

OPTIMISING MARKER PLACEMENT FOR GAIT ANALYSIS

Michael Buckingham

Bachelor of Engineering
Software Engineering



Department of Engineering
Macquarie University

November 6, 2017

Supervisor: Dr Danè Turner
Co-Supervisor: Associate Professor Franck Cassez



ACKNOWLEDGMENTS

I would like to acknowledge Dr Danè Turner for all her help and guidance throughout my thesis project.

I would also acknowledge the motivation and support of my parents, whom without their guidance would not have the confidence to be who I am today.

I would like to thank my girlfriend Taylor, for proof reading this document and putting up with me in my stressed, sleep deprived state.

Finally, I would like to thank my good friend Vishnu, for lending me his time and computer for testing of the program.



STATEMENT OF CANDIDATE

I, Michael Buckingham, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Software Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment at any academic institution.

Student's Name: Michael Buckingham

Student's Signature: Michael Buckingham (Electronic Signature)

Date: November 6, 2017



ABSTRACT

Gait analysis is a well-defined area in mechanical engineering. However, in biomedical and mechatronic engineering, gait analysis of the human body is a lot newer with many issues to overcome. One of the main problems faced by researchers is the scaling of models for comparison.

This project will investigate gait analysis scaling for the human body, and produce a program to speed up the scaling process.

This completed project will help assist in defining a more accurate process for scaling gait models and aid research in future to make further discoveries and advancements.



Contents

Acknowledgments	iii
Abstract	vii
Table of Contents	ix
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Aim	1
2 Background and Related Work	3
2.1 What is Gait Analysis	3
2.2 Scaling in Gait Analysis	4
2.2.1 Manual Scaling	4
2.2.2 Linear Scaling	4
2.2.3 Anatomical Landmark Scaling	5
2.2.4 Kinematical Scaling	5
2.3 Optimisation Technique	6
2.3.1 Evolutionary Optimisation	6
2.3.2 Stochastic Optimisation	7
2.4 Methods Implemented in OpenSim	7
2.4.1 Computing Scale factor	7
2.4.2 Scaling the Model's Geometry	8
2.4.3 Scaling Mass and Inertial Properties	8
2.4.4 Scaling Muscles and Other Model Components	8
2.4.5 Placing Markers	8
3 System Requirements	9
3.1 Functional Requirements	9
3.2 Non-Functional Requirements	9
3.2.1 Availability	9

3.2.2	Maintainability	9
3.3	Design and Implementation Requirements	9
3.3.1	System Interfaces	9
3.3.2	User Interfaces	10
3.3.3	Hardware Interfaces	10
3.3.4	Software Interfaces	10
3.3.5	Communications Interfaces	10
4	System Design	11
4.1	Selected Method	11
4.2	Method Justification	11
4.3	Algorithm Explanation	11
5	Algorithm Correctness	13
6	Algorithm Complexity	15
7	Testing Method	17
7.1	Method 1	17
7.2	Method 2	18
7.3	Method 3	18
7.4	Method 4	18
7.5	Method 5	19
8	Results and Analysis	21
8.1	Testing Method 1	21
8.2	Testing Method 2	24
8.3	Testing Method 3	26
8.4	Testing Method 4	27
8.5	Testing Method 5	28
9	Conclusions and Future Work	33
9.1	Conclusions	33
9.2	Future Works	34
10	Abbreviations	35
A	Software Requirements	37
B	Source Code	39
B.1	OpenSim_Scaling.py	39
B.2	GetFiles.py	45
B.3	ModifyMarker.py	48
B.4	ModifySetup.py	49
B.5	MultipleScale.py	55

<i>CONTENTS</i>	xi
B.6 Scale.py	56
B.7 SelectScaleFile.py	58
B.8 Script.ps1	59
C Tabulated Results	63
D Meeting Attendance Form	67
Bibliography	67



List of Figures

2.1	Example of Gait Model in Vicon.	3
2.2	Example of an OpenSim Model.	4
2.3	Schematic overview of linear model scaling.	5
2.4	Schematic Overview of anatomical landmark scaling.	5
2.5	Schematic overview of kinematical scaling.	6
2.6	Basic evolutionary algorithm cycle.	6
2.7	Weighted least square mean formula implemented by OpenSim.	8
4.1	Software flow diagram.	12
8.1	Comparison of runtime data.	21
8.2	Comparison of error values.	22
8.3	Comparison of marker placements on the initial generic models for subject 20170223MB. Blue markers are the original file while the pink markers are the modified model.	23
8.4	Comparison of marker placements on the scaled models for subject 20170223MB. Blue markers are the original file while the pink markers are the modified model.	24
8.5	Comparison scaling times.	25
8.6	Comparison of scaling with the manual marker weighting.	26
8.7	Comparison error from static pose for all test subjects.	27
8.8	Comparison error from static pose for test subject 2070704PT.	28
8.9	Close up of torso markers for subject 20170704PT. The pink marker is from the script, The blue marker is from JN, the black marker is from JM, the green marker is from PT, and the yellow marker is from TN.	29
8.10	Front and rear marker comparison for the manually scaled models along with the script scaled model for subject 20170704PT. The pink markers are from the script, the blue markers are from JN, the black markers are from JM, the green markers are from PT and the yellow markers are from TN.	30

8.11 Left and Right marker comparison for the manually scaled models along with the script scaled model for subject 20170704PT. The pink markers are from the script, the blue markers are from JN, the black markers are from JM, the green markers are from PT, and the yellow markers are from TN.	31
8.12 Comparison of error obtained during inverse kinematics.	32
D.1 Meeting Attendance Form	68

List of Tables

8.1	Comparing the weighted least square method and developed script	28
A.1	Functional Requirements	37
A.2	Availability Requirements	37
A.3	Maintainability Requirements	37
A.4	System Interface Requirements	38
A.5	User Interface Requirements	38
A.6	Software Interface Requirements	38
C.1	Comparison of script and manual scaling error	63
C.2	Comparison of script with manual marker weighting and manual scaling error	64
C.3	Comparison of Scaling Time	65
C.4	Comparison of Scaling Method to Static Pose	66
C.5	Comparison of Scaling Methods Inverse Kinematic Error	66



Chapter 1

Introduction

Gait analysis is the study of the movement of an object. This process is done by placing markers on the subject and using a set of cameras to recording 3D coordinates of the markers as the subject moves along a set path. Currently, Macquarie University is conducting a study of the effectiveness of knee replacement surgery by analysing the gait of a subject before and after the surgery. This data is then manually modified to better correspond to the standard model. This manual manipulation lowers the accuracy and validity of the results as there is an increases chance human error and the manual manipulation decreases the likelihood that of repeatable results.

1.1 Aim

The aim of this project to automate, or semi-automate the process of scaling the test data, such that it is the best fit to the standard model. This process will allow for more accurate gait analysing of a human knee. A successful project will allow for more significant research into the effectiveness of current knee replacement techniques and possibly allow for future advancements in knee replacement surgery.

Chapter 2

Background and Related Work

2.1 What is Gait Analysis

Gait analysis is the study of the movement of an object. This process is achieved by placing markers on a subject and using a set of cameras to record the 3D coordinates of the markers as the subject moves along a set course, as shown in Figure 2.1. This data is scaled to a generic model for interpretation, ensuring an accurate comparison between each test subject. Figure 2.2 shows an example of a scaled model.

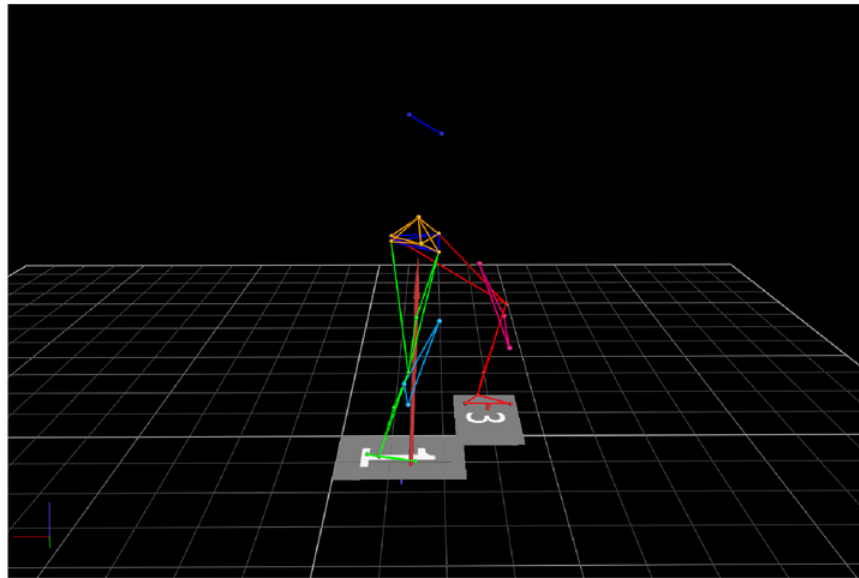


Figure 2.1: Example of Gait Model in Vicon.

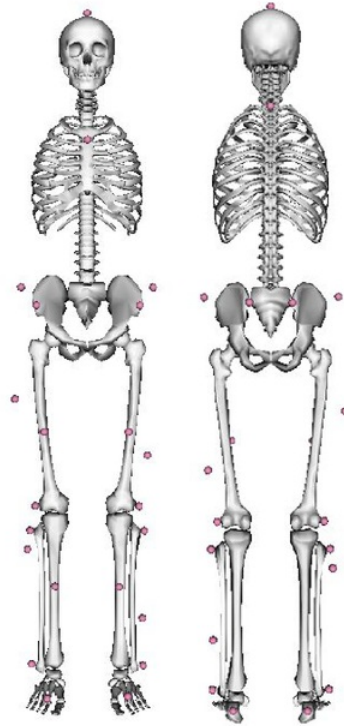


Figure 2.2: Example of an OpenSim Model.

2.2 Scaling in Gait Analysis

2.2.1 Manual Scaling

Manual scaling is where the operator adjusts the generic model to represent the subject. This scaling is usually achieved by the operator adjusting the marker position in computer modelling software like OpenSim. This manual manipulation inherently has accuracy issues as the finished model varies based on the operator.

2.2.2 Linear Scaling

Linear scaling models are the current standard method for musculoskeletal scaling. It relies on the motion trail to linearly scale body segment and calibrate the position of the markers on the subject [1]. Figure 2.3 demonstrates this process.

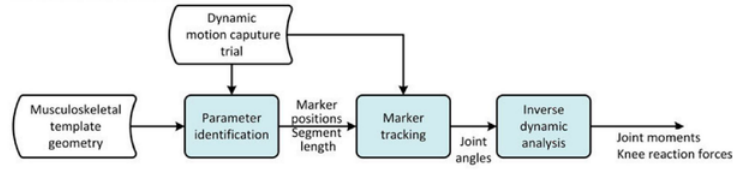


Figure 2.3: Schematic overview of linear model scaling.

2.2.3 Anatomical Landmark Scaling

Anatomical scaling uses a similar concept to linear scaling but instead of scaling the generic model straight to the motion trail, the static pose is used. The use of a static pose allows for the calculation of the joint parameters based on markers placed on the subjects anatomical landmarks.

Once the calculations are finished, the generic model is non-linearly transformed such that it has the same joint parameters and body segment values and the static pose. Figure 2.4 shows this process.

