A HUMAN MOTION DATABASE: THE COGNITIVE AND PARAMETRIC SAMPLING OF HUMAN MOTION

by

ARNAB BISWAS

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON
DECEMBER 2010

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Gutemberg Guerra-Filho for guiding me and having the patience to drive me towards the completion of this thesis. My interactions with him have been more like a friend rather than an advisor. I am really grateful to him for giving me the oppurtunity to pursue my M.S. with thesis at University of Texas at Arlington under him.

I would like to thank Dr. Manfred Huber and Dr. Gian Luca Mariottini for serving in my committee. I would also like to thank Dr. Filia Makedon for the access to the Heracleia Lab and the permission to use the motion capture system.

I am happy to have made such wonderful friends over here without whom the journey would have been really bland. I would like to specially mention Venkata Dinesh Jammulla from whom I learnt so much. Also I would like to mention Anirban Maiti, Arjun Dasgupta, Debanjana Maiti, Gauri Vakde, Jyothi Keshavan, Mahashweta Das, Kausik Kayal, Roman Arora and Senjuti Basu Roy, I had many memorable times with all of them.

Finally I will acknowledge my family including my grandmother Mrs. Usha Rani Maji and my belated grandfather Mr. Narendranath Maji whose influences are the greatest in me, my father Dr. Abani Biswas for giving me unconditional support. I would like to thank my sister Mrs. Moumita Sinha and brother in law Mr. Ritwik Sinha for believing in me during difficult times. Last but not the least, I must mention the tiny Nishaant Sinha for providing unlimited comedic relief.

November 16, 2010

ABSTRACT

A HUMAN MOTION DATABASE: THE
COGNITIVE AND PARAMETRIC
SAMPLING OF HUMAN
MOTION

Arnab Biswas, M.S.

The University of Texas at Arlington, 2010

Supervising Professor: Gutemberg Guerra-Filho

Motion databases have a strong potential to guide progress in the field of machine recognition and motion-based animation. Existing databases either have a very loose structure that do not sample the domain according to any controlled methodology or too few action samples which limits their potential to quantitatively evaluate the performance of motion-based techniques. The controlled sampling of the motor domain in the database may lead investigators to identify the fundamental difficulties of motion cognition problems and allow the addressing of these issues in a more objective way. In this thesis, we describe the construction of our Human Motion Database using controlled sampling methods (parametric and cognitive sampling) to obtain the structure necessary for the quantitative evaluation of several motion-based research problems. The Human Motion Database is organized into several components: the praxicon dataset, the cross-validation dataset, the generalization dataset, the compositionality dataset, and the interaction dataset. The main contributions of this thesis include (1) a survey of human motion databases describing data sources related to motion synthesis and analysis problems, (2) a sampling methodology that takes advantage of a

systematic controlled capture, denoted as cognitive sampling and parametric sampling, (3) a novel structured motion database organized into several datasets addressing a number of aspects in the motion domain, (4) a study of the design decisions needed to build a custom skeleton to generate joint angle data from marker data, and (5) a study of the motion capture technologies and the general optical motion capture workflow including capturing and post processing data.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

LIST OF ILLUSTRATIONS ix

LIST OF TABLES xi

Chapter Page

1. INTRODUCTION 1

2. RELATED WORK 5

2.1 Existing Mocap Databases 7

2.1.1 CMU Mocap Database 7

2.1.2 IEMOCAP Database 8

2.1.3 HumanEva Database 9

2.1.4 CMU Motion of Body Database 11

2.1.5 HDM05 Motion Capture Database 11

2.1.6 Biological Motion Library 12

2.1.7 Korea University Gesture Database 13

2.1.8 Human Identification at Distance Database at Georgia Tech 14

2.1.9 ICS Action Database 15

2.2 Analysis and Features of the Human Motion Database 16

3. EQUIPMENT & SUBJECTS 20

3.1 Motion Capture Systems 20

3.1.1 Types of motion capture systems 20

3.1.2 Optical Motion Capture System 24

3.2 Equipment 24
3.2.1 The Vicon optical motion capture system 24
3.2.2 Camera Configuration 27
3.2.3 Capture Room 28
3.2.4 Marker Configuration 29
3.2.5 Software provided by Vicon 29
3.3 Volunteers 30
3. 4 Organization in terms of actions 32
4. CAPTURE AND POST-PROCESSING 34
4.1 Capture 36
4.1.1 Calibration 36
4.1.1.1 Extrinsic and Intrinsic Camera Calibration 36
4.1.1.2 Euclidean Calibration 37
4.1.1.3 Subject Calibration 38
4.2 Capture Session 39
4.2 Post Processing 41
4.2.1 Reconstruct and Auto Labeling 41
4.2.2 Clean data 41
4.2.3 Generate Output 42
4.2.4 Types of Capture Errors 44
4.2.5 Techniques Used to Correct Errors 49
4.2.6 Generation of output data 54
5. SKELETON DESIGN 55
5.1 Skeleton Design Considerations 56
5.2 Skeleton Hierarchy and Bones 58

5.3 Automatic Construction of Skeleton 60
6. METHODOLOGY 65
6.1 Understanding Human Motion 65
6.2 Database Structure 67
6.2.1 Praxicon 67
6.2.2 Range of skeletal structures 69
6.2.3 Multiple ways to perform each action 69
6.2.4 Complex Motion 71
6.3 Research Problems 72
6.3.1 Classification and Identification 72
6.3.2 Retargeting 74
6.3.3 Generalization 75
6.3.4 Transitioning 77
6.3.5 Splicing 77
7. CONCLUSION 79

APPENDIX

A. WORLKFLOW TO GENERATE BVH FILES IN VICON BLADE 81
B. SCRIPT TO GENERATE CUSTOM SKELETON FROM THE MARKER DATA 84
C. LIST OF ACTIONS 107
REFERENCES 117
BIOGRAPHICAL INFORMATION 119

LIST OF ILLUSTRATIONS

Figure Page

2.1 Existing Motion Databases. 6

3.1 Architecture of the Vicon system. 25

3.2 Vicon mocap devices. 26

3.3 Camera setup in the capture room. 27

3.4 Camera configuration in the capture room. 28

3.5 The distribution of height and weight for all volunteers as a Voronoi diagram. 31

3.6 Distribution of subject parameters. Male subjects in red and female subjects in blue. 32

4.1 The marker configuration. 35

4.2 The Capture workflow. 36

4.3 The Extrinsic/Intrinsic Camera Calibration workflow. 37

4.4 The Euclidian Calibration workflow. 38

4.5 Subject standing in T pose. 39

4.6 The Subject Calibration workflow. 39

4.7 The Capture Session workflow. 40

4.8 The time taken in minutes for the cleaning of actions per subject. 42

4.9 Time taken to complete the first iteration of cleaning for each (a) action (b) subject. 43

4.10 Trajectories of markers show artifacts. 44

4.11 Various artifacts such as (a) disappearing markers (b) falling markers 45

4.12 Markers being mislabeled 46

4.13 The same frame before and after the cleaning process. 50

5.1 Typical markers used to calculate the parameters of each bone. 57

5.2 The hierarchy used build our custom skeleton to generate bvh files. 58

5.3 Step by step construction of skeleton. 62

6.1 Meaningful, observable, voluntary actions: jump, kick, and step up. 68

6.2 Target points in 3D space for actions in the generalization dataset (a) reach (b) kick 69

6.3 Interaction database showing actions handshake and passing a suitcase 73

6.4 Different skeletal structures. 74

6.5 Reach motion data showing nine target points of the whole dataset 76

6.6 Transitioning data showing a Walk action, a Jump action and the ground truth data for a Walk and Jump action 77

6.7 Splicing shows a walk action and a wave action and then ground truth of a walk and wave action. 78

LIST OF TABLES

Table Page

2.1 Summary of features of existing databases 17

5.1 The set of markers associated with each bone to set up the location, direction and oriention of each bone 60

CHAPTER 1

INTRODUCTION

Understanding the fundamentals of human motion involves the study of various aspects of this phenomenon. The same action can be performed in various manners. Human beings are equally adept at combining actions sequentially and concurrently to produce more complex activities. While actions require coordination between different body parts of the same subject, interactive actions require coordination among two or more subjects. The study of human motion must encompass the full repertoire of these variations.

The advancement of motion capture technology in recent years has paved the way for good quality motion data which can be used to study human motion precisely. This technology has been used for the construction of benchmark databases with attributes relevant to several human motion problems. These standard databases should create an even measuring ground for the quantitative evaluation of methods by all researchers. However, the existing motion capture databases either have a very loose structure or too few action samples which limits their potential to quantitatively evaluate the performance of motion-based techniques.

The correctness of an algorithm for motion synthesis or analysis is mostly assessed visually. This is very subjective in nature and does not provide a uniform method of evaluation. On the other hand, the performance of any algorithm can be quantitatively evaluated when they are tested with precise data providing sampling of all motion variation in a principled controlled fashion.

Motion databases contain data samples used to train and test recognition and generation algorithms. Consequently, these databases are crucial for a proper evaluation of motion synthesis and analysis methods. For these reasons, motion databases have a strong

potential to guide progress in the field of machine recognition and motion-based animation. Existing databases do not sample the domain according to any structured controlled methodology or have very limited resources. The controlled sampling of the motor domain in the database may lead investigators to identify the fundamental difficulties of motion cognition problems and allow the addressing of these issues in a more objective way. Therefore, there is a need for motion databases built with a controlled methodology. In this thesis, we describe the construction of our Human Motion Database (HMD) using controlled sampling methods (parametric and cognitive sampling) to obtain the structure necessary for the quantitative evaluation of several motion-based research problems.

The unique features of the Human Motion Database are:

• The praxicon dataset, a corpus of human motion from a single subject with a wide range of more than 350 commonly performed actions. This organized vocabulary of usual actions (no specific domain) is designed to aid the training and testing phases of motion indexing, automatic animation, and action recognition problems.

• The cross-validation dataset, a subset of 70 actions in the praxicon performed by a large set of 50 different subjects. The cross-validation dataset provides a range of skeletal structures distributed over height, weight, gender, and age of the subjects.

• The generalization dataset, a set of representative actions where each action is performed several times according to different manners represented by specific parameters. For instance, the reach action is executed for all different discrete 3D target locations in front of the subject, the walk action is obtained for 8 different directions (N, NE, E, SE, S, SW, W, NW) while facing forward (i.e., N), and the sitting action is capture for a range of seat heights.

• The compositionality dataset, a set of representative pairs of individual actions that may be composed sequentially or concurrently into a more complex action.

• The interaction dataset, a praxicon with more than 140 actions where two different subjects interact with each other.

The main contributions of this thesis include (1) a survey of human motion databases describing data sources related to motion synthesis and analysis problems, (2) a sampling methodology that takes advantage of a systematic controlled capture, denoted as cognitive sampling and parametric sampling, (3) a novel structured motion database organized into several datasets addressing a number of aspects in the motion domain, (4) a study of the design decisions needed to build a custom skeleton to generate joint angle data from marker data, and (5) a study of the motion capture technologies and the general optical motion capture workflow including capturing and post processing data.

The rest of this thesis is organized as follows.

Chapter 2 describes related work with a survey of existing human motion databases. We discuss and review several motion capture databases, referred as mocap databases from here on, which have been made available for research purposes by educational institutions. In this discussion, we identify, analyze, and evaluate features and principles of these existing databases. We also consider the potential issues and limitations of these databases that lead to the creation of the Human Motion Database. The databases considered in this chapter demonstrate that such repositories of motion data are essential for the advancement in a number of fields. They are the inspiration and guidelines for creating a database to the advancement of automated animation, robotics, surveillance, human-machine interfaces, and gait recognition.

