Value Decomposition of a $3N$x6 matrix to produce the common point that is the rotation point. If the solution fails to come up with an answer, as in the case of no motion or planar motion, then the closest point between the two coordinate systems is

used. This technique relies heavily on orientation information that is not always available. In their study, magnetic motion tracking devices are used which contain both position and orientation. This is akin to solving for the best-fit sphere around a center. The state of the science for sphere fitting was presented in Section 4.1 but the improvement upon the state is now presented in Section 5.1.

Chapter 5 PROPOSED SOLUTION

The proposed solution to provide a quick method to draw a skeleton from motion capture data is presented in this chapter. The main contribution to the state of the science is the thorough analysis of the previously fastest sphere-fitting technique and the subsequent improvement of the answer.

5.1 Unbiased Generalized Delogne-Kása Estimator

The estimator explained in this chapter is asymptotically unbiased. Asymptotically unbiased is defined as an inversely proportional relationship with the sample count:

(170) $$E(\hat{q}) = q_0 + O\left(\frac{1}{N}\right)$$

where q_0 is the parameter that the estimator is trying to estimate. This basically says that the estimator is expected to get closer to the true answer if more samples are taken. From the previous discussion in Section 4.1.5, it was shown that the GDKE estimators for center and radius do not satisfy this requirement. Our algorithm uses a simple substitution that turns the GDKE into one with a diminishing bias. It involves the use of an a-priori estimate of the measurement error in the samples. This is not that unreasonable since most systems of measurement have some kind of estimate to the measurement error. Since the sample covariance matrix **C** is the only biased term in the equation for the GDKE center, this is what will be altered.

5.1.1 Derivation

The derivation stems from a simple substitution. The substitution

(171) $$\mathbf{C}' = \mathbf{C} - \hat{\Sigma}$$

has the convenient property of

(172) $$E(\mathbf{C}') = \mathbf{C}_0 + \Sigma - \hat{\Sigma}$$

If the true measurement covariance is the same as the estimated measurement covariance then

(173) $$E(\mathbf{C}') = \mathbf{C}_0$$

The covariance of the new sample covariance matrix is the same as the old one since we are just adding a constant to the variable.

(174) $$Cov(\mathbf{C}'_{ij}, \mathbf{C}'_{mn}) = Cov(\mathbf{C}_{ij}, \mathbf{C}_{mn})$$

if the error estimate is the same as the truth error covariance.

Inserting Equation (171) will produce the following equations for the solution of a hyper-sphere

(175) $$\hat{c}' = \bar{x} + \tfrac{1}{2}\mathbf{C}'^{-1}\mathbf{S}$$

The radius estimator can be similarly compensated for its bias by

(176) $$\hat{r}' = \sqrt{\tfrac{N-1}{N} Tr(\mathbf{C}') + (\bar{x} - \hat{c}')^T (\bar{x} - \hat{c}')}$$

where *Tr*(*) means the trace of the matrix.

So where does this estimate of the measurement covariance come from? An un-biased estimator for the trace of the measurement covariance is presented below which relies on known information about the sphere being measured.

(177) $$Tr(\hat{\Sigma}) = \tfrac{1}{N}\sum_{i=1}^{N}(x_i - c_0)^T (x_i - c_0) - r_0^2$$

During calibration (i.e. finding the measurement error), a known sphere with radius and center can be measured, providing the estimate of the trace of the measurement covariance.

5.1.2 Statistical Properties

Now it is desirable to find the statistical properties of the new estimators and compare them to the old GDKE. If the estimate is equal to the true sample covariance then these new equations have the following statistical properties. The altered covariance is expanded by the Leontief inverse as

$$\mathbf{C}'^{-1} = \mathbf{C}_0^{-1} + \sum_{k=1}^{\infty} \left(\mathbf{I} - \mathbf{C}_0^{-1}\mathbf{C}'\right)^k \mathbf{C}_0^{-1} \tag{178}$$

Using this expansion, the altered center estimator is

$$\hat{c}' = \bar{x} + \tfrac{1}{2}\mathbf{C}_0^{-1}\mathbf{S} + \tfrac{1}{2}\sum_{k=1}^{\infty} \left(\mathbf{I} - \mathbf{C}_0^{-1}\mathbf{C}'\right)^k \mathbf{C}_0^{-1}\mathbf{S} \tag{179}$$

The expectation of the altered center estimator is

$$E\left(\hat{c}'\right) = \bar{\mu} + \tfrac{1}{2}\mathbf{C}_0^{-1}\mathbf{S}_0 + \tfrac{1}{2}\sum_{k=1}^{\infty} E\left(\left(\mathbf{I} - \mathbf{C}_0^{-1}\mathbf{C}'\right)^k \mathbf{C}_0^{-1}\mathbf{S}\right) \tag{180}$$

Using the same analysis for the GDKE in the previous section, the new estimator has an expectation of

$$E\left(\hat{c}'\right) = c_0 - \tfrac{1}{2}\mathbf{C}_0^{-1}\Delta\mathbf{S} + \tfrac{1}{2}\sum_{k=2}^{\infty} \left(-\mathbf{C}_0^{-1}\Delta\mathbf{C}\right)^k \mathbf{C}_0^{-1}\left(\mathbf{S}_0 + \Delta\mathbf{S}\right) \tag{181}$$

Then, the bias of the new estimator is exposed as

$$E\left(\hat{c}'\right) = c_0 + O\left(\tfrac{1}{\sqrt{N}}\sigma\mathbf{C}_0^{-1}\sqrt{\mathbf{F}_0}\right) \tag{182}$$

The covariance of the new center estimator, with similar analysis, is

$$Cov\left(\hat{c}',\hat{c}'^{T}\right)=\tfrac{1}{N}\sigma^{2}\mathbf{C}_{0}^{-1}\left(8\mathbf{F}_{0}+Tr\left(\mathbf{F}_{0}\right)-4Tr\left(\mathbf{C}_{0}\right)\mathbf{C}_{0}-Tr\left(\mathbf{C}_{0}\right)^{2}\right)\mathbf{C}_{0}^{-1}+O\left(\tfrac{\sigma^{4}}{N}\right) \tag{183}$$

The expectation of the square of the new radius estimator is

$$E\left(\hat{r}'^{2}\right)=r_{0}^{2}+O\left(\tfrac{1}{\sqrt{N}}\sigma\overline{\mu}^{T}\mathbf{C}_{0}^{-1}\sqrt{\mathbf{F}_{0}}\right) \tag{184}$$

The variance of the radius estimator comes to

$$Var\left(\hat{r}'^{2}\right)=O\left(\tfrac{1}{N}\Sigma\right) \tag{185}$$

The expectation of the measurement covariance estimator is simply

$$E\left(Tr\left(\hat{\Sigma}\right)\right)=Tr\left(\Sigma\right) \tag{186}$$

The variance of the measurement covariance estimator is found from the expectation of the square. Starting from the square

$$\begin{aligned}Tr\left(\hat{\Sigma}\right)^{2}=&\left(r_{0}^{2}-c_{0}^{T}c_{0}\right)^{2}+4\left(r_{0}^{2}-c_{0}^{T}c_{0}\right)c_{0}^{T}\overline{x}+4c_{0}^{T}\overline{x}\overline{x}^{T}c_{0}-4c_{0}^{T}\tfrac{1}{N^{2}}\sum_{i=1}^{N}\sum_{j=1}^{N}x_{j}x_{i}^{T}x_{i}\\&+2\left(c_{0}^{T}c_{0}-r_{0}^{2}\right)\tfrac{1}{N}\sum_{i=1}^{N}x_{i}^{T}x_{i}+\tfrac{1}{N^{2}}\sum_{i=1}^{N}\sum_{j=1}^{N}x_{i}^{T}x_{i}x_{j}^{T}x_{j}\end{aligned} \tag{187}$$

The expectation of this square is

$$\begin{aligned}E\left(Tr\left(\hat{\Sigma}\right)^{2}\right)=&\left(r_{0}^{2}-c_{0}^{T}c_{0}-Tr\left(\Sigma\right)+4c_{0}^{T}\overline{\mu}-2\tfrac{1}{N}\sum_{i=1}^{N}\mu_{i}^{T}\mu_{i}\right)\left(r_{0}^{2}-c_{0}^{T}c_{0}-Tr\left(\Sigma\right)\right)\\&+4c_{0}^{T}\overline{\mu}\left(\overline{\mu}^{T}c_{0}-\tfrac{1}{N}\sum_{i=1}^{N}\mu_{i}^{T}\mu_{i}\right)+4\tfrac{1}{N}c_{0}^{T}\Sigma c_{0}-8\tfrac{1}{N}c_{0}^{T}\Sigma\overline{\mu}\\&+\left(\tfrac{1}{N}\sum_{i=1}^{N}\mu_{i}^{T}\mu_{i}\right)^{2}\\&+2\tfrac{1}{N}Tr\left(\Sigma^{2}\right)+4\tfrac{1}{N^{2}}\sum_{i=1}^{N}\mu_{i}^{T}\Sigma\mu_{i}\end{aligned} \tag{188}$$

This expectation turns the variance into

(189) $$Var\left(Tr\left(\hat{\Sigma}\right)\right)=2\frac{1}{N}\left(Tr\left(\Sigma^2\right)+2\frac{1}{N}\sum_{i=1}^{N}\left(\mu_i-c_0\right)^T\Sigma\left(\mu_i-c_0\right)\right)$$

If the measurement covariance is expected to be diagonal ($\Sigma=\sigma^2\mathbf{I}$) then the variance of the variance estimator is

(190) $$Var\left(\hat{\sigma}^2\right)=2\frac{1}{ND^2}\sigma^2\left(D\sigma^2+2r_0^2\right)$$

where *D* is the dimension of the measurement. A typical use of these new estimators can be displayed using Mathematica, with exactly the same data as was displayed for the GDKE in Figure 17.

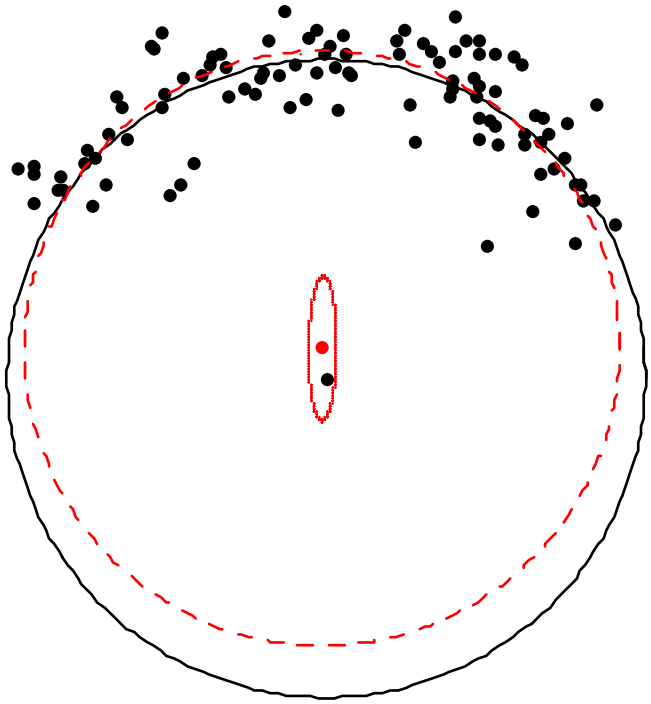

Figure 18 UGDK Error Ellipse

As can be seen in Figure 18, the true center is within the error ellipsoid. This data contains 100 points generated with a diagonal measurement covariance with all diagonals

equal to 0.05^2. The Leontief condition ($\rho < 1$) is satisfied with the spectral radius in question equal to 0.557238.

These equations show that there still is a bias, but it is asymptotically unbiased. Figure 19 shows a Monte-Carlo run that explicitly shows the $1/\sqrt{N}$ dependency. The error in the estimate is compared with how many points were analyzed for a particular joint in some motion capture data.

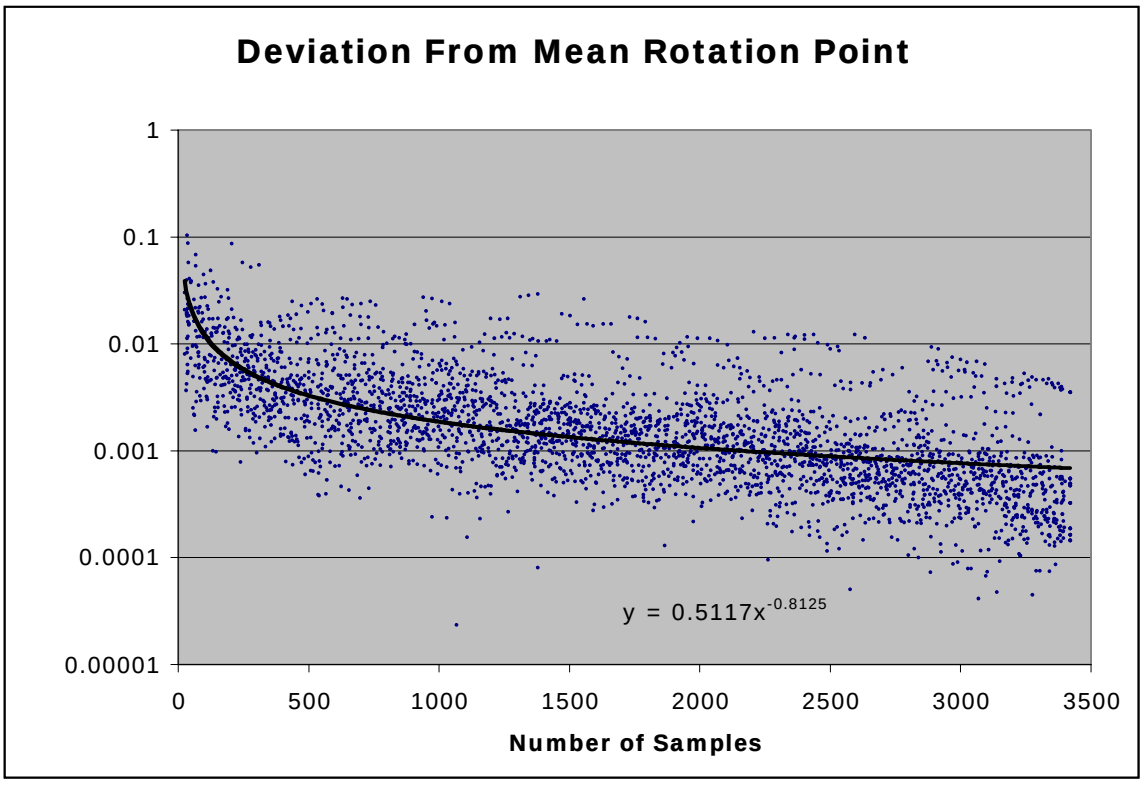

Figure 19 Sample Size Dependency of Deviation

An example analysis using MLE, UGDK and GDKE is presented in Figure 20. The figure clearly shows the improvement over the GDKE. The figure shows the bias is removed using the UGDK.

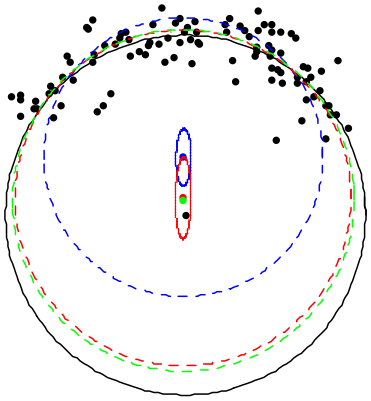

Figure 20 One hundred samples comparison of MLE (green), UGDK (red), GDKE (blue)

There is a case where this new method fails to bring an improvement. The case is not very common and should not be of concern. The following graph shows the problem.

Error!

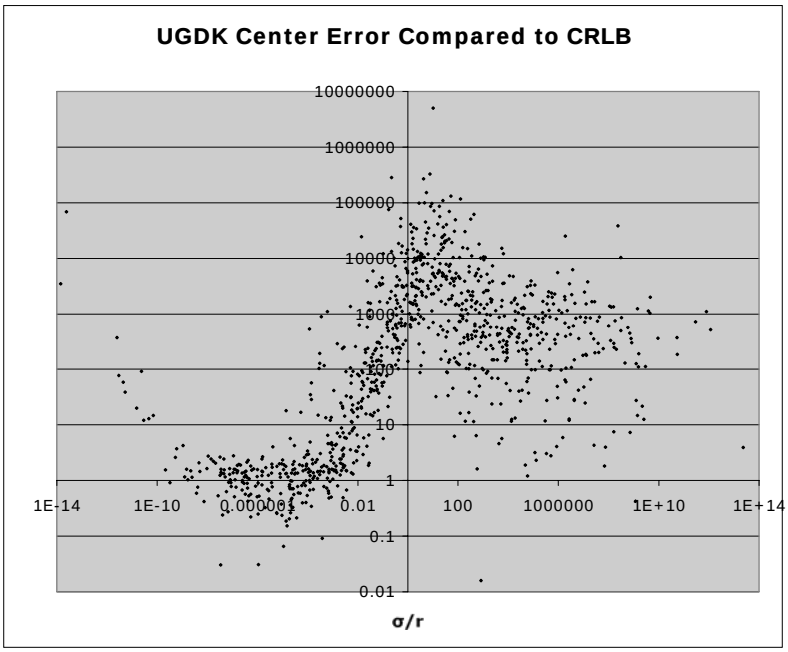

Figure 21 UGDK Compared to CRLB

There is no concern here since all of the methods have troubles in this area. This situation is a bit impractical, as no one wants a system whose measurement error is actually bigger than the item being measured?

That brings up the point - when should a measurement system be trusted? A cursory glance would say

(191) $$\sigma < r_0$$

is when the system should be accepted. Upon further analysis, the answer comes from the world of economics [101] - the Leontief inverse.

(192) $$\left(\mathbf{I}-\mathbf{A}\right)^{-1} = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 + \mathsf{L}$$

This series expansion is convergent when the spectral radius (i.e. largest absolute eigenvalue) is

(193) $$\rho\left(\mathbf{A}\right) < 1$$

The matrix of interest in our application is the sample covariance matrix **C**. The matrix can be expanded two ways. The first way is of practical importance and that is when

(194) $$\rho\left(\mathbf{C}_0^{-1}\Sigma\right) < 1$$

which allows us to expand the inverse of the expectation of the covariance matrix into

(195) $$\left(\mathbf{C}_0 + \Sigma\right)^{-1} = \mathbf{C}_0^{-1} - \mathbf{C}_0^{-1}\Sigma\mathbf{C}_0^{-1} + \left(\mathbf{C}_0^{-1}\Sigma\right)^2\mathbf{C}_0^{-1} - \cdots$$

If the spectral radius is greater than one, then the expansion turns into

(196) $$\left(\mathbf{C}_0 + \Sigma\right)^{-1} = \Sigma^{-1} - \Sigma^{-1}\mathbf{C}_0\Sigma^{-1} + \left(\Sigma^{-1}\mathbf{C}_0\right)^2\Sigma^{-1} - \mathsf{L}$$

So what is this magical turning point? If the measurement covariance Σ is diagonal with equal variances in all directions (i.e. $\Sigma=\sigma^2 I$) then the spectral radius can be evaluated as

(197) $$\rho\left(\mathbf{C}_0^{-1}\Sigma\right)=\frac{\sigma^2}{\lambda_{\min}}$$

The where $\lambda_{\min}$ is the smallest eigenvalue of the sample covariance matrix $\mathbf{C}_0$. This is still not very useful because eigenvalues are mathematically intensive to solve. Let us set up a scenario to produce an answer for the limiting case of too many data points. We want to develop an equation for the eigenvalues when the hypersphere is partially covered by sample points.