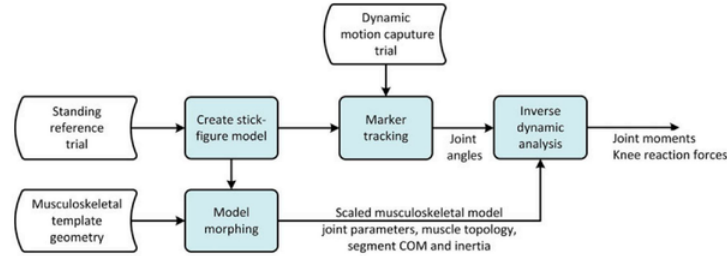


Figure 2.4: Schematic Overview of anatomical landmark scaling.

2.2.4 Kinematical Scaling

Kinematical scaling advances on anatomical landmark scaling through the use of dynamic functional trails. This advancement eliminates the dependency that correct marker placement places on anatomical landmark scaling.

Using a gait trial as the dynamic functional trial allows for the specific joint parameters to be calculated through the use of an optimised-based parameter identification method. This optimisation method can be similar to the multi-joint method proposed by Reinbolt et al [2]. Figure 2.5 outlines this process.

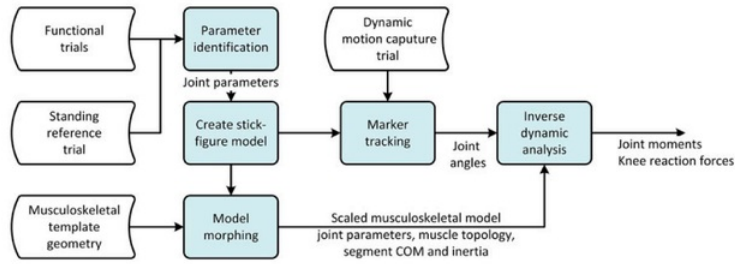


Figure 2.5: Schematic overview of kinematical scaling.

2.3 Optimisation Technique

2.3.1 Evolutionary Optimisation

Evolutionary optimisation bases itself on similar concepts to organic evolution. It aims to find the optimal system configuration within set bounded conditions. This optimisation is achieved through continuous modification of the system (or sub-system) and keeping the best outcome [3]. The basic design of an evolutionary algorithm is shown below in Figure 2.6 [4].

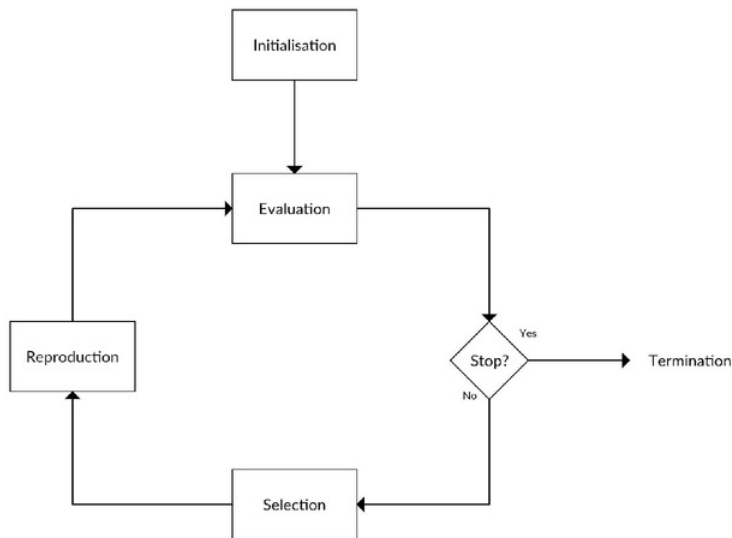


Figure 2.6: Basic evolutionary algorithm cycle.

Explanation of Stages

Initialisation: The initial set-up of the system.

Evaluation: Comparing the systems current state to the desired state

Stop: Terminate the loop if system is in optimal configuration

Selection: Selecting the optimal component in the system

Reproduction: Modifying the sub-optimal components

2.3.2 Stochastic Optimisation

Stochastic optimisation refers to any optimisation method where randomness is present. This randomness often presents itself in the objective function or constraints [5]. Stochastic algorithms function by generating random solutions to a problem and testing how well they fit the solution. More advanced algorithms will incorporate a limiting function which will create tighter bounds for the random solution, the closer the function gets to an optimal solution [6].

Stochastic optimisation can be split into two sub-categories, single-stage stochastic optimisation and multi-stage stochastic optimisation [7]. These differ in the number of random decisions needed for a single solution. As its name implies, single-stage stochastic optimisation focuses on optimising a problem with a single random objective function or constraints. This method is contrasted by multi-stage stochastic optimisation, where there is a need to find a sequence of random variables to solve a given problem [5].

2.4 Methods Implemented in OpenSim

During the scaling process, OpenSim implements a combination of three scaling methods to scale the test data to a given standard model. These methods are manual scaling, linear scaling and inverse kinematics. The full scaling process is broken down into the steps below [8].

2.4.1 Computing Scale factor

Computing the scale factor is the first step in the scaling process. Calculating the scale factor is achieved by averaging the length of each marker set in the given .trc file. These lengths are then divided by the segment length of the generic model, giving the scale factor for each segment. The only exception to this is if a manual scale factor is specified. In this case, the specified scale factor is given priority over the computed scaled factor. Each segment scale factor is then averaged to give the overall scale factor [8].

2.4.2 Scaling the Model's Geometry

Once the scale factor has been computed, OpenSim's scaling tool uses these scale factors to scales the model's geometry. This geometric scaling is done by scaling muscle attachment points, mass centre location, joint frame locations and force application points [8].

2.4.3 Scaling Mass and Inertial Properties

Next, the computed scale factors are used to scale the size of each body segment. Additionally, the mass of each segment is adjusted such that the mass of the subject is conserved.

There are two different ways that OpenSim scales the segment mass. The first approach is to preserve mass distribution. This approach scales the segment mass of the subject in such a way that it is proportional to the generics model segment mass. The second approach is to independently scale each segment based on the segments scale factor. [8].

2.4.4 Scaling Muscles and Other Model Components

Finally, OpenSim scales any components in the model that is dependent on its segment length. These components include ligaments and muscle actuators. For this scaling, a new scale factor is calculated. It is the ratio of a segment before and after scaling, this new scale factor is used to scale any length dependent component [8].

2.4.5 Placing Markers

Once the above scaling process is complete, OpenSim tries to move the markers on the generic model to match those on the experimental model's static pose. The experimental static pose is calculated by the implementation of the inverse kinematic algorithm used in the Inverse Kinematic tool [8] [9]. The marker error is generated by using a weighted least square mean formula as stated in figure 2.7 [10]. Where \mathbf{q} is the vector of generalized coordinates being solved for, $\mathbf{x}_i^{\text{exp}}$ is the experimental position of marker i , $\mathbf{x}_i(\mathbf{q})$ is the position of the corresponding marker on the model (which depends on the coordinate values), q_j^{exp} is the experimental value for coordinate j [10].

$$\min_{\mathbf{q}} \left[\sum_{i \in \text{markers}} w_i \left\| \mathbf{x}_i^{\text{exp}} - \mathbf{x}_i(\mathbf{q}) \right\|^2 + \sum_{j \in \text{unprescribed coords}} \omega_j \left(q_j^{\text{exp}} - q_j \right)^2 \right]$$

$$q_j = q_j^{\text{exp}} \text{ for all prescribed coordinates } j$$

Figure 2.7: Weighted least square mean formula implemented by OpenSim.

Chapter 3

System Requirements

For this section, *the system* shall refer to the scaling program in development.

3.1 Functional Requirements

With *The system's* role being to automate scaling of models for gait analysis, its functions can be broken down into three distinct categories. Those categories being data input, data processing, and data output. Table A.1 states the complete list of the functional requirements.

3.2 Non-Functional Requirements

3.2.1 Availability

With the project solely being a software component, a working version of the program must be available at all times. Table A.2 states the Availability requirements.

3.2.2 Maintainability

The system must be able to support updates to the source code. Table A.3 displays the Maintainability requirements.

3.3 Design and Implementation Requirements

3.3.1 System Interfaces

The system will need to interface with a computing device. In particular, the system will need to be able to interface with any operating system currently supported by Microsoft. Table A.4 lists the System interface requirements.

3.3.2 User Interfaces

The user will need to be able to select multiple input files that *the system* will use for scaling. Table A.5 shows all the User interface requirements.

3.3.3 Hardware Interfaces

The computing device's operating system ensures a well-defined interface with the hardware, such that there are no additional hardware requirements.

3.3.4 Software Interfaces

Along with the operating system requirement stated in section 3.3.1, *the system* will need to interface with three different software libraries. These software libraries being Python, .NET and OpenSim's inbuilt API's. Table A.6 states all software interface requirements.

3.3.5 Communications Interfaces

As *the system* will run locally on the user's device, it does not need any need any communication interfaces.

Chapter 4

System Design

4.1 Selected Method

Evolutionary optimisation has been selected for the initial design of this project. Section 4.2 explains the reasoning on for the selection of this method.

4.2 Method Justification

Evolutionary optimisation has been chosen as it is simpler to design, implement and test. This is due to the randomness introduced by a stochastic method. This randomness adds extra uncertainty to the optimisation algorithms implementation.

Another factor to take into consideration is that time complexity of both optimisation methods. Theoretically, in an evolutionary algorithm, the time complexity should stay constant on multiple runs on the same data set. This complexity is contrasted by stochastic optimisation where the time complexity changes based on the randomness in the random number generation function.

4.3 Algorithm Explanation

The automated scaling program shall have the following structure:

1. The program will take input files of the generic model shown in Figure show in Figure 2.2, the test subject's static pose, the scale settings from OpenSim and a list of markers and what order they are to be scaled in.
2. The program will scale the generic model to the subject's static pose using Opensim's scaling tool.
3. The program will calculate the maximum error and the marker where the maximum error occurs.

4. The program will check if the maximum error is within its set error limit.
5. If the maximum error is outside the set error limit, the program will move the markers on the generic model
6. The program will repeat steps 2-5 until the scaled model is within the set error limit or if there are 20 cycles without an improvement in error.
7. When the scaled model is within the set error limit or reaches its cycle limit, the program will output the scaled model in OpenSim and then terminate.

This process is outlined in Figure 4.1.

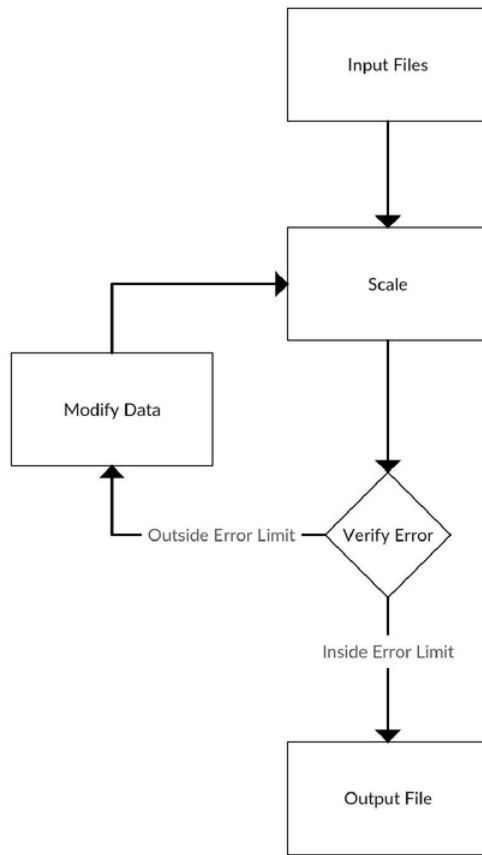


Figure 4.1: Software flow diagram.

Chapter 5

Algorithm Correctness

The program is considered to be correct if the maximum error of the scaled generic model is less than two centimetres.