Chapter 3 describes the equipment, setup, and human resources used for the capture of the Human Motion Database. First, we discuss different kinds of motion capture systems that can be used to record human motion. Next, the chapter describes the motion capture hardware used to create the HMD. Finally, the details on the anthropomorphic distribution of subjects in HMD are discussed. We present our sampling methodology and the organization of our database

In Chapter 4, we discuss the capture and post processing stages of the creation of the database. Capture is the process by which a phenomenon of interest is recorded and stored digitally. Motion capture primarily records the movements of a human or in some cases animals or birds to recreate the motion for various applications. This chapter goes through the various workflows for the capturing process such as camera calibration, volume calibration, subject calibration, and capture sessions. Next, the chapter discusses the various errors present in the captured raw data and the techniques by which the data was cleaned and post processed to finally obtain the joint angle data output.

In chapter 5, we first discuss the challenges and design approach of constructing a skeleton from a marker set. We also describe the creation of the hierarchy of the skeletal structure. Finally, we discuss the theory, methods, and code used to implement our automatic skeleton building process.

Chapter 6 introduces several problems in motion synthesis and analysis that may benefit of this database. It describes the structure of the Human Motion Database and the methodology developed to collect several aspects of human motion. First, we discuss and justify the need for a database like the Human Motion Database. Then, we introduce the goals and unique features in the database. Finally, we enumerate which areas, problems, and applications the database is targeted.

Chapter 7 contains our conclusions where we mention what value the Human Motion Database brings to the research community. We go over the problems that it intends to solve and how it achieves its goals.

CHAPTER 2

RELATED WORK

In this chapter, we discuss and review different motion capture databases, referred as mocap databases from here on, which have been made available for research purposes by educational institutions. In this discussion, we identify, analyze, and evaluate features and principles of these existing databases. We also consider the potential issues and limitations of these databases that lead to the creation of the Human Motion Database. The databases considered in this chapter demonstrate that such repositories of motion data are essential for the advancement in a number of fields. They are the inspiration and guidelines for creating a database to the advancement of automated animation, robotics, surveillance, human-machine interfaces, and gait recognition.

Motion capture databases have been constructed to serve as input for various research problems including gesture recognition [10], identification of motion properties [11], motion blending and synthesizing [12], pose estimation and tracking [14], interactive communication [5]. Some mocap databases were built as a general repository to address various problems [1].

The construction of a mocap database creates a level ground and standardizes the metric for the evaluation of many open problems so that the research community is able to compare different approaches. A widely used and accepted database is a necessary condition to determine and evaluate the state of the art in current research. The performance of different methods is evaluated using the same datasets as a benchmark and, consequently, a feasible comparison between several approaches is achieved. In this sense, the design and sampling method of the mocap database is an important characteristic to guide the progress of research efforts.

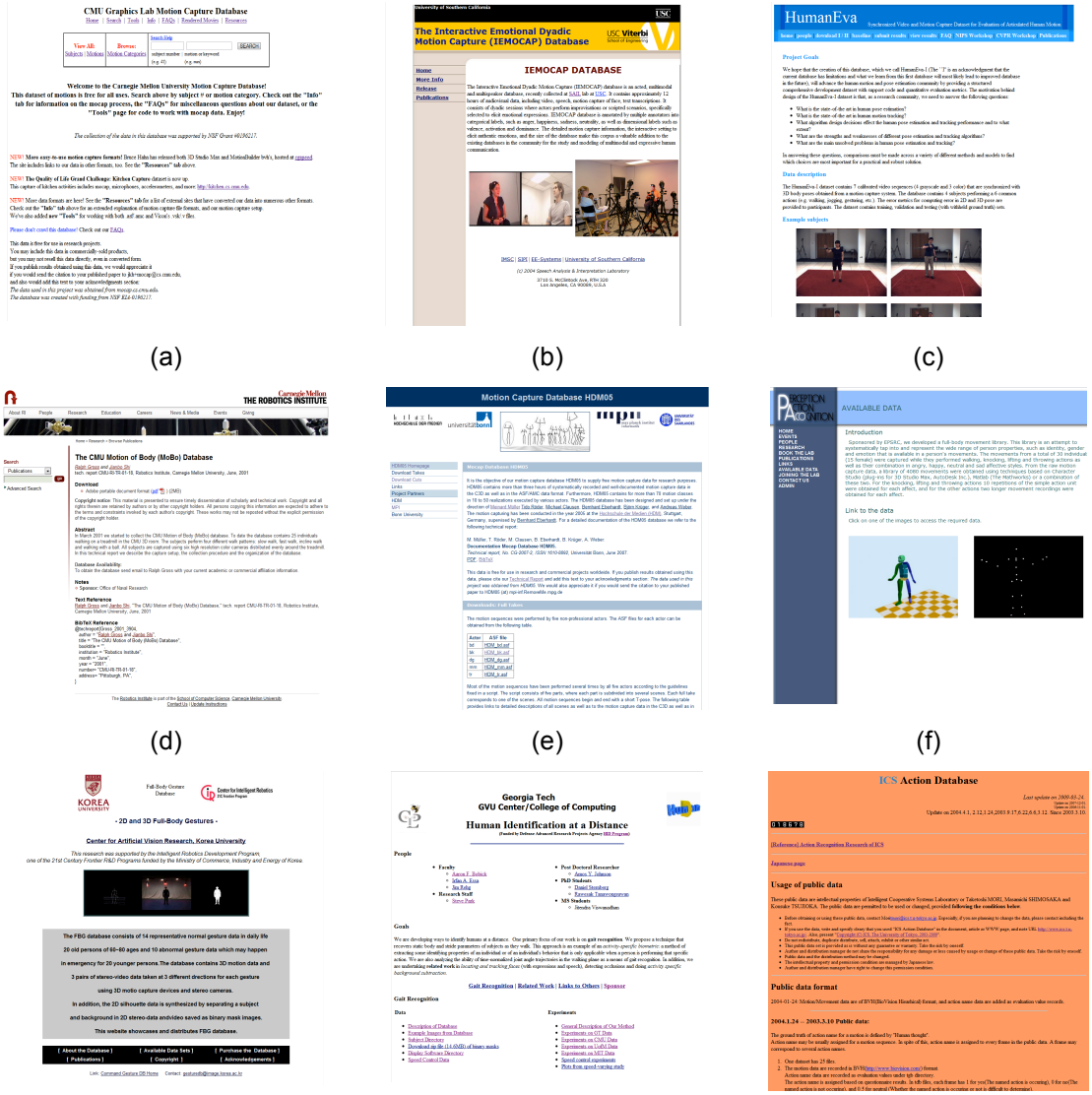


Figure 2.1: Existing Motion Databases: (a) CMU MoCap Database, (b) IEMOCAP Database, (c) HumanEva Database, (d) CMU Motion of Body Database, (e) HDM05 Motion Capture Database, (f) Biological Motion Library, (g) Korea University Gesture Database, (h) HID Database at Georgia Tech, (i) ICS Action Database

The media formats of motion databases vary from video data of the whole body performing various actions, video data of facial expressions, motion capture data of the whole body, and motion capture data of the face. Synchronous motion capture data and video data are also supplied as ground truth and test data pairings.

2.1 Existing Mocap Databases

2.1.1 CMU Mocap Database

The CMU motion capture database [1] is one of the earliest and most widely used motion databases in the research community. It contains over 6 hours of full body motion capture data with 2605 different motion clips (or trials) organized in 6 categories and 23 subcategories. The database is a large repository of many different kinds of motions divided into the following 6 major categories: (1) Human Interaction, (2) Interaction with Environment, (3) Locomotion, (4) Physical Activities & Sports, (5) Situations & Scenarios, and (6) Test Motions. The subcategories include classes such as two subjects, playground, path with obstacles, running, jumping, basketball, dance, boxing, pantomime, communication gestures and signals to name a few.

The actions have been performed by 144 subjects (some subjects are the same person) which correspond to 144 sessions. The database has no formal structure or goal to address a particular problem and was built just to provide a free source of animation data for the research community. Most sessions have a different set of actions. For example, while one set may have a set of similar or slightly different walk actions, another set contains both walk and run actions, and another set may contain basketball moves. Even the same action performed across different sets may or may not have been performed in the same way. For example, a soccer ball can be kicked in many different ways and it depends on how the subject performs the motion.

The CMU mocap database has been applied to many research problems in motion-based animation and beyond. More recently, the database has been enhanced with a set of

kitchen activities. This kitchen dataset has 43 subjects performing 5 different sequences of actions to cook 5 different recipes. The mocap data in the kitchen dataset is also synchronized with video, audio, and other sensor data such as accelerometers and other internal measurement units. This section of the CMU mocap database contains well-defined actions as well as data of special cases of anomalies like sudden fire, smoke, or falling dishes. The definition of the action concerns how rigidly the action is executed in terms of subject training. Natural variation happens when action definition is left to the subject's interpretation.

The lack of structure of the action sampling brings variation which is both the advantage and disadvantage of this database. While variation is natural in human motion, actions which have the same meaning semantically (such as reach) can be very different when performed by different people as well as where the person is reaching. The action variation, lacking structure and proper parameterization, is a realistic sampling of the generalized nature of the corpus of human motion, where one action can be instantiated by many different motions or by a conjugation of a simpler set of motions. On the other hand, the use of such an unstructured database requires additional effort from the user to subjectively evaluate semantically similar motions by manually going through the motion frame by frame. For example, in the action recognition problem which concerns the identification of a particular query action such as run, it would be beneficial to collect well-defined motion data. A well-defined motion is performed according to precise performance guidelines such as accurate descriptions and demonstrations. If natural variation is required for that particular action, it is essential that there is an uniform distribution of the variations so that the learning algorithms do not introduce any bias in the classification process. Herein lies the limitaions of the CMU Mocap database as there is no uniformity in the sampling of variations or consistency across the same motion performed by differernt subjects. Hence, there is a need to create a subset of the database by manually evaluating every action for the purpose of any research problem. The HMD solves this problem by providing well-defined action sets and a uniform distribution of variation.

2.1.2 IEMOCAP Database

The IEMOCAP database [5], collected at the University of South California, contains audio as well as motion capture data of the human face, head, and hands. They assume the emotion of a human being is expressed by a combination of multiple features or channels which include various properties of the speech, facial expressions, gestural movements of the hand, motion of the head and torso, and gaze. The IEMOCAP database was created to provide the first integrated datasets of the above mentioned channels which will assist in the study of modeling "expressive human communication".

The IEMOCAP database is a well-structured database developed according to the requirements for the study of emotional expression. Initially, the design and construction of the database considered the limitations and scope of already existing datasets. Special importance was given to have a large corpus of emotions performed by many (10) trained subjects or experienced actors to get realistic data. The emotions expressed in the database include happiness, anger, sadness, frustration, neutral, disgust, fear, excitement, and surprise. The parameters taken into account in the designing of the database include scope (number and distribution of properties such as gender and variation of emotions), naturalness (sessions performed by actors), context (5 minute long dialogs to capture how one emotion can change from one to another), descriptors (supplying additional meta data to describe the performance in terms of parameters such as intensity, variability, and others).

The systematic and planned nature of the database is one of its great advantages. However, the database fails with respect to the number of subjects involved. Only ten subjects are used over 5 dyadic sessions, where two subjects are communicating with each other. These sessions give very little scope of distribution of data over parameters such as gender and culture. Our HMD has a much larger distribution over 50 subjects and the scope of capturing motion variations is much higher which enables the learning and discovery of more robust methods.