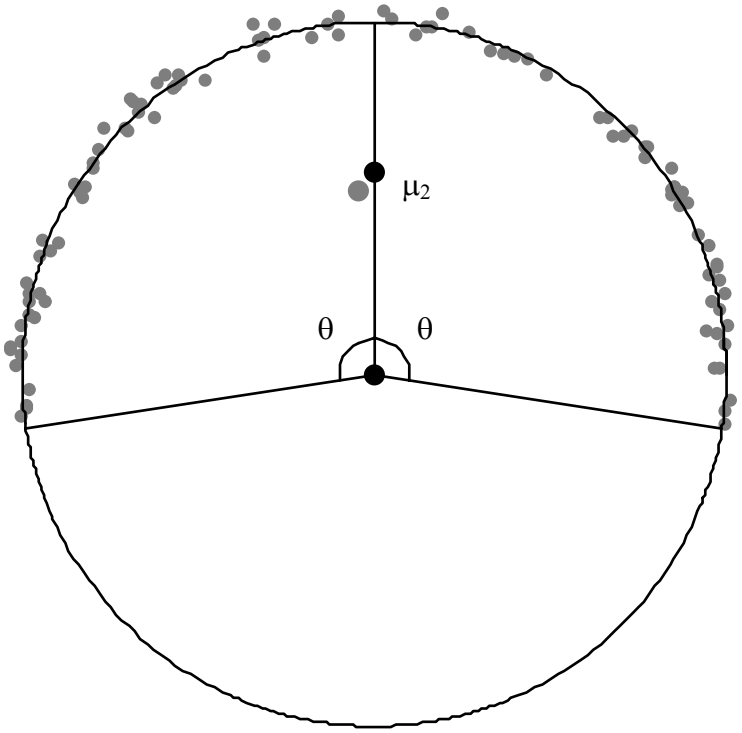

Figure 22 Circle with Constrained Data

First is the case for a two-dimensional hypersphere (i.e. circle). The points are confined to an angular distance θ from the y-axis. Each point on the circle can be expressed as

(198) $$v_2 = \begin{pmatrix} r_0 \sin a \\ r_0 \cos a \end{pmatrix}$$

where *a* is the angle from the y-axis. The mean has an asymptote as the number of samples increase.

(199) $$\mu_2 = \lim_{N\to\infty} \tfrac{1}{N} \sum_{i=1}^{N} v_2 = \tfrac{1}{\phi_2} \int_{-\theta}^{\theta} v_2 da$$

where

(200) $$\phi_2 = \int_{-\theta}^{\theta} da = 2\theta$$

This gives us the answer of the asymptotic average being on the y-axis at

(201) $$\mu_2 = r_0 \frac{\sin\theta}{\theta} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Similarly, the asymptotic covariance can be found as the integral

(202) $$\mathbf{C}_2 = \tfrac{1}{\phi_2} \int_{-\theta}^{\theta} (v_2 - \mu_2)(v_2 - \mu_2)^T da = \begin{pmatrix} \lambda_{\max 2} & 0 \\ 0 & \lambda_{\min 2} \end{pmatrix}$$

This matrix happens to be diagonal containing all of the eigenvalues which turn out to be

(203) $$\lambda_{\min 2} = r_0^2 \frac{\theta(2\theta + \sin(2\theta)) - 4\sin^2\theta}{4\theta^2} \text{ and}$$

(204) $$\lambda_{\max 2} = r_0^2 \frac{\theta - \cos\theta \sin\theta}{2\theta}$$

The three-dimensional hypersphere (i.e. sphere) is similarly evaluated only this time, the solid angle must be integrated. The points are confined to an angular distance θ from the z-axis. The point on the sphere is

(205) $$v_3 = \begin{pmatrix} r_0 \sin a \cos b \\ r_0 \sin a \sin b \\ r_0 \cos a \end{pmatrix}$$

The asymptotic limit of the average point is then

(206) $$\mu_3 = \tfrac{1}{\phi_3} \int_0^{\theta} \int_0^{2\pi} v_3 \sin a \, db \, da$$

where

(207) $$\phi_3 = \int_0^{\theta} \int_0^{2\pi} \sin a \, db \, da = 2\pi(1 - \cos\theta)$$

The asymptotic limit of the average lies on the z-axis at

(208) $$\mu_3 = r_0 \cos^2\left(\frac{\theta}{2}\right) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Similarly, the asymptotic covariance is evaluated as

(209) $$\mathbf{C}_3 = \tfrac{1}{\phi_3} \int_0^{\theta} \int_0^{2\pi} (v_3 - \mu_3)(v_3 - \mu_3)^T \sin a \, db \, da = \begin{pmatrix} \lambda_{\max 3} & 0 & 0 \\ 0 & \lambda_{\max 3} & 0 \\ 0 & 0 & \lambda_{\min 3} \end{pmatrix}$$

The diagonals of this matrix have two equal eigenvalues (the largest) and one smallest value.

(210) $$\lambda_{\min 3} = r_0^2 \tfrac{1}{3} \sin^4\left(\frac{\theta}{2}\right) \text{ and}$$

(211) $$\lambda_{\max 3} = r_0^2 \tfrac{1}{3}(2+\cos\theta)\sin^2\left(\frac{\theta}{2}\right)$$

Following this answer, a good measurement condition for the sphere is when

(212) $$\sigma < r_0 \frac{1}{\sqrt{3}}\sin^2\left(\frac{\theta}{2}\right)$$

whereas the best condition to get a good answer for a circle is when

(213) $$\sigma < r_0 \frac{\sqrt{\theta(2\theta+\sin(2\theta))-4\sin^2\theta}}{2\theta}$$

5.2 Cylindrical Joint Solution

Special considerations are needed when a cylindrical joint is suspected. In this case, a point on the upper segment will always draw a circular arc thereby remaining planar. Any method for finding a sphere where only a circle exists will fail. The planar condition can be discovered when the sample covariance matrix **C** becomes near singular:

(214) $$|\mathbf{C}| \approx 0$$

This occurs when the rank of **C** is less than the number of dimensions. In essence, one dimension must be removed from the sphere equation to get the circle estimation. The sample covariance matrix can be rewritten using its eigensystem values.

(215) $$\mathbf{C} = \sum_{i=1}^{D} \lambda_i v_i v_i^T$$

where λ_i are the eigenvalues and v_i are the corresponding unit eigenvectors. The inverse of the matrix can be similarly expanded with the same eigensystem as

(216) $$\mathbf{C}^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} v_i v_i^T$$

The GDKE solution to the circle in 3D space can be calculated two ways. The first way does not involve inverting small values. This is achieved by removing the eigenvalues that are below a certain limit. This reduced matrix can then be used in the GDKE formula producing

(217) $$\hat{c} = \bar{x} + \tfrac{1}{2} \sum_{\lambda_i > \varepsilon} \frac{1}{\lambda_i} v_i v_i^T \mathbf{S}$$

(218) $$\hat{r} = \sqrt{\sum_{\lambda_i > \varepsilon} \frac{1}{4\lambda_i^2} \left(v_i^T \mathbf{S} \right)^2 + \tfrac{N-1}{N} \lambda_i}$$

This new center is the best-fit estimate of a circle for the data. This equation is successful even if the sample covariance is exactly singular. This formula can be used to calculate the center and radius of curvature for curvilinear paths in any dimensional space. For the rotation point of a joint, it has a flaw. A cylindrical joint rotates about an axis. The center that is calculated lies on this axis but might not be the volumetric center of the cylinder. To overcome this, two markers on either side of the center can be averaged.

5.3 Multiple Marker Solution

For multiple markers going around the same center of rotation, another formula can be achieved by the same analysis of least squares. This technique is good to use when more than one marker is available. It has the ability to average out errors when one marker is too close to the rotation point or has other systematic problems. The gradient of the sum of variances (cf. Equation (85)) is

(219) $$\nabla s_m^2 = \sum_{p=1}^{M} 4\left(2\mathbf{C}_p\left(\hat{c}_m - \bar{x}_p\right) - \mathbf{S}_p\right)$$

where M is the number of markers. Setting this to zero will provide a solution for multiple markers:

(220) $$\hat{c}_m = \left(\sum_{p=1}^{M} \mathbf{C}_p\right)^{-1} \sum_{p=1}^{M} \mathbf{C}_p \bar{x}_p + \tfrac{1}{2}\mathbf{S}_p$$

and the individual radius becomes

(221) $$\hat{r}_p = \sqrt{\tfrac{N-1}{N} Tr\left(\mathbf{C}_p\right) + \left(\bar{x}_p - \hat{c}_m\right)^T \left(\bar{x}_p - \hat{c}_m\right)}$$

where M is the number of markers and the subscript p indicates values that utilize the single marker's positions.

The same unbiased analysis applies to this multiple marker version and results in

(222) $$\hat{c}'_m = \left(\sum_{p=1}^{M} \mathbf{C}'_p\right)^{-1} \sum_{p=1}^{M} \mathbf{C}'_p \bar{x}_p + \tfrac{1}{2}\mathbf{S}_p \text{ and}$$

(223) $$\hat{r}'_p = \sqrt{\tfrac{N-1}{N} Tr\left(\mathbf{C}'_p\right) + \left(\bar{x}_p - \hat{c}'_m\right)^T \left(\bar{x}_p - \hat{c}'_m\right)}.$$

The matrix that is to be inversed here is still a positive-definite matrix since positive-definite matrices added together still produce a positive-definite matrix. This allows for the speedier Cholesky decomposition just like before. The singular values can be excluded just like in Chapter 5.2. An example of the MGDK method is presented in Figure 23.

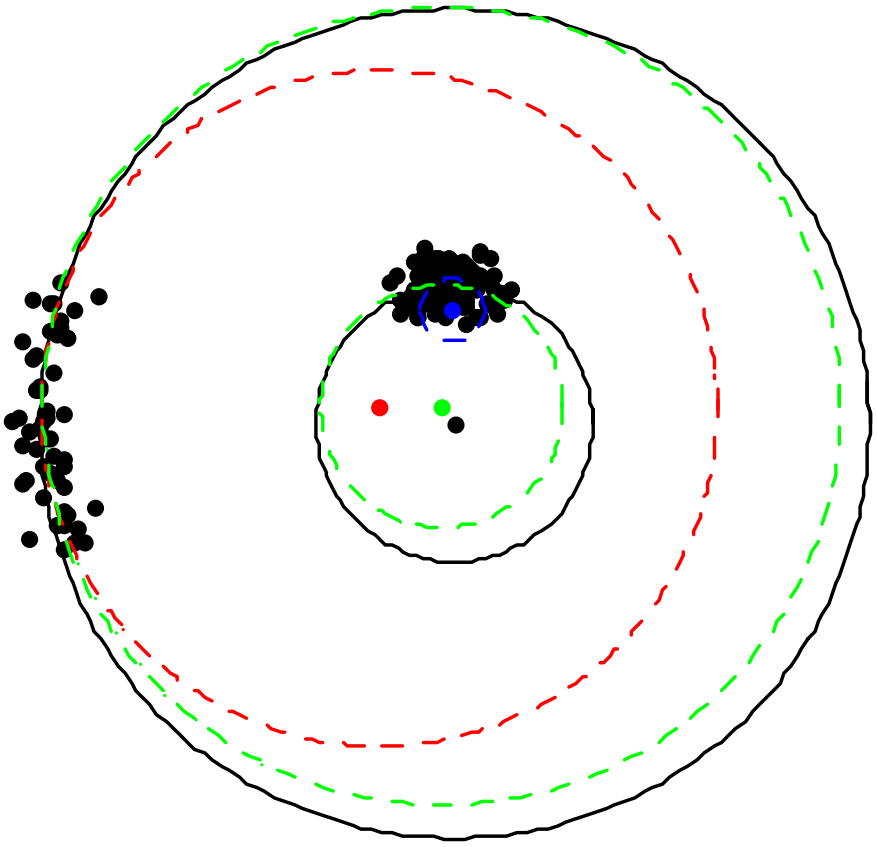

Figure 23 MGDK example

This example shows what happens when the individual circles are compared to that when combined in the MGDK. The outer circle solution is drawn in red; the inner circle solution is drawn in blue, and the MGDK solution is drawn in green. The example shows a dramatic improvement over both of the individual circle calculations.

5.4 Incrementally Improved Solution

A more refined answer can be achieved when using a incremental improvement formula. The idea here is a group of samples are collected and an answer is retrieved

from the GDKE or UGDK formulae. Then a new sample is added, refining the previous answer for the center. Starting off simple, the mean has a recursion of

$$\bar{x}_{n+1} = \bar{x}_n + \frac{x_{n+1} - \bar{x}_n}{n+1},\ \bar{x}_1 = x_1 \tag{224}$$

One definition will make the following equations smaller:

$$\delta_n \equiv x_n - \bar{x}_n \tag{225}$$

$$\delta_{n+1} = n\left(\bar{x}_{n+1} - \bar{x}_n\right) \tag{226}$$

The sample covariance matrix has a recurrence relationship of

$$\mathbf{C}_{n+1} = \frac{n-1}{n}\mathbf{C}_n + \frac{n+1}{n^2}\delta_{n+1}\delta_{n+1}^T,\ \mathbf{C}_1 = \mathbf{0} \tag{227}$$

The unbiased improvement of the sample covariance matrix is

$$\mathbf{C}'_{n+1} = \frac{n-1}{n}\mathbf{C}'_n - \frac{1}{n}\hat{\Sigma} + \frac{n+1}{n^2}\delta_{n+1}\delta_{n+1}^T,\ \mathbf{C}'_1 = \mathbf{0} \tag{228}$$

The inverse of the covariance matrix also has a recurrence - without doing an additional matrix inverse

$$\mathbf{C}_{n+1}^{-1} = \frac{n}{n-1}\mathbf{C}_n^{-1}\left(\mathbf{I} + \frac{\delta_{n+1}\delta_{n+1}^T\mathbf{C}_n^{-1}}{\frac{n(n-1)}{n+1} + Tr\left(\delta_{n+1}\delta_{n+1}^T\mathbf{C}_n^{-1}\right)}\right) \tag{229}$$

Putting these together produces a recurrence relationship for the center of the hypersphere

$$\mathbf{S}_{n+1} = \tfrac{n-1}{n}\mathbf{S}_n + \tfrac{1}{n}\left(-2\mathbf{C}_{n+1} - Tr\left(\mathbf{C}_{n+1}\right) + \tfrac{(n+1)(n+2)}{n^2}\delta_{n+1}\delta_{n+1}^T\right)\delta_{n+1} \tag{230}$$

$$\hat{c}_{n+1} = \bar{x}_{n+1} + \tfrac{1}{2}\mathbf{C}_{n+1}^{-1}\mathbf{S}_{n+1} \tag{231}$$

$$\hat{r}_{n+1}^2 = \tfrac{n}{n+1}Tr\left(\mathbf{C}_{n+1}\right) + \left(\bar{x}_{n+1} - \hat{c}_{n+1}\right)^T\left(\bar{x}_{n+1} - \hat{c}_{n+1}\right) \tag{232}$$

The cost of the incremental improvement to produce the new center and radius is measured in the FLOP count:

(233) $$FLOPs = 17\tfrac{1}{2}D^2 + 23\tfrac{1}{2}D + 12$$

For a sphere, the FLOPs are 123 for every new point. When compared to the FLOPs for the GDKE method, this incremental approach is about four times slower. This makes the incremental approach a last resort when a few extra points need to be added to a previously calculated center and radius. This new estimator has the distinct advantage of constant memory requirements no matter how many points are analyzed.

5.5 Hierarchical Skeleton Solution

5.5.1 Arbitrary Figure

An arbitrary figure can be drawn at each time frame if certain criteria are met for the data at that time frame. An arbitrary figure is made up of linked segments stored in a single parent-multiple children tree structure. The drawing of the figure consists of connecting the rotation points from parent to children. Each line drawn would then be fixed inside the segment. The process would continue from the root segment to the leaf segments. The leaf segments would have to just draw lines to the existing data points on that segment.

Starting with the original data, each assigned to their corresponding segment ($x_{s\,i}$), the point must be transformed to the coordinate system fixed to the parent of the segment s.

$$y_{si} = \mathbf{M}_{s-1}^{T}\left(x_{si} - p_{s-1}\right) \tag{234}$$

These new vectors ($y_{s\,i}$) are what are passed to the spherical center estimator formula. This formula remains valid and allows the process to continue to the children only if both p and $\mathbf{M}$ can be determined for the parent. This amounts to being able to construct three axes and a point that are fixed on a segment. There are three cases that are acceptable and are the criteria for being able to draw the entire skeleton at a particular time frame.

5.5.1.1 Case 1 – Three or more points

This is the easiest case to determine the segment's coordinate system. First, pick three of the available points, ($x_{s\,1}$, $x_{s\,2}$, $x_{s\,3}$). One of the points must be chosen as the center of the coordinate system

(235) $$p_s = x_{s1}$$

The three coordinate axes must then be constructed in a mutually perpendicular fashion (right-handed coordinate system). Two points can determine the first direction:

(236) $$\hat{x} = \frac{x_{s2} - x_{s1}}{\left| x_{s2} - x_{s1} \right|}$$

A second direction can be determined from the third point and the first direction:

(237) $$\hat{z} = \frac{x_{s3} \times \hat{x}}{\left| x_{s3} \times \hat{x} \right|}$$

The third direction can be determined from the previous two directions:

(238) $$\hat{y} = \hat{z} \times \hat{x}$$

The floating point operations involved in calculating one coordinate system is

(239) $$FLOP_3 = 39$$

Drawing a skeleton using solely this case of markers involves calculating the centers of rotation for all non-leaf segments in the hierarchical figure.

(240) $$FLOPs = N_S 39 + \left(N_S - N_L\right) 18$$

where N_S is the number of segments and N_L is the number of leaf segments. A typical human figure with fourteen segments and five leaf segments produces

(241) $$FLOPs = 708$$

5.5.1.2 Case 2 – Two points

This relies on some previously calculated information to determine the coordinate system. Three points are needed and some point must be divined that is fixed on the segment. Luckily, the common rotation point between the segment and its parent has been previously calculated because of the tree traversal. The three available points are now ($x_{s\,1}$, $x_{s\,2}$, c_s). One of these points must be chosen as the center of the coordinate system

(242) $$p_s = x_{s1}$$

The center of rotation for this segment is determined from the parent's coordinate system with

(243) $$c_s = p_{s-1} + \mathbf{M}_{s-1}\tilde{c}_s$$

where $\tilde{c}_s$ is a constant vector in the parent's coordinate system pointing at the child's rotation point. The three coordinate axes must then be constructed in a mutually perpendicular fashion (right-handed coordinate system). Two points can determine the first direction:

(244) $$\hat{x} = \frac{x_{s2} - x_{s1}}{\left| x_{s2} - x_{s1} \right|}$$

A second direction can be determined from the third point and the first direction:

(245) $$\hat{z} = \frac{c_s \times \hat{x}}{\left| c_s \times \hat{x} \right|}$$

The third direction can be determined from the previous two directions:

(246) $$\hat{y} = \hat{z} \times \hat{x}$$

The floating point operations involved in calculating one coordinate system is

(247) $$FLOP_2 = 57 + FLOP_P$$

where $FLOP_P$ is the FLOPs needed to calculate the coordinate matrix M of the parent. Drawing a skeleton using solely this case of markers involves calculating the centers of rotation for all non-leaf segments in the hierarchical figure.

(248) $$FLOPs = N_S 39 + (N_S - N_L + N_2)18$$

where N_2 is the number of segments with two data points with a parent with three.

5.5.1.3 Case 3 – One Point

This is the easiest case to determine the segment's coordinate system. First, pick three of the available points, ($x_{s\,1}$, c_s). One of the points must be chosen as the center of the coordinate system

(249) $$p_s = x_{s1}$$

The three coordinate axes must then be constructed in a mutually perpendicular fashion (right-handed coordinate system). Two points can determine the first direction:

(250) $$\hat{x} = \frac{c_s - x_{s1}}{\left| c_s - x_{s1} \right|}$$

(251) $$c_s = p_{s-1} + \mathbf{M}_{s-1}\tilde{c}_s$$

A second direction can be determined from the null vector previously calculated:

(252) $$\hat{z} = v_s$$

(253) $$v_s = \mathbf{M}_{s-1}\tilde{v}_s$$

The third direction can be determined from the previous two directions:

(254) $$\hat{y} = \hat{z} \times \hat{x}$$

The rotation matrix $\mathbf{M}_s$ can now be constructed by placing the three coordinate axes as columns in the matrix.

(255) $$\mathbf{M}_s = \begin{pmatrix} \hat{x}_x & \hat{y}_x & \hat{z}_x \\ \hat{x}_y & \hat{y}_y & \hat{z}_y \\ \hat{x}_z & \hat{y}_z & \hat{z}_z \end{pmatrix}$$

These three cases will allow the reconstruction of an entire skeleton as long as the root segment has at least three markers (Case 1). The recursive nature of the other cases precludes them from the root.

The floating point operations involved in calculating one coordinate system is

(256) $$FLOP_1 = 54 + FLOP_P$$

where $FLOP_P$ is the FLOPs needed to calculate the coordinate matrix M of the parent. Drawing a skeleton using solely this case of markers involves calculating the centers of rotation for all non-leaf segments in the hierarchical figure.

(257) $$FLOPs = N_S 39 + (N_S - N_L + N_1)18$$

where N_1 is the number of segments with one data points with a parent with three.