For the marker with the max error, if there is a marker location between 50% and 0.9% of the error value that reduces the overall error, the algorithm will select this modification. From testing, it has been evaluated that the algorithm is correct 80% of the time, for the given test subjects.

Chapter 6

Algorithm Complexity

The developed algorithm is a Monte-Carlo algorithm, such that it will always output a scaled model, but it may not be a correct scaled model.

Since the algorithm has no set termination but terminates based off having not improved the error value or being inside a predefined bound. There is no definable complexity in natural time. That being said if the subject is at an optimal scale but outside the set bound, the lower bound complexity is defined as:

$$o(f(n)) = \sum_{n=1}^{\text{Number of scaling bodies}} 20x_n$$

Where x is the time taken to scale a select body of markers.

Chapter 7

Testing Method

The tests below are the methods used to verify and validate the developed script. These methods were chosen as they best validate the efficiency and correctness of the script when compared to manual scaling.

7.1 Method 1

The first testing method implemented aims to test the runtime of the script along with the consistency of the script across different devices. The script will run ten times on three devices of different specifications. For each run, identical script settings were used to ensure that valid results. These settings are, first scaling with the markers on the pelvis (RASI, LASI, LPSI, RPSI, LTOR, RTOR). Then scaling with the markers on the bony landmarks (HEAD, LKNE, RKNE, RFIB, LFIB, RANK, LANK, RHEE, LHEE, RASI, LASI, LPSI, RPSI, LTOR, RTOR) and finally scaling with all the markers on the test subject.

The final setting defined is the marker weighting. These are predefined as all the markers located on the pelvis having a marker weight of 5, while the other markers have a marker weighting of 1.

The order and weighting specified above is selected due to the overall error being more sensitive to any error present in the pelvis. This sensitivity causes any error in the pelvis to be compounded throughout the rest of the model.

The specifications of the three devices used are defined below:

1. Desktop PC:

- Operating System: Windows 10 Pro
- CPU: AMD Ryzen 7 1700 @ 3.6Ghz
- Memory: 16GB DDR4 2400MHz
- Hard Drive: 960GB SSD
- GPU: NVIDIA GTX 1080ti EVGA FTW3 11GB

2. HP Spectre Pro Laptop:

- Operating System: Windows 10 Enterprise
- CPU: Intel Core i7-4500U @ 1.8Ghz
- Memory: 8GB DDR3 1600MHz
- Hard Drive: 128GB SSD
- GPU: Intel Dedicated Graphics

3. Microsoft Surface Pro 4:

- Operating System: Windows 10 Pro
- CPU: Intel Core i5-6300U @ 2.4Ghz
- Memory: 8GB DDR3 1867MHz
- Hard Drive: 256GB SSD
- GPU: Intel Dedicated Graphics

7.2 Method 2

The previous test aimed to prove that the script runs identically on different devices. The second testing method aims to compare the scripts error to the error of the manual scalers. This comparison was achieved by running the script with the same marker weightings as the manually scaled model.

7.3 Method 3

The third testing method aims to compare the accuracy of the different scalers final model to the script. This test was conducted by taking the marker locations on the manually scaled model and the model scaled by the script. These markers are then compared by calculating their distance to the marker location in the test subjects static pose.

7.4 Method 4

The fourth test aims to compare the errors obtained when running inverse kinematics on a manually scaled model and a model scaled by the script. This comparison shall allow for conclusions to be made on the accuracy of scaled models as any error present will be compounded in the inverse kinematic process.

7.5 Method 5

The fifth test aims to try and compare the results of Mohboobins weighted least square mean method [11] to the developed script. This method tries to replicate the test performed by Mohboobin.

Chapter 8

Results and Analysis

8.1 Testing Method 1

The graph in Figure 8.1 displays a comparison of the script runtime between the different devices. In this graph, the number of cycles for each test subject has been included to show the relation between runtime and number of iterations. In this graph, the subject 20170223MB's generic model has been manually modified to better represent the initial marker placement on the subjects body.

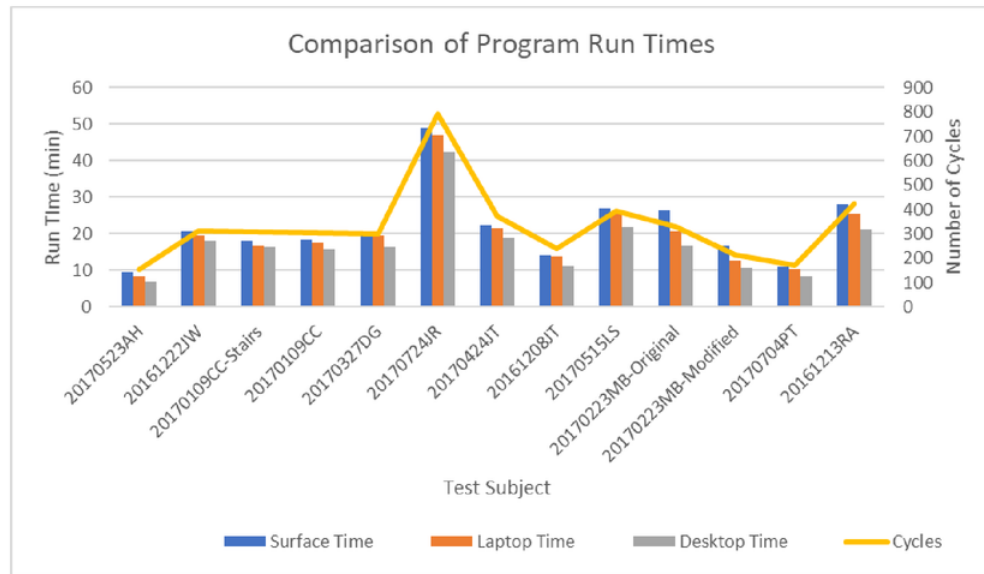


Figure 8.1: Comparison of runtime data.

Figure 8.2 is a visual representation of the data in Table C.1. This graph compares the errors from the manually scaled test subjects to test subjects scaled by the script. Again, in this graph, the number of cycles has been included to show any relation between the final error and iterations. An error line has also been added, where any value below that mark is considered as an acceptable scaled model. Similar to Figure 8.1, subject 20170223MBs modified generic model has been included.

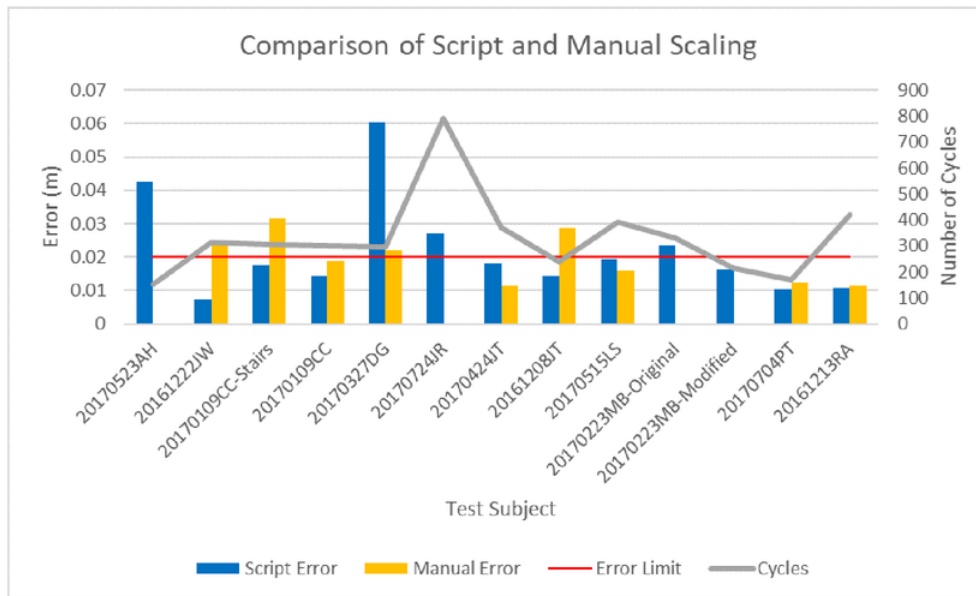


Figure 8.2: Comparison of error values.

From both Figure 8.1 and Figure 8.2, it can be observed that there is a correlation between the runtime and the number of cycles. However, there is no correlation between final error and number of cycles, and by extension no relation between runtime and final error. Figure 8.1 also displays that the scripts main limitation is the processing speed of the CPU. Such that, the better the CPUs single and multi-core performance, the faster the script will complete the scaling process.

It can also be observed from the data, that 75% of the twelve test subjects can be scaled to have an acceptable error value. Furthermore, out of the nine manually scaled subjects, only three manually scaled subjects have a lower error.

Due to the script having difficulty scaling a large number of incorrectly placed markers on the test subject, this higher error can be caused by inaccurate marker placement. Two sources of incorrectly placed markers are, large amounts of soft tissue on the subjects body making it difficult for the bony landmarks to be identified. The other source of marker error is where markers are incorrectly placed due to human error.

In Figure 8.2, it can be observed that a subject's final error can be reduced by initially modifying the generic model to represent a more accurate marker placement on the test subject. This observation is evident with subject 20170223MB, with their final error being reduced by 30% by modifying a few markers on the subjects pelvis and leg. This modification to the generic model can be seen in Figures 8.3, with the scaled models shown in Figures 8.4.

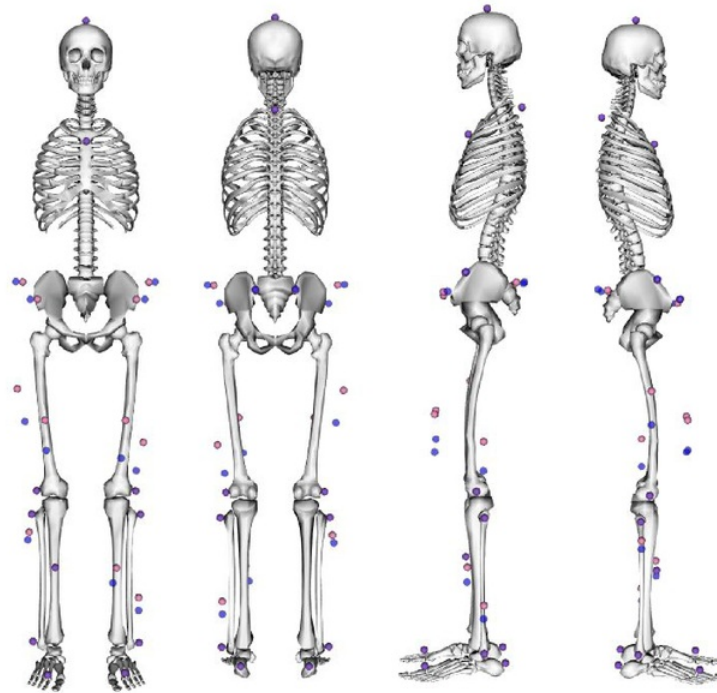


Figure 8.3: Comparison of marker placements on the initial generic models for subject 20170223MB. Blue markers are the original file while the pink markers are the modified model.