2.1.3 HumanEva Database

The HumanEva database [14] is an effort to provide data that will assist in pose estimation and tracking of human motion and establish quantitative evaluation metrics. The HumanEva database was motivated by the lack of quantitative analysis of pose estimation and tracking algorithms as a limitation in the current state-of-the-art algorithms dealing with the problem. The correctness of an algorithm was mostly assessed visually which was very subjective in nature and did not provide a uniform method of evaluation. The HumanEva database, in an effort to address this issue, provides ground truth data to enable the testing of pose estimation and tracking algorithms. For every video of a human figure performing some action, there is the corresponding motion capture data of the same performance. For any algorithm trying to estimate pose using various vision techniques, the result can be readily compared quantitatively to the true pose of the body. By using motion capture technology to provide the ground truth, the subjects were able to perform more flexible actions. In earlier databases, the actions were performed in a constrained way in order to provide the ground truth. For example, instead of having predefined restrictions on the performance of certain actions such as traversing a circular groove of known dimension on a table with the tip of the hand [9], the HumanEva database used a much more relaxed action definition. The HumanEva dataset provides real noise and real data with a degree of accuracy very close to synthetic data. The dataset also provides a standard evaluation metric which can be used to evaluate algorithms using the data.

There are 6 actions performed by 4 subjects in the HumanEva dataset. There is a tradeoff between the realism of the video data and the accuracy of the ground truth mocap data. The subjects had to wear "natural clothing" in order to provide the "complexity posed by moving clothing". Motion capture with markers is typically done while wearing tight fitting bodysuits to minimize the effect of moving clothes. The use of realistic clothing leads to inaccurate motion capture data, as reconstructed motion data will contain artifacts such as movements which are

not part of the body but from the clothes. The HumanEva database has undergone "minimal post-processing". Errors in the motion capture process, such as missing or mislabeled markers, have been identified and not included in the quantitative evaluation. The limitation of the HumanEva database is the errors in the motion capture data provided as ground truth. On the other hand, our HMD contains motion capture data which has gone through extensive post processing to maintain its quality in terms of accuracy.

2.1.4 CMU Motion of Body Database

The CMU Motion of Body (MoBo) Database [8] was one of the first motion databases to advance research on human gait from a biometric point of view. Biometrics usually focuses on identification of human from their gait characteristics. The database contains four different styles of walk (slow, fast, inclined, and with a ball in the hand) performed on a treadmill by 25 subjects. The data is in the form of video data from six different cameras placed at six different locations around the subject. A disadvantage of this database is the fact that only images are available. The frame rate is 30 fps which is quite low when compared to motion capture databases currently available. However, being one of the first such databases, it contains a good variation and range in terms of number of subjects. Although there are only four motions captured, they are relevant to the problem of subject identification by gait. Our HMD has motion capture data with a much higher sampling rate recorded at 120 fps. Furthermore, besides gait, the HMD contains a set of 70 different actions performed by 50 different subjects.

2.1.5 HDM05 Motion Capture Database

The Hochschule der Medien (HDM05) database [12] provides data "that can be freely used for systematic research on motion analysis, synthesis and classification". The HDM05 database was motivated by the lack of data for research in reusing and synthesizing mocap data. More specifically, motion data for editing, morphing, and blending. These problems require a large variety of motions to be able learn from, morph, and blend. The HM05 database is categorically divided into groups describing the general kind of motion such as locomotion,

grabbing, sports, and others. These categories are again subdivided into motion classes which describe one particular motion. The HDM05 dataset consists of around 100 different motion classes performed by 5 different actors. The trials were captured with detailed instruction on how to perform them. Each trial may contain more than one motion concatenated one after another. However, the trials are edited and divided into 1,457 smaller motion clips where each motion clip contains only one type of motion. These different motion clips combined amount to each motion class having between 10 and 50 realizations of the same action. These realizations may be slightly different to each other but semantically have the same representation. The multiple realizations of the same action are necessary to learn and classify the motion.

The data is available in both animation asf/amc format as well as raw mocap data c3d format. The database has a good range of actions as well as variation of same action such as walking with different weights. The repeated actions of the same class are very good for training. The HDM05 database is very well structured where all the motions are well-defined (*i.e.*, performed according to a certain script). One of the limitations of this database is the low number of 5 actors. This may lead to non scalable classifiers with respect to the size and shape of the skeleton or to the variation of execution over a broader range of subjects. Another limitation is the lack of symmetry of the actions performed by the actors. As mentioned above, each motion class has between 10 and 50 realizations or repetitions. However, not all subjects contribute equally to the class. For example, some classes such as shuffle do not have the performance from one subject altogether. The distribution of our HMD is much more uniform as 50 subjects perform the same set of 70 actions.

2.1.6 Biological Motion Library

The Biological Motion library [11] has been built by the University of Glasgow, Scotland in order to analyze and identify motion properties or features such as gender, identity, and affects in the form of emotions (*e.g.*, angry, sad, happy, neutral) in a motion. The Biological Motion Library was inspired by a lack of data to study the perception of human motion. This

dataset is structured and evenly distributed across the parameters mentioned above. There are 15 male and 15 female actors who perform three types of actions: walk, a set of arm motions (knocking, throwing and lifting), and sequences of a walk performed between two arm actions. The walk actions and sequenced actions are performed twice while the arm actions have five iterations each. The whole set of actions is performed according to four different emotional affecters. One of the important aspects of capturing the data has been noted as not demonstrating the actions for the actors. On the other hand, the actors were asked to perform with the aid of a script describing the action as well as the emotion. This enables variability of performance which is natural between different people performing the same action and does not induce any bias in the performance.

The commonly used motion capture formats such as bvh and asf/amc are not provided. The data is in the form of marker data which stores the point coordinates of individual markers in Character Studio csm motion capture file format. Hence, only translational data is provided while rotational data can only be obtained by designing a skeleton model and developing a special purpose method. This limits the applicability of the library in other fields although the sequence data can be used as ground truth for synthetically generated data to be applied to motion-based animation problems. Since the goal of the library is to solely identify properties such as gender, identity, and affects, there was no need for a huge corpus of motions, rather it has the same motions performed with an even distribution of gender and affects over the 30 subjects. The range of actions covered by the database is not enough for a motion library trying to encompass the whole gamut of human motion. Our HMD provides over 350 different actions that are distributed over gender, age, and size of skeleton.

2.1.7 Korea University Gesture Database

The Korea University Gesture (KUG) database [10] was created to assist human tracking and gesture recognition problems. It contains motion capture data along with synchronized video data from multiple stereoscopic cameras from different directions. The KUG

database contains 20 actors evenly distributed with respect to gender and between the ages of 60 and 80 years old. The set of actions or gestures captured are divided into three parts: 14 normal gestures (such as sitting on a chair, walking at a place, or other "human gestures in everyday life"), 10 abnormal gestures (different forms of falling such as forward, backward, or from a chair), and 30 command gestures (well-defined and commonly used in gesture based studies such as yes, no, pointing, and drawing numbers). The stereoscopic cameras provide noisy depth information while the motion capture data provides very accurate 3D data.

The flexibility of this database lies in its ability to be utilized in multiple approaches. Both the mocap as well as the stereoscopic data can be used to aid gesture recognition in various approaches. The mocap data can also act as an evaluation by providing the ground truth for pose estimation. The KUG database was inspired by the lack of data concerning various gestures or actions performed specifically by a large number of subjects. Hence, the database aims at providing a structured action set performed by 20 actors. However, the selection of actions for the database is a drawback. There is no concrete reason given to choose the sets of actions in the normal and abnormal subsets. The choice of 14 normal actions which are generally performed in everyday life is very subjective and no explanation is given as to why certain actions were chosen over others. The abnormal dataset also contains actions involving fall which is an involuntary action and very difficult to replicate in a natural way. The actors are also wearing motion capture suits and not normal clothing. This is undesirable for approaches which use vision techniques on the video data as the actors are not wearing realistic clothing. Our HMD has a larger number of subjects and a wider variety of actions which are chosen using novel techniques to ensure a good distribution and capture variability of motion.

2.1.8 Human Identification at Distance Database at Georgia Tech

The Human Identification at Distance (HID) database [3] at Georgia Tech has been built with the primary goal of gait recognition. It has been applied in biometrics to identify a person from properties in the motion while doing a specific action. The database has 20

subjects performing walk motions. The motion capture data is supplemented with video data of the subjects performing the action walk. The video data includes video of the subject wearing the motion capture suit in the motion capture volume, video of the subject in normal clothing performing the same action indoors from two different angles, and video of the subject in normal clothing performing the same action outdoors captured from the side from two different distances. A number of 15 out of the 20 subjects captured have a complete set of data. A complete set consists of 6 trials each of the two angles of the outdoors video and 3 trials each of the two distances of the indoor video.

The HID database can be used to learn the difference of execution of the same action by different subjects. However, the database is limited in terms of the range of actions. As the database contains only walk motion, the identifying unique properties or parameters that the database aims to detect will be a very small subset of the actual parameters that a subject can have. For example, a subject can have a very typical walk but a very unique run action or could be easily identified by how the person sits. Having a larger set of actions would definitely help the subject classes to be trained much better and have more knowledge of the subject. Variations of particular actions are just a subset of the HMD whose flexibility and scope is much greater than HID.

2.1.9 ICS Action Database

The ICS Action Database [2] at the University of Tokyo has been used for human action recognition and segmentation. The ICS database contains a set of 25 different actions with each action having five trials making a total of 125 trials of motion capture data in the bvh file format. The information about the subjects performing the actions is not given such as whether the 5 sets of 25 actions are performed by different subjects.

The uniqueness of this database is that each trial is labeled frame by frame to belong to a set of actions. This way, every frame can belong to multiple actions. Each trial has a set of 25 files corresponding to the 25 actions in the database. Each line in the file corresponds to a

particular frame and whether it is part of the action. For a pair of frame and action, there are three possible modes {1.0, 0.5, 0.0}, where 1.0 means true (*i.e.*, the frame appears to be part of the said action), 0.0 means false (*i.e.*, the frame appears not to be part of the said action), and 0.5 is neutral (*i.e.*, the frame cannot be adjudged whether it is part of the said action). For example, the action "get up" in the database starts with the subject lying down. Thus, the initial frames of "get up" motion file are labeled as 1.0 in the "get up_lying on back" label file and labeled as 0.0 in the "get up_get up" label file. The latter file shows true frames once the actual motion of get up starts and goes on till the subject is in a sitting position. Then, the "get up_sitting" file starts to show true frames. This frame by frame labeling enables very good action definition. Our HMD provides a much wider range of subjects and actions than this database.

Table 2.1 shows the summarized information for all the mocap databases surveyed in this chapter. The table shows the information regarding the type of data, the categories of motion, the number of different subjects, the advantages, and disadvantages of each database.

2.2 Analysis and Features of the Human Motion Database

From the databases discussed in the previous section, a set of guidelines can be formulated when creating a new database. At the basic level, a database can be semi-structured or well-structured. Semi-structured databases, such as the widely popular CMU mocap database, do not have any specific goal but the sheer size of the database makes it possible to have different applications. However, the semi-structured nature (*i.e.*, lack of consistency across the database with regards to distribution of motions and subjects) of a database requires some time to find a specific subset of actions for a particular experiment. Well-structured databases, like the HDM05 database, are created to cater to a specific need in the research community with a very specific goal.