5.5.2 Predefined Marker Association

During motion capture, markers are placed over the body and tracked by one of several methods available. The animator of the tracked data either has to have previous knowledge of the marker's associated segment or come up with an algorithm to do the association. For those with no a-priori knowledge, the algorithm can get time consuming. Some authors [60] use this method by classifying markers together that don't move relative to each other. The grouping methods work when there is more than one marker per segment and the hierarchy of segments is well defined in the data. These methods are very slow though.

In order to figure out which markers are associated with which segments, the names of the markers within the data were analyzed for normal naming conventions. The association was then hand written to a text file that is read after the data is read. Naming conventions are fairly straightforward for most sets of data. Examples of normal naming conventions and their segments can be found in Table 1.

Table 1 Marker Associations

Marker Name	Associated Segment	Location on Segment
RFHD	Head	anterior right top
LFHD	Head	anterior left top
RSHO	Chest	mid right top
RFTShould	Chest	anterior right top

RRRShould	Chest	posterior right top
LSHO	Chest	mid left top
LFTShould	Chest	anterior left top
LRRShould	Chest	posterior left top
CLAV	Chest	anterior mid top
RUPA	Upper Right Arm	mid right
LUPA	Upper Left Arm	mid left
RELB	Upper Right Arm	mid right bottom
LELB	Upper Left Arm	mid left bottom
RARM	Lower Right Arm	mid right
LARM	Lower Left Arm	mid left
RWRB	Lower Right Arm	posterior right top
RWRA	Lower Right Arm	anterior right bottom
LWRB	Lower Left Arm	posterior left top
LWRA	Lower Left Arm	anterior left bottom
RFIN	Right Hand	mid right anterior

LFIN	Left Hand	mid left posterior
STRN	Chest	anterior mid bottom
RTHI	Upper Right Leg	mid right top
RGTR	Upper Right Leg	mid right top
LTHI	Upper Left Leg	mid left
LGTR	Upper Left Leg	mid left top
RKNE	Upper Right Leg	mid right bottom
LKNE	Upper Left Leg	mid left bottom
RLEG	Lower Right Leg	mid right top
RTIB	Lower Right Leg	mid right
LLEG	Lower Left Leg	mid left
LTIB	Lower Left Leg	mid left
RANK	Lower Right Leg	mid right bottom
LANK	Lower Left Leg	mid left bottom
RMT5	Right Foot	anterior right top
LMT5	Left Foot	anterior left top

RTOE	Right Foot	anterior left top
LTOE	Left Foot	anterior right top
RHEE	Right Foot	posterior mid top
LHEE	Left Foot	posterior mid top
RBHD	Head	posterior mid right
LBHD	Head	posterior mid left
C7	Chest	posterior mid top
R10	Chest	posterior right top
RBAC	Chest	posterior right top
T10	Chest	posterior mid
RFWT	Hips	anterior right top
LFWT	Hips	anterior left top
RBWT	Hips	posterior mid top
LBWT	Hips	posterior mid top
RPelvis	Hips	posterior right mid

This is a fairly complete list of abbreviations but there are many operators and systems that use either more extension collections of markers or don't even follow the naming convention. A particular association for a dataset can be initially guessed and then with trial-and-error, the association can be improved. This thesis has created association files for each dataset that was analyzed. The files are simple texts files that relate the name given to the marker in the dataset and the "standard" name as given in Table 1. The format is explained in Chapter 7.3.1.

Chapter 6 RESULTS

6.1 Case Study of CMU Data 60-08

The CMU Graphics Lab produced a one minute long motion capture data-set of a salsa dance in 60-08. The data file contains 3421 time slices for 41 markers on two figures. This case study will concentrate on analyzing the performance of the UGDK in determining the rotation points in the female subject. Four passes on the data will collect rotation point calculations, each pass randomly removing from 0 to 99% of the time frames in increments of 1%. 400 calculated rotation points were collected for each segment modeled. The calculated constants are the relative rotation points as referenced in each segment's parent's coordinate system. The 400 calculations were averaged and the standard deviations were calculated as well. These values are presented in the tables below.

Table 2 Table of Means of Rotation Points

Rotation Point	Mean x (m)	Mean y (m)	Mean z (m)

Waist	0.13324824	-0.063898286	0.130439024
Neck	-0.077279042	-0.003811468	-0.010261576
Left Ankle	0.437510805	-0.032639212	0.040255145
Left Wrist	0.110981565	-0.075779216	0.014702334
Left Elbow	0.283393628	-0.060979142	0.012893845
Left Knee	-0.244076283	-0.07952933	0.009066338
Right Elbow	0.253621623	-0.146993545	-0.001828167
Right Knee	0.188950453	-0.080176833	-0.003163181
Right Ankle	0.241527156	-0.065156728	0.006255086
Right Wrist	0.211446183	-0.069707086	-0.019905695
Left Shoulder	0.01607591	-0.069950812	-0.119403856
Left Hip	0.003814737	-0.019484536	-0.198284078
Right Shoulder	-0.00219989	-0.069832041	0.132929455
Right Hip	0.263017638	-0.022520684	-0.202218743

Table 3 Table of Standard Deviations of Rotation Points

Rotation Point	σ_x	σ_y	σ_z
Waist	0.003896258	0.004605284	0.014676666
Neck	0.001768695	0.001355136	0.001206322
Left Ankle	0.01981794	0.011053845	0.011233588
Left Wrist	0.002071725	0.002539415	0.000947775
Left Elbow	0.015518996	0.004050952	0.00398332
Left Knee	0.011752322	0.002762332	0.003553895
Right Elbow	0.017223326	0.00815392	0.004313482
Right Knee	0.016813465	0.003744518	0.003104466
Right Ankle	0.129159355	0.034143233	0.020772927
Right Wrist	0.00163169	0.000935295	0.000834657
Left Shoulder	0.001960198	0.000852649	0.00314934
Left Hip	0.001577703	0.001635193	0.004697627
Right Shoulder	0.000891381	0.00158762	0.003755364
Right Hip	0.001447724	0.00128305	0.004635324

Most of the standard deviations are less than one centimeter, but there are some significant outliers like the right ankle. Further analysis of the calculated points for the ankles and elbows show that the four runs produced two answer due to different orientations of the parent's reference frame. Therefore the standard deviation presented above for the ankles and elbows are erroneously calculating the deviation from the average of two distinct means. It is more appropriate to calculate the standard deviation from a single mean. When these outliers are removed from the calculation of the deviation, a very informative graph can be produced below. Every calculation for every segment is presented below as a deviation from the single mean rotation point versus the number of sample.

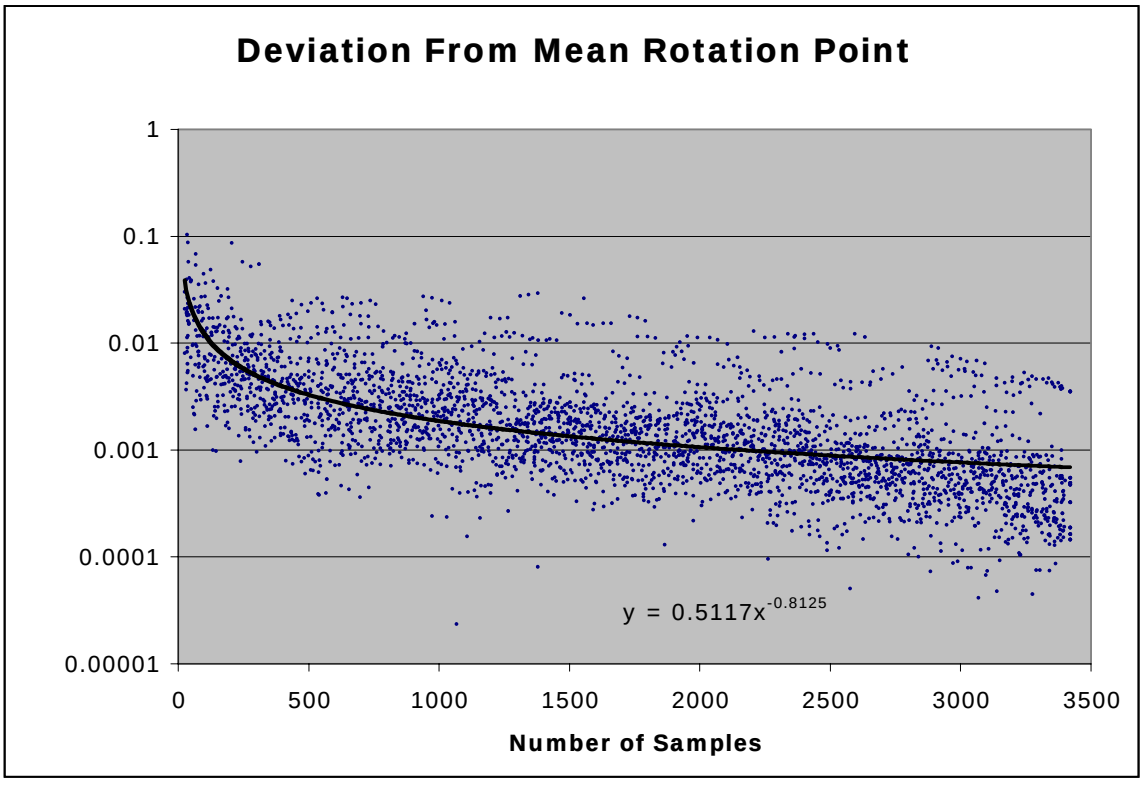

Figure 24 Inverse Power Law for Rotation Point Calculation

As can be readily seen from the above graph, a statistically significant amount of calculations are within one centimeter of accuracy when analyzing more than about 200 samples. The accuracy gets better on average with a power law close to $1/\sqrt{N}$.

6.2 Case Study of Eric Camper Data

The motion capture data in the ericcamper.c3d was analyzed to produce a skeleton. The results of the drawing produced favorable results due to the range of motion (ROM) involved in the exercise recorded. The subject did various martial arts maneuvers that moved every joint involved in drawing. One time frame is presented in the following figure. Although this is a purely qualitative analysis, the picture shows what appears to be a natural pose for all joints during a karate exercise.

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Figure 25 Eric Camper Skeleton

6.3 Comparison

The following table provides a succinct view of all of the methods studied in this thesis. It provides the positive and negative aspects of the methods. This table makes it easier to choose which solution is right for their particular situation.

Table 4 Comparison of Center Estimators

Method	Advantages	Disadvantages
MLE	accepted theory of best solution	slow, initial guess, may not converge
LLS	semi fast O(236*N*), no initial guess	loss of significant figures
GDKE	fast O(26*N*), no initial guess	biased O(σ)
UGDK*	fast O(26*N*), no initial guess, asymptotically unbiased	slightly biased O($\sigma/\sqrt{N}$)
SGDK*	removes singularities matrices	slower – need eigenvalues
MGDK*	handles multiple markers, averages out errors	slightly biased O($\sigma/\sqrt{N}$)

IGDK*	space O(1), time O(123*N*)	uses GDK first

* new solutions developed in this thesis

6.4 Speed

A Monte-Carlo experiment was set up to determine the speed of the various sphere-fit algorithms. Up to a million samples were chosen on a sphere with varying measurement error, confinement angle, sphere center and sphere radius. The measurement error varied from $1x10^{-11}$ to $1x10^{12}$. The confinement angle varied from 0 to 180°. The sphere center varied as much as 2 around the origin. The radius varied 0 to 37. The linear algebra algorithms were all implemented from well-accepted implementations presented in Numerical Recipes in C [94]. The code was compiled optimized for a PowerPC G4 processor and run on a 1GHz Apple PowerBook 12". The next graph presents the timing for the four algorithms.

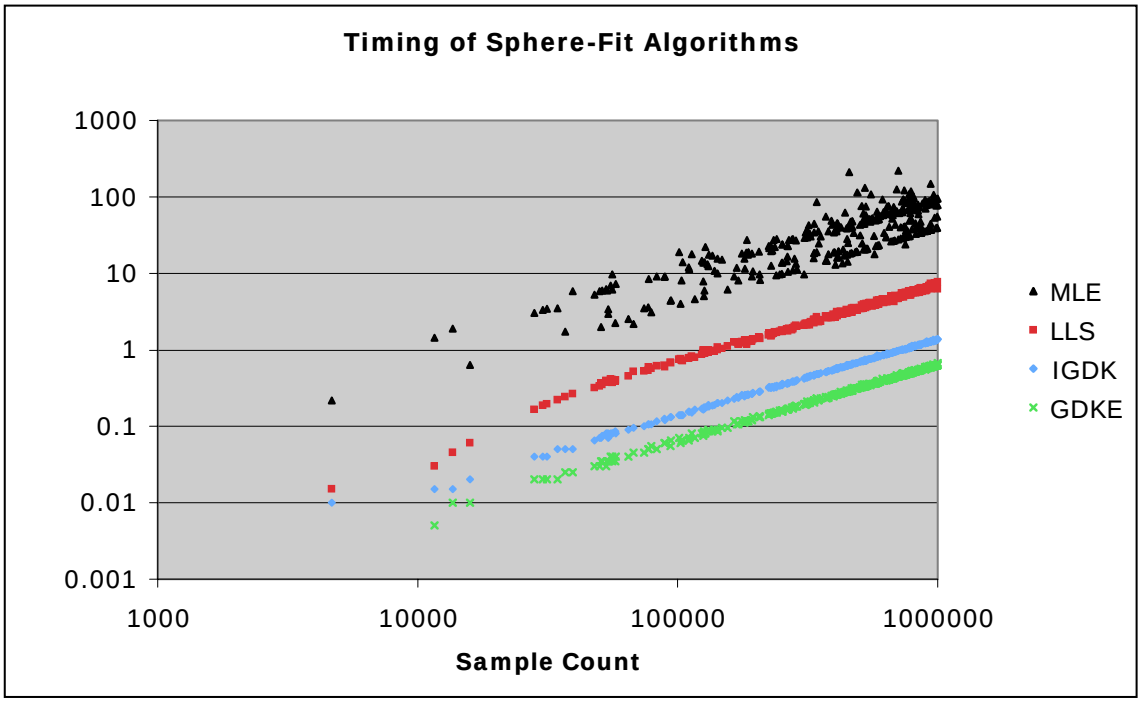

Figure 26 Timing of Algorithms

The previous graph shows that all of these algorithms are linearly dependent on sample count (i.e. O(*N*)). It further shows the MLE as messier with two distinct multiplication factors to this dependency. More information can be retrieved if the times are compared to the GDKE's time. The next graph shows this ratio compared to the measurement error.

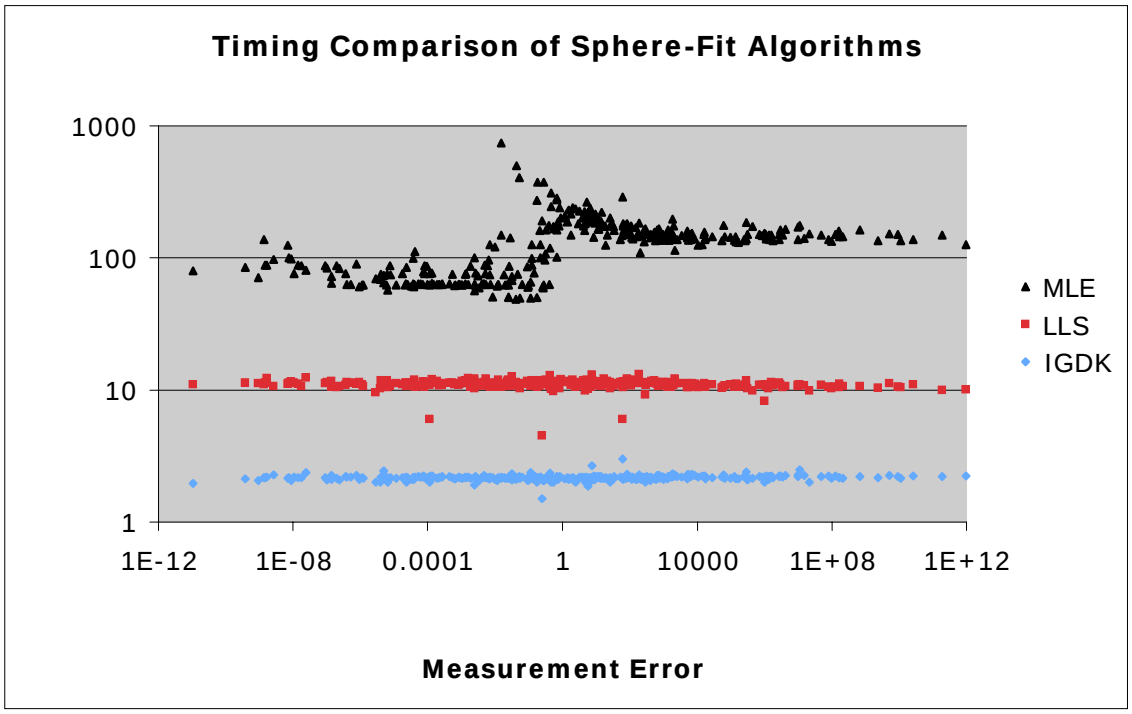

Figure 27 Timing Comparison to GDKE

As can be seen, the GDKE is always faster. The GDKE is 2.17 times faster than IGDK on average. The GDKE is 11.05 times faster than the LLS method and about 80 times faster than the MLE when the measurement error is less than one. The FLOP count predicts this closely with LLS = 236/26 = 9.07 and IGDK = 123/26 = 4.73. The differences between the FLOP count and experiment can be accounted for in the hardware and software overhead that FLOP count never accounts for.

6.5 Conclusion

This five year research effort into trying to speed up motion capture animation resulted in the discovery of the vital low-level hole that needed filling. That hole is the speedy calculations of rotation points directly from motion capture data. This thesis explains the fastest known general method for calculating these rotation points and can be

as much as ten times faster than the next fastest method. An international community of computer scientists has accepted the initial results of this research earlier this year as a full paper [63]. This thesis demonstrates the mathematics, probabilities, and implementations for this new method for determining the center of a hypersphere. The UGDK method has further impact in a vast collection of fields as diverse as character recognition to nuclear physics where an algorithm is needed for the speedy recovery of the center of a circle or sphere. The UGDK is expected to play a vital role in the process of determining a skeleton from motion capture data. The MGDK adds robustness to the equations allowing to use every bit of available data. The IGDK further has the application of being an ideal algorithm to burn into a silicon chip whose memory requirements are constrained.

6.6 Important Contributions

First and foremost, the best contribution to the science for this dissertation is the fastest, asymptotically unbiased estimator of a hypersphere

$$\hat{c}' = \bar{x} + \tfrac{1}{2}\left(\mathbf{C} - \hat{\Sigma}\right)^{-1}\mathbf{S}$$

$$\hat{r}' = \sqrt{\tfrac{N-1}{N}Tr\left(\mathbf{C} - \hat{\Sigma}\right) + \left(\bar{x} - \hat{c}'\right)^{T}\left(\bar{x} - \hat{c}'\right)}$$

and for multiple markers

$$\hat{c}'_m = \left(\sum_{p=1}^{M}\left(\mathbf{C}_p - \hat{\Sigma}\right)\right)^{-1}\sum_{p=1}^{M}\left(\left(\mathbf{C}_p - \hat{\Sigma}\right)\bar{x}_p + \tfrac{1}{2}\mathbf{S}_p\right).$$

These made it possible for a closed form solution of a skeleton from generic, noisy motion capture data. Understanding when the best measurement conditions are for reducing the risks of measurements is important. This paper presents an easy to measure limit to

strive for when trying to calculate the center of a sphere with partial coverage. The relationship

$$\sigma < r\frac{1}{\sqrt{3}}\sin^2\left(\frac{\theta}{2}\right)$$

must be satisfied in order to achieve good results using our algorithm.