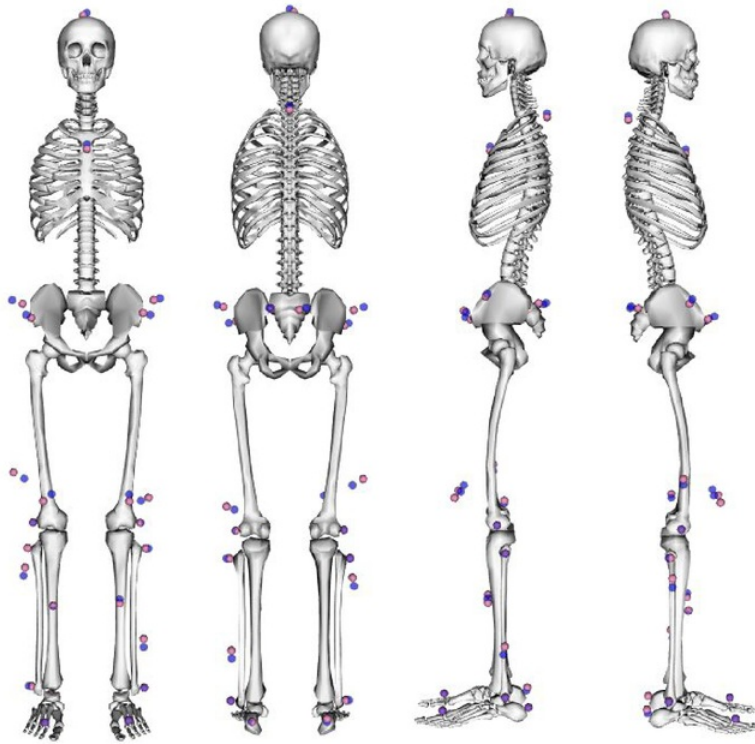


Figure 8.4: Comparison of marker placements on the scaled models for subject 20170223MB. Blue markers are the original file while the pink markers are the modified model.

8.2 Testing Method 2

Figure 8.5 displays the data shown in Table C.3. This graph compares the estimated time taken to manually scale a test subject to the time taken for the same test subject to be scaled by the developed script. The estimate manual scaling time is a rough estimate based on the scalier recollection. The estimate was based on the lower bound scaling time, assuming for eight hour days and that the scalier work solely on scaling the whole time.

The graph displayed in Figure 8.6 compares the error obtained when the test subject was scaled manually and when the same subject was scaled with the script, using the same marker weights as the manual scaling. Similar to Figure 8.2, an error line has been added to display what is considered an acceptable scaling.

In Figure 8.5, it can be identified that on average the script is 99% faster the current manual scaling method. This massive reduction is due to the script being able to modify

the generic model and efficiently evaluate the modification multiple time per second. This reduction is contrasted by the manual method which can take up to 5 minutes to make single scaling iteration.

From Figure 8.6, it can be concluded that using the same weighting as the manually scaled subjects, nine out of the twelve subjects have an acceptable final error. It can also be observed that manually scaled model had a lower error five out of the 12 models. This result can be caused by manual scalers tuning the marker weightings mid scaling to better fit their proposed modified generic model. This manual tuning could lessen the accuracy of the results, as it can obfuscate any errors present in the proposed model.

Furthermore, the only test subject to have a manually scaled error lower than either automatically scaled model was subject 20170424JT. As hypothesised above, this could be caused by inaccurate marker placement, or manual tuning of the marker weights to better suit the proposed model.

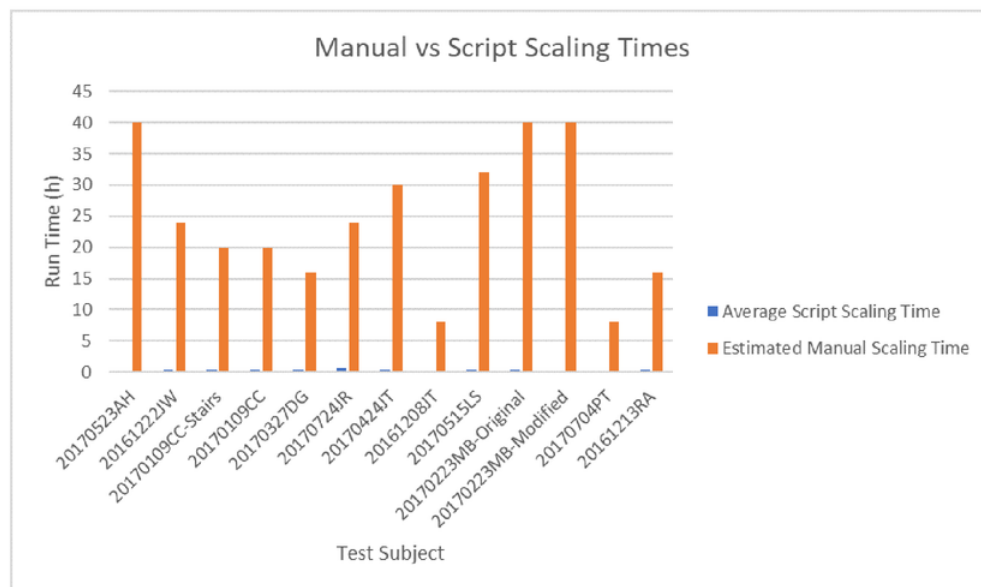


Figure 8.5: Comparison scaling times.

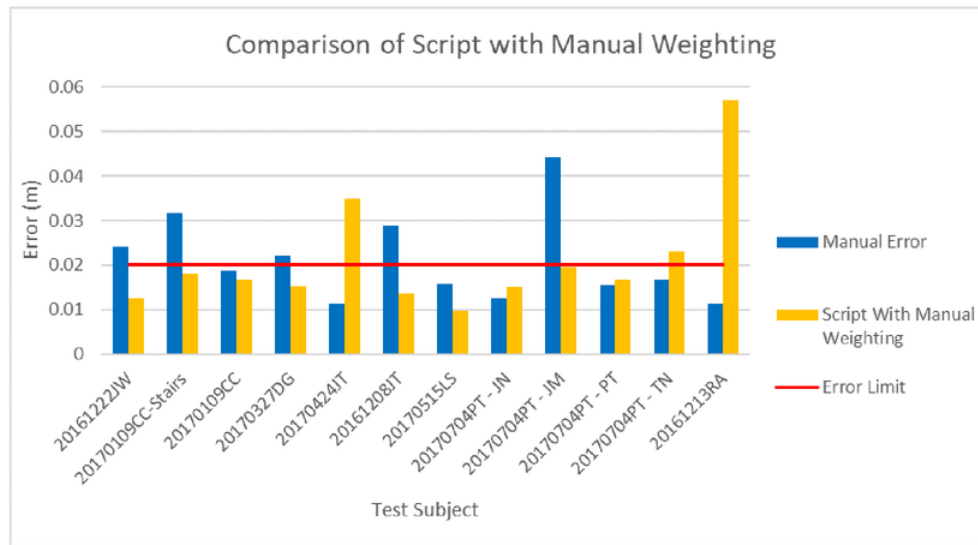


Figure 8.6: Comparison of scaling with the manual marker weighting.

8.3 Testing Method 3

Figure 8.7 shows the average error for each scaling method when compared to the static pose. This has been broken down further in Figure 8.8, where the error for each manual scaler is being compared for a single test subject.

In Figures 8.7 and 8.8, the distance between the scaled model and static poses marker location has been calculated and used as a measure of scaling accuracy. From observing the figures and the data in Tables C.4, there is a general trend to the script having a lower average marker distance. But it is difficult to that any one scaling method is more accurate than another due to the variation in scaled marker differences being negligible. Only one case having a variation that was more than one millimetre.

Upon further visual comparisons of the marker placements for subject 20170704PTs scaled models as shown in Figures 8.10 and 8.11. There have been instances of markers being incorrectly move. This is evident in Figure 8.9, where the CLAV marker is not placed in the centre of the model with the grouping of the manually scaled CLAV markers.

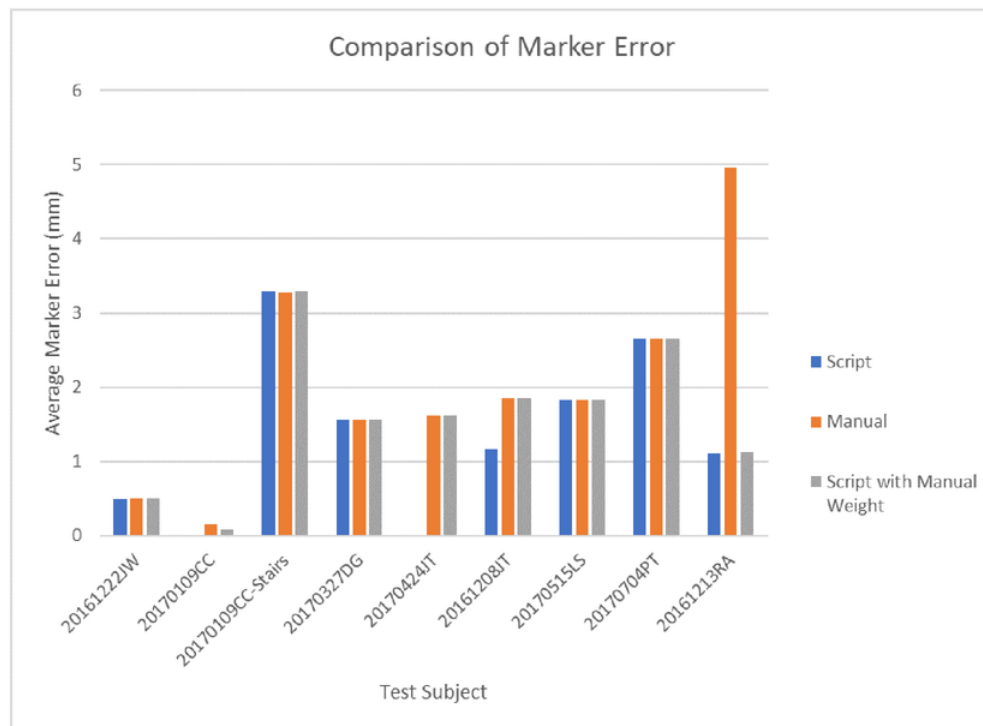


Figure 8.7: Comparison error from static pose for all test subjects.

8.4 Testing Method 4

The graph displayed in Figure 8.12 displays a comparison of the average errors obtained when running three different inverse kinematic tests on a manually scaled model for a subject and a model scaled by the script, both of subject 20161208JT. The three inverse kinematic test performed where, step-up, step-down and walking.

From observing the data, it can be observed that for the step-up and step-down test, the variation in error is negligible, with the exception of the max error. But there is a more notable difference in the walking test with the script scaled model having the lower errors. This result demonstrates that for subject 20161208JT, the script can produce a model that is close to the manually scaled model.

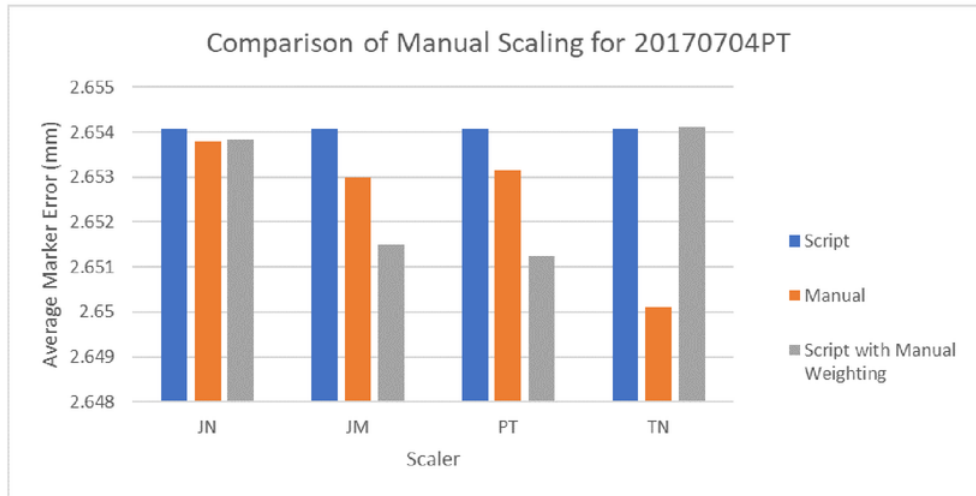


Figure 8.8: Comparison error from static pose for test subject 2070704PT.