Table 2.1. Summary of features of existing databases

Database	Goal	Number of Subjects	Number of Actions	Additional Data	Advantages	Disadvantages
CMU Mocap	Repository to aid research in animation	144 in general, 5 in the kitchen DB	23 broad categories in general,5 sets in the kitchen DB	Audio, video and accelerometers only in the kitchen DB	Lots of actions	Lack of structure
IEMOCAP	Model expressive human communication	10 actors total, mocap of only 5 actors	9 different emotions	Audio	Very well structured	Low number of subjects
HumanEva	Establish Quantitative Evaluation for pose estimation and human tracking	4	6	Video	Structured	Lower Range of subjects and actions. Motion capture data has bad quality
HDM05	Motion analysis synthesis and classification	5	100	None	Very well structured. Same motion performed in many ways	Only 5 actors. Motion realizations not evenly distributed among actors
Biological Motion Library	Analyze and identify motion properties or features such as gender, identity, and affects in the form of emotions	15 male and 15 female	4 basic actions in 4 different emotions and sequences of the 4 actions	None	Structured, actions performed over a range of 4 emotional affects	Focus is on emotions and not on the range of actions
KUG	Gesture recognition and human tracking	20 with age ranging from 60 to 80	54	Stereoscopic video data	Well structured and defined range of motions	No explanation why certain actions chosen over others. Video data is not ideal as subjects are wearing mocap suits
HID	Gait recognition	20	1	Video data from multiple angles	High number of subjects with data in both video and mocap form	Range of actions very low
ICS	Human action recognition and segmentation	5 Different sets	25	Mocap data is labeled frame by frame	Very well-defined actions, particularly when there are multiple actions within a particular trial	Not much information about subjects, 30 fps data rate

The goal of the database determines the following important factors:

- Type of motion data: full body motion capture, just motion capture of the face, a combination of both, or mocap associated with data such as video, audio, and others.
- Range of motion: a wide range of motion or a limited number of motions performed in different ways or repeated in multiple realizations.
- Range of subjects: wide distribution of features and more variation important or a small set of subjects.
- Definition and execution of the motions: trained actors perform the actions or untrained subjects.
- Output data: animation formats such as asf/amc and bvh or point light display format.

These are important decisions that form the basis of the database design. Except the CMU mocap database, all the other databases have multiple subjects performing the same set of actions. The actions are always defined to varying degrees of detail and there is some rigidity on how the actions are executed. Repeated actions and multiple realizations are an essential part of learning actions and training algorithms. Most of the databases have proper annotation. In a case where there are multiple motions in the same file, the files are split up to contain only one type of motion. The only exception to this is some trials which must have multiple actions to satisfy one of the goals of the database: transition. Some databases put a lot of effort into assuring that the motion captured is real and natural and not overly scripted or simulated. There are databases that even use actors (who are trained to express emotions) to perform scenarios in the most realistic way.

One first goal of our Human Motion Database (HMD) is to be a corpus of human motion. The Human Motion Database has been constructed to have the most complete set of human actions. Besides being a structured human motion corpus, the Human Motion Database is designed serve as a benchmark for motion-based animation problems such as motion indexing, retargeting, splicing, transitioning, and generalization. Although there exist data in

other databases which can be used to address these problems, none of them are structured specifically for these problems. The HMD makes addressing these above mentioned problems its secondary goal. This enables a structure which caters to the motion-based animation problems. The database has a large range of actions (over 350 actions from a single subject and 150 interactions between two subjects), a large range of subjects (50) each performing a set of 70 actions. Furthermore, the 50 actors are distributed over a range of anthropomorphic features such as gender, age, height, and weight. Each motion class is performed by every subject for at least 10 realizations. All the motion clips are carefully annotated with the actions clearly defined. The output of the data is the standard animation format of bvh.

Ground truth data is provided for motion-based animation problems, such as synthesized motions in transitioning and splicing, where it is necessary to quantitatively evaluate the results. Additionally, HMD also provides data of interaction between two subjects with motion data of both humans in a synchronize manner. This enhances the corpus of human motion with actions which were previously not available in any other database. The whole structure of the database is explained in more detail in Chapter 6.

CHAPTER 3

EQUIPMENT & SUBJECTS

This chapter describes the equipment, setup, and human resources used for the capture of the Human Motion Database. First, we discuss different kinds of motion capture systems that can be used to record motion in Section 3.1. Next, the chapter discusses the motion capture hardware used to create the HMD in Section 3.2. Finally, Section 3.3 gives details of the anthropomorphic distribution of subjects in HMD. One of the primary goals of this chapter is to elaborate on how the Human Motion Database is structured according to the set of volunteers.

3.1 Motion Capture Systems

Motion Capture systems have initially been developed in history to study the human gait [13]. In recent years, it is been used extensively in the animation and gaming industry for the synthesis of more realistic, natural movements. Motion capture, as the term implies, is the digital recording of motion performed by a subject and then reconstructed in a 3-dimensional volume [9]. Various motion capture systems based on different technologies can be used to capture motion. The most prevalent ones commercially available use optical, magnetic, inertial, mechanical, time-of-flight or a combination of these principles. These systems are explained in the next section.

3.1.1 Types of motion capture systems

There are several types of motion capture systems, each one having its own advantages and disadvantages. These systems may also vary in terms of the data collected (*i.e.*, points in space or joint angle data) depending on the type of application. Different research

areas such as psychology require point data, while animation research generally uses joint angle data. Next, we present a representative set of common types of motion capture systems.

Optical Motion Capture systems use sensors to capture images, which are processed to track human motion. The basic principles of optical motion capture systems can be summarized as using multiple cameras to observe the same space, then using stereo matching techniques on the 2D images to estimate 3D features. Stereo matching techniques use the camera parameters and common tracking points in the images to correlate and calculate 3D information. Where most optical motion capture systems vary is the technique to identify the common tracking points across the images. Various camera sensor technologies can be employed to capture the images and correspondingly many techniques of computer vision and pattern recognition are used to take benefit of the unique raw data generated by the sensors. These systems are considered to have a high degree of accuracy and to give adequate freedom of movement. However, they suffer from occlusion (when an object is not observed because another object comes between the object and the sensors' line of sight) and the capture is restricted to a volumetric space where the sensors need to be installed and calibrated. Optical Motion capture systems can be can be placed in two broad classifications:

- **Marker based systems**: These systems rely on detection of specific tracking points in the form of markers which are placed on the skin of the subject being captured. The markers used are system specific and they depend on what kind of technology the sensors use and on which techniques are used for detecting the markers. For example, a system using near infrared source of light may use retro-reflective markers (*e.g.*, Vicon), while a sensor capturing frequencies in the visible range may use white patches on black clothing [9], and yet others will have active markers which emit a unique frequency such that sensors are tuned to capture only those frequencies. The techniques implemented to extract 3D motion information will be custom made to use the specific raw data each system generates. Generally, these

systems minimize the cost of pose estimation by focusing on filtering the marker data to build skeletal models of the subject. Marker-based capture is considered to be more accurate and to show real-time results.

- **Markerless systems**: Markerless systems depend solely on vision based pose estimation techniques to build the model of the subject. These systems also requires at least two cameras, are more sensitive to error and require more strict capture conditions (like restrictions on the background of the capture space) than the marker based system.

Magnetic Motion Capture systems work with magnetic field sensors coupled with magnetic field transmitters both having three orthogonally placed coils which serve as axes. The magnetic sensors are placed on the body, which detects low frequency magnetic fields from the transmitter. As the strength of a magnetic field is proportional with distance, the sensors can provide data that is used to calculate to the position and orientation of each sensor in a 3-dimensional space. The human body does not affect magnetic fields and thus these systems do not have occlusion problems. However, the presence of any ferromagnetic substance in the capture space can induce error in the data. The transmitter also has a certain range within which the sensors can detect the magnetic fields generated by it. Another disadvantage is that the wires associated with the sensors on the body of the subject can restrict freedom of movement.

Inertial motion capture systems use inertial sensors such as accelerometers and gyroscopes to calculate the body pose. Like markers of the optical motion systems, the sensors are put on the human body in specific locations according to a skeletal model. The sensors can detect parameters such as orientation, angular velocity, and angular acceleration which can be used to calculate the motion of the whole body figure by calculating the relative transformations between connected links in a skeletal model. These systems do not suffer from occlusion and can be used in a more flexible capture volume as long as the receivers are within range of the

sensors. However, the sensors are very sensitive to errors such as slight movement of the sensors on the skin. All the errors always add up leading to progressively worse quality data till a new calibration is performed. In practice, although it allows a subject to perform actions such as rolling on the ground (which is very difficult in optical motion systems due to occlusion), the quality of data is not good due to the high sensitivity of the sensors. Also the sensors require a power source to function which means carrying batteries on the subject while performing, which does affect the natural performance of an action. Thus, even if these systems give more flexibility of use, the overall quality of data is not as accurate as the optical systems.

Time of Flight systems use pulses from different wave sources (such as acoustic or infrared) to get the depth information of the scene using time of flight principles. A pulse generated from a source reflects back off objects and sensors can calculate the distance of these objects from the time it took for the pulse to come back. This way, a system can generate 3-dimensional depth fields of the observed space. The data can then be used to track human movement. Time of flight cameras are sensors which perform in real-time. They require low processing power compared to stereoscopic systems and can provide a very high frame rate data. However, the current state-of-the-art suggests that using the depth data to get the human motion in the form of joint angles takes more processing power and generates at best 4-10 frames per second [7]. Thus, time of flight real-time systems still cannot match other systems described above in terms of frame rate. Other problems include multiple time of flight cameras can cause interference and multiple reflections can cause false depth information.

Mechanical systems fit an exoskeleton on the subject. The exoskeleton consists of articulated joints, which are controlled by the movement of the subject. Sensors such as goniometers are attached to the system that measures the joint angles of the articulated mechanical parts. These systems have no limit on capture volume and do not suffer from occlusion. However, the exoskeleton may either restrict the movement or alter the performance of the action. The accuracy of the captured data is considered to be less than other systems.

Hybrid systems use a combination of two or more technologies mentioned above to use the advantages of both systems as well as to compliment each others data quality. Such a system using ultra sonic time of flight and inertial motion sensors has been discussed in [15]. That system produces more accurate data than inertial systems by supplementing the data with time of flight data. It also provides good portability and can be used in everyday locations without the use of a capture volume. However, these are still emerging technology and no commercial solutions are available yet which uses such a configuration.

3.1.2 Optical Motion Capture System

As described in section 3.2.1, optical motion capture systems provide very accurate data and unrestricted freedom of movement for most human actions. However, they also suffer from problems such as occlusion and restricted capture volume. The intention of the Human Motion Database was to collect very high quality data in terms of accuracy. Marker-based optical systems provide accuracy better than any other type of existing motion capture system. On the other hand, occlusions cause an extra overhead of cleaning the data and restrict some actions such as rolling on the ground. However, most of these disadvantages can be overcome by capturing multiple takes to get the best possible take as the cost of capturing data is trivial compared to post processing data.

3.2 Equipment

The equipment used to construct the Human Motion Database is a Vicon optical motion capture system. It consists of near infrared cameras, base stations, a capture space and a desktop with software to help in the collection of data. The individual components are described in the following sections.