Another important contribution is the full analysis of the Cramér-Rao Lower Bound for the confined points on a hypersphere. The circle CRLB is

$$CRLB_2 = \tfrac{1}{N}\sigma^2 \frac{2\theta}{2\theta(\theta+\cos\theta\sin\theta)-4\sin^2\theta}\begin{pmatrix} \frac{2\theta(\theta+\cos\theta\sin\theta)-4\sin^2\theta}{\theta-\cos\theta\sin\theta} & 0 & 0 \\ 0 & 2\theta & -2\sin\theta \\ 0 & -2\sin\theta & \theta+\cos\theta\sin\theta \end{pmatrix}$$

and the spherical CRLB is

$$CRLB_3 = \tfrac{1}{N}\sigma^2 \frac{3}{\sin^4\left(\frac{\theta}{2}\right)}\begin{pmatrix} \frac{\sin^2\left(\frac{\theta}{2}\right)}{(2+\cos\theta)} & 0 & 0 & 0 \\ 0 & \frac{\sin^2\left(\frac{\theta}{2}\right)}{(2+\cos\theta)} & 0 & 0 \\ 0 & 0 & 1 & -\cos^2\left(\frac{\theta}{2}\right) \\ 0 & 0 & -\cos^2\left(\frac{\theta}{2}\right) & \frac{3\cos^4\left(\frac{\theta}{2}\right)+\sin^4\left(\frac{\theta}{2}\right)}{3} \end{pmatrix}$$

These covariances of the estimators allow us to determine the best possible error in any estimator for the sphere.

The contributions of this thesis cover every aspect of generating arbitrary skeletons from motion capture data in a speedy fashion without compromising accuracy.

6.7 Further Research

The UGDK estimators are an improvement on existing science but they are strictly dependent on a-priori knowledge of the measurement error. It has been shown that the measurement error trace can itself be estimated but not the whole measurement covariance matrix. An estimator for the whole matrix would be most ideal but was not found in the course of this study. In addition, the statistical properties were not explored in this paper when the points do not have the same measurement covariance. This will surely introduce an additional error or bias in the estimators' answers.

Chapter 7 APPENDIX

7.1 Mathematical Proofs

7.1.1 Moments of Multivariate Normal

The probability analysis presented in this thesis relies on determining the moments of the multivariate normal probability distribution. A multivariate normal deviate has the probability density function of

$$f_N(x;\mu,\Sigma)=\frac{1}{\sqrt{(2\pi)^N|\Sigma|}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{258}$$

for a deviate *x* defined by

$$x=\begin{pmatrix}x_1 & x_2 & \cdots & x_N\end{pmatrix}^T \tag{259}$$

and the distribution parameters of

$$\mu=\begin{pmatrix}\mu_1 & \mu_2 & \cdots & \mu_N\end{pmatrix}^T \tag{260}$$

$$\Sigma=\begin{pmatrix}\Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1N}\\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2N}\\ \vdots & \vdots & \ddots & \vdots\\ \Sigma_{N1} & \Sigma_{N2} & \cdots & \Sigma_{NN}\end{pmatrix} \tag{261}$$

The moment generating function is

$$M_N(t;\mu,\Sigma)=e^{\mu^T t+\frac{1}{2}t^T\Sigma t} \tag{262}$$

where

$$t=\begin{pmatrix}t_1 & t_2 & \cdots & t_N\end{pmatrix}^T \tag{263}$$

The moments can be retrieved from the derivatives of the moment generating function by

(264) $$E\left(x_i^k x_j^l x_k^m \cdots\right) = \frac{\partial^k}{\partial t_i^k}\frac{\partial^l}{\partial t_j^l}\frac{\partial^m}{\partial t_k^m}\cdots M_N\Big|_{t=0}$$

That leads to the moments of the elements of the vector *x*

(265) $$E\left(x_i\right) = \mu_i$$

(266) $$E\left(x_i x_j\right) = \mu_i\mu_j + \Sigma_{ij}$$

(267) $$E\left(x_i x_j x_k\right) = \mu_i\mu_j\mu_k + \Sigma_{ij}\mu_k + \Sigma_{ik}\mu_j + \Sigma_{jk}\mu_i$$

(268) $$\begin{aligned} E\left(x_i x_j x_k x_l\right) = {} & \mu_i\mu_j\mu_k\mu_l + \Sigma_{ij}\Sigma_{kl} + \Sigma_{ik}\Sigma_{lj} + \Sigma_{il}\Sigma_{jk} \\ & + \Sigma_{ij}\mu_k\mu_l + \Sigma_{ik}\mu_j\mu_l + \Sigma_{il}\mu_j\mu_k + \Sigma_{jk}\mu_i\mu_l + \Sigma_{jl}\mu_i\mu_k + \Sigma_{kl}\mu_i\mu_j \end{aligned}$$

(269) $$\begin{aligned} E\left(x_i x_j x_k x_l x_m\right) = {} & \mu_i\mu_j\mu_k\mu_l\mu_m \\ & + \Sigma_{ij}\left(\mu_k\mu_l\mu_m + \Sigma_{kl}\mu_m + \Sigma_{km}\mu_l + \Sigma_{lm}\mu_k\right) \\ & + \Sigma_{ik}\left(\mu_j\mu_l\mu_m + \Sigma_{jl}\mu_m + \Sigma_{jm}\mu_l + \Sigma_{lm}\mu_j\right) \\ & + \Sigma_{il}\left(\mu_j\mu_k\mu_m + \Sigma_{jk}\mu_m + \Sigma_{jm}\mu_k + \Sigma_{km}\mu_j\right) \\ & + \Sigma_{im}\left(\mu_j\mu_k\mu_l + \Sigma_{jk}\mu_l + \Sigma_{jl}\mu_k + \Sigma_{kl}\mu_j\right) \\ & + \Sigma_{jk}\mu_i\mu_l\mu_m + \Sigma_{jl}\mu_i\mu_k\mu_m + \Sigma_{jm}\mu_i\mu_k\mu_l \\ & + \Sigma_{jm}\Sigma_{kl}\mu_i + \Sigma_{jk}\Sigma_{lm}\mu_i + \Sigma_{jl}\Sigma_{mk}\mu_i \\ & + \Sigma_{kl}\mu_i\mu_j\mu_m + \Sigma_{km}\mu_i\mu_j\mu_l + \Sigma_{lm}\mu_i\mu_j\mu_k \end{aligned}$$

(270)

$$
\begin{aligned}
E\left(x_i x_j x_k x_l x_m x_n\right) = {} & \mu_i\mu_j\mu_k\mu_l\mu_m\mu_n \\
& + \Sigma_{ij}\begin{pmatrix} \mu_m\mu_n\mu_k\mu_l + \Sigma_{mn}\Sigma_{kl} + \Sigma_{mk}\Sigma_{nl} + \Sigma_{kn}\Sigma_{lm} \\ + \Sigma_{kl}\mu_m\mu_n + \Sigma_{km}\mu_l\mu_n + \Sigma_{kn}\mu_l\mu_m + \Sigma_{lm}\mu_k\mu_n \\ + \Sigma_{nl}\mu_k\mu_m + \Sigma_{mn}\mu_k\mu_l \end{pmatrix} \\
& + \Sigma_{ik}\begin{pmatrix} \mu_j\mu_l\mu_m\mu_n + \Sigma_{jl}\Sigma_{mn} + \Sigma_{jm}\Sigma_{nl} + \Sigma_{jn}\Sigma_{lm} \\ + \Sigma_{jl}\mu_m\mu_n + \Sigma_{jm}\mu_l\mu_n + \Sigma_{jn}\mu_l\mu_m + \Sigma_{lm}\mu_j\mu_n \\ + \Sigma_{nl}\mu_j\mu_m + \Sigma_{mn}\mu_j\mu_l \end{pmatrix} \\
& + \Sigma_{il}\begin{pmatrix} \mu_j\mu_k\mu_m\mu_n + \Sigma_{jk}\Sigma_{mn} + \Sigma_{jm}\Sigma_{kn} + \Sigma_{jn}\Sigma_{km} \\ + \Sigma_{jk}\mu_m\mu_n + \Sigma_{jm}\mu_k\mu_n + \Sigma_{jn}\mu_k\mu_m + \Sigma_{km}\mu_j\mu_n \\ + \Sigma_{kn}\mu_j\mu_m + \Sigma_{mn}\mu_j\mu_k \end{pmatrix} \\
& + \Sigma_{im}\begin{pmatrix} \mu_j\mu_k\mu_l\mu_n + \Sigma_{jk}\Sigma_{nl} + \Sigma_{jl}\Sigma_{kn} + \Sigma_{jn}\Sigma_{kl} \\ + \Sigma_{jk}\mu_l\mu_n + \Sigma_{jl}\mu_k\mu_n + \Sigma_{jn}\mu_k\mu_l + \Sigma_{kl}\mu_j\mu_n \\ + \Sigma_{kn}\mu_j\mu_l + \Sigma_{nl}\mu_j\mu_k \end{pmatrix} \\
& + \Sigma_{in}\begin{pmatrix} \mu_j\mu_k\mu_l\mu_m + \Sigma_{jk}\Sigma_{lm} + \Sigma_{jl}\Sigma_{km} + \Sigma_{jm}\Sigma_{kl} \\ + \Sigma_{jk}\mu_l\mu_m + \Sigma_{jl}\mu_k\mu_m + \Sigma_{jm}\mu_k\mu_l + \Sigma_{kl}\mu_j\mu_m \\ + \Sigma_{km}\mu_j\mu_l + \Sigma_{lm}\mu_j\mu_k \end{pmatrix} \\
& + \mu_i\begin{pmatrix} \Sigma_{jk}\left(\mu_l\mu_m\mu_n + \Sigma_{lm}\mu_n + \Sigma_{\ln}\mu_m + \Sigma_{mn}\mu_l\right) \\ + \Sigma_{jl}\left(\mu_k\mu_m\mu_n + \Sigma_{km}\mu_n + \Sigma_{kn}\mu_m + \Sigma_{mn}\mu_k\right) \\ + \Sigma_{jm}\left(\mu_k\mu_l\mu_n + \Sigma_{kl}\mu_n + \Sigma_{kn}\mu_l + \Sigma_{\ln}\mu_k\right) \\ + \Sigma_{jn}\left(\mu_k\mu_l\mu_m + \Sigma_{kl}\mu_m + \Sigma_{km}\mu_l + \Sigma_{lm}\mu_k\right) \\ + \Sigma_{kl}\mu_j\mu_m\mu_n + \Sigma_{km}\mu_j\mu_l\mu_n + \Sigma_{kn}\mu_j\mu_l\mu_m \\ + \Sigma_{kn}\Sigma_{lm}\mu_j + \Sigma_{kl}\Sigma_{mn}\mu_j + \Sigma_{km}\Sigma_{nl}\mu_j \\ + \Sigma_{lm}\mu_j\mu_k\mu_n + \Sigma_{\ln}\mu_j\mu_k\mu_m + \Sigma_{mn}\mu_j\mu_k\mu_l \end{pmatrix}
\end{aligned}
$$

If all indices are equal, the expectations simplify to

(271)
$$E\left(x_i\right) = \mu_i$$

(272)
$$E\left(x_i^2\right) = \mu_i^2 + \Sigma_{ii}$$

(273)
$$E\left(x_i^3\right) = \mu_i^3 + 3\Sigma_{ii}\mu_i$$

(274)
$$E\left(x_i^4\right) = \mu_i^4 + 6\Sigma_{ii}\mu_i^2 + 3\Sigma_{ii}^2$$

(275) $$E\left(x_i^5\right)=\mu_i^5+10\Sigma_{ii}\mu_i^3+15\Sigma_{ii}^2\mu_i$$

(276) $$E\left(x_i^6\right)=\mu_i^6+15\Sigma_{ii}\mu_i^4+45\Sigma_{ii}^2\mu_i^2+15\Sigma_{ii}^3$$

The vector moments are

(277) $$E\left(x\right)=\mu$$

(278) $$E\left(xx^T\right)=\mu\mu^T+\Sigma$$

(279) $$E\left(x^Tx\right)=\mu^T\mu+Tr\left(\Sigma\right)$$

(280) $$E\left(xx^Tx\right)=\mu\mu^T\mu+Tr\left(\Sigma\right)\mu+2\Sigma\mu$$

(281) $$E\left(xx^Txx^T\right)=\mu\mu^T\mu\mu^T+2\Sigma^2+\Sigma Tr\left(\Sigma\right)+Tr\left(\Sigma\right)\mu\mu^T+2\mu\mu^T\Sigma+2\Sigma\mu\mu^T+\Sigma\mu^T\mu$$

(282) $$E\left(xx^Tax^T\right)=\mu\mu^Ta\mu^T+\Sigma\mu^Ta+\Sigma a\mu^T+\mu a^T\Sigma$$

(283) $$E\left(x^Txx^Tx\right)=\mu^T\mu\mu^T\mu+2Tr\left(\Sigma^2\right)+Tr\left(\Sigma\right)^2+2Tr\left(\Sigma\right)\mu^T\mu+4\mu^T\Sigma\mu$$

(284) $$\begin{aligned}E\left(xx^Txx^Tx\right)=&\mu\mu^T\mu\mu^T\mu+2Tr\left(\Sigma\right)\mu\mu^T\mu+4\mu\mu^T\Sigma\mu+4\Sigma\mu\mu^T\mu\\&+2Tr\left(\Sigma^2\right)\mu+Tr\left(\Sigma\right)^2\mu+4Tr\left(\Sigma\right)\Sigma\mu+8\Sigma^2\mu\end{aligned}$$

(285) $$\begin{aligned}E\left(xx^Txx^Tax^T\right)=&\mu\mu^T\mu\mu^Ta\mu^T+\Sigma a^T\mu\mu^T\mu+\Sigma Tr\left(\Sigma\right)a^T\mu+Tr\left(\Sigma\right)\mu\mu^Ta\mu^T\\&+2\Sigma a^T\Sigma\mu+\Sigma a\mu^T\mu\mu^T+2\Sigma a\mu^T\Sigma+\Sigma a\mu^TTr\left(\Sigma\right)+2\Sigma\mu a^T\mu\mu^T\\&+2\Sigma\mu a^T\Sigma+2\Sigma\Sigma a^T\mu+2\Sigma\Sigma a\mu^T+\mu\mu^T\mu a^T\Sigma+2\mu a^T\mu\mu^T\Sigma\\&+2\mu a^T\Sigma\Sigma+\mu a^T\Sigma Tr\left(\Sigma\right)+2\mu a^T\Sigma\mu\mu^T\end{aligned}$$

(286) $$\begin{aligned}E\left(xx^Tab^Txx^T\right)=&\mu\mu^Tab^T\mu\mu^T+\Sigma a^T\Sigma b+\Sigma ab^T\Sigma+\Sigma ba^T\Sigma+\Sigma\mu^Tab^T\mu\\&+\Sigma ab^T\mu\mu^T+\Sigma ba^T\mu\mu^T+\mu\mu^Tba^T\Sigma+\mu\mu^Tab^T\Sigma+\mu b^T\Sigma a\mu^T\end{aligned}$$

(287) $$\begin{aligned}E\left(xx^TAxx^T\right)=&\mu\mu^TA\mu\mu^T+\Sigma Tr\left(A\Sigma\right)+\Sigma A\Sigma+\Sigma A^T\Sigma+\Sigma\mu^TA\mu\\&+\Sigma A\mu\mu^T+\Sigma A^T\mu\mu^T+\mu\mu^TA^T\Sigma+\mu\mu^TA\Sigma+Tr\left(A^T\Sigma\right)\mu\mu^T\end{aligned}$$

$$
\begin{aligned}
E\left(xx^T xx^T xx^T\right) = {} & \mu\mu^T\mu\mu^T\mu\mu^T + \Sigma Tr\left(\Sigma\right)^2 + 2\Sigma Tr\left(\Sigma^2\right) + 8\Sigma^3 + 4\Sigma^2 Tr\left(\Sigma\right) \\
& + 4\Sigma\mu^T\Sigma\mu + 8\Sigma\mu\mu^T\Sigma + 4\Sigma^2\mu^T\mu + 8\Sigma^2\mu\mu^T + 8\mu\mu^T\Sigma^2 \\
& + 4\mu\mu^T\Sigma Tr\left(\Sigma\right) + 4\Sigma Tr\left(\Sigma\right)\mu\mu^T + 2Tr\left(\Sigma^2\right)\mu\mu^T + Tr\left(\Sigma\right)^2\mu\mu^T \\
& + 2\Sigma Tr\left(\Sigma\right)\mu^T\mu + 4\Sigma\mu\mu^T\mu\mu^T + 4\mu\mu^T\mu\mu^T\Sigma + 4\mu\mu^T\Sigma\mu\mu^T \\
& + 2\mu\mu^T\mu\mu^T Tr\left(\Sigma\right) + \Sigma\mu^T\mu\mu^T\mu
\end{aligned} \tag{288}
$$

The vector covariances are calculated with the definition of

$$Cov\left(a,b\right) = E\left(ab\right) - E\left(a\right)E\left(b\right) \tag{289}$$

to produce the following

$$Cov\left(x,x^T\right) = \Sigma \tag{290}$$

$$Cov\left(x^T,x\right) = Tr\left(\Sigma\right) \tag{291}$$

$$Cov\left(x,x^T x\right) = 2\Sigma\mu \tag{292}$$

$$Cov\left(xx^T,x\right) = Tr\left(\Sigma\right)\mu + \Sigma\mu \tag{293}$$

$$Cov\left(x,x^T xx^T\right) = 2\Sigma^2 + \Sigma Tr\left(\Sigma\right) + 2\Sigma\mu\mu^T + \Sigma\mu^T\mu \tag{294}$$

$$Cov\left(xx^T,xx^T\right) = \Sigma^2 + \Sigma Tr\left(\Sigma\right) + Tr\left(\Sigma\right)\mu\mu^T + \mu\mu^T\Sigma + \Sigma\mu\mu^T + \Sigma\mu^T\mu \tag{295}$$

$$Cov\left(xx^T x,x^T\right) = 2\Sigma^2 + \Sigma Tr\left(\Sigma\right) + 2\mu\mu^T\Sigma + \Sigma\mu^T\mu \tag{296}$$

$$Cov\left(x,x^T ax^T\right) = \Sigma\mu^T a + \Sigma a\mu^T \tag{297}$$

$$Cov\left(xx^T,ax^T\right) = \Sigma\mu^T a + \mu a^T\Sigma \tag{298}$$

$$Cov\left(xx^T a,x^T\right) = \Sigma\mu^T a + \mu a^T\Sigma \tag{299}$$

$$Cov\left(x^T,xx^T x\right) = 2Tr\left(\Sigma^2\right) + Tr\left(\Sigma\right)^2 + Tr\left(\Sigma\right)\mu^T\mu + 2\mu^T\Sigma\mu \tag{300}$$

$$Cov\left(x^T x,x^T x\right) = 2Tr\left(\Sigma^2\right) + 4\mu^T\Sigma\mu \tag{301}$$

$$Cov\left(x^T xx^T,x\right) = 2Tr\left(\Sigma^2\right) + Tr\left(\Sigma\right)^2 + Tr\left(\Sigma\right)\mu^T\mu + 2\mu^T\Sigma\mu \tag{302}$$

(303) $$Cov\left(x,x^T xx^T x\right)=4\Sigma\mu\mu^T\mu+4Tr(\Sigma)\Sigma\mu+8\Sigma^2\mu$$

(304) $$Cov\left(xx^T,xx^T x\right)=\left(\left(Tr(\Sigma)+3\Sigma\right)\left(\mu\mu^T+Tr(\Sigma)\right)+2\mu\mu^T\Sigma+2Tr\left(\Sigma^2\right)+6\Sigma^2\right)\mu$$

(305) $$Cov\left(xx^T x,x^T x\right)=4\mu\mu^T\Sigma\mu+2\Sigma\mu\mu^T\mu+2Tr\left(\Sigma^2\right)\mu+2Tr(\Sigma)\Sigma\mu+8\Sigma^2\mu$$

(306) $$\begin{aligned}Cov\left(xx^T xx^T,x\right)&=Tr(\Sigma)\mu\mu^T\mu+2\mu\mu^T\Sigma\mu+\Sigma\mu\mu^T\mu\\&\quad+2Tr\left(\Sigma^2\right)\mu+Tr(\Sigma)^2\mu+3Tr(\Sigma)\Sigma\mu+6\Sigma^2\mu\end{aligned}$$

(307) $$\begin{aligned}Cov\left(x,x^T xx^T xx^T\right)&=\Sigma Tr(\Sigma)^2+2\Sigma Tr\left(\Sigma^2\right)+8\Sigma^3+4\Sigma^2 Tr(\Sigma)\\&\quad+4\Sigma\mu^T\Sigma\mu+8\Sigma\mu\mu^T\Sigma+4\Sigma^2\mu^T\mu+8\Sigma^2\mu\mu^T\\&\quad+4\Sigma Tr(\Sigma)\mu\mu^T+2\Sigma Tr(\Sigma)\mu^T\mu+4\Sigma\mu\mu^T\mu\mu^T+\Sigma\mu^T\mu\mu^T\mu\end{aligned}$$