8.5 Testing Method 5

The data in Table 8.1 is a summary of the comparison between the developed script and the WLSM method. The data shows that Mohboobin's method is more effective in reducing error in gait model "Gait2354". Since this model is different from the model "gait2392" which the previous tests used, it is difficult to draw a direct conclusion from between this test and the previous tests.

Table 8.1: Comparing the weighted least square method and developed script

Method	Max Error (cm)
WLSM	0.04
Script	2.5

It is difficult to validate the accuracy of this result. This difficulty is because we are unable to verify that the example models provided by OpenSim have not been modified since Mohboobin's testing in 2013.

Another area of difficulty comes from the source code not being available, such that we are unable to do further testing comparing the two methods.

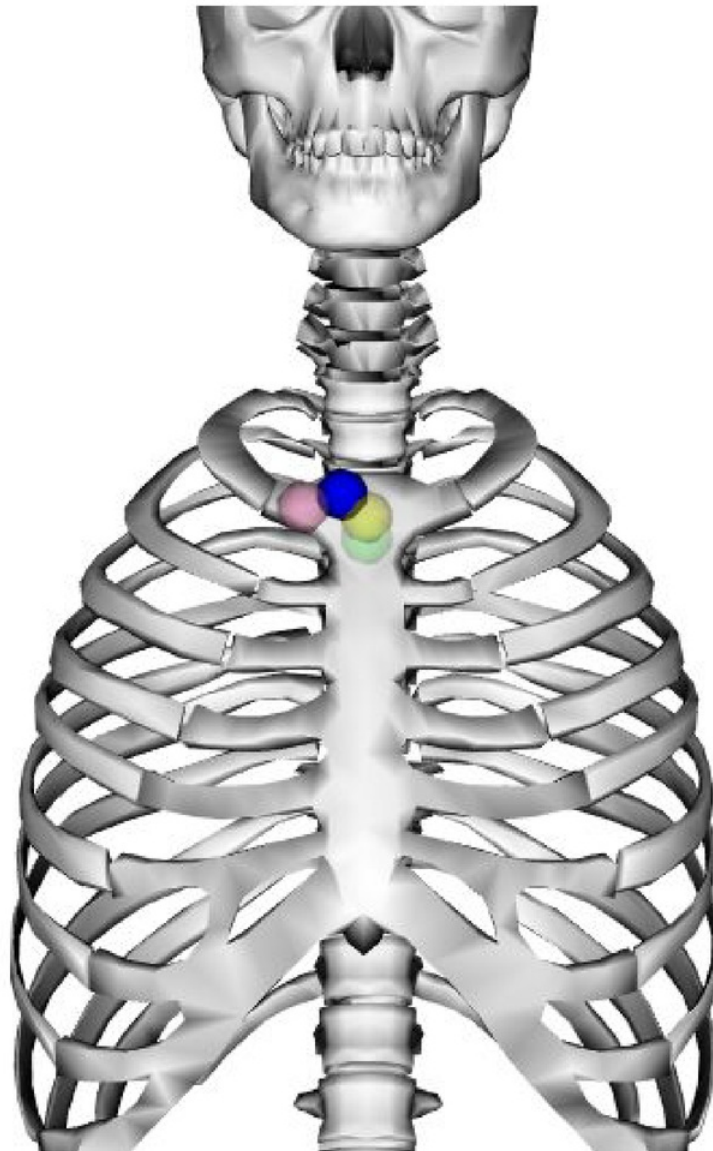


Figure 8.9: Close up of torso markers for subject 20170704PT. The pink marker is from the script, The blue marker is from JN, the black marker is from JM, the green marker is from PT, and the yellow marker is from TN.

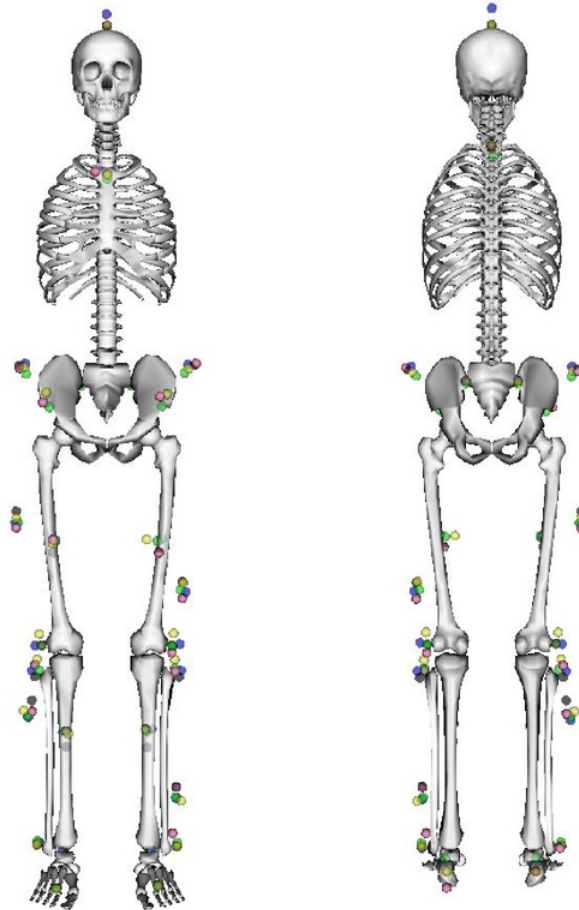


Figure 8.10: Front and rear marker comparison for the manually scaled models along with the script scaled model for subject 20170704PT. The pink markers are from the script, the blue markers are from JN, the black markers are from JM, the green markers are from PT and the yellow markers are from TN.

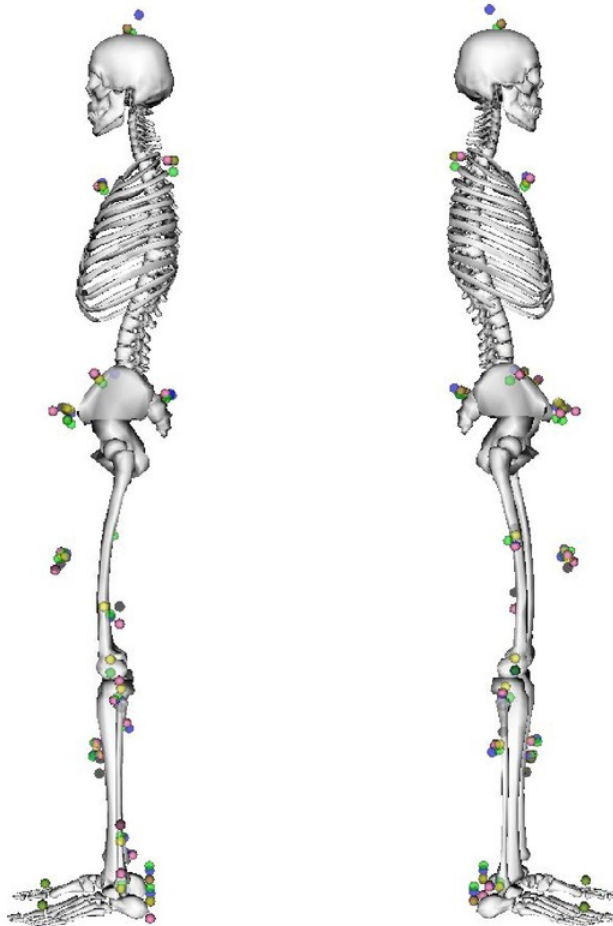


Figure 8.11: Left and Right marker comparison for the manually scaled models along with the script scaled model for subject 20170704PT. The pink markers are from the script, the blue markers are from JN, the black markers are from JM, the green markers are from PT, and the yellow markers are from TN.

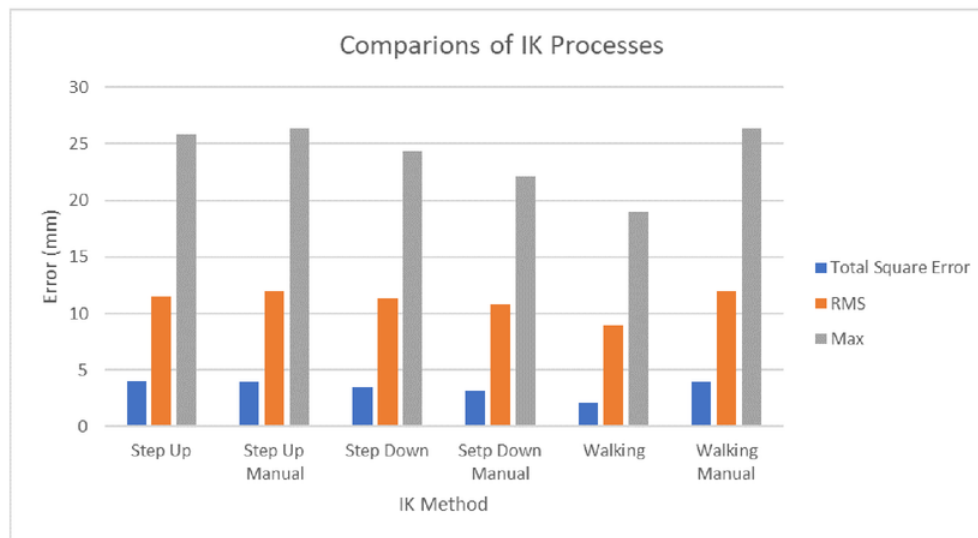


Figure 8.12: Comparison of error obtained during inverse kinematics.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

The shown above in Chapter 8 and Appendix C, it can be concluded that 75% of the subjects scaled by the script have an acceptable error value. It can also be observed that without any manual intervention, the script produced a lower error than manual scaling 50% of the time with this percentage increasing with minimal manual intervention. Further investigation found that manual manipulation of the initial model can reduce the final error and can speed up the scaling process. A similar result can be seen in Mohboobins paper, where he saw a reduction in the maximum error from 6 cm down to 0.04cm in his test sample [11]. His method differs from the script in this project, as Mohboobin method focus on trying to find a solution for the weighted least square mean formula stated in Figure 2.7, while this projects script focuses on manipulating OpenSim's generic model to be a better representation of the test subject.

Unfortunately, it is difficult to compare the accuracy of the two different methods. This difficulty is due to there currently is no method for determining the accuracy of models. Further testing will also be needed for a conclusion to be made on which method is better.

From the test data obtained it can be calculated that time taken to scale a test subject with the script was 99% less than the current manual process, with the script's main bottleneck being the CPU.

From the results summarised above, it can be concluded that it is not optimal to solely scale using the developed scaling script, but to use the script in conjunction with manual user intervention. Unfortunately, due to the validation of the marker placement being outside the scope of this project, it is difficult to state with complete confidence which method is superior.

In conclusion, the script developed in this project satisfies all the predefined requirement as outlined in Chapter 3 and the project aim outlined in Section 1.1. Such that the project was successful in its goal of development of a program to optimise marker placement for gait analysis.

9.2 Future Works

There are several areas that can be investigated to either improve scaling or further validate the results obtained in this project. One of the areas for further research is to conduct a research project validating accuracy of the scaled marker placements. This can be achieved by performing a CT scan on a test subjects to get an accurate position of the marker placements and comparing it to the final scaled model.

Due to the project scope focusing on developing an optimisation algorithm to speed up the scaling process and not comparing the efficiency of an evolutionary implementation verse a stochastic implementation. An area of future research would be to develop a stochastic optimisation algorithm and compare its time complexity to the evolutionary optimisation algorithm. This can also be extended to find ways to optimise the methods implemented by the program developed in this project.