3.2.1 The Vicon optical motion capture system

The system architecture of the Vicon system used has been shown in Fig.3.1. The cameras are placed around the capture volume looking towards the intended center of the volume. All the cameras are connected with data cables to the central Ultranet HD core system,

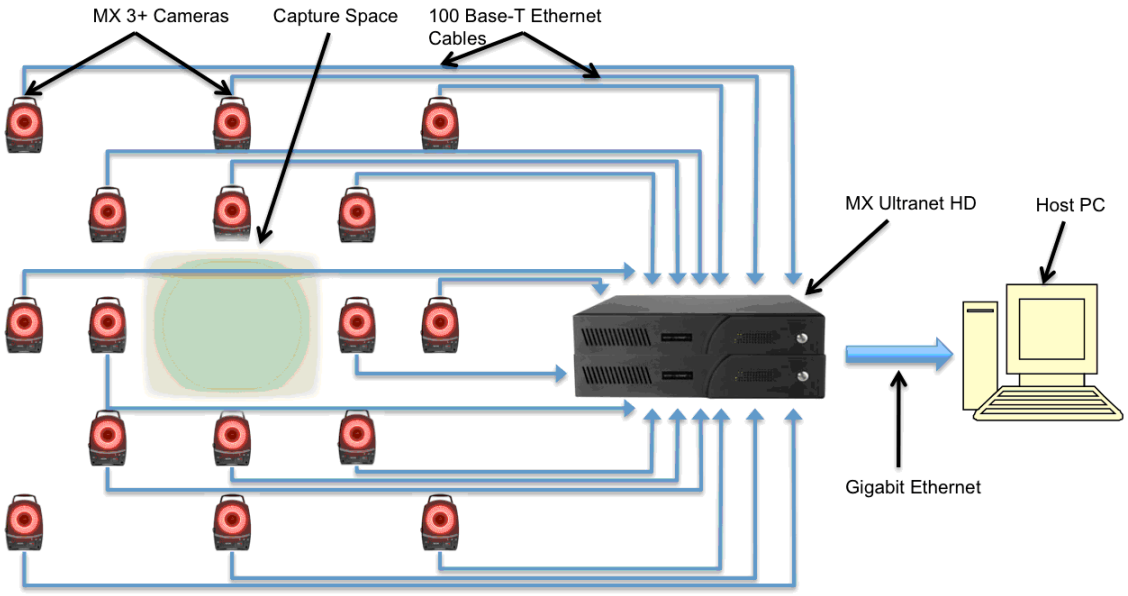


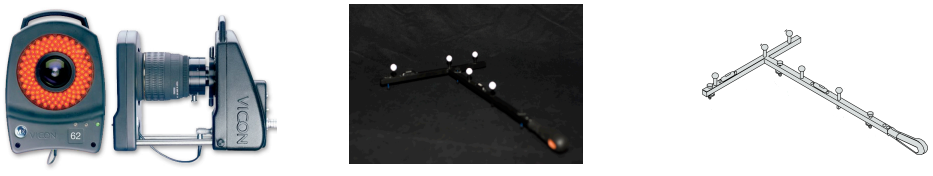
Figure 3.1: Architecture of the Vicon system.

which in turn is connected to the Host PC which contains all the software. The system consists of the following major components:

- Cameras: 16 MX 3+ cameras were used to look at the capture volume (see Fig. 3.2(a)). The cameras have a resolution of 0.3 megapixels (659 x 494 pixels) and can operate at a maximum rate of 240 frames per second. There were 8 cameras that were placed on trusses, which had Pentax TV lens with a focal length of 8.5mm. The other 8 cameras, placed on tripods closer to the capture volume, had Fujinon lenses with focal length of 6mm. The lens of the cameras consists of sensors capable of detecting only near infrared light. The near infrared light is provided from strobes attached around each of the cameras emitting light of wavelength 780 nm. The cameras can detect markers from its field of view and send their pixel coordinates to the Vicon software.
- Calibrating device: A calibrating device is needed to setup the system at the start of each capture session. The Vicon calibration device is an L shaped wand with five 14 mm markers placed on it at specific positions (see Fig. 3.2(b)). It has a handle to

hold it while doing a dynamic calibration and spirit levels with adjustable feet to be used during static calibration. The calibration process will be described in detail in Chapter 4.

- Markers: These are spherical objects of radius 14 mm that are covered with retro-reflective paint or material. The Vicon cameras can detect these markers. These markers are put on the subject to be captured.
- MX Ultranet HD: This is the core unit that connects the cameras to the Host PC. It can supply power, synchronizations, and communications for up to 10 MX cameras. It can also synchronize with other third party hardware such as video cameras, electromyography equipment, and force plates. In case more than 10 MX cameras are used, more than one such unit is used. However, only one primary unit will be connected to the Host PC while all secondary units are connected to this primary unit. In our system of 16 cameras, there were two MX Ultranet HD units with both the primary and secondary connected to 8 cameras each.
- Host PC: The MX Ultranet HD is connected to a PC for communication and data transfer with the help of a gigabit Ethernet wire. The PC, loaded with Windows XP, contains the software to control the system.
- Tripods, trusses and pan-tilt head: Manfrotto tripods were used to place 8 cameras closer to the capture volume. The heights of the tripods ranged from 92 cm to 141



(a) (b)

Figure 3.2: Vicon mocap devices: (a) Vicon MX 3+ camera, (b) Vicon L-frame calibration wand

cm. The other 8 cameras were placed on trusses at two heights of approximately 263 cm and 290 cm. Manfrotto pan-tilt heads were used to mount all the cameras to their respective truss or tripod. The pan tilt heads offered three degrees of freedom of pitch, yaw, and roll which helped in pointing the cameras towards the intended direction.

- Software: Vicon provided the two softwares to control the system: Vicon Nexus and Vicon Blade. Vicon Nexus has been developed to cater to the kinesiological applications and has facilities to capture data from force plate and EMG data. Vicon Blade is more useful for animation applications and has more tools to generate animation formats such as bvh and asf-amc easily.

3.2.2 Camera Configuration

The system has sixteen cameras. The cameras are arranged at four different levels around the capture volume in order to make every section of the capture volume equally covered. The camera setup in the capture room is shown in Fig. 3.3. Eight cameras were placed on horizontally hanging trusses while eight more were placed on tripods on the ground. The cameras were attached to both the truss and tripods using pan-tilt head mounts. The trusses, shown in blue and orange in Fig. 3.3, were built at two different heights 103.5 inches and 89.5 inches. The tripods were adjusted to fit the pan-tilt head at approximately three varying heights from 36.5 inches to 55.5 inches.

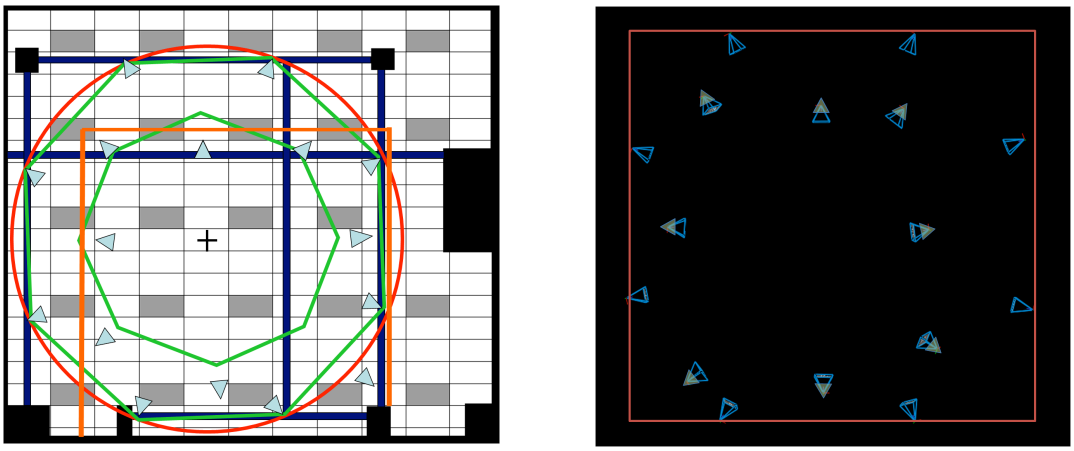
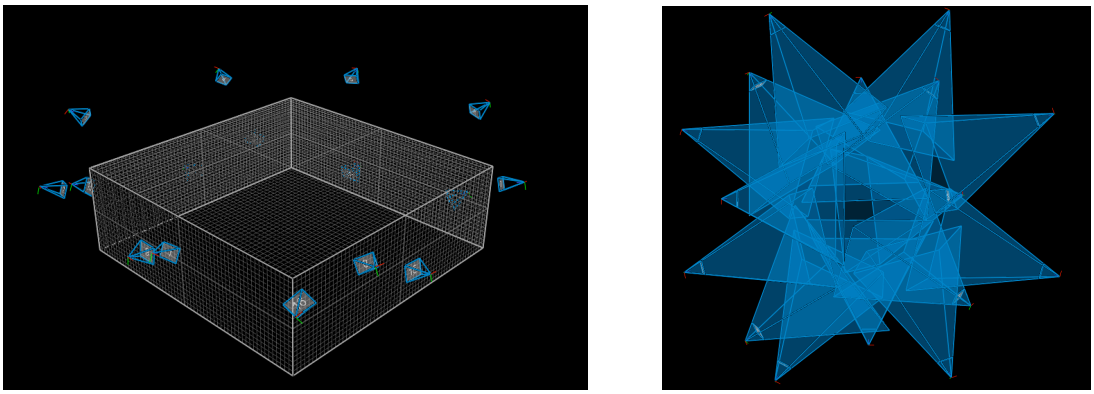


Figure 3.3: Camera setup in the capture room.

3.2.3 Capture Room

The capture volume is defined as the space where all the subjects can perform actions such that the system can record the motion. It is a calibrated space where the system can calculate the exact positions of 3-dimensional points from the 2-dimensional data from each of a subset of the cameras. The capture volume does not have rigid boundaries although it does have defined edges within which a point will always be spotted and reconstructed with a high degree of accuracy provided the point is not occluded. It is recommended not to collect data beyond these edges, although the data collected may be of good quality depending on where in the volume the rules are broken. Figure 3.4(a) shows the capture volume as generated by Vicon Nexus program. The bounding box, as shown in the figure, is not rigid as mentioned above. Figure 3.4(b) shows the frustum or field of view of all 16 cameras from the top view. We can get an idea of the real capture volume from this figure as any space where the frustums of 2 or more cameras overlap. Subjects were always asked to perform inside the predefined edges during capture session.



(a) (b)

Figure 3.4: Camera configuration in the capture room: (a) Capture volume as shown in Vicon Nexus, (b) Camera frustums

3.2.4 Marker Configuration

The cameras are able to detect only near infrared light having wavelength 780 nm. The system is designed to detect special markers, which are covered with retro-reflective material. As the strobes are placed around the lenses of the cameras, the retro-reflective markers reflect most of the light incident on it back to the direction of the lenses. The markers however are to be placed on specific locations on the body according to the skeletal model, known as Plug-in Gait model, used in the reconstruction of the human skeleton in the software. The markers are placed on specific bones on the body according to specific points described in the model. The model is a set of independent rigid structures moving with respect to each other and attached at articulated joints. The system reconstructs the markers present in the capture volume to the virtual capture space. These markers are then automatically labeled and made to fit into the skeletal model. The fitting process helps in creating the joints using inverse kinematics with the information of the assumed rigid structures of the model. Such a model is needed also to aid in post-processing tasks such as marker tracking and generating joint angle data.

The Plug-in Gait model provided by Vicon is included in the Nexus software. Given a set of captured markers, the Plug-in Gait model can generate the angular joint kinematics and

kinetics of the subject. There are many variations of the model and the one used to capture the Human Motion Database is the Plug-In Gait Full-body model. The marker configuration is explained in detail in Chapter 6.

3.2.5 Software provided by Vicon

The software used for the modeling, reconstruction, and post-processing is the Vicon Nexus 1.4.115.43300. Some steps of post processing such as generation of joint angle data using a custom skeleton was done in Vicon Blade 1.6.214.46031. The Vicon Nexus program is used mainly by the researchers studying the kinesiological aspect of motion and has the facility to collect data from force plates and muscle EMGs along with the optical data. The Vicon Blade program is more suited for animation purposes and has more features enabling generating a bvh and creating a custom skeleton.

3.3 Volunteers

The Human Motion Database collected motion data from a total of 51 subjects. Volunteers were asked to respond from both university postings as well as local postings. A set of volunteers were selected from the pool of applicants. Our main objective in terms of subject selection was to have a diverse distribution and uniform coverage of anthropomorphic (*e.g.*, height, weight) parameters as described below. An special effort was made to get, as much as possible, a diverse distribution and uniform coverage of subjects in terms of:

• Age: The age of the volunteers ranged from 7 to 82 years. Approximately half of the volunteers (28) are in the age range between 7 to 21 years because variation of skeletal structure, height, and weight is more pronounced in this range.

• Gender: Approximately half of the volunteers were selected from both genders: male and female. There are 26 male and 24 female subjects in the database.

• Height: The height of the subjects captured varies from 45.8 inches to 74.0 inches. A special effort was given to get both male and female representatives at most height levels.

• Weight: The weight of the subjects captured varies from 42.2 lbs to 262.8 lbs. Again, a special effort was given to get both male and female representatives at most weight levels.