(308) $$\begin{aligned}Cov\left(xx^T,xx^T xx^T\right)&=\Sigma Tr(\Sigma)^2+2\Sigma Tr\left(\Sigma^2\right)+6\Sigma^3+3\Sigma^2 Tr(\Sigma)\\&\quad+4\Sigma\mu^T\Sigma\mu+6\Sigma\mu\mu^T\Sigma+3\Sigma^2\mu^T\mu+6\Sigma^2\mu\mu^T+6\mu\mu^T\Sigma^2\\&\quad+4\mu\mu^T\Sigma Tr(\Sigma)+3\Sigma Tr(\Sigma)\mu\mu^T+2Tr\left(\Sigma^2\right)\mu\mu^T+Tr(\Sigma)^2\mu\mu^T\\&\quad+2\Sigma Tr(\Sigma)\mu^T\mu+3\Sigma\mu\mu^T\mu\mu^T+\mu\mu^T\mu\mu^T\Sigma+2\mu\mu^T\Sigma\mu\mu^T\\&\quad+\mu\mu^T\mu\mu^T Tr(\Sigma)+\Sigma\mu^T\mu\mu^T\mu\end{aligned}$$

(309) $$\begin{aligned}Cov\left(xx^T x,x^T xx^T\right)&=\Sigma Tr(\Sigma)^2+2\Sigma Tr\left(\Sigma^2\right)+8\Sigma^3+4\Sigma^2 Tr(\Sigma)\\&\quad+4\Sigma\mu^T\Sigma\mu+4\Sigma\mu\mu^T\Sigma+4\Sigma^2\mu^T\mu+8\Sigma^2\mu\mu^T+8\mu\mu^T\Sigma^2\\&\quad+2\mu\mu^T\Sigma Tr(\Sigma)+2\Sigma Tr(\Sigma)\mu\mu^T+2Tr\left(\Sigma^2\right)\mu\mu^T+2\Sigma Tr(\Sigma)\mu^T\mu\\&\quad+2\Sigma\mu\mu^T\mu\mu^T+2\mu\mu^T\mu\mu^T\Sigma+4\mu\mu^T\Sigma\mu\mu^T+\Sigma\mu^T\mu\mu^T\mu\end{aligned}$$

The vector central moments are

(310) $$E\left(x-\mu\right)=0$$

(311) $$E\left(\left(x-\mu\right)\left(x-\mu\right)^T\right)=\Sigma$$

(312) $$Cov\left(\left(x-\mu\right),\left(x-\mu\right)^T\right)=\Sigma$$

(313) $$E\left(\left(x-\mu\right)^T\left(x-\mu\right)\right)=Tr(\Sigma)$$

(314) $$Cov\left(\left(x-\mu\right)^T,\left(x-\mu\right)\right)=Tr(\Sigma)$$

(315) $$E\left((x-\mu)(x-\mu)^T(x-\mu)\right)=0$$

(316) $$E\left((x-\mu)(x-\mu)^T(x-\mu)(x-\mu)^T\right)=2\Sigma^2+\Sigma Tr(\Sigma)$$

(317) $$Cov\left((x-\mu)(x-\mu)^T,(x-\mu)(x-\mu)^T\right)=\Sigma^2+\Sigma Tr(\Sigma)$$

(318) $$E\left((x-\mu)^T(x-\mu)(x-\mu)^T(x-\mu)\right)=2Tr(\Sigma^2)+Tr(\Sigma)^2$$

(319) $$Cov\left((x-\mu)^T(x-\mu),(x-\mu)^T(x-\mu)\right)=2Tr(\Sigma^2)$$

(320) $$E\left((x-\mu)(x-\mu)^T(x-\mu)(x-\mu)^T(x-\mu)\right)=0$$

(321) $$E\left((x-\mu)(x-\mu)^T(x-\mu)(x-\mu)^T(x-\mu)(x-\mu)^T\right)=8\Sigma^3+4\Sigma^2Tr(\Sigma)+2\Sigma Tr(\Sigma^2)+\Sigma Tr(\Sigma)^2$$

(322) $$Cov\left((x-\mu)(x-\mu)^T(x-\mu),(x-\mu)^T(x-\mu)(x-\mu)^T\right)=8\Sigma^3+4\Sigma^2Tr(\Sigma)+2\Sigma Tr(\Sigma^2)+\Sigma Tr(\Sigma)^2$$

(323) $$E\left((x-\mu)^T(x-\mu)(x-\mu)^T(x-\mu)(x-\mu)^T(x-\mu)\right)=8Tr(\Sigma^3)+6Tr(\Sigma^2)Tr(\Sigma)+Tr(\Sigma)^3$$

Now, we will consider when there are multiple measurements with the same covariance but different mean. The k^{th} measurement then has the properties of

(324) $$E(x_k)=\mu_k \text{ and}$$

(325) $$Cov(x_k,x_k^T)=\Sigma$$

The vector expectations of the averages of N independent measurements are

(326) $$E(\bar{x})=\bar{\mu}$$

(327) $$E(x_k\bar{x}^T)=\mu_k\bar{\mu}^T+\tfrac{1}{N}\Sigma$$

(328) $$E(x_kx_l^T\bar{x})=\mu_k\mu_l^T\bar{\mu}+\tfrac{1}{N}Tr(\Sigma)\mu_k+\tfrac{1}{N}\Sigma\mu_l$$

(329) $$E\left(x_k x_k^T \bar{x}\right) = \left(\mu_k \mu_k^T + \Sigma\right)\bar{\mu} + \frac{1}{N} Tr(\Sigma)\mu_k + \frac{1}{N}\Sigma\mu_k$$

(330) $$E\left(\bar{x}\bar{x}^T \bar{x}\right) = \left(\bar{\mu}\bar{\mu}^T + \frac{1}{N}\Sigma\right)\bar{\mu} + \frac{1}{N} Tr(\Sigma)\bar{\mu} + \frac{1}{N}\Sigma\bar{\mu}$$

(331) $$E\left(x_k \bar{x}^T \bar{x}\right) = \mu_k\left(\bar{\mu}^T\bar{\mu} + \frac{1}{N} Tr(\Sigma)\right) + \frac{2}{N}\Sigma\bar{\mu}$$

(332) $$E\left(\bar{x}\bar{x}^T x_k\right) = \left(\bar{\mu}\bar{\mu}^T + \frac{1}{N}\Sigma\right)\mu_k + \frac{1}{N} Tr(\Sigma)\bar{\mu} + \frac{1}{N}\Sigma\bar{\mu}$$

(333) $$E\left(\bar{x}x_k^T x_k\right) = \bar{\mu}\left(\mu_k^T\mu_k + Tr(\Sigma)\right) + \frac{2}{N}\Sigma\mu_k$$

(334) $$E\left(\bar{x}x_k^T x_l\right) = \bar{\mu}\mu_k^T\mu_l + \frac{1}{N}\Sigma\mu_k + \frac{1}{N}\Sigma\mu_l$$

The sample vector central moments are

(335) $$E\left(x_k - \bar{x}\right) = \mu_k - \bar{\mu}$$

(336) $$E\left(\left(x_k - \bar{x}\right)\left(x_l - \bar{x}\right)^T\right) = \left(\mu_k - \bar{\mu}\right)\left(\mu_l - \bar{\mu}\right)^T - \frac{1}{N}\Sigma$$

(337) $$E\left(\left(x_k - \bar{x}\right)^T\left(x_l - \bar{x}\right)\right) = \left(\mu_k - \bar{\mu}\right)^T\left(\mu_l - \bar{\mu}\right) - \frac{1}{N} Tr(\Sigma)$$

(338) $$E\left(\left(x_k - \bar{x}\right)\left(x_l - \bar{x}\right)^T\left(x_l - \bar{x}\right)\right) = \left(\mu_k - \bar{\mu}\right)\left(\mu_l - \bar{\mu}\right)^T\left(\mu_l - \bar{\mu}\right) + \frac{N-1}{N} Tr(\Sigma)\left(\mu_k - \bar{\mu}\right) - \frac{2}{N}\Sigma\left(\mu_l - \bar{\mu}\right)$$

Unbiased estimators of the vector central moments from *N* samples are

(339) $$\mathbf{C} = \frac{1}{N-1}\sum_{k=1}^{N}\left(x_k - \bar{x}\right)\left(x_k - \bar{x}\right)^T$$

(340) $$E(\mathbf{C}) = \mathbf{C}_0 + \Sigma$$

(341) $$Cov\left(\mathbf{C}_{ij}, \mathbf{C}_{mn}\right) = \frac{1}{N-1}\left(\Sigma_{im}\Sigma_{nj} + \Sigma_{in}\Sigma_{jm} + \Sigma_{im}\mathbf{C}_{0nj} + \Sigma_{in}\mathbf{C}_{0\,jm} + \Sigma_{jm}\mathbf{C}_{0in} + \Sigma_{jn}\mathbf{C}_{0im}\right)$$

(342) $$D = \frac{1}{N-1}\sum_{k=1}^{N}\left(x_k - \bar{x}\right)^T\left(x_k - \bar{x}\right)$$

(343) $$E(D) = Tr\left(\mathbf{C}_0 + \Sigma\right)$$

(344) $$Cov(D,D) = \frac{1}{N-1} Tr\left(2\Sigma^2 + 4\mathbf{C}_0\Sigma\right)$$

(345) $$\mathbf{S} = \tfrac{1}{N-1}\sum_{k=1}^{N}\left(x_k - \bar{x}\right)\left(x_k - \bar{x}\right)^T\left(x_k - \bar{x}\right)$$

(346) $$E\left(\mathbf{S}\right) = \mathbf{S}_0$$

(347) $$Cov\left(\mathbf{S},\mathbf{S}^T\right) = \tfrac{1}{N-1}\left(\Sigma^3 + \Sigma^2\mathbf{C}_0 + \Sigma\mathbf{C}_0\Sigma + \mathbf{C}_0\Sigma^2 + \mathbf{F}_0\Sigma + \Sigma\mathbf{F}_0\right)$$

(348) $$\mathbf{F} = \tfrac{N}{(N-1)^2}\sum_{k=1}^{N}\left(x_k - \bar{x}\right)\left(x_k - \bar{x}\right)^T\left(x_k - \bar{x}\right)\left(x_k - \bar{x}\right)^T$$

(349) $$E\left(\mathbf{F}\right) = \mathbf{F}_0 + 2\Sigma^2 + \Sigma Tr\left(\Sigma\right)$$

(350) $$G = \tfrac{N}{(N-1)^2}\sum_{k=1}^{N}\left(x_k - \bar{x}\right)^T\left(x_k - \bar{x}\right)\left(x_k - \bar{x}\right)^T\left(x_k - \bar{x}\right)$$

(351) $$E\left(G\right) = Tr\left(\mathbf{F}_0\right) + 2Tr\left(\Sigma^2\right) + Tr\left(\Sigma\right)^2$$

7.1.2 Positive-Semidefinite Sample Covariance

A positive-semidefinite matrix **A** is defined as having the property $h^{\mathrm{T}}\mathbf{A}h \geq 0$ for an arbitrary vector $h \neq \mathbf{0}$. The sample covariance matrix is

(352) $$\mathbf{C} = \tfrac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)\left(x_i - \bar{x}\right)^T$$

Then applying an arbitrary vector h

(353) $$h^T\mathbf{C}h = \tfrac{1}{N-1}\sum_{i=1}^{N}h^T\left(x_i - \bar{x}\right)\left(x_i - \bar{x}\right)^T h$$

Now defining $q_i \equiv \left(x_i - \bar{x}\right)^T h$ simplifies the sum to

(354) $$h^T\mathbf{C}h = \tfrac{1}{N-1}\sum_{i=1}^{N}q_i^2$$

which is always positive for an arbitrary set of values q_i except for one non-trivial case. The exception is when the points are coplanar. In the coplanar case, $q_i = 0$ for all i when h is normal to the plane and therefore $h^T\mathbf{C}h=0$.

7.2 Inertial Properties of a Tetrahedron

This paper does not deal with inertia but in the process of doing this research, various physical models were created. One model was the tetrahedral mesh along with its physical properties. Inertia properties were not easy to calculate so this section was left in for the convenience of future simulation work. Inertial properties can be quite difficult to calculate but can be exactly determined. The moment of inertia and angular momentum is taken from standard analytical mechanics books as

(355) $$I = \int |\hat{w} \times r|^2 \, dm$$

(356) $$L = \int r \times (w \times r) dm$$

where $dm = \rho \, dV$, the density times the differential volume.

The moment of inertia and angular momentum are very desirable traits to follow in any dynamic simulation. They both involve integrating over the tetrahedral volume. I have derived them below using parameterized coordinates. A point inside the tetrahedron can be uniquely determined by

(357) $$r = a_0 + (a_1 + (a_2 + a_3 t_0) t_1) t_2$$

where

(358) $$a_0 = v_2$$

(359) $$a_1 = v_3 - v_2$$

(360) $$a_2 = v_0 - v_3$$

(361) $$a_3 = v_1 - v_0$$

and v_i are one of the tetrahedron vertices

The differential volume is determined by the parameter space change formula

(362) $$dV = \frac{\partial r}{\partial t_2} \cdot \left(\frac{\partial r}{\partial t_1} \times \frac{\partial r}{\partial t_0} \right) dt_0 dt_1 dt_2$$

(363) $$\frac{\partial r}{\partial t_2} \cdot \left(\frac{\partial r}{\partial t_1} \times \frac{\partial r}{\partial t_0} \right) 6 t_1 t_2^2 V_{Tet}$$

Each parameter t_i varies from zero to one for inside the tetrahedron so the entire integral results in

(364) $$I = 6m \int_0^1 \int_0^1 \int_0^1 \left| \hat{w} \times r \right|^2 t_1 t_2^2 dt_0 dt_1 dt_2$$

where $\hat{w}$ is the angular velocity unit vector (i.e. the spin axis) and m is the mass of the entire tetrahedron. The density of the tetrahedron is kept constant and thus the mass is brought out of the integrand as the density times the volume. Performing the triple integral produces a double sum. The integral turns into the double sum

(365) $$I = 6m \sum_{i=0}^{3} \sum_{j=0}^{3} \left(\hat{w} \times a_i \right) \cdot \left(\hat{w} \times a_j \right) \int_0^1 \int_0^1 \int_0^1 t_0^{n_{0ij}} t_1^{n_{1ij}} t_2^{n_{2ij}} dt_0 dt_1 dt_2$$

where

(366) $$n_{kij} = \left\lfloor \frac{i+k}{3} \right\rfloor + \left\lfloor \frac{j+k}{3} \right\rfloor + k$$

Integrating the double sum and expanding the a_i terms into v_i terms produces the equation

(367) $$I = m\sum_{i=0}^{3}\sum_{j=0}^{3} q_{ij}a_{ij}$$

(368) $$a_{ij} = \left(\hat{w}\times v_i\right)\cdot\left(\hat{w}\times v_j\right)$$

(369) $$q_{ij} = \begin{cases} \dfrac{1}{20} & j \neq i \\ \dfrac{1}{10} & j = i \end{cases}$$

Since both q_{ij} and a_{ij} are symmetric then Equation (367) reduces to ten terms.

The angular momentum is very similar with the integral

(370) $$L = 6m\int_0^1\int_0^{t_2}\int_0^{t_1} r\times\left(w\times r\right)t_1t_2^2\,dt_0\,dt_1\,dt_2$$

(371) $$L = m\sum_{i=0}^{3}\sum_{j=0}^{3} q_{ij}b_{ij}$$

(372) $$b_{ij} = v_i\times\left(w\times v_j\right)$$

In this case, b_{ij} is not symmetric so all sixteen terms must be calculated. A much simpler form of these equations occurs when a body has a coordinate system that is centered on its center of mass and is aligned to its principal axes. This simplification will be approached in the next section. The following matrix equations separate out the purely geometric quantity Q from the spinning quantity $\hat{w}$.

(373) $$I_{tet} = m_{tet}\hat{w}^T Q(c)\hat{w}$$

(374) $$L_{tet} = m_{tet}Q(c)w$$

(375) $$Q_{tet}(c) = Q(c_m - c) + \frac{1}{20}\sum_{i=0}^{3} Q(v_i - c_m)$$

(376) $$Q(\rho) = \rho^T \rho - \rho\rho^T = \begin{pmatrix} \rho^2 - x^2 & -xy & -xz \\ -yx & \rho^2 - y^2 & -yz \\ -zx & -zy & \rho^2 - z^2 \end{pmatrix}$$

The time complexity of Q_{tet} is 30 multiplications + 54 additions. For each of the above equations, the vectors are relative to the center of rotation. To generalize, one must subtract the center of rotation from the four vertex vectors.

7.3 File Formats

7.3.1 Marker Association Format

This file is a simple text file that associates the marker identification string stored in a C3D file with a known marker label. The known marker label is identified in a marker-set file that clearly associates it with a particular segment on the body. This file allows for quick association of the data with a segment. The only other alternative is to do grouping analysis for all of the data that is very time consuming. The file contains the following format:

```
46 WANDS

MARKERSET vicon512.txt

TestSubjectProdMS-V2:Root = unknown
TestSubjectProdMS-V2:LFWT = LFWT
TestSubjectProdMS-V2:RFWT = RFWT
TestSubjectProdMS-V2:LBWT = LBWT
TestSubjectProdMS-V2:RBWT = RBWT
TestSubjectProdMS-V2:LKNE = LKNE
TestSubjectProdMS-V2:LTHI = LTHI
TestSubjectProdMS-V2:LANK = LANK
TestSubjectProdMS-V2:LSHN = LLEG
TestSubjectProdMS-V2:LHEE = LHEE
…
```

7.3.2 Articulated Tetrahedral Model Format

The Articulated Tetrahedral Model (ATM) file format was created in order to bring together the necessary information to produce a jointed figure. It is a text format that references other the individual segments' mesh files. The text file has keywords followed by values and sub-fields. All words are separated by spaces. The following are the list of available keywords:

```
FIGURE myFigure
```

This keyword defines the name of the entire figure.

```
SEGMENTS 16
```

The number of segments of the figure follows:

```
MESH 0 chest.mesh
```

mesh identifies the mesh file that follows the index value. The file is identified as a tetrahedral or triangular mesh in the *.mesh format defined by the freeware MEDIT tool for editing meshes and explained in Chapter 7.3.3. The filename is for a file that is located in the same folder as the ATM file.

```
NAME Chest
```

The name of the segment follows this keyword.

```
MASS 30.0 // kg
```

The mass of the segment follows this keyword.

```
SCALE_XYZ 0.2 0.2 1.0
TRANSLATE_X 1.5
TRANSLATE_Y 1.5
```

```
TRANSLATE_Z 1.5
TRANSLATE_XYZ 1.5 2.5 3.5
ROTATE_X 35.2
PARENT 4 // hips
```

The index of the parent segment is defined here.

```
JOINT Waist
  3 0.994765 1.002953 1.084350
DEGREES_OF_FREEDOM 1 0
  VALUE 0.0
  3 -0.257042 0.966392 -0.003915
  FROM -93.199994 TO 29.000000
```

The name of the joint that is proximal to this segment is defined here followed by joint parameters. The center of rotation is defined immediately following the JOINT keyword. First a count of how many dimensions for the vector is specified (usually 3). Then, each coordinate, e.g. x y z. After the joint position, its freedoms are defined. DEGREES_OF_FREEDOM is followed by two integers. The first is the count of rotational freedoms, and the second is the count of translational freedoms. Then comes the list of freedom parameters for which there are three for each freedom. The current value is defined, then the freedom axis, then the freedom limits. Rotational limits and values are in degrees and translational ones are in meters.

```
CHILDREN 3
  1 // Neck
  2 // Upper Left Arm
  3 // Upper Right Arm
```

The list of child indices are defined here

```
ENDMESH
```

Mandatory ending of the segment information

`ENDFIGURE`

Mandatory ending of the figure information

7.3.3 MESH Format

The MESH format is the easiest format I have found for tetrahedral meshes. It is the default format for the free Internet program MEDIT available from http://www.ann.jussieu.fr/~frey/logiciels/medit.html. Pascal J. Frey created this format to replace the inadequacies of prior formats. I have found it necessary to add some key-words to extend the format capabilities. This format is composed of a single (binary or text) data file. Its structure is organized as a series of fields identified by keywords. The blanks, newlines or carriage returns, and tabs are considered as item separators. A comment line starts with the character # and ends at the end of the line. The comments are placed exclusively between the fields. The mesh file must start with the descriptor:

```
MeshVersionFormatted 1
Dimension 3
```

The other fields supported by MEDIT are either required or facultative. The required fields correspond to the geometry (*i.e.* the coordinates) and to the topology description (*i.e.* the mesh entities). In the following tables, the term v_i indicates a vertex number (*i.e.* the i^{th} vertex in the vertex list), e_i is an edge number, t_i is a triangle number, T_i is a tetrahedron number, and q_i is a quadrilateral number. Notice that the vertices coordinates are real numbers in single precision.