A third area of research is test how marker weighting and the order of marker weights affect the final scaling error. As the program developed in this project can have markers weights and scaling order modified without any modification to the underlying source code, this testing can utilise the developed program to minimise human error and bias on the results.

Another area of future research is to further investigate Mohboobin's method and compare to the method developed in this project.

A final possible area of research is to implement a method that uses the marker location of the subjects static pose, or image recognition to better align the markers. This method can also be extended to use a similar process to further verify the marker placement on the scaled model.

Chapter 10

Abbreviations

API	Application Programming Interface
C7	7th cervical vertebra
CLAV	Clavicle
CT	Computed Tomography
EA	Evolutionary algorithm
EO	Evolutionary optimisation
GUI	Graphical User Interface
HEAD	Top of the Head
IK	Inverse kinematic
LANK	Left Ankle
LASI	Left Anterior Superior Iliac Spine
LFIB	Left Fibula
LHEE	Left Heel
LKNE	Left Knee
LPSI	Left Posterior Superior Iliac Spine
LQUA	Left Quadriceps
LSHI	Left Shin
LTHI	Left Thigh
LTIB	Left Tibia
LTOE	Left Toe
LTOR	Left Torso
OS	Operating system
RANK	Right Ankle
RASI	Right Anterior Superior Iliac Spine
RFIB	Right Fibula
RHEE	Right Heel
RKNE	Right Knee
RPSI	Right Posterior Superior Iliac Spine
RQUA	Right Quadriceps
RSHI	Right Shin

RTHI	Right Thigh
RTIB	Right Tibia
RTOE	Right Toe
RTOR	Right Torso
SA	Stochastic algorithm
SO	Stochastic optimisation
UI	User interface
WLSM	Weighted least square mean

Appendix A

Software Requirements

Table A.1: Functional Requirements

Requirement Number	Requirement
FR-1	<i>The system</i> must be able to accept input files.
FR-2	<i>The system</i> must be able to process the input files.
FR-3	<i>The system</i> must be able to scale the generic model.
FR-4	<i>The system</i> must be able to output the scaled file.
FR-5	The scaled file must have a maximum error less than 0.02m.
FR-7	<i>The system</i> must be faster than the current manual scaling method

Table A.2: Availability Requirements

Requirement Number	Requirement
AR-1	A working version of <i>the system</i> shall be available 100% of the time.

Table A.3: Maintainability Requirements

Requirement Number	Requirement
MR-1	<i>the system</i> must be able to be updated when a new update is released.

Table A.4: System Interface Requirements

Requirement Number	Requirement
SI-1	The user must be able to use the system on a Windows 7, Windows 8/8.1 or Windows 10 machine.

Table A.5: User Interface Requirements

Requirement Number	Requirement
UI-1	The user must be able to select a generic model to scale.
UI-2	The user must be able to select the subject's static pose trial.
UI-3	The user must be able to select the subject's scaling file.
UI-4	The user must be able to select a scaling body file.
UI-5	The user must be able to select a custom scaling weights or the default scaling weight.

Table A.6: Software Interface Requirements

Requirement Number	Requirement
SWI-1	<i>The system</i> must be able to interface with Python's libraries.
SWI-2	<i>The system</i> must be able to interface Microsoft's native .NET libraries.
SWI-2	<i>The system</i> must be able to interface OpenSim's inbuilt libraries.

Appendix B

Source Code

B.1 OpenSim_Scaling.py

```
# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#
# Permission is hereby granted, free of charge, to any person
obtaining a copy of
# this software and associated documentation files (the "
Software"), to deal in
# the Software without restriction, including without limitation
the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the
# Software, and to permit persons to whom the Software is
furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND
, EXPRESS OR IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION
```

```
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

from GetFiles import *
from ModifyMarker import *
from ModifySetup import *
from Scale import *
from MultipleScale import *
from SelectScaleFile import *
import time
import os
import datetime

def log(model, line):
    # INPUT: Location of patient Files and string
    # OUTPUT: appends the string to the log file and print the
    #          string to console
    print line
    file = model.FilePath + '/Log/Log.txt'
    f = open(file, 'a')
    f.write(line + '\n')
    f.close()

class File:
    # Creates the "File" object
    def __init__(self, name, path, ):
        self.FileName = name
        self.FilePath = path
        self.FullFileName = path + '/' + name

if __name__ == "__main__":
    # Main script, initialises all variables and runs scaling
    # program
    bound = 0.000005

    summary = []
    ST = str(datetime.datetime.now())
    files, customScaling, runArray = GetFiles()
```

```

start = time.time()
location = os.getcwd()

# creates file object from GetFiles() output
for f in files:
    if f['FileType'] == 'Pose':
        pose = File(f['FileName'], f['FilePath'])
    if f['FileType'] == 'Model':
        model = File(f['FileName'], f['FilePath'])
    if f['FileType'] == 'Setting':
        setting = File(f['FileName'], f['FilePath'])

# Creation of log file
directory = model.FilePath + '/Log'
if not os.path.exists(directory):
    os.makedirs(directory)
file = model.FilePath + '/Log/Log.txt'
f = open(file, 'a')
f.write(pose.FileName + "_" + ST + '\n')
f.close()

# Initial Scaling of markers
InitModifySetup(setting, model, pose, customScaling)
errorMarker, initErrorVal = Scale(location, setting.
    FullFileName)
scaleValue = float(initErrorVal) / 2
log(model, "INIT:_" + str(errorMarker) + "_" + str(
    initErrorVal) + "_" + str(scaleValue))

# Check if initial scale is within the predefined bounds
if float(initErrorVal) > bound:
    # Loops the program depending on the number of scaling
    # bodys defined in test file
    for runs in runArray:
        run = runs.get('run')
        scaleMarkers = runs.get('markerList')
        log(model, "Scale_" + run)
        prevError = float(initErrorVal)

        # creates multiple setup files
        ModifySetup(setting, model, customScaling,
            scaleMarkers)
        errorMarker, initErrorVal = Scale(location, setting.

```

```

FullFileName)
scaleValue = float(initErrorVal) / 2
count = 1
sameCount = 0
# Modifies markers and validates if scaled model
meets specifications
while prevError > bound:
    ModifyMarker(model.FullFileName, errorMarker,
                 scaleValue)

    lowMarker, lowValue, lowIndex = MultipleScale(
        location, setting)
    log(model,
         str(lowMarker) + ", " + str(lowValue) + ", "
         + str(lowIndex) + ", " + str(count) + ",
         " + str(
             sameCount))
    count += 1
    if float(lowValue) < prevError:
        sameCount = 0
        SelectScaleFile(model.FullFileName, lowIndex
                        )
        prevError = float(lowValue)
        errorMarker = lowMarker

    elif sameCount == 10:
        log(model,
             "Resetting Scale Variance (" + str(
                 scaleValue) + " -> " + str(float(
                     prevError) / 2) + ")")
        scaleValue = float(prevError) / 2
        sameCount += 1

    elif sameCount < 20:
        sameCount += 1
        log(model, "halving value (" + str(
            scaleValue) + " -> " + str(float(
                scaleValue) / 2) + ")")
        scaleValue = float(scaleValue) / 2
    else:
        log(model,
             "Unable to scale " + run + " further, "
             exiting loop" + str(

```

```

        prevError) + "\n" + errorMarker + "\n"
        + str(
            sameCount))

    break

log(model, "Finished\n" + run + "\nin\n" + str(count) +
    "\ncycles.")

dict = {'run': run, 'runCount': count, 'error':
    prevError, 'marker': errorMarker}
summary.append(dict)
tree = et.ElementTree(file=model.FullFileName)
writeFile = model.FilePath + "/scaled" + run + ".
    osim"
tree.write(writeFile)

# Runs the final scale on all scalable markers
ModifySetupFinal(setting, model, customScaling)
errorMarker, initErrorVal = Scale(location, setting.
    FullFileName)
scaleValue = float(initErrorVal) / 2
log(model, "FINAL:\n" + str(errorMarker) + "\n" + str(
    initErrorVal) + "\n" + str(scaleValue))
log(model, "Scale_Final")
prevError = float(initErrorVal)

count = 1
sameCount = 0
while prevError > bound:
    ModifyMarker(model.FullFileName, errorMarker,
        scaleValue)

    lowMarker, lowValue, lowIndex = MultipleScale(
        location, setting)
    log(model, str(lowMarker) + "\n" + str(lowValue) + "\n" + str(
        lowIndex) + "\n" + str(count) + "\n" + str(
            sameCount))
    count += 1
    if float(lowValue) < prevError:
        sameCount = 0
        SelectScaleFile(model.FullFileName, lowIndex)
        prevError = float(lowValue)

```

```

        errorMarker = lowMarker

    elif sameCount == 10:
        log(model, "Resetting Scale Variance_" + str(
            scaleValue) + "_->" + str(float(prevError) /
            2) + ")")
        scaleValue = float(prevError) / 2
        sameCount += 1

    elif sameCount < 20:
        sameCount += 1
        log(model, "halving value_" + str(scaleValue) +
            "_->" + str(float(scaleValue) / 2) + ")")
        scaleValue = float(scaleValue) / 2
    else:
        log(model, "Unable to scale body further,
            exiting loop" + str(prevError) + "_" + str(
            sameCount))
        break

log(model, "Finished body in_" + str(count) + "_cycles."
    )

finalCount = count
finalError = prevError
finalMarker = errorMarker
tree = et.ElementTree(file=model.FullFileName)
writeFile = model.FilePath + '/scaledFinal.osim'
tree.write(writeFile)

# Deletes all temp files
RemoveOldFiles(model, setting)
finish = time.time()
totalCycles = finalCount
# Finalises the log file
for line in summary:
    totalCycles += line.get('runCount')
    log(model, line.get('run') + "::Error:" + str(line
        .get('error')) + "_Marker:" + line.get(
        'marker') + "_Cycles:" + str(line.get('runCount'
        )))
    # log(model, "Body:: Error: " + str(bodyError) + "
        Marker: " + bodyMarker + " Cycles: " + str(

```



```

        bodyCount))
    log(model, "Final::Error:" + str(finalError) + " "
        Marker:" + finalMarker + "Cycles:" + str(
        finalCount))
    log(model, "RunTime:" + str((finish - start) / 60) + "
        minCycles:" + str(totalCycles) + '\n\n')

    # Opens scaled model in OpenSim
    OpenOSIMFile(model)
    # Terminates the code
    exit(0)

```

B.2 GetFiles.py

```

# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of
# this software and associated documentation files (the "
# Software"), to deal in
# the Software without restriction, including without limitation
# the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or
# sell copies of the
# Software, and to permit persons to whom the Software is
# furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND
# , EXPRESS OR IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
# , FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
# WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN

```

*CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.*

```
import Tkinter, tkFileDialog, tkMessageBox
import ntpath

def GetFiles():
    # INPUT: None
    #Output: List of files, Boolean defining custom scaling
    values, array of scale bodies.
    files = []

    # Select file body segments
    root = Tkinter.Tk()
    root.withdraw()
    fullFilePath = tkFileDialog.askopenfilename(filetypes=(("
        Marker_List", "*.txt"), ("All_files", "*.*")), title='
        Select_Marker_List')

    # processes body segments and appends them to an array
    f = open(fullFilePath)
    runArray = []
    for line in f:
        runName = ""
        markerList = []
        split = line.split(":")
        if split[0] != "":
            runName = split[0].strip()
            markers = split[1].split(",")
            for marker in markers:
                markerList.append(marker.strip())
            dict = {'run': runName, 'markerList': markerList}
            runArray.append(dict)

    # Select File static pose (.trc)
    root = Tkinter.Tk()
    root.withdraw()
    fullFilePath = tkFileDialog.askopenfilename(filetypes=(("
        Template_files", "*.trc"), ("All_files", "*.*")), title='
        Select_Static_Pose_File')
    poseFileName = ntpath.basename(fullFilePath)
```

```

poseFilePath = ntpath.dirname(fullFilePath)
dictValue = {'FileType': 'Pose', 'FileName': poseFileName, '
    FilePath': poseFilePath}
files.append(dictValue)

# Select File generic model (.osim)
root = Tkinter.Tk()
root.withdraw()
fullFilePath = tkFileDialog.askopenfilename(filetypes=(("
    Template_files", "*.osim"), ("All_files", "*.*")), title=
    'Select_Generic_Model_File')
modelName = ntpath.basename(fullFilePath)
modelFilePath = ntpath.dirname(fullFilePath)

dictValue = {'FileType': 'Model', 'FileName': modelName,
    'FilePath': modelFilePath}
files.append(dictValue)

# Select File Initial Setup (.xml)
root = Tkinter.Tk()
root.withdraw()
fullFilePath = tkFileDialog.askopenfilename(filetypes=(("
    Template_files", "*.xml"), ("All_files", "*.*")), title=
    'Select_Scale_Settings_File')
setupFileName = ntpath.basename(fullFilePath)
setupFilePath = ntpath.dirname(fullFilePath)

dictValue = {'FileType': 'Setting', 'FileName':
    setupFileName, 'FilePath': setupFilePath}
files.append(dictValue)

# Selects if script uses weights defined in settings file
result = tkMessageBox.askquestion("Scaling", "Use_scaling_
    weights_from_file?")
if result == 'yes':
    customScaling = True
else:
    customScaling = False

# returns list of files and boolean for custom scaling and
    bodysegment array
return files, customScaling, runArray

```

B.3 ModifyMarker.py

```
# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#
# Permission is hereby granted, free of charge, to any person
obtaining a copy of
# this software and associated documentation files (the "
Software"), to deal in
# the Software without restriction, including without limitation
the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the
# Software, and to permit persons to whom the Software is
furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND
, EXPRESS OR IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

import xml.etree.cElementTree as et

def ModifyMarker(markerFile, mark, value):
    # INPUT: Marker file, marker to modify, value to modify
    marker by
    # OUTPUT: Modified marker files
```

```

datafile = markerFile
# repeats the modification of marker file for each of the 6
# possible combinations
for i in range(0, 6):
    tree = et.ElementTree(file=datafile)
    datafileMod = datafile.replace('osim', '') + str(i) + '.
        osim'
    # iterates through all markers, modify's marker passed
    for elem in tree.iter(tag='Marker'):
        if elem.attrib['name'] == mark:
            for loc in elem:
                if loc.tag == 'location':
                    a = loc.text.lstrip()
                    split = a.split("_")
                    x = float(split[0])
                    y = float(split[1])
                    z = float(split[2])
                    if i == 0:
                        loc.text = str(x + value) + "_" +
                            str(y) + "_" + str(z)
                    if i == 1:
                        loc.text = str(x - value) + "_" +
                            str(y) + "_" + str(z)
                    if i == 2:
                        loc.text = str(x) + "_" + str(y +
                            value) + "_" + str(z)
                    if i == 3:
                        loc.text = str(x) + "_" + str(y -
                            value) + "_" + str(z)
                    if i == 4:
                        loc.text = str(x) + "_" + str(y) + "
                            _" + str(z + value)
                    if i == 5:
                        loc.text = str(x) + "_" + str(y) + "
                            _" + str(z - value)
    tree.write(datafileMod)

```

B.4 ModifySetup.py

```

# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#

```

```

# Permission is hereby granted, free of charge, to any person
# obtaining a copy of
# this software and associated documentation files (the "
# Software"), to deal in
# the Software without restriction, including without limitation
# the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or
# sell copies of the
# Software, and to permit persons to whom the Software is
# furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND
# , EXPRESS OR IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
# , FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
# WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```

```

import xml.etree.cElementTree as et
from OpenSim_Scaling import log
exclude = []

```

```

def InitModifySetup(setting, model, pose, custom):
    pelvis = ["RASI", "LASI", "LPSI", "RPSI", "LTOR", "RTOR"]
    datafile = setting.FullFileName
    fileName = model.FileName
    pose = pose.FileName
    # repeates the modification of marker file for each of the 6
    # possible combinations

    tree = et.ElementTree(file=datafile)

```

```

# iterates through all markers, modify's marker passed
for elem in tree.iter(tag='GenericModelMaker'):
    for loc in elem:
        if loc.tag == 'model_file':
            loc.text = fileName
for elem in tree.iter(tag='ModelScaler'):
    for loc in elem:
        if loc.tag == 'marker_file':
            loc.text = pose
for elem in tree.iter(tag='MarkerPlacer'):
    for loc in elem:
        if loc.tag == 'marker_file':
            loc.text = pose
for elem in tree.iter(tag='IKMarkerTask'):
    markerName = elem.attrib['name']
    for loc in elem:
        if loc.tag == 'apply':
            if loc.text == 'false':
                exclude.append(markerName)
tree.write(datafile)
print exclude

# input: setting and marker files.
# output: 6 different settings files for scaling method.
def ModifySetupPelvis(setting, model, custom):
    pelvis = ["RASI", "LASI", "LPSI", "RPSI", "LTOR", "RTOR"]
    datafile = setting.FullFileName
    fileName = model.FileName
    # repeats the modification of marker file for each of the 6
    # possible combinations
    for i in range(0, 6):
        tree = et.ElementTree(file=datafile)
        datafileMod = datafile.replace('xml', '') + str(i) + '.xml'
        # iterates through all markers, modify's marker passed
        for elem in tree.iter(tag='GenericModelMaker'):
            for loc in elem:
                if loc.tag == 'model_file':
                    # print 'Before ' + loc.text
                    loc.text = fileName.replace('osim', '') + str(i) + '.osim'
                    # print 'After ' + loc.text

```

```

    for elem in tree.iter(tag='IKMarkerTask'):
        if elem.attrib['name'] in pelvis:
            for loc in elem:
                if loc.tag == 'apply':
                    loc.text = 'true'
                    if not custom:
                        if loc.tag == 'weight':
                            loc.text = '5'
            else:
                for loc in elem:
                    if loc.tag == 'apply':
                        loc.text = 'false'
    tree.write(datafileMod)

def ModifySetupBody(setting, model, custom):
    datafile = setting.FullFileName
    fileName = model.FileName
    body = ["HEAD", "LKNE", "RKNE", "RFIB", "LFIB", "RANK", "
    LANK", "RHEE", "LHEE"]

    for i in range(0, 7):
        if i < 6:
            tree = et.ElementTree(file=datafile)
            datafileMod = datafile.replace('xml', '') + str(i) +
            '.xml'
            # iterates through all markers, modify's marker
            passed
            for elem in tree.iter(tag='GenericModelMaker'):
                for loc in elem:
                    if loc.tag == 'model_file':
                        a = loc.text.lstrip()
                        # print 'Before ' + loc.text
                        loc.text = fileName.replace('osim', '')
                        + str(i) + '.osim'
                        # print 'After ' + loc.text
        else:
            tree = et.ElementTree(file=datafile)
            datafileMod = datafile
            # iterates through all markers, modify's marker
            passed
            for elem in tree.iter(tag='GenericModelMaker'):
                for loc in elem:

```



```

        if loc.tag == 'model_file':
            # print 'Before ' + loc.text
            loc.text = fileName
            # print 'After ' + loc.text
    for elem in tree.iter(tag='IKMarkerTask'):
        if elem.attrib['name'] in body:
            for loc in elem:
                if loc.tag == 'apply':
                    loc.text = 'true'
                if not custom:
                    if loc.tag == 'weight':
                        loc.text = '1'
    tree.write(datafileMod)

def ModifySetupFinal(setting, model, custom):
    datafile = setting.FullFileName
    fileName = model.FileName

    for i in range(0, 7):
        if i < 6:
            tree = et.ElementTree(file=datafile)
            datafileMod = datafile.replace('xml', '') + str(i) +
                '.xml'
            # iterates through all markers, modify's marker
            # passed
            for elem in tree.iter(tag='GenericModelMaker'):
                for loc in elem:
                    if loc.tag == 'model_file':
                        # print 'Before ' + loc.text
                        loc.text = fileName.replace('osim', '')
                            + str(i) + '.osim'
                        # print 'After ' + loc.text
        else:
            tree = et.ElementTree(file=datafile)
            datafileMod = datafile
            # iterates through all markers, modify's marker
            # passed
            for elem in tree.iter(tag='GenericModelMaker'):
                for loc in elem:
                    if loc.tag == 'model_file':
                        a = loc.text.lstrip()
                        # print 'Before ' + loc.text

```

```

        loc.text = fileName
        # print 'After ' + loc.text
    for elem in tree.iter(tag='IKMarkerTask'):
        if elem.attrib['name'] not in exclude:
            for loc in elem:
                if loc.tag == 'apply':
                    loc.text = 'true'
                    if i == 0:
                        line = elem.attrib['name'] + "└" +
                            loc.text
                        log(model, line)
                if not custom:
                    if loc.tag == 'weight':
                        loc.text = '1'
            else:
                for loc in elem:
                    if loc.tag == 'apply':
                        loc.text = 'false'
                    if i == 0:
                        line = elem.attrib['name'] + "└" +
                            loc.text
                        log(model, line)
    tree.write(datafileMod)

def ModifySetup(setting, model, custom, markerList):
    datafile = setting.FullFileName
    fileName = model.FileName
    # repeats the modification of marker file for each of the 6
    possible combinations
    for i in range(0, 6):
        tree = et.ElementTree(file=datafile)
        datafileMod = datafile.replace('xml', '') + str(i) + '.xml'
        # iterates through all markers, modify's marker passed
        for elem in tree.iter(tag='GenericModelMaker'):
            for loc in elem:
                if loc.tag == 'model_file':
                    # print 'Before ' + loc.text
                    loc.text = fileName.replace('osim', '') +
                        str(i) + '.osim'
                    # print 'After ' + loc.text
    for elem in tree.iter(tag='IKMarkerTask'):

```

```

        if elem.attrib['name'] in markerList:
            for loc in elem:
                if loc.tag == 'apply':
                    loc.text = 'true'
                    if i == 0:
                        line = elem.attrib['name'] + "└" +
                            loc.text
                        log(model, line)
                if not custom:
                    if loc.tag == 'weight':
                        loc.text = '1'
            else:
                for loc in elem:
                    if loc.tag == 'apply':
                        loc.text = 'false'
                    if i == 0:
                        line = elem.attrib['name'] + "└" +
                            loc.text
                        log(model, line)
    tree.write(datafileMod)

```

B.5 MultipleScale.py

```

# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#
# Permission is hereby granted, free of charge, to any person
  obtaining a copy of
# this software and associated documentation files (the "
  Software"), to deal in
# the Software without restriction, including without limitation
  the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or
  sell copies of the
# Software, and to permit persons to whom the Software is
  furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
  included in all
# copies or substantial portions of the Software.
#

```

```
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND
# , EXPRESS OR IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
# , FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
# WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
from Scale import *
import sys
```