As subjects were included in the database during the selection process, a Voronoi diagram as shown in Fig 3.5 was constructed using the weight and height parameters as 2-dimensional coordinates. The Voronoi diagram was used to find out interesting data points (subjects) that would increase diversity (points further from existing points in the diagram) and increase coverage of the anthropomorphic parametric space. This guided the selection criteria of new subjects to enter the database construction.

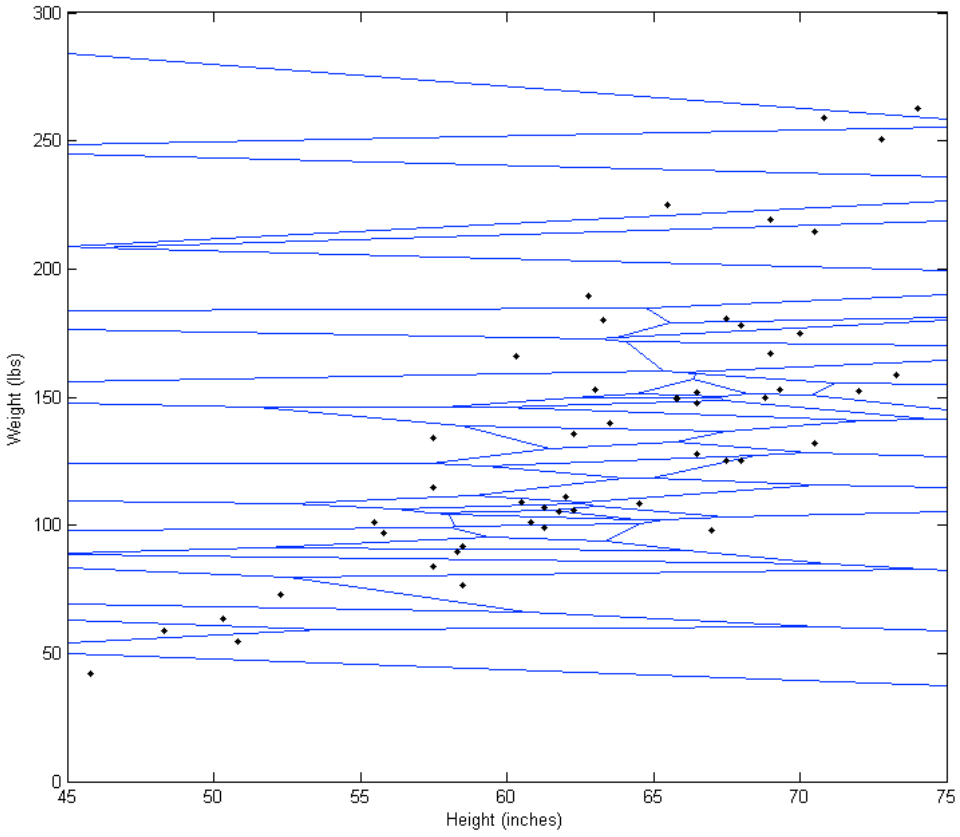
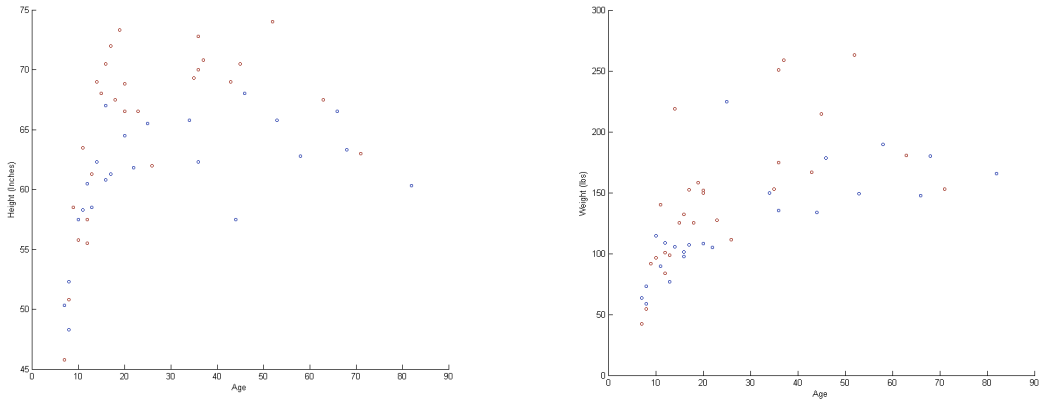


Figure 3.5: The distribution of height and weight for all volunteers as a Voronoi diagram.



(a) (b)

Figure 3.6: Distribution of subject parameters. Male subjects in red and female subjects in blue: (a) Age against height, (b) Age against weight

The distribution of height and the distribution of weight with respect to age are given in Fig 3.6(a) and 3.6(b), respectively. These figures show that there are both male (depicted as red points) and female (depicted as blue points) subjects at most height and weight levels as well as age categories. The more uniform distribution of data points in Fig 3.5 justifies the effort to capture 28 subjects under the age of 21 years which resulted in a better variety of skeletal structure.

3.4 Organization in terms of actions

One of the most important design goals (as explained in more details later) to create this database was to have a proper structure and to have metadata associated with the actual mocap data in order to solve current problems in human motion synthesis and analysis. The novel way of using cognitive and parametric sampling brings distribution in the database which is supplemented with metadata, such as the anthropomorphic data of the subjects. The sampling methodology is discussed in detail in Chapter 6. The subject data shown here is provided as a text format file along with the database.

Since the actions captured in the construction of the database were everyday general actions and did not involve any special skills, there was no need to hire professional actors.

Rather paid volunteers with any background were used and asked to perform the said corpus of motion. The main selection criteria was guided by the need for even distribution and coverage on the above mentioned parameters as well as the actors ability to perform all the said actions on direction or demonstration. We have to mention that some older subjects could not perform certain actions, such as jump, in a consistent manner, thus sometimes those actions could not be captured or have been performed differently from the rest of the subjects in the database. This information can help in identifying change of motion characteristics due to age.

Each of the 50 subjects performed 70 actions in a single 2-3 hour long session. They were given vocal instructions to maintain the consistent performance over the range of subjects. Two out of the 50 actors perform an additional set of actions to address the problems of transitioning, splicing, generalization, and praxicon. There is also a set of synchronized actions performed by two subjects doing interactive motions. The interactive motions are performed together although the data for the subjects are placed in different files. The data of both subjects is synchronized with each other. The interaction set of data has some actions that are acted out and not real such as stabbing, as capturing those actions realistically is impossible.

CHAPTER 4

CAPTURE AND POST-PROCESSING

In this chapter, we will discuss capture and post processing stages of the creation of the database. Capture is the process by which a phenomenon of interest is recorded and stored digitally. Motion capture primarily records the actions of a human or in some cases animals or birds to recreate the motion for various applications. This chapter describes the various workflows for the capturing process such as camera calibration, volume calibration, subject calibration, and capture sessions. Next, the chapter discusses the various capture errors present in the raw data and the techniques by which the data was cleaned and post processed to finally obtain the joint angle data output.

Except for subjects 001, 007 and 030, all subjects were captured in one session in 2.5-3.5 hours. There was provision for breaks during a session, but most subjects completed the session without stops. Subject 001 was involved in six different sessions, while subject 007 was involved in two sessions. Both subjects 001 and 007 have extra sets of actions as described later in chapter 6.

Motion capturing is comprised of many stages. Considering that the hardware is already fixed and the capture room is setup, the system still needs multiple calibrations such as intrinsic calibration, extrinsic calibration, Euclidean calibration, and subject calibration. The intrinsic calibration, extrinsic calibration, and the Euclidean calibration must be done to calibrate the cameras. All these processes were done at the beginning of every capture day because the chances of the camera locations shifting from the previously calibrated locations were higher under no supervision. Once the cameras are calibrated, multiple subjects and multiple sessions

can be captured every day with the same camera calibrations as long as the camera locations are not accidently changed.

The subject must wear a body fitting mocap suit. Then the subject is measured and anthropomorphic data collected. This data serves as valuable annotation for the database and demonstrates the good distribution of subjects that is so essential to the dataset. Then retro-reflective markers are placed on the subject according to Figure 4.1. The subject then undergoes a subject calibration followed by the reconstruction and labeling of markers. Before the capture of every action, the action is described to the subject, demonstrated by the staff, and practiced for a few realizations such that the captured action conforms to the definition. Then the action is recorded for at least 10 repetitions. All actions are captured in individual trials. Sometimes when the required number of realizations is not covered in a single trial, multiple trials of the same action are captured for the subject. The capture workflow is shown in

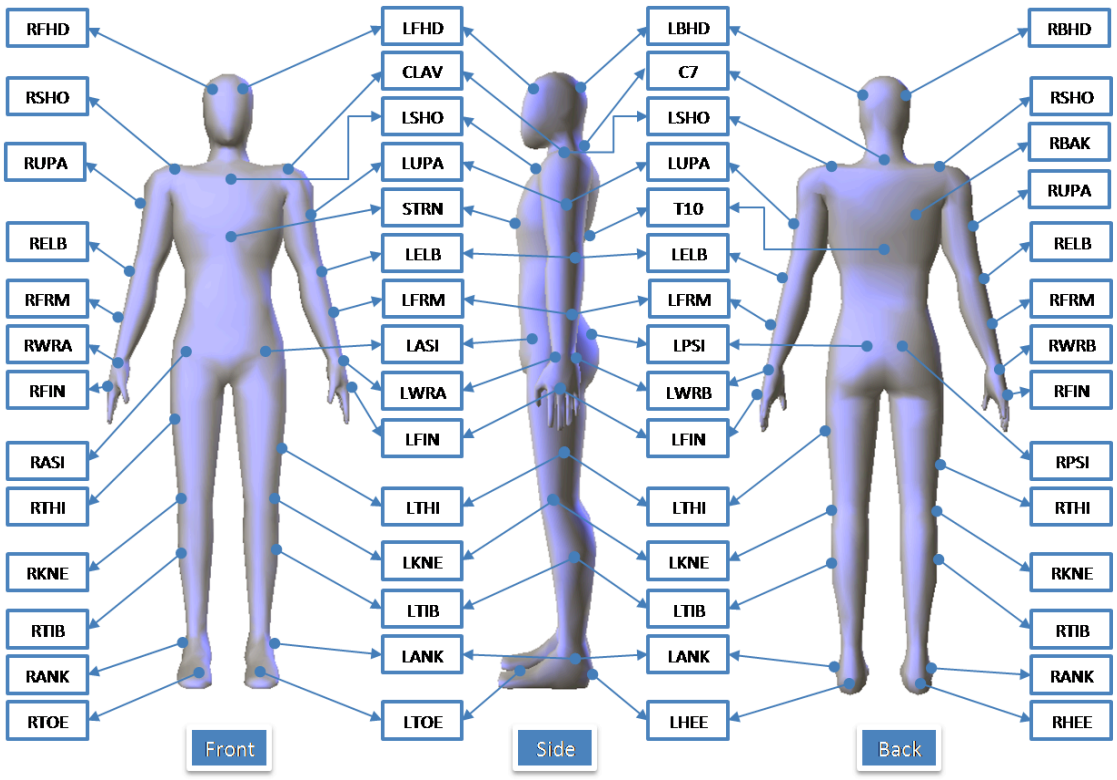


Figure 4.1: The marker configuration.

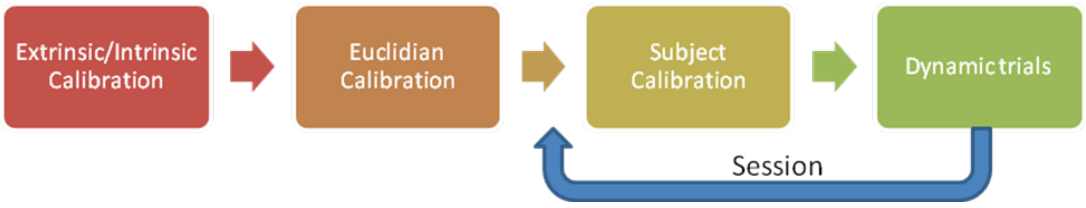


Figure 4.2: The capture workflow.