Table 5 Primitives in MESH Format

Keyword	Card.	Syntax	Range
Vertices	*np*	$x_i\ y_i\ z_i\ ref_i$	i:[1,*np*]
Edges	*ne*	$v^1_i\ v^2_i\ ref_i$	i:[1,*ne*], v^j_i:[1,*np*]
Triangles	*nt*	$v^j_i\ ref_i$	i:[1,*nt*], j:[1,3], v^j_i:[1,*np*]
Quadrilaterals	*nq*	$v^j_i\ ref_i$	i:[1,*nq*], j:[1,4], v^j_i:[1,*np*]
Tetrahedra	*ntet*	$v^j_i\ ref_i$	i:[1,*ntet*], j:[1,4], v^j_i:[1,*np*]
Hexaedra	*nh*	$v^j_i\ ref_i$	i:[1,*nh*], j:[1,8], v^j_i:[1,*np*]

The description of constrained entities or singularities follows. In particular, a corner (keyword `Corner`) is a C^0 continuity point (this type of item is necessarily a mesh vertex). By analogy, a `Ridge` is an edge where there is a C^0 continuity between the adjacent faces. The fields of type `Requiredxx` make it possible to specify any type of entity that must be preserved by the meshing algorithm.

Table 6 Preservation Adjectives in MESH Format

Keyword	Card.	Syntax	Range
Corners	*nc*	v_i	i:[1,*nc*], v_i:[1,*np*]
RequiredVertices	*nrv*	v_i	i:[1,*nrv*], v_i:[1,*np*]
Ridges	*nr*	e_i	i:[1,*nr*]
RequiredEdges	*nre*	e_i	i:[1,*nre*]

As mentioned above, it is also possible to specify normals and tangents to the surface. The normals (respectively tangents) are given as a list of vectors. The normal at a vertex, keyword `NormalAtVertices`, is specified using the vertex number and the index of the corresponding normal vector. The normal at a vertex of a triangle, `NormalAtTriangleVertices`, corresponds to the combination of the triangle number, the index of the vertex in the triangle and the index of the normal vector at this vertex. The keyword `NormalAtQuadrilateralVertices` is handled similarly. The tangent vectors are described in the same way.

Table 7 Optional Adjectives of Primitives in MESH Format

Keyword	Card.	Syntax	Range
Normals	*nn*	$x_i\ y_i\ z_i$	i:[1,*nn*]
Tangents	*nnt*	$x_i\ y_i\ z_i$	i:[1,*nnt*]
NormalAtVertices	*nv*	$v_i\ n_i$	i:[1,*nv*], v_i:[1,*np*],

			n_i:[1,*nn*]
NormalAtTriangleVertices	*ntv*	$t_i\ v_i\ n_i$	*i*:[1,*ntv*], t_i:[1,*nt*], v_i:[1,*np*], n_i:[1,*nn*]
NormalAtQuadrilateralVertices	*nqv*	$q_i\ v_i\ n_i$	*i*:[1,*nqv*], q_i:[1,*nq*], v_i:[1,*np*], n_i:[1,*nn*]
TangentAtEdges	*te*	$e_i\ v_i\ t_i$	*i*:[1,*te*], e_i:[1,*ne*], v_i:[1,*np*], t_i:[1,*nnt*]
***Colors**	*ncc p*	$r_i\ g_i\ b_i\ a_i$	*i*:[1,*ncc*], *p*:[3,4], a_i only for p=4
***ColorAtVertices**	*ncv*	$v_i\ c_i$	*i*:[1,ncv], v_i:[1,*np*], c_i:[1,*ncc*]
***NeighborAtTriangleEdges**	*nte*	$t^1_i\ f\ t^2_i$	*i*:[1,*nte*], *f*:[1,3], t^j_i:[1,*nt*]
***NeighborAtTetrahedronFaces**	*ntf*	$T^1_i\ f\ T^2_i$	*i*:[1,*ntf*], *f*:[1,4], T^j_i:[1,*ntet*]

* Keywords added for this research.

Finally, the data structure must end with the keyword: `End`. The new color keywords are handy for coloring the mesh. The neighbor keyword reduces the amount of time it takes to traverse a mesh. Knowing the neighbor at each face of a tetrahedron can

determine which tetrahedron to go when moving through a mesh. If there is no neighbor, then the tetrahedron is on the outside.

7.3.4 PLY Format

The Polygon File Format (PLY) is another popular meshing format but only for 2D (surface) meshes. It is also known as the Stanford Triangle Format. It can be ASCII or binary and contains almost all that is needed for triangulated meshes. The format is originally from Stanford and is documented with examples and source code there (http://graphics.stanford.edu/data/3Dscanrep). This research program has incorporated the ability to read PLY files for the figure animation.

7.3.5 C3D Format

The Coordinate 3D (C3D) file format is only used for storing the raw motion capture data. It is a good format and is the oldest in the business. It originally came in 1986 from Dr. Andrew Danis but is now part of a non-profit organization at http://www.c3d.org. The file format is a subset of a more general (ADTECH) format that is binary and stores both parameter information and raw data. The data can be stored in little-endian (Intel), big-endian (MIPS, PPC), and DEC binary formats for 16-bit integers and 32-bit floats. Both analog and digital information can be stored along with their descriptions and measurement units. All of the data that comes from Carnegie Melon Graphics Lab is stored in C3D format.

7.4 User's Guide to Program

The following sections describe the use of the application that was built for the research. The application was designed using the Xcode 1.5 integrated development envi-

ronment available for free from Apple and comes with the MacOSX 10.4 operating system. There are 45270 lines of code with a total McCabe's Cyclomatic Complexity Number of 8112. The application follows the recommended Macintosh human interface and is compatible with MacOSX 10.1 through 10.4.8 though only tested on 10.3 and 10.4. The engine that drives the calculations is written entirely in standard portable C++ and the graphics panes are written in OpenGL. The user interface is written in ObjectiveC that compliment the NIB files for Interface Builder. Everything except the user interface is portable. The code was compiled for the PowerPC G4 processor and was tested on a PowerBook G4.

7.4.1 Menu

The File Menu provides the basic file saving and retrieving capabilities some of which are available in the Options Pane as well. A picture of the File Menu open is in

Figure 28.

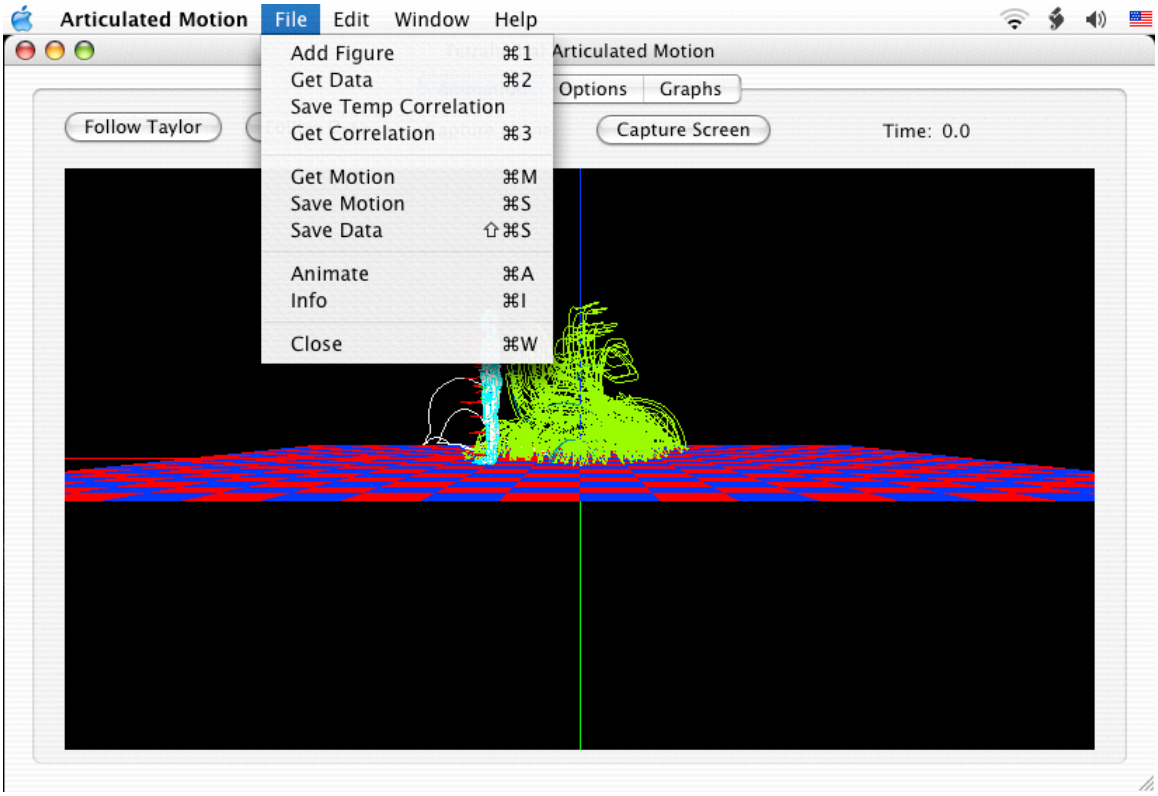

Figure 28 File Menu GUI

Add Figure

The Add Figure menu item (optionally use □-1) allows the user to add a figure to the graphics. A figure can be an Articulated Tetrahedral Mesh (ATM) file (as explained in Chapter 7.3.2); a MESH formatted file (both tetrahedral and triangulated); and a PLY formatted file. Only the ATM file can be used for the hierarchical analysis of the motion capture data. When this menu item is picked, a standard Open dialog is presented for choosing a file. When a file is chosen, the file is parsed and presented in the Animation Pane. The Options Pane text for the figure is filled out with the path of the file and the height of the figure is filled out.

Get Data

This menu item (optionally use □-2) will present a standard Open dialog to allow the user to choose a C3D motion capture data file. The selected file will associate the data with the currently active figure (see the Options Pane). Once chosen, the file is parsed and connected lists of data frames are filled out.

Save Temp Correlation

This menu item allows the user to save a temporary correlation text file associated with the C3D data file that was previously chosen. The text file can then be edited to allow a hierarchical correlation of the data markers with segments on the figure. The number of markers is stated on the first line (e.g. 41 WANDS). The marker set file name is stated on the second line (e.g. MARKERSET vicon512.txt). Then comes a list of marker correlations (e.g. Jim:RTHI = RTHI Right thigh). The left side of the equals sign is the single word that the marker is labeled as in the data (e.g. Jim:RTHI). The right side has a single word that corresponds to a marker in the above-mentioned marker set file followed by a free-form description.

Get Correlation

This menu item (optionally use □-3) presents the user with an Open dialog to retrieve a correlation file. The file is parsed and the left side is looked up in the C3D data and the right side is looked up in the marker set file. The marker set file determines which segment to attach the marker and where on the segment. If the marker is not to be used in the correlation, just put some uncorrelated name (e.g. Unknown) on the right hand side.

Get Motion

This menu item retrieves motion files that have been previous saved or manually created. Motion files allow the user to create motion of an articulated figure based on a Taylor expansion of any or all degrees of freedom.

Save Motion

This menu item will save the Taylor expansions that were created when the "Analyze" button is pressed in the Graphs Pane.

Save Data

This menu item will save the joint angles that were created when the "Analyze" button is pressed in the Graphs Pane.

Animate

This menu item will start the animation.

Info

This menu item will enable some graphics hardware information to be displayed on the Animation Pane.

Close

This menu item will close the window.

7.4.2 Options Pane

The Options plane contains all of the figure, data, and animation options available. On the right are all of the files that are used in the current analysis. The “Marker Set Dir” is the directory that all marker set files are located for the correlation. The “Figure” file is the currently active articulated model. The “Motion Data” file is the current C3D being analyzed. The “Correlate Data” file is the correlation file that binds the data with the hierarchical figure. The “replace” checkbox will replace the current figure with the opened figure if it is on. Otherwise the opened file will be added to the scene. The “ft->m” checkbox will convert the motion data from feet to meters if on. The “x2” button will multiply the sampling rate by two. The last two buttons are there for convenience since the data does not always specify the correct units or sampling rate. Once the data is read in, the height and sampling rate text can be edited to whatever size and the internal data will be updated. The same goes for the Figure information. The height of the figure can be edited.

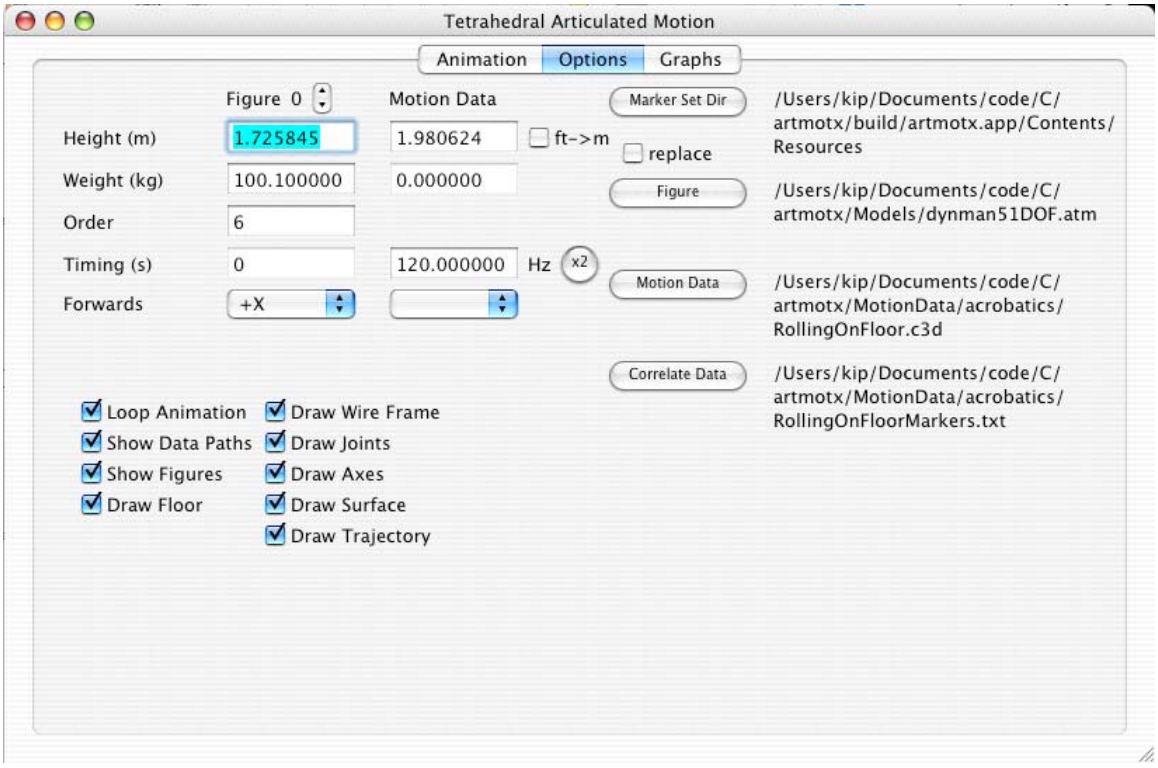

Figure 29 Options Pane GUI

Drawing Options

The many checkboxes allow the selection of various things to be drawn in the Animation Pane. "Loop Animation" will make the animation repeat when the maximum time of the data is reached. "Show Data Paths" will display the paths taken for each marker in the data. The paths will be colored lighter if the marker has been correlated to the figure. "Show Figures" will display all figures that have been read in. "Draw Floor" will ... draw the floor?! The other checkboxes draw extra little goodies on the figure.

7.4.3 Animation Pane

This pane is where all of the action happens. Select this pane once all of the files and options are set in the Options Pane. The time of the animation is displayed in the upper right in seconds. The Capture Screen button will take an exact snapshot of the

OpenGL window and allow the user to save it as a TIFF file. The TIFF file is a 32-bit ABGR8888 color file. The "Capture Movie" checkbox is similar except that it saves many TIFF files at 0.1 second intervals for the entire loop. The "Follow Data" starts the animation process that draws the moving figure and the skeleton in the data. There are options for panning the scene around. If the Command key (□) is held down while the mouse button is held down and the screen is dragged by the mouse, then the scene is dragged up/down or left/right. If the Option key is held down with the mouse button, the user can zoom in or out of the scene with the mouse. With just the mouse button and the mouse dragging, the user can rotate the scene around the (0,0,0) point.

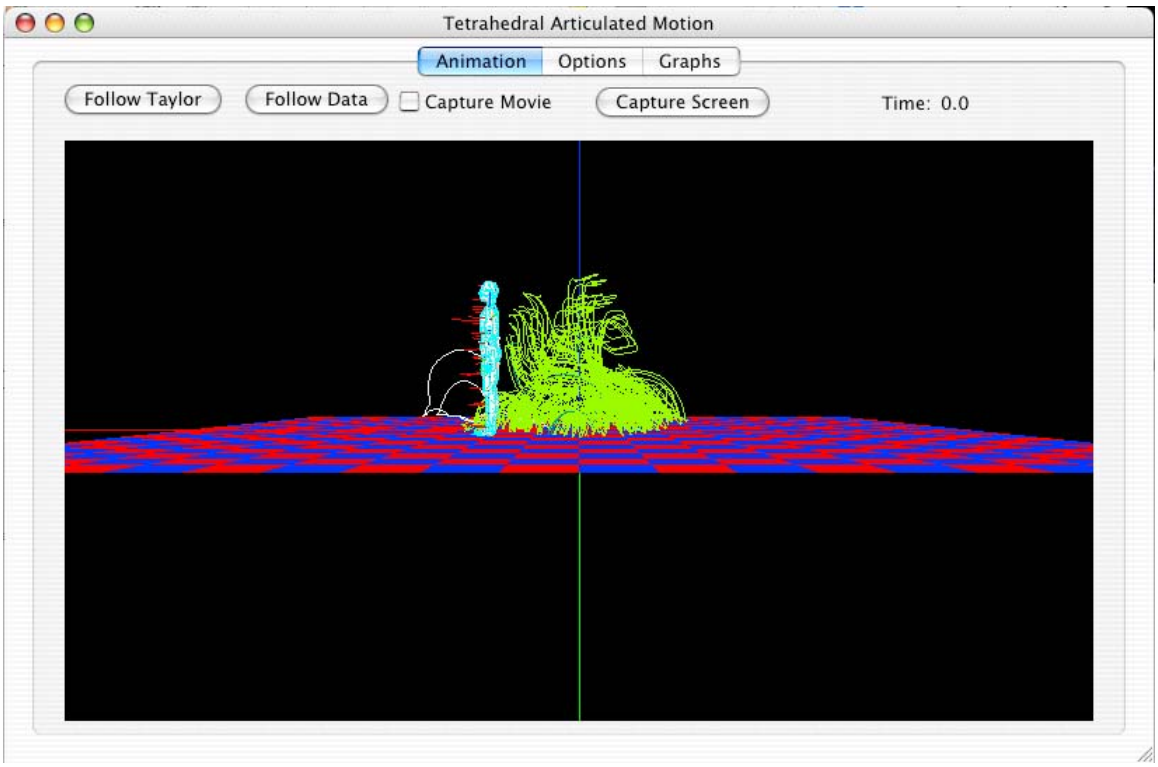

Figure 30 Animation Pane GUI

7.4.4 Graphs Pane

This pane was added for analysis of joint angles was just a visualization aid for research. The analyze button currently tries to determine the angles which are needed to rotate in order to follow the current dataset. Once the analysis is done, a joint can be picked from the pop-up. When the user slides the horizontal time bars left and right, a new Taylor expansion will be generated for the time window. Taylor expansions help in analyzing the predictive-ness of the angle paths.