```
def MultipleScale(location , baseFile):
    # INPUT: Location of script and base settings file
    # OUTPUT: marker with the lowest error, lowest error and
    #         file index of lowest error
    lowestErrorMarker = ""
    lowestErrorValue = float(sys.maxsize)
    lowestErrorIndex = -1
    for i in range(0, 6):
        file = baseFile.FullFileName.replace('xml', '') + str(i)
        + '.xml'
        errorMarker , errorVal = Scale(location , file)
        if (float(errorVal) < lowestErrorValue):
            lowestErrorMarker = errorMarker
            lowestErrorValue = float(errorVal)
            lowestErrorIndex = i

    return lowestErrorMarker , lowestErrorValue , lowestErrorIndex
```

B.6 Scale.py

```
# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of
```

```
# this software and associated documentation files (the "  
Software") , to deal in  
# the Software without restriction , including without limitation  
the rights to use ,  
# copy , modify , merge , publish , distribute , sublicense , and/or  
sell copies of the  
# Software , and to permit persons to whom the Software is  
furnished to do so ,  
# subject to the following conditions :  
#  
# The above copyright notice and this permission notice shall be  
included in all  
# copies or substantial portions of the Software .  
#  
# THE SOFTWARE IS PROVIDED "AS IS" , WITHOUT WARRANTY OF ANY KIND  
, EXPRESS OR IMPLIED ,  
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY  
, FITNESS FOR A  
# PARTICULAR PURPOSE AND NONINFRINGEMENT . IN NO EVENT SHALL THE  
AUTHORS OR COPYRIGHT  
# HOLDERS BE LIABLE FOR ANY CLAIM , DAMAGES OR OTHER LIABILITY ,  
WHETHER IN AN ACTION  
# OF CONTRACT , TORT OR OTHERWISE , ARISING FROM , OUT OF OR IN  
CONNECTION WITH THE  
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE .
```

```
import subprocess
```

```
def Scale(location , scaleFile):  
    # INPUT: location of scaling script and settings file to  
scale with  
    # OUTPUT: marker and error of marker with the greatest error  
    scaleScript = u'&"' + location + '/Script.ps1" _-Filename_ ' +  
        "' + scaleFile + '"'  
  
    # Opens a powershell window and runs the scaling script  
    p = subprocess.check_output(  
        [u'C:/WINDOWS/system32/WindowsPowerShell/v1.0/powershell  
        .exe' , scaleScript])  
  
    output = p.splitlines()
```

```

errorMarker = output[0]
errorVal = output[1]
return errorMarker, errorVal

```

```

def OpenOSIMFile(model):
    # INPUT: Final scaled model.
    # OUTPUT: Opens OpenSim with the final scale model loaded.
    openScript = r'&"C:\OpenSim_3.3\bin\opensim64.exe" --
        console_suppress' + ''' + model.FullFileName + '''

    p = subprocess.check_output([u'C:/WINDOWS/system32/
        WindowsPowerShell/v1.0/powershell.exe', openScript])

```

B.7 SelectScaleFile.py

```

# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of
# this software and associated documentation files (the "
# Software"), to deal in
# the Software without restriction, including without limitation
# the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or
# sell copies of the
# Software, and to permit persons to whom the Software is
# furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
# included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND
# , EXPRESS OR IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
# , FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

```

```

    WHETHER IN AN ACTION
    # OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
    CONNECTION WITH THE
    # SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

import xml.etree.cElementTree as et
import os

def SelectScaleFile(file , index):
    # INPUT: base generic model and the index of the best scaled
    generic model
    # OUTPUT: overites base generic model with best scaled
    generic model
    datafileMod = file.replace('osim', '') + str(index) + '.osim'
    ,
    tree = et.ElementTree(file=datafileMod)
    tree.write(file)

def RemoveOldFiles(model, setting):
    # INPUT: generic model and settings file
    # OUTPUT: deletes the temp generic model and settings files.
    for i in range(0, 6):
        os.remove(model.FullFileName.replace('osim', '') + str(i)
            + '.osim')
        os.remove(setting.FullFileName.replace('xml', '') + str(
            i) + '.xml')

```

B.8 Script.ps1

```

# This file is part of OpenSim_Scaling
#
# Copyright (c) 2017, Michael Buckingham
#
# Permission is hereby granted, free of charge, to any person
# obtaining a copy of
# this software and associated documentation files (the "
# Software"), to deal in
# the Software without restriction, including without limitation
# the rights to use,
# copy, modify, merge, publish, distribute, sublicense, and/or

```

```

    sell copies of the
# Software, and to permit persons to whom the Software is
    furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be
    included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND
    , EXPRESS OR IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
    , FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
    WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
    CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```

```

[CmdletBinding()]
Param(
    [Parameter(Mandatory=$true)]
    [string]$Filename
)
$marker = $null
$errorVal = $null
$errorLine = $null

try{
    $Scale = scale -S $Filename

    foreach($line in $Scale){
        if($line -like "Frame_ at_(t=*)"){
            $errorLine = $line
        }
    }
    if($errorLine -ne $null){
        $max = $errorLine.split(",").get(2).trim()
        $Marker = $max.split("_").get(1).replace("(", "").replace(
            ")", "")
    }
}

```



```
        $errorVal = $max.split(" ").get(0).replace("max=", "")
        return $marker, $errorVal
    }
    else{
        return "Scale_Null"
    }
}
catch{
    return "Scaling_Error"
}
```


Appendix C

Tabulated Results

Table C.1: Comparison of script and manual scaling error

Subject	Script Error (m)	Manual Error (m)	Cycles
20170523AH	0.0425	No Data	152
20161222JW	0.0072	0.0242	313
20170109CC-Stairs	0.0176	0.0317	306
20170109CC	0.0143	0.0188	302
20170327DG	0.0604	0.0221	299
20170724JR	0.0271	No Data	793
20170424JT	0.0181	0.0114	373
20161208JT	0.0143	0.0289	241
20170515LS	0.0196	0.0159	393
20170223MB-Original	0.0237	No Data	331
20170223MB-Modified	0.0162	No Data	214
20170704PT	0.0105	0.0125	169
20161213RA	0.0109	0.0113	422

Table C.2: Comparison of script with manual marker weighting and manual scaling error

Subject	Manual Error (m)	Script With Manual Weighting (m)
20161222JW	0.0242	0.0125
20170109CC-Stairs	0.0317	0.0182
20170109CC	0.0188	0.0166
20170327DG	0.0221	0.0153
20170424JT	0.0114	0.035
20161208JT	0.0289	0.0136
20170515LS	0.0159	0.0098
20170704PT - JN	0.0125	0.0151
20170704PT - JM	0.0443	0.0195
20170704PT - PT	0.0155	0.0168
20170704PT - TN	0.0166	0.0231
20161213RA	0.0113	0.057

Table C.3: Comparison of Scaling Time

Subject	Surface Time (min)	Laptop Time (min)	Desktop Time (min)	Average Time (h)	Manual Time (h)	Cycles
20170523AH	9.44	8.28	6.95	0.14	40*	152
20161222JW	20.7	19.56	18.25	0.33	24	313
20170109CC-Stairs	17.97	16.79	16.38	0.28	20	306
20170109CC	18.29	17.44	15.94	0.29	20	302
20170327DG	20.24	19.4	16.52	0.31	16	299
20170724JR	49.03	47.05	42.45	0.77	24*	793
20170424JT	22.31	21.48	18.96	0.35	30	373
20161208JT	14.1	13.89	11.15	0.22	8	241
20170515LS	26.83	25.9	21.73	0.41	32	393
20170223MB-Original	26.4	20.58	16.73	0.35	40	331
20170223MB-Modified	16.68	12.71	10.74	0.22	40	214
20170704PT	10.88	10.43	8.45	0.17	8	169
20161213RA	27.97	25.66	21.34	0.42	16	422

* Manual scaling incomplete

Table C.4: Comparison of Scaling Method to Static Pose

Subject	Script	Manual	Script with Manual Weight
20161222JW	0.4939	0.5016	0.5049
20170109CC	0.0083	0.146	0.0882
20170109CC-Stairs	3.2867	3.2809	3.2873
20170327DG	1.5546	1.5598	1.5612
20170424JT	0.0007	1.6104	1.609
20161208JT	1.1678	1.8498	1.8495
20170515LS	1.826	1.829	1.8247
20170704PT	2.6541	2.6501*	2.6541*
20170704PT - JN	2.6541	2.6538	2.6538
20170704PT - JM	2.6541	2.653	2.6515
20170704PT - PT	2.6541	2.6532	2.6512
20170704PT - TN	2.6541	2.6501	2.6541
20161213RA	1.1158	4.9643	1.1183

* Average based of individual scaler results

Table C.5: Comparison of Scaling Methods Inverse Kinematic Error

Error Type	Total Square Error	RMS	Max
Step Up	3.9991	11.5465	25.8688
Step Up Manual	3.973	11.9903	26.3344
Step Down	3.5404	11.3412	24.3074
Step Down Manual	3.1298	10.8413	22.119
Walking	2.1106	8.9139	18.9861
Walking Manual	3.973	11.9903	26.3344

Appendix D

Meeting Attendance Form

Consultation Meetings Attendance Form










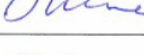



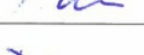
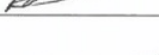
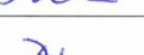



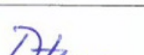






Week	Date	Comments (if applicable)	Student's Signature	Supervisor's Signature
2	8/8/17	Review of scaling process		
3	15/8/17	review of scaling and automation methodology		
4	24/8/17	Merger manipulation script		
5	01/09/17	review of test thesis documentation progress		
6	04/09/17	Review of progress report expectations		
7	12/09/17	Review of test scaling code progress		
MS1	19/09/17	Review of code and testing progress via email		
MS2	26/09/17	Review of testing progress		
8	03/10/17	Review of runtime data		
9	10/10/17	Review of script errors		
10	17/10/17	Review of scaled model comparison		
11	24/10/17	Review of document progress		
12	03/11/17	Thesis document, poster and presentation expectations		

Figure D.1: Meeting Attendance Form

Bibliography

- [1] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.
- [2] J. A. Reinbolt, J. F. Schutte, B. J. Fregly, B. I. Koh, R. T. Haftka, A. D. George, and K. H. Mitchell, "Determination of patient-specific multi-joint kinematic models through two-level optimization," *Journal of Biomechanics*, vol. 38, no. 3, pp. 621–626, 2005.
- [3] J. A. Egea, R. Mart, and J. R. Banga, "An evolutionary method for complex-process optimization," *Computers & Operations Research*, vol. 37, no. 2, pp. 315–324, 2010.
- [4] R. Wehrens and L. M. Buydens, "Evolutionary optimisation: a tutorial," *TrAC Trends in Analytical Chemistry*, vol. 17, no. 4, pp. 193–203, 1998.
- [5] L. A. Hannah, "Stochastic optimization," *International Encyclopedia of the Social & Behavioral Sciences*, pp. 473–481, 2015.
- [6] J. J. Schneider and S. Kirkpatrick, *Stochastic optimization*. Springer, 2006.
- [7] K. Marti, *Stochastic optimization methods*, 2nd ed. Springer, 2010.
- [8] "How scaling works," 2017. [Online]. Available: <https://simtk-confluence.stanford.edu:8443/display/OpenSim/How+Scaling+Works>
- [9] "Inverse kinematics tasks for scale," 2017. [Online]. Available: <https://simtk-confluence.stanford.edu:8443/display/OpenSim/Inverse+Kinematics+Tasks+for+Scale>
- [10] "How inverse kinematics works," 2017. [Online]. Available: <https://simtk-confluence.stanford.edu:8443/display/OpenSim/How+Inverse+Kinematics+Works>
- [11] A. Mohboobin, "An automated iterative method for adjusting virtual model markers in an opensim model," Ph.D. dissertation, University of Pittsburgh, 2013.