Figure 4.2. The raw data recorded during each session is stored and kept for the post processing step. The post processing stage includes reconstruction of the raw data, labeling the data, and the cleaning stage. All these stages are explained in detail in the following sections.

4.1 Capture

4.1.1 Calibration

The capture of a session is preceded by calibration. Calibration is required by a motion capture system to compute the extrinsic (position and orientation) properties and the intrinsic properties (focal length, geometric image distortion) of cameras [9] and to set up the axes and Euclidean metrics of the capture volume. A calibration method optimizes the system for maximum accuracy of the captured data. Generally the calibration of vision-based motion capture systems relies on observing an object of known dimensions and using it as the reference object. This reference object is known as a calibration device. The calibration device serves a way for the system to measure its own accuracy. The calibration device used in the system is a five marker L frame wand as shown in Figure 3.2(b). Euclidean calibration and extrinsic/intrinsic calibration were performed once every day. Subject calibration was performed every time a subject starts a new session, where a session consists of a set of trials each consisting of only one type of action. The calibration of an optical motion capture system has the following phases:

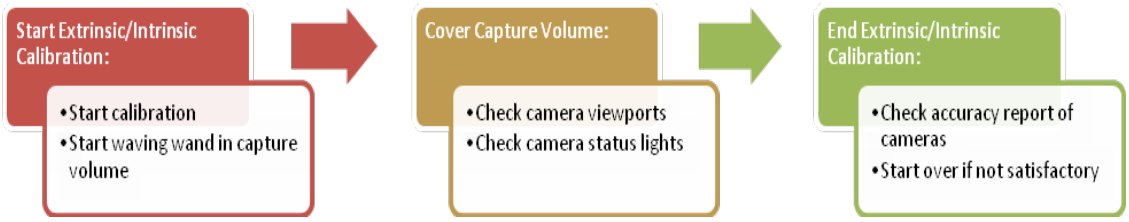


Figure 4.3: The extrinsic and intrinsic camera calibration workflow.

4.1.1.1 Extrinsic and Intrinsic Camera Calibration

During this process, the calibration device is waved in the capture volume such that the cameras can observe the features on the calibration device. The calibration device is moved in such a way to cover the whole capture volume. As some sections of the volume are only covered by a subset of the cameras, the calibration device must cover all the cameras equally. For this reason there is a feedback, which has real time detected features for all the cameras showing the cumulative calibration data. Each camera can be set to have a threshold for the number of features captured in the calibration data. The threshold set for our calibrations was 1000 features per camera. Once the number of features for each camera hits the minimum threshold, the calibration procedure is finished. When all the cameras have adequate calibration data, the reconstruction accuracy of the system is determined. The extrinsic and intrinsic camera calibration workflow is shown in Figure 4.3. If the accuracy is acceptable, we continue to the next step, which is the Euclidean calibration.

4.1.1.2 Euclidean Calibration

The Euclidean calibration is a process to calibrate the axes of the capture volume and the Euclidean metrics of the World coordinate system. Setting up the extrinsic parameters of the cameras in the previous calibration phase already establishes a coordinate system. However, this coordinate system cannot identify the up direction of the capture volume. The Euclidean calibration aligns the global coordinate system of the capture volume in the system with that of the capture volume in the lab. The Euclidean calibration defines the axes of the system effectively specifying the up direction and the other directions. A specific five marker L frame

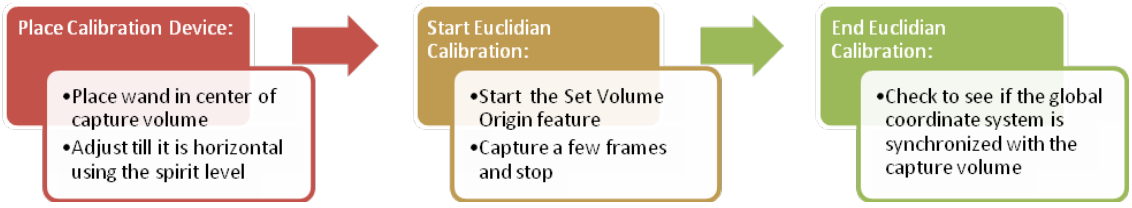


Figure 4.4: The Euclidian calibration workflow.

calibration device was used in this step. The Euclidean calibration device has sprit level indicators, which can be used to adjust it to be horizontal on the ground. The calibration device is placed on the floor in the middle of the capture volume at the location which becomes the origin of the coordinate system. The intersection of the two sections forming the L frame calibration device defines the origin of the global coordinate system. The hand of the device defines the Z axis while the other horizontal bar defines the X axis. The up direction, which is also the Y axis, is calculated from the X and Z axes. When the Euclidean calibration process is started, the system captures a few frames and does the calculations as mentioned above. The Euclidian calibration workflow is shown in Figure 4.4.

4.1.1.3 Subject Calibration

The camera and Euclidean calibration is followed by the calibration of the subject. Subject calibration is required to develop a model for the subject so that it can be used to label the capture trials. The calibration procedure followed uses a Gait full body model. The model is based on a specific marker configuration which is used create a skeleton model for the subject. This model contains the topology of the markers and bone segments. The subject is required to wear a bodysuit such that markers do not move with the clothing. The subject anthropomorphic measurements are taken. The specific measurements taken are the leg length, knee width, ankle width, shoulder offset, elbow width, wrist width, hand thickness for both right and left parts of the body. Additional measurements include the height and weight of the subject. These measurements are part of the subject description. The markers are placed on the subject at specific locations according to the marker configuration model as shown in Figure 4.1. Then the

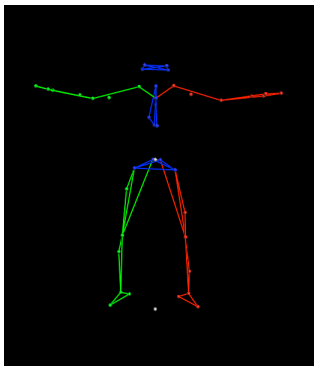


Figure 4.5: Subject standing in T pose.

subject takes a static neutral pose such that the recording is of good quality and there is no occlusion. All subjects were instructed to stand in a T pose at the center of the capture volume for this purpose (see Figure 4.5). A T pose is defined as the subject standing with both legs straight and stretching out the hands on the side such that they are parallel to the floor. After that a short static trial is captured with the subject in T pose. The recorded data is reconstructed and labeled manually according to the model. All the markers should be reconstructed and labeled in the whole length of this static trial. Then this trial is used to define the rigid body segments of the subject such as the joint centers, bone lengths, and bone orientations. The result of this step is a custom skeleton model for the subject which is generated by scaling the skeleton template to match the values obtained in the subject calibration step. This skeleton model is used in the capture trials to automatically label the subject provided they have the same marker set. The subject calibration workflow is shown in Figure 4.6.

4.1.2 Capture Session

The capture session consists of one subject performing a set of trials, each trial corresponding to one action repeated several times. A capture consists of the following steps:

- Each action is explained to the subject and sometimes demonstrated by the staff to follow strict guidelines of action definition. The aim was to make the realizations of the same action similar and consistent throughout the database for all subjects.

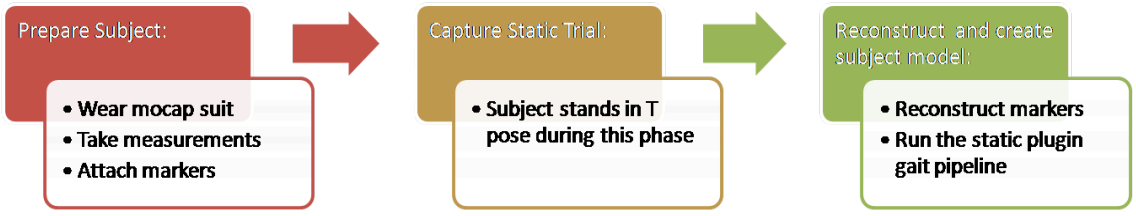


Figure 4.6: The subject calibration workflow.

- In some actions, props, such as a football, a soccer ball, a table, chairs, and a hammer, were used. However, the motion of the props was not captured. Some actions like shake hands and throw-catch ball required another human to be involved, although only the primary subject was captured. The actions of the other person in these cases were mostly reflective of the subject being captured.
- The subject practiced the action till it conformed to the action definition. If required, additional instructions were given. Otherwise, the capture starts by asking the subject to perform the action.
- The subject performs the action for at least 10 repetitions. While the action is being performed, the action is observed for any execution errors. Errors, explained in detail in the next section, include major variation of action performance over time, misplacement of markers, large sections of data being missing due to occlusion. In case the recording is not satisfactory for the first time, multiple trials are taken till ten good repetitions are recorded.
- The capture is stopped and the raw data is stored with the action name. This is later used for post processing.
- Open a new trial and go through the same steps.

The capture session workflow can be seen in Figure 4.7. Some capture sessions involved interactions where two subjects perform actions together in the capture space. The subject calibration procedure to enable the capturing of two subjects is different. In this case,

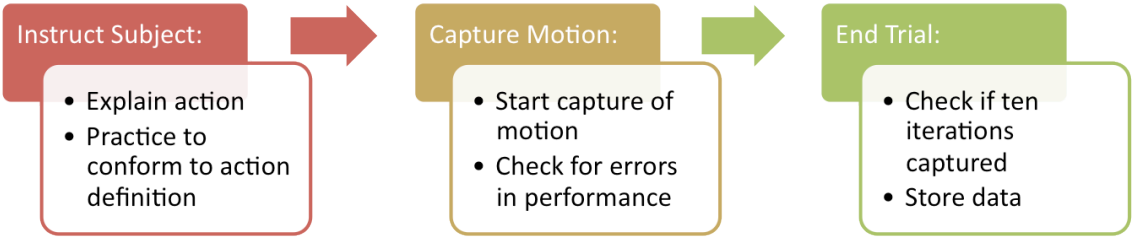


Figure 4.7: The capture session workflow.

both the subjects are prepared according to the marker configuration model. Once a new session is opened, both subjects are added consecutively. The first subject is created and the measurements are entered into the subject description as discussed in the single subject calibration. Then the subject is asked to stand in a T pose and a small trial is captured for the calibration operations to process. Next the second subject is added to the session and the above mentioned procedure is also followed for the second subject. After this, both subjects can be captured in trials with the same procedure as the single subject capture.

4.2 Post Processing

The data recorded during the capture process may contain errors and artifacts. Post-processing is the stage where the stored data is converted to a more useful, consistent form with the aim of only enhancing the data while taking care not to alter correctly captured good data. The final product is known as clean data. The post-processing steps can be broadly described as follows:

4.2.1 Reconstruct and Auto Labeling

The raw data captured is reconstructed. Using the calibration information, the motion capture system tries to automatically label the markers according to the marker configuration model. This builds a relationship between the markers as well as creates marker trajectories.