Figure 31 Graphs Pane GUI

7.5 Programmer's Reference

7.5.1 Class Diagrams

The articulated figure designed for this research is for general-purpose articulated modeling. The original attempt was to use it for physical based calculations. With the advent of the Minimum Variance Method, the model has been resigned to a simple organizational tool for the hierarchical tree traversals. The entire implementation is explained here for future use. The C++ object model is set up in a hierarchical manner for the articulated figure. An articulated figure (ArtFigure.cpp) is made of segments (Segment.cpp). A segment is a rigid body (RigidBody.cpp) that is linked to another by a joint. The rigid body is a shape that can be translated and rotated, but not molded. The rigid body is made of a mesh of tetrahedrons (TetrahedralMesh.cpp). The mesh is a face-connected list of tetrahedrons. Only faces on the surface have no connected tetrahedrons and are available for drawing. A tetrahedron (Tetrahedron.cpp) is a four sided figure with four vertices assigned. Each side of the tetrahedron is a triangle (Triangle.cpp). Since adjacent tetrahedrons share vertices, the tetrahedrons will contain only references to its points. All of the points for the tetrahedral mesh will be stored in a linear array inside the TetrahedralMesh instance. This allows easy access to the points for quick translations and rotations without duplication of effort. Figure 32 is a UML Diagram describing the relationships between objects.

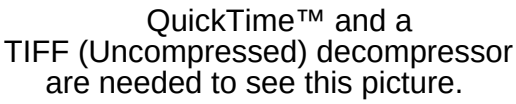

Figure 32 UML Diagram of Articulated Figure

7.6 C++ Implementations

This section contains the crucial implementations for the novel algorithms in this thesis. It starts with the implementation of the UGDK, followed by IGDK. The next algorithm involves the collecting of the raw data and turning them into relative positions. The calculation of the rotation point follows.

7.6.1 Unbiased Generalized Delogne-Kása Method

This snippet of C++ code implements the core of the Unbiased Generalized Delogne-Kása Method. Some of the standard linear algebra operations are hidden inside of the Vector and Matrix classes.

```
void VectorDataSet::init( const Vector * v,
                          unsigned int n,
                          double eps )
{
  unsigned int i;
  Vector d;
  double d2;

  if( n == 0 )
  {
    cerr << "ERROR: VectorDataSet::init"
         << " no data to init set"
         << endl;
    return;
  }
  init( v->numElements );
  for( i=0; i<n; i++ )
  {
    moment1 += v[i];
  }
  moment1 /= n;
  for( i=0; i<n; i++ )
  {
    d = v[i]-moment1;
    covariance += d.SquareTranspose();
    d2 = d*d;
    moment3 += d*d2;
    moment2 += d2;
  }
```

```
  moment2 /= n;
  if( n > 1 )
  {
    covariance /= n-1;
    moment3 /= n-1;
    covariance -= eps*eps; // compensate for bias
  }
  else
  {
    covariance = 0.0;
    moment3 = 0.0;
  }
  Matrix decomp = covariance.CholeskyDecomposition();
  covarianceInverse = decomp.CholeskyInverse();
  center = moment1 + 0.5*decomp.CholeskyBacksubstitution(moment3);
  centerDiff = 1.0e20;
  difMoment1 = 1.0e20;
  radius = sqrt(((n-1.0)/n)moment2+(center-moment1)*(center-moment1));
  numData = n;
  numCenterData = 1;
  centerAvg = center;
}
```

7.6.2 Incrementally Improved Generalized Delogne-Kása

This code snippet adds a vector to the data set and improves the various values that were previously calculated in Section 7.6.1.

```
void VectorDataSet::operator+=( const Vector& a )
{
  unsigned int i;
  Vector b,d;
  double d2,q1,qn1,q2,qp1;
  Matrix m;
  if( numData > 0 )
  {
    difMoment1 += a - lastValue;
  }
  lastValue = a;
  q1 = (numData/(numData+1.0)); // < 1
  qn1 = (numData-1.0)/numData;  // < 1
  q2 = (numData+2.0)/numData;   // > 1
  qp1 = (numData+1.0)/numData;  // > 1
  difMoment1 = (a-moment1)/(numData+1.0);
  moment1 += difMoment1; // average
  d = a-moment1;
  d2 = d*d
  b = covarianceInverse*d;
  moment2 += (qp1*d2 - moment2)/(numData+1.0);
  covarianceInverse += (covarianceInverse-(numData/((numData-
        1.0)*q1+(d*b)))*b.SquareTranspose())/(numData-1.0);
```

```
  covariance += (qp1*d.SquareTranspose() - covariance)/numData;
  centerDiff = (covarianceInverse*d)*(d*(a-center) -
        0.5*(moment2+d2))/(numData+1.0);
  center += centerDiff;
  centerAvg += (center-centerAvg)/(numCenterData+1.0);
  moment3 += ((-2.0*covariance-moment2*qp1+d2*(qp1*q2))*d-
        moment3)/numData;
  radius = sqrt(qn1*moment2+(center-moment1)*(center-moment1));

  for( i=0; i<dimension; i++ )
  {
    if( a[i] < minimum[i] )
    {
      minimum[i] = a[i];
    }
    if( a[i] > maximum[i] )
    {
      maximum[i] = a[i];
    }
  }
  numData++;
  numCenterData++;
}
```

7.6.3 Collecting the Raw Data for Rotation Point

This function collects the raw data and creates an array of positions relative to the parent segment. The inputs are the time window beginning and end. The outputs are the collection of relative points and whether the function succeeded.

```
bool Segment::CollectRelativeDataPoints( Units::second t1,
                                         Units::second t2 )
{
  unsigned int i,k,n=1,n1=n;
  double maxLen = 0;
  PositionVector dataCenter2;
  Matrix dataAxes2;
  Units::second t3 = 0,deltaT,lastTime;
  EventFrame::ElementType evl[1] = {NULL};
  Event * e[n];
  EventFrame events;
  unsigned int best = 0;
  Vector pi(3);

  // the parent must exist and have data
  if( numCorrelated == 0 ||
      parent == NULL )
  {
    return false;
  }
  // find the farthest data point
  for( i=0; i<correlatedFrame.numElements; i++ )
```

```
  {
    for( k=i+1; k<correlatedFrame.numElements; k++ )
    {
      if( correlatedFrame.lengths[i][k] > maxLen )
      {
        best = i;
        maxLen = correlatedFrame.lengths[i][k];
      }
    }
  }
  evl[0] = correlatedFrame[best];
  if( evl[0] == NULL )
  {
    return false;
  }
  events.setEvents(evl,n);
  deltaT = events.minTimeBetweenEvents;
  if( t1 < events.minTime )
  {
    t1 = events.minTime;
  }
  if( t2 > events.maxTime )
  {
    t2 = events.maxTime;
  }
  t3 = t1 - deltaT; // so first frame is retrieved
  pathToFollow.RemoveAll();
  while( events.NextAbsoluteStableFrame(n1,e,t3) && (t3 <= t2) )
  {
    if( !parent->GetSameDataAxes( t3, dataAxes2, dataCenter2 ) )
    {
      // bad frame, continue to next
      continue;
    }
    for( i=0; i<n; i++ )
    {
      if( e[i] != NULL )
      {
        pi = (e[i]->position - dataCenter2)*dataAxes2;
        pathToFollow.InsertEvent(pi,
                                 e[i]->time,
                                 e[i]->accuracy,
                                 e[i]->time_accuracy);
      }
    }
  }
  return true;
}
```

7.6.4 Rotation Point Calculation of Segment

This function takes the collection of relative data points and calculates the relative rotation point in the data. It also calculates the best fit planar normal of the data which is needed for marker sets of one marker on a segment. It returns the relative rotation point.

```
PositionVector Segment::GetDataRotationPoint()
{
  Event * e = NULL;
  unsigned int j;
  Vector p(3),pi(3),p3(3),mean(3),pip(3),pir(3);
  Vector b,r;
  Matrix A(3,3),P2(3,3),a,Ainv;
  //Vector::ElementType p2=0.0,q,L=0.0;
  double err(0);
  static const int N = 100000;
  int n;
  VectorDataSet dataSet;
  Vector vs[N];

  kalmanFilterFailed = true;
  if( CollectRelativeDataPoints() )
  {
    n = 0;
    e = pathToFollow.beginning;
    while( e && n < N )
    {
      vs[n] = e->position;
      e = e->next;
      n++;
    }
    dataSet.init(vs,n,pathToFollow.beginning->accuracy);
    while( e && !dataSet.sphereHasConverged(0.01) )
    {
      dataSet += e->position;
      e = e->next;
      n++;
    }
    if( n > 0 )
    {
      j = n;
      r = dataSet.center;
      double cn = dataSet.covariance.ConditionNumber();
      Vector v = dataSet.covariance.NullSpace(1.000001/cn).Column(0);
      relativeDataParentOneDOFaxis = v;
      Vector r1 = r + v*((p-r)*v);

      if( cn > 10000.0 ) // project onto plane
      {
         r = r1;
      }
    }
```

```
  }
  return PositionVector(r);
}
```

7.6.5 Constants Calculation of Hierarchical Articulated Data

This function calculates the constants necessary to draw the skeleton at any time frame. There are no inputs except for the available raw data. The outputs are the constants that are set for each segment. This function must be called from the root segment or the hierarchy will break.

```
void Segment::SetDataRelativeStuff() // call from root
{
  Matrix dataAxes = Matrix::Identity(3);
  PositionVector dataCenter,rot,c,v;
  unsigned int i;
  Units::second t = 0;
  Event * e[numCorrelated];

  unsigned int n = 0,n1=numCorrelated;
  if( correlatedFrame.NextFrame(n1,e,t) )
  {
    c = 0.0;
    for( i=0; i<numCorrelated; i++ )
    {
      if( e[i] != NULL )
      {
        c += e[i]->position;
        n++;
      }
    }
    if( n > 0 )
    {
      c /= n;
    }
  }
  if( parent )
  {
    // calculate rotation point from available data
    relativeDataParentRotationPoint = GetDataRotationPoint();
    if( parent->numCorrelated == 0 ) // set previous segment's
    {
      parent->relativeDataParentRotationPoint =
              relativeDataParentRotationPoint;
    }
    t = 0.0;
    if( parent->GetDataAxes(t,dataAxes,dataCenter) )
    {
      rot = dataCenter + dataAxes*relativeDataParentRotationPoint;
```

```
      v = dataAxes*relativeDataParentOneDOFaxis;
      relativeDataJointAxesZero =
                  dataAxes.Transpose()*Matrix::Identity(3);
    }
    else
    {
      cerr << "No parent data axes for " << name << endl;
      rot = PositionVector(0.0,0.0,0.0);
      relativeDataJointAxesZero = Matrix::Identity(3);
    }
    t = 0.0;
    // set segment's relative stuff to markers
    if( GetDataAxes(t,dataAxes,dataCenter) )
    {
      relativeDataBodyAxes = dataAxes.Transpose()*Matrix::Identity(3);
      relativeDataJointAxes = dataAxes.Transpose()*Matrix::Identity(3);
      relativeDataRotationPoint = (rot - dataCenter)*dataAxes;
      relativeDataCentroid = (c - dataCenter)*dataAxes;
      relativeDataOneDOFaxis = dataAxes.Transpose()*v;
    }
    else
    {
      cerr << "No self data axes for " << name << endl;
    }
  }
  else // root
  {
    // no rotation point so just get the centroid of data
    t = 0.0;
    if( !GetDataAxes(t,dataAxes,dataCenter) )
    {
      cerr << "Damn, root really needs >= 3 correlated markers"
           << endl;
    }
    relativeDataCentroid = (c - dataCenter)*dataAxes;
    relativeDataRotationPoint = relativeDataCentroid;
    relativeDataOneDOFaxis = DirectionVector(1.0,0.0,0.0);
    relativeDataParentOneDOFaxis = DirectionVector(1.0,0.0,0.0);
    relativeDataParentRotationPoint = PositionVector(0.0,0.0,0.0);
    relativeDataJointAxes = dataAxes.Transpose()*Matrix::Identity(3);
    relativeDataJointAxesZero=dataAxes.Transpose()*Matrix::Identity(3);
    relativeDataBodyAxes = dataAxes.Transpose()*Matrix::Identity(3);
  }
  //cout << "DataGrouping " << CheckDataGrouping() << endl;
  for( i=0; i<numChildren; i++ )
  {
    children[i]->SetDataRelativeStuff(); // recursive
  }
}
```

7.6.6 Calculation of fixed axes of data

This function calculates the center and the three axes for a segment based upon the previously calculated constants and the current raw data. The input t is the time at which to extract the data. The function outputs the axes in the form of a 3x3 matrix and the center in absolute coordinates if successful. If not successful, the function returns false.

```
bool Segment::GetDataAxes( Units::second& t,
                           Matrix& dataAxes,
                           PositionVector& dataCenter ) const
{
  Event * e[3] = {NULL,NULL,NULL};
  Units::second oldT = t;

  // a segment without axes, use parent's
  if( parent && axesFrame.numElements == 0 )
  {
    if( !parent->GetDataAxes( t, dataAxes, dataCenter ) ) // recursive
    {
      return false;
    }
  }
  // root and all segments with many points gets in here
  else if( axesFrame.numElements == 3 )
  {
    unsigned int n = 3;
    if( !axesFrame.NextAbsoluteFrame(n,e,t) )
    {
      if( t < axesFrame.maxTime )
      {
        cerr << "Missing events for " << jointName
             << " t " << t << endl;
      }
      return false;
    }
    dataCenter = e[0]->position;
    dataAxes = Matrix::Axes(dataCenter,e[1]->position,e[2]->position);
  }
  else if( !parent ) // segment must have parent beyond this point
  {
    return false;
  }
  else if( axesFrame.numElements == 2 ) // need parent's stuff set
  {
    unsigned int n = 2;
    Matrix pDataAxes;
    PositionVector pDataCenter;
    if( !axesFrame.NextAbsoluteFrame(n,e,t) )
```

```
    {
      return false;
    }
    // recursive
    if( !parent->GetSameDataAxes( t, pDataAxes, pDataCenter ) )
    {
      return false;
    }
    dataCenter = pDataCenter+pDataAxes*relativeDataParentRotationPoint;
    dataAxes = Matrix::Axes(dataCenter,e[0]->position,e[1]->position);
  }
  else if( axesFrame.numElements == 1 ) // need parent's stuff set
  {
    unsigned int n = 1;
    Matrix pDataAxes;
    PositionVector pDataCenter,r2;

    if( !axesFrame.NextAbsoluteFrame(n,e,t) )
    {
      return false;
    }
    // recursive
    if( !parent->GetSameDataAxes( t, pDataAxes, pDataCenter ) )
    {
      return false;
    }
    dataCenter = pDataCenter+pDataAxes*relativeDataParentRotationPoint;
    r2 = dataCenter + pDataAxes*relativeDataParentOneDOFaxis;
    dataAxes = Matrix::Axes(dataCenter,r2,e[0]->position);
  }
  else
  {
    return false;
  }
  return true;
}
```

7.6.7 Drawing Rotation Points with Constants of Motion

This function is called after the rotation points have been calculated and the hierarchical skeleton has been defined. It retrieves the current raw 3D positional data and calculates where the rotation points and axes are supposed to be based on the previous analysis. The radius input is for drawing a sphere around the current raw data. The t input is the time at which the data is retrieved. There are no results except those that are drawn on the OpenGL window.

```
void Segment::DrawNearestEvents( double radius,
                                 Units::second t )
{
  unsigned int i;
  Matrix dataAxes,dataAxes0,dataAxes1;
  PositionVector dataCenter,dataCenter0,dataCenter1,rot,rot1,pos,ep;

  if( t<lastTime )
  {
    count = 0;
  }
  lastTime = t;

  Color::Green.glColor();
  correlatedFrame.Draw(radius*0.5,t); // draw dots for data
  if( parent && parent->GetDataAxes( t, dataAxes0, dataCenter0 ) )
  {
    rot = dataCenter0 + dataAxes0*relativeDataParentRotationPoint;
  }
  else if( parent != NULL )
  {
    return;
  }
  t -= 0.01;
  if( GetDataAxes( t, dataAxes, dataCenter ) )
  {
    if( parent == NULL )
    {
      rot = dataCenter + dataAxes*relativeDataCentroid;
    }
  }
  else
  {
    return;
  }
  Color::White.glColor();
  glBegin(GL_LINES);
  for( i=0; i<numChildren; i++ )
  {
    if( children[i]->numCorrelated ||
        children[i]->numChildren > 0 )
    {
      rot.glVertex();
      rot1 = datacenter + dataAxes*children[i]->
                                  relativeDataParentRotationPoint;
      rot1.glVertex();
    }
  }
  if( numCorrelated > 0 &&
      numChildren == 0 )
  {
    rot.glVertex();
    (dataCenter + dataAxes*relativeDataCentroid).glVertex();
  }
  glEnd();
  if( numChildren == 0 ) // end effector
  {
```

```
    correlatedFrame.DrawLines(t);
  }
  count++;
}
```

Chapter 8 BIBLIOGRAPHY

[1] AAMAS. *ACUMEN: Amplifying Control and Understanding of Multiple ENtities*, Bologna, Italy, July 15-19 2002. ACM 1-58113-480-0/02/0007.

[2] J.K. Aggarwal, Q. Cai, and W. Liao, "Nonrigid motion analysis: articulated and elastic motion" Computer Vision and Image Understanding 70, no. 2 (May 1998) : 142-56.

[3] J. M. Allbeck and N. I. Badler. Avatars á la snow crash. *Computer Animation*, pages 19–24, June 1998.

[4] J. M. Allbeck and N. I. Badler. Consistent communication with control. Workshop on Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents, Autonomous Agents, 2001.

[5] J. M. Allbeck and N. I. Badler. Toward representing agent behaviors modified by personality and emotion. *Embodied Conversational Agents at AAMAS'02*, July 15-19 2002.

[6] K. Ashida, S.-J. Lee, J. M. Allbeck, H. Sun, N. I. Badler, and D. Metaxas. Pedestrians: Creating agent behaviors through statistical analysis of observation data. *Proc. Computer Animation*, 2001.

[7] J. Assa, Y. Caspi, D. Cohen-Or, "Action Synopsis : Pose selection and illustration" *ACM Transaction on Graphics (SigGraph 2005)* 24(3) (July 2005) : 667-676.

[8] C. Babski and D. Thalmann. Real-time animation and motion capture in web human director. *VRML*, 2000.

[9] N. Badler, J. Allbeck, L. Zhao, and M. Byun. Representing and parameterizing agent behaviors. January 4 2002.

[10] N. Badler, M. Costa, L. Zhao, and D. Chi. To gesture or not to gesture: What is the question? In *Proceedings of Computer Graphics International*, pages 3–9, Geneva, Switzerland, June 2000. IEEE CS.

[11] N. I. Badler. *Dance Technology: Current Applications and Future Trends*, chapter A Computational Alternative to Effort Notation, pages 23–44. National Dance Association, va edition, 1989.

[12] N. I. Badler, K. H. Manoochehri, and D. Baraff. Multi-dimensional input techniques and articulated figure positioning by multiple constraints. *Workshop on Interactive 3D Graphics*, pages 151–169, October 23 1986.

[13] N. I. Badler. Animation 2000++. *IEEE Computer Graphics and Applications*, pages 28–29, January/February 2000.

[14] N. I. Badler and J. M. Allbeck. *Deformable Avatars*, chapter Towards Behavioral Consistency in Animated Agents, pages 191–205. Kluwer Academic Publishers, 2001.

[15] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. *IROS*, 1998.

[16] S. Bandi and D. Thalmann. A configuration space approach for efficient animation of human figures. *NRAM*, 1997.

[17] M. Berman and D. Culpin. The statistical behaviour of some least squares estimators of the centre and radius of a circle. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(2):183–196, 1986.

[18] R. Bindiganavale, W. Shuler, J. M. Allbeck, N. I. Badler, A. K. Joshi, and M. Palmer. Dynamically altering agent behaviors using natural language instructions. *Autonomous Agents*, pages 293–300, June 2000.

[19] A. Bloomfield, Y. Deng, J. Wampler, P. Rondot, D. Harth, M. McManus, and N. I. Badler. A taxonomy and comparison of haptic actions for disassembly tasks. *Proceedings of the IEEE Virtual Reality Conference*, pages 225–231, March 2003.

[20] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella. The process of motion capture: Dealing with the data. In D. Thalmann and M. van de Panne, editors, *Eurographics Animation and Simulation*, pages 3–18, Wien, Sept 1997. Springer-Verlag.

[21] B. Bodenheimer, C. Rose, S. Rosenthal, J. Pella, "The Process of Motion Capture: Dealing with the Data" *Eurographics* CAS (Sept. 1997): 3-18

[22] D.A. Bolt, "Two Stage Control for High Degree of Freedom Articulated Figures" Ph.D. thesis for Computer Science, University of Colorado in Colorado Springs, (2000).

[23] R. Boulic, P. Fua, L. Herda, M. Silaghi, J.-S. Monzani, L. Nedel, and D. Thalmann. An anatomic human body for motion capture. In *EMMSEC*, pages 1–7, Bordeaux, France, September 1998.

[24] R. Boulic, Z. Huang, N. Magnenat-Thalmann, and D. Thalmann. Goal oriented design and correction of articulated figure motion with the track system. 1994.

[25] R. Boulic and R. Mas. Hierarchical kinematic behaviors for complex articulated figures. 1997.

[26] R. Boulic, R. Mas, and D. Thalmann. Position control of the center of mass for articulated figures in multiple support. 1995.

[27] R. Boulic, R. Mas, and D. Thalmann. Interactive identification of the center of mass reachable space for an articulated manipulator. 1997.

[28] R. Boulic and D. Thalmann. Combined direct and inverse kinematic control for articulated figure motion editing. *CGF*, 1992.

[29] R. Boulic and D. Thalmann. Complex character positioning based on a compatible flow model of multiple supports. *IEEE Transactions on Visualization and Computer Graphics*, 3(3), July–September 1997.

[30] L. Bretzner and T. Lindeberg. Qualitative multiscale feature hierarchies for object tracking. *Journal of Visual Communication and Image Representation*, 11:115–129, 2000.

[31] M. Burr, A. Cheng, R. Coleman and D. Souvaine. "Transformations and Algorithms for Least Sum of Squares Hypersphere Fitting." *16th Canadian Conference on Compu-*

tational Geometry, 2004, pp 104-107.

[32] J. Chai, J. Hodgins "Performance Animation from Low Dimensional Control-Signals" *ACM Transaction on Graphics (SigGraph 2005)* 24(3) (July 1005) : 686-696.

[33] L. Y. Chang, N. S. Pollard, "Constrained least-squares optimization for robust estimation of center of rotation", *Journal of Biomechanics*, (accepted 7 May 2006)

[34] N. Chernov and C. Lesort. "Least squares fitting of circles". *Journal of Mathematical Imaging and Vision*, 23:239–252, 2005.

[35] D. Chi, M. Costa, L. Zhao, and N. I. Badler. Emote model for effort and shape. *Proceedings of SIGGRAPH*, 2000.

[36] K. Choi, S. Park, H. Ko. Processing motion capture data to achieve positional accuracy. *Graphical Models and Image Processing*, 61(5):260–273, September 1999.

[37] K. Choi and H. Ko. On-line motion retargetting. *Journal of Visualization and Computer Animation*, 11(5):223–235, June 2000.

[38] M.-H. Choi and J. F. Cremer. Interative manipulation of articulated objects with geometry awareness. *Proceedings of the IEEE International Conference on Robotics & Automation*, pages 592–598, May 1999.

[39] S. Chopra-Khullar and N. I. Badler. Where to look? automating attending behaviors of virtual human characters. 1999.

[40] C. A. Corral and C. S. Lindquist. On implementing Kása's circle fit procedure. *IEEE Transactions on Instrumentation and Measurement*, 47(3):789–795, June 1998.

[41] P. Delogne. "Computer Optimization of Deschamps' Method and Error Cancellation in Reflectometry," Proceedings of the IMEKO Symposium on Microwave Measurements, *Budapest, Hungary,* May 1972, pp. 117-129.

[42] Q. Delamarre and O. Faugeras. 3D Articulated Models and Multiview Tracking with Physical Forces. *Computer Vision and Image Understanding*, 81:328–357, 2001.

[43] E-Factory. *Jack human modeling and simulation - Virtual people, virtual places, real solutions*.

[44] R. M. Ehrig, W. R. Taylor, G. N. Duda, M. O. Heller, "A survey of formal methods for determining the centre of rotation of ball joints", Journal of Biomechanics (accepted 3 October 2005)

[45] P.J. Frey "MEDIT: An interactive mesh visualization software" Rapport technique n°0253, December 3, 2001, 41 pages.

[46] P. Fua, R. Plänkers, and D. Thalmann. From synthesis to analysis: Fitting human animation models to image data. *CGI*, 1999.

[47] W. Gander, G.H. Golub, R. Strebel. "Least-Squares Fitting of Circles and Ellipses". *BIT Numerical Mathematics* 34, Springer 1994, pp 558-578.

[48] C. R. Henderson. Estimation of variance and covariance components. *Biometrics*, 9(2):226–252, June 1953.

[49] L. Herda, P. Fua, R. Plänkers, R. Boulic, and D. Thalmann. Using skeleton-based tracking to increase the reliability of optical motion capture. *HMS*, 2001.

[50] L. Herda, P. Fua, R. Plankers, R Boulic, D. Thalmann, "Skeleton-based motion capture for robust reconstruction of Human Motion", Computer Graphics Lab (LIG) EPFL:Lausane, Switzerland (2000).

[51] R.J. Holt, T.S. Huang, and A.N. Netravali, R.J. Qian, "Determining articulated motion from perspective views: a decomposition approach" *Pattern Recognition*: 30, no. 9, (1997): 1435.

[52] S. Jung and K. Wohn. Tracking and motion estimation of the articulated object: a hierarchical Kalman filter approach. *Real-Time Imaging*, 3:415–432, 1997.

[53] T. Jung. An algorithm with logarithmic time complexity for interactive simulation of hierarchically articulated bodies using a distributed-memory architecture. *Real-Time Imaging*, 4:81–96, 1998.

[54] P. E. Jupp and J. T. Kent. Fitting smooth paths to spherical data. *Applied Statistics*, 36(1):34–46, 1987.

[55] K. Kanatani. "Cramér-Rao lower bounds for curve fitting." *Graphical Models and Image Processing*, 60(2):93–99, March 1998.

[56] K. Kanatani. "Ellipse Fitting with Hyperaccuracy." *IEICE Transactions on Information and Systems*, E89-D (10): 2653-2660, October 2006.

[57] I. Kása. A circle fitting procedure and its error analysis. *IEEE Transactions on Instrumentation and Measurement*, 25:8–14, March 1976.

[58] C. Kervrann and F. Heitz. A hierarchical markov modeling approach for the segmentation and tracking of deformable shapes. *Graphical Models and Image Processing*, 60(3):173–195, May 1998.

[59] P. Kiriazov and H. Ko. On control design in simulation of human motion. 1998.

[60] A. G. Kirk, J. F. O'Brien, and D. A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2005.

[61] A. G. Kirk and O. Arikan. Real-time ambient occlusion for dynamic character skins. 2007.

[62] J. K. Knight. Probability density function determination from ordered statistics. Masters thesis, California State University, Fullerton, August 1995.

[63] J. K. Knight and S. K. Semwal, Fast Skeleton Estimation from Motion Capture Data using Generalized Delogne-Kása method. In *15th International Conference in Central Europe on Computer Graphics Visualization and Computer Vision*. Full Papers Proceedings of WSCG, ISBN 978-80-86943-01-5, Feb 2007.

[64] H. Ko, J. Cremer, "VRLOCO: Real-Time Human Locomotion from Positional Input Stream" *Presence* 5(4) (1996): 367-380.

[65] H. Ko and N. I. Badler. Animating human locomotion in real-time using inverse dynamics, balance and comfort control. *IEEE Computer Graphics and Applications*,

16(2):50–59, March 1996.

[66] L. Kovar, M. Gleicher, F. Pighin, "Motion Graphs" *Proceedings of SIGGRAPH 2002*: 473-482.

[67] H. K. Kwangjin Choi, Sanghyun Park. Processing motion capture data to achieve positional accuracy. *Graphical Models and Image Processing*, 61(5):260–273, September 1999.

[68] S. P. Lee, J. B. Badler, and N. I. Badler. Eyes alive. *Proceedings of ACM SIGGRAPH*, 21(3):637–644, July 2002.

[69] J. Lee, S.Y. Shin, "A Hierarchical Approach to Interactive Motion Editing for Human-like Figures" *Proceedings of SIGGRAPH 1999*.

[70] J. Lee, J. Chai, P.S.A Reitsma, "Interactive Control of Avatars Animated with Human Motion Data" *Proceedings of SIGGRAPH 2002*: 491-500.

[71] C.K. Liu, Z. Popović, "Synpaper of Complex Dynamic Character Motion from Simple Animation" *Proceedings of SIGGRAPH 2002*: 408-416.

[72] Z. Lui, S. Gortler, M. Cohen, "Hierarchical Spacetime Control" Computer Graphics (*SIGGRAPH 1994 Proceedings*).

[73] G. Lukács, A.D. Marshall, R.R. Martin, "Geometric Least-Squares Fitting of Spheres, Cylinders, Cones, and Tori" RECCAD Deliverable Documents 2 and 3 Copernicus Project No. 1068 Reports on basic geometry and geometric model creation, etc. Edited by Dr. R. R. Martin and Dr. T. Varady Report GML 1997/5, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1997.

[74] N. Magnenat-Thalmann and D. Thalmann. Motion control of synthetic actors: An integrated view of human animation. *MCAAF*, 1989.

[75] F. Marina, E. Stella, A. Branca, N. Veneziani, and A. Distante. Specialized hardware for real-time navigation. *Real-Time Imaging*, 7:97–108, 2001.

[76] E. G. Miller. A new class of entropy estimators for multi-dimensional densities. *International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[77] T. Molet, R. Boulic, and D. Thalmann. A real time anatomical converter for human motion capture. *EGCAS*, 1996.

[78] J.-S. Monzani, P. Baerlocher, R. Boulic, and D. Thalmann. Using an intermediate skeleton and inverse kinematics for motion retargeting. *Eurographics*, 19(3), 2000.

[79] A. Moreno, J. Umerez, and J. Ibañez. Cognition and life: The autonomy of cognition. pages 107–129, 1997.

[80] Motion Lab Systems, Inc. "C3D Format User Guide", PDF document at http://www.motion-labs.com, Motion Lab Systems, Baton Rouge, LA, (2003) (98 pages)

[81] M. Muller, T. Roder, M. Clausen, "Efficient Content-Based Retrieval of motion capture Data" *ACM Transaction on Graphics (SigGraph 2005)* 24(3) (July 2005) : 677-685.

[82] Y. Nievergelt. Perturbation analysis for circles, spheres, and generalized hyperspheres fitted to data by geometric total least-squares. *Mathematics of Computation*, 73(245):169–180, August 2003.

[83] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *SIGGRAPH Computer Graphics Proceedings, Annual Conference Series*, 2002.

[84] J. F. O'Brien, R. E. Bodenheimer Jr., G. J. Brostow, and J. K. Hodgins. *Automatic Joint Parameter Estimation from Magnetic Motion Capture Data*, pages 53–60, Montreal, Quebec, Canada, May 15-17 2000. Graphics Interface.

[85] J. F. O'Brien, P. R. Cook, and G. Essl. Synthesizing sounds from physically based motion. *SIGGRAPH Computer Graphics Proceedings, Annual Conference Series*, 2001.

[86] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. *Computer Graphics Proceedings, Annual Conference Series*, 1999.

[87] J. F. O'Brien and J. K. Hodgins. Animating fracture. *Communications of the ACM*, 43(7), July 2000.

[88] J. F. O'Brien, C. Shen, and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. *ACM SIGGRAPH Symposium on Computer Animation*, 2002.

[89] G.E. Plassman, "A Survey of Singular Value Decomposition Methods and Performance Comparison of Some Available Serial Codes" NASA CR-2005-213500, July 2005.

[90] N.S. Pollard, P.S.A. Reitsma, "Animation of Humanlike Characters: Dynamic Motion Filtering with a Physically Plausible Contact Model" *Yale Workshop on Adaptive and Learning Systems* (2001).

[91] Z. Popović, A. Witkin, "Physically Based Motion Transformation" *Computer Graphics Proceedings*, Annual Conference Series, 1999.

[92] I. Potucek. Automatic image stabilization for omni-directional systems.

[93] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21(4):145–152, July 1987.

[94] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "Numerical Recipes in C: The Art of Scientific Computing" 2nd.ed., Cambridge University Press: 1992.

[95] S. Robinson, "Fitting Spheres by the Method of Least Squares" Communications of the ACM, Volume 4, No. 11; November 1967, p. 491.

[96] P. L. Rosin. Further five point ellipse fitting. *Graphics Models and Image Processing*, 61:245–259, 1999.

[97] A. Safonova, J. K. Hodgins, N. S. Pollard. "Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behavior-Specific Spaces". *ACM Transactions on Graphics*, Vol. 23 No. 3, pp. 514-521, Aug 2004.

[98] S. Semwal, M.J. Parker, "An Animation System for Biomechanical Analysis of Leg Motion and Predicting Injuries during Cycling" *Real-Time Imaging*: 5 (1999): 109-

123.

[99] S.K. Semwal, R. Hightower, S. Stansfield, "Closed Form and Geometric Algorithms for Real-Time Control of an Avatar" *IEEE Proceedings of VRAIS 1996*.

[100] C.M. Shakarji "Least-Squares Fitting Algorithms of the NIST Algorithm Testing System" *J. of Research of the National Institute of Standards and Technology*: 103, No. 6 (1998): 633.

[101] A. Simonovits *Econometrica*, Vol. 43, No. 3 (May, 1975), pp. 493-498

[102] J. Shi, N. I. Badler, and M. B. Greenwald. Joining a real-time simulation: Parallel finite-state machines and hierarchical action level methods for mitigating lag time. 2000.

[103] M.-C. Silaghi, R. Plänkers, R. Boulic, P. Fua, and D. Thalmann. Local and global skeleton fitting techniques for optical motion capture. In N. Magnenat-Thalmann and D. Thalmann, editors, *Modelling and Motion Capture Techniques for Virtual Environments*, volume 1537 of *Lecture Notes in Artificial Intelligence*, pages 26–40, Berlin, Nov 1998. Proceedings of CAPTECH, Springer.

[104] M.-C. Silaghi, R. Plänkers, R. Boulic, P. Fua, and D. Thalmann. Local and global skeleton fitting techniques for optical motion capture. *Captech*, 1998.

[105] H. Späth. Least-square fitting with spheres. *Journal of Optimization Theory and Applications*, 96(1):191–199, January 1998.

[106] A. Strandlie, J. Wroldsen, R. Frühwirth and B. Lillekjendlie. Track Fitting on the Riemann Sphere. *International Conference on Computing in High Energy and Nuclear Physics*, Padova, Italy; February, 2000.

[107] S. Tak and H. Ko. Example Guided Inverse Kinematics. *Proceedings of the International Conference on Computer Graphics and Imaging*, pages 19–23, 2000.

[108] S.M. Thomas and Y.T. Chan. "Cramer-Rao Lower Bounds for Estimation of a Circular Arc Center and Its' Radius". *CVGIP: Graphics Model and Image Processing*, Vol. 57, No. 6, pages 527-532, 1995.

[109] D. Tolani, A. Goswami, N.I. Badler, "Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs" *Graphical Models*: 62 (2000): 353-388.

[110] D. Umbach and K. N. Jones. A few methods for fitting circles to data. *IEEE Transactions on Instrumentation and Measurement*, 52(6):1881–1885, December 2003.

[111] M. van de Panne, "From Footprints to Animation" *Computer Graphics Forum*: 16, no. 4 (1997): 211-223.

[112] G. A. Watson. Least squares fitting of circles and ellipses to measured data. *BIT*, 39(1):176–191, 1999.

[113] Y. Wei, S. Xia, D. Zhu. "A Robust Method for Analyzing the Physical Correctness of Motion Capture Data". *VRST*: pp. 338–341, Nov 1-3, 2006, Cyprus.

[114] G. Welch and G. Bishop. An introduction to the Kalman filter. 2004.

[115] D. J. Wiley and J. K. Hahn. "Interpolation synthesis of articulated figure motion."

IEEE Computer Graphics and Applications, 17(6):39–45, November/December 1997.

[116] L. de Witte. “Least Squares Fitting of a Great Circle Through Points on a Sphere.” Communications of the ACM, Volume 3, No. 11, November 1960, pp. 611-613.

[117] L. Yang, F. Albregtsen, and T. Taxt. “Fast computation of three-dimensional geometric moments using a discrete divergence theorem and a generalization to higher dimensions.” *Graphical Models and Image Processing*, 59(2):97–108, March 1997.

[118] J. Znamenáček and M. Valášek. An efficient implementation of the recursive approach to flexible multibody dynamics. *Multibody System Dynamics*, 2(3):227–251, 1998.

[119] E. E. Zelniker and I. V. L. Clarkson. A Generalisation of the Delogne-Kása Method for Fitting Hyperspheres. Pacific Grove, California, November 2004. Thirty-Eighth Asiomar Conference on Signals, Systems and Computers.

[120] E. E. Zelniker and I. V. L. Clarkson. A Statistical Analysis Least-Squares Circle-Centre Estimation. In 2003 IEEE International Symposium on Signal Processing and Information Technology, December 2003, Darmstadt, Germany, pp 114-117

[121] V.B. Zordan, A. Majkowaska, B. Chiu, M. Fast, “Dynamic Response for the Motion Capture Animation” *ACM Transaction on Graphics (SigGraph 2005)* 24(3) (July 2005) : 686-701.

[122] V.B. Zordan, N. C. Van Der Horst, “Mapping Optical Motion Capture Data to Skeletal Motion Using a Physical Model” *Eurographics/SIGGRAPH Symposium on Computer Animation* (2003) : 245-250.

Chapter 9 INDEX

A

angular momentum 112, 114
a-priori knowledge 49, 87
articulated figure ... xiii, xv, xvi, 1, 2, 3, 4, 7, 126, 131
articulated motion 1
asymptotic 74, 75
asymptotically unbiased 65, 70, 97, 100

B

Badler 17
biased 44, 46, 47, 50, 51, 65, 97
Bodenheim 17

C

C3D 115, 122, 125, 127
center estimator 35, 42, 43, 45, 46, 47, 48, 67, 68, 82
center of rotation ..... xvii, 84, 115, 117
CMU Graphics Lab ..... xiv, 91, 123
complexity 5, 115
coordinate system 82, 83, 84, 85, 91, 114
Cramér-Rao Lower Bound 23
Cyclomatic Complexity 123
cylindrical joint 76

D

Danis 122
degrees of freedom 1, 4, 126
Delogne ..... xviii, 21, 41, 133
digital cameras 7

E

eigenvalue 55, 72, 73
expectation 46, 47, 48, 50, 51, 52, 54, 55, 56, 59, 67, 68, 72

F

flops 40, 44, 81
Forward kinematics 6
Forward kinetics 6
Frey 118

G

Generalized Delogne-Kåsa Estimator .... xviii, 21
genetic algorithms xiii
gradient 42, 43

H

hierarchical 124, 125, 127, 131, 142
human shoulder 4
humerus 4, 5
hypersphere 44, 45, 61, 66, 73, 74, 80, 100
hysteresis 12

I

incremental improvement 81
Inverse kinematics 7, 17
Inverse kinetics 6

K

Kåsa ..... xviii, 21, 41, 133
Kirk 12
Knight 17

L

least-squares fit ..... xvi, 3
Leontief 46, 55, 67, 70, 72
Levenberg-Marquardt 35, 37
linear least-squares ..... xvi, 3, 38, 63

M

Maximum-Likelihood Estimator ..... xvi, 3, 21, 34
measurement covariance ... 46, 50, 54, 66, 67, 68, 69, 72
MEDIT 116, 118
MESH 116, 118, 119, 120, 124
moment of inertia 112
Monte-Carlo 19, 35, 43, 70
Movie 129
multivariate normal 20, 50, 51, 103

N

Newton-Euler differential equation ..... xiv, 1
NIST 34
non-linear fitting ..... xvi, 3

O

O'Brien 14, 63
OpenGL 123, 129, 143
orientation ..... xvi, 3, 63

P

physics ..... xiii, xiv, 1, 6, 100
Polygon File Format 122
Positional errors 12
positive-semidefinite 111

R

radius estimator 41, 42, 47, 48, 66, 68
recurrence 80

S

sample covariance . 43, 45, 46, 50, 51, 55, 65, 67, 72, 73, 76, 80, 111

sample mean 46
scapula 4
segment ..... xiii, 1
Singular Value Decomposition 39, 40, 63
skeleton xviii, 6, 11, 12, 63, 82, 86, 95, 100, 129, 138, 142
spectral radius 55, 70, 72
Stanford Triangle Format 122

T

tetrahedral mesh 116
tetrahedron 112, 113, 119, 122, 131
third central moment 43, 46, 47, 56, 57, 59
TIFF 129
translational freedom 4, 5
triangular mesh 116

U

UML 131

V

Vicon Peak 8, 9
video 7, 8

W

Wiley 17

Z

Zelniker 41, 45
Znamenáček 18