4.2.2 Clean data

The trials are manually checked for errors and artifacts and corrected or improved using various techniques. Some techniques involved some individual functions while others used

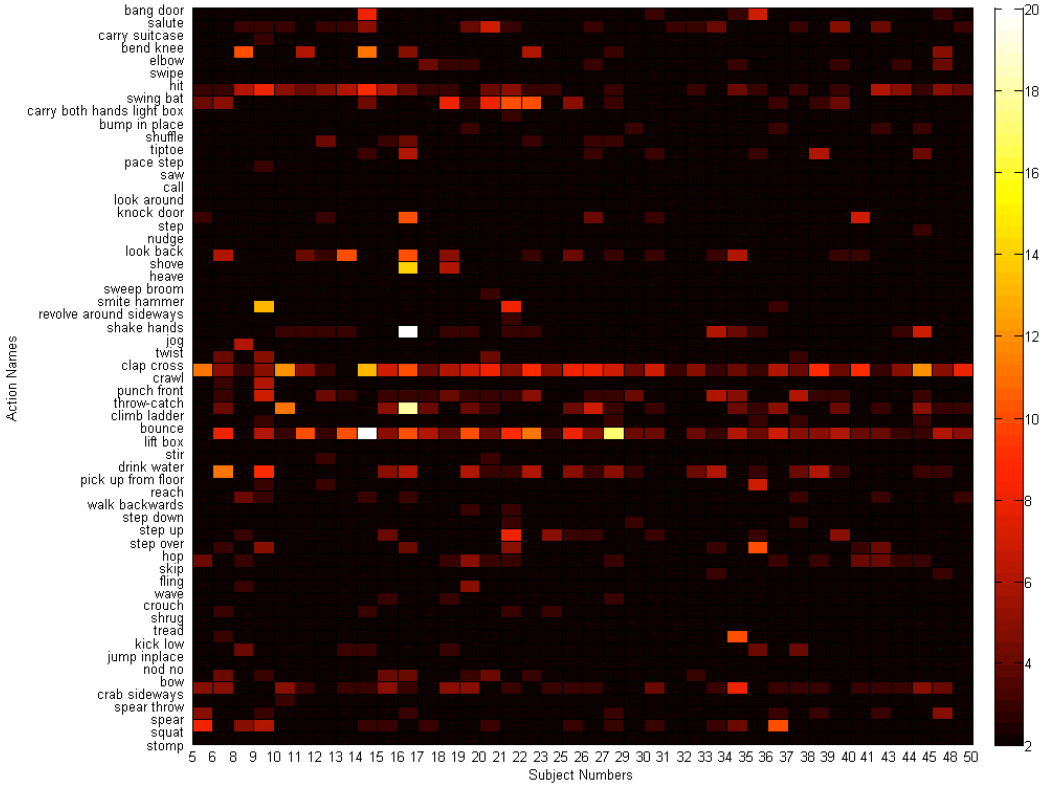
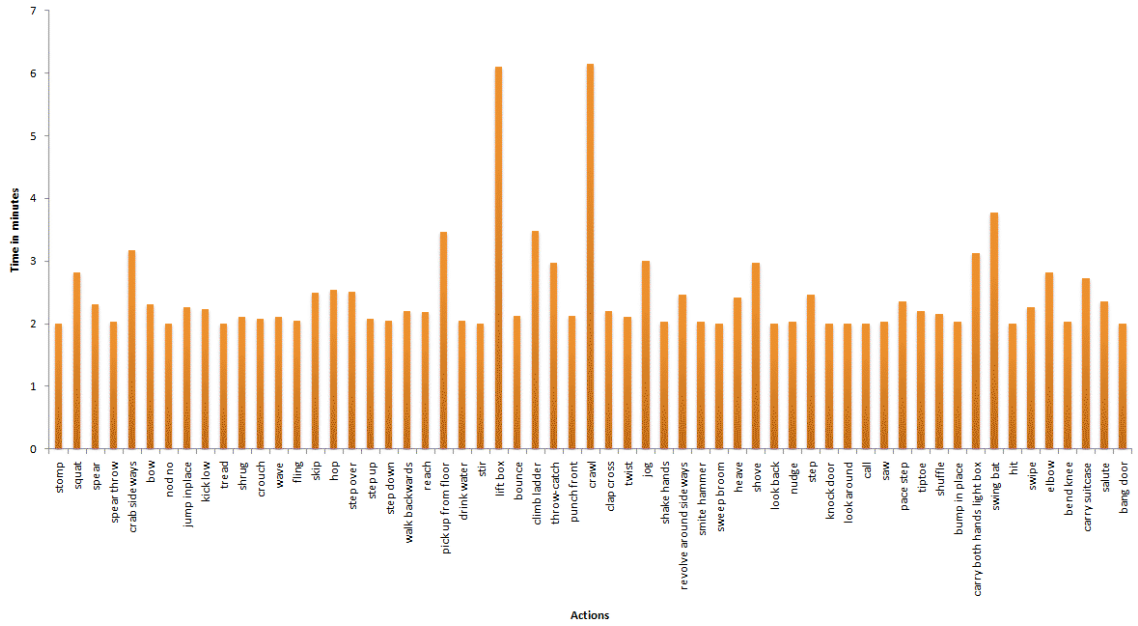


Figure 4.8: The time taken in minutes for the cleaning of actions per subject.

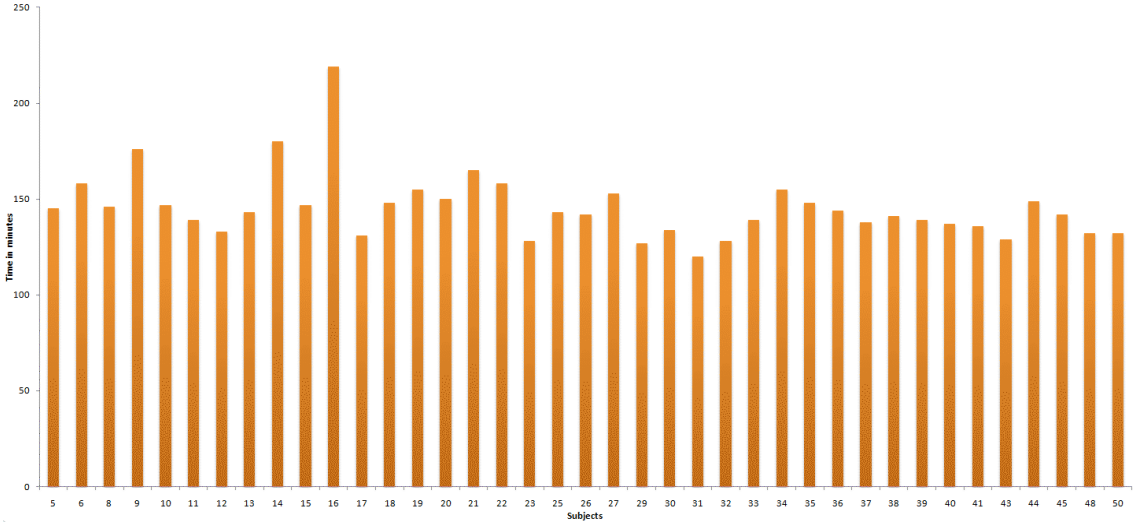
custom macros and scripts. The process of cleaning data was iterative where, after every round of cleaning, outputs were generated and the joint angle data is processed through our Matlab scripts to check for errors. These Matlab scripts were used iteratively as a guide to go into the next round of cleaning.

Figure 4.8 shows the time taken to clean the data of the entire database for the first iteration. Note that the data shown in Figure 4.8 cover only 39 subjects and 59 actions. Thisdata shows that some actions, such as lift box and crawl (see Fig. 4.9(a)), and some subjects, like subject 16 (see Fig. 4.9(b)), took more time to post-process than others. Successive iterations took more time on each trial because of more attention to detail but showed a similar trend with respect to distribution of workload throughout the database. The interactions dataset also took a

lot of time to clean due the occurrence of more occlusions resulting from two subjects being in the capture volume.



(a)



(b)

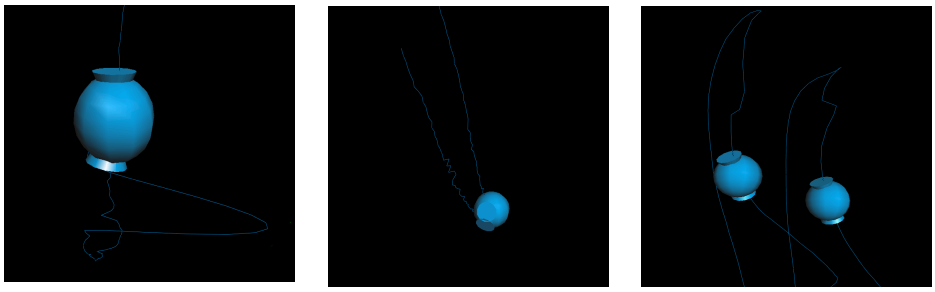
Figure 4.9: Time taken to complete the first iteration of data cleaning: (a) Cleaning time per action, (b) Cleaning time per subject

4.2.3 Generate Output

The outputs generated by a motion capture system were of two formats: c3d and csv files. The c3d is a binary file containing the 3D coordinate positions of the markers for every frame. The csv is a comma separated variable file containing both the trajectory data and the joint angle data generated according to the marker configuration model. The csv files were used as input for the Matlab scripts to check for errors.

4.2.4 Types of Capture Errors

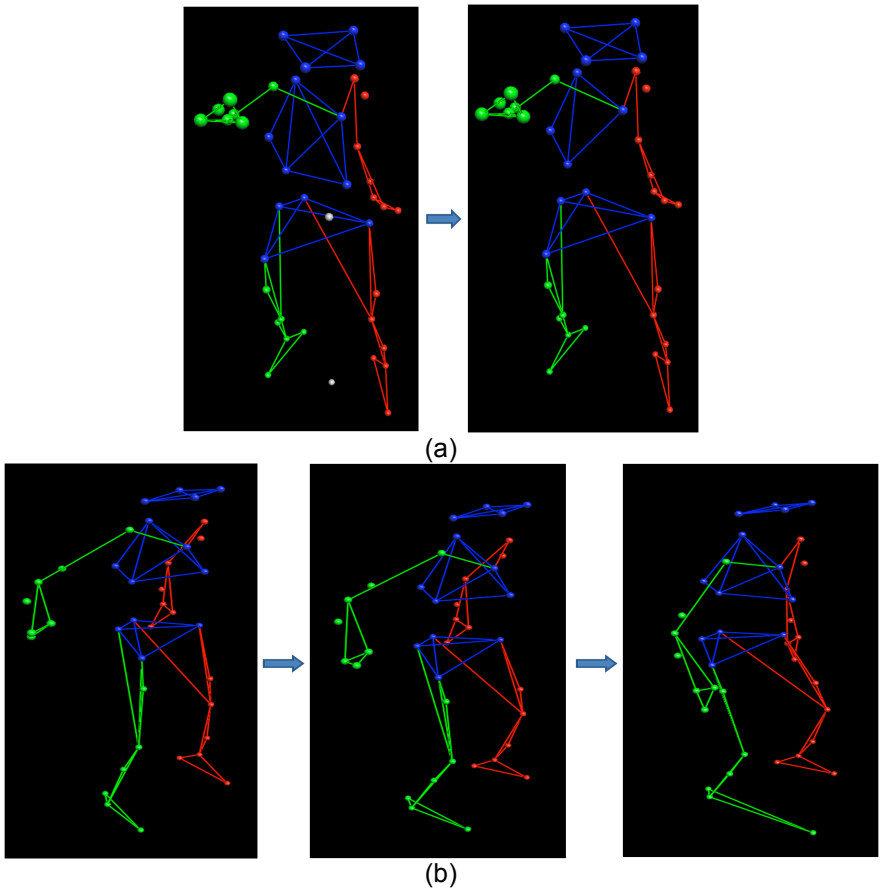
Errors or artifacts are defined as undesirable noisy data, which do not belong to the intended captured motion (see Fig. 4.10). Errors can be detected on a manual frame by frame inspection of the markers. The cleaning of the data as mentioned was an iterative process. The rounds of cleaning in the iterative process varied from one to another. Initially the volume of data being observed and checked was very high. At that stage all the frames of all trials were checked. The main checking performed in the first step was if the auto labeling was correct throughout the trial and to detect big artifacts in the data. Progressively in successive iterations, using the Matlab scripts as guide, the volume of the data to be checked became lower. This enabled more detailed inspection of certain sections of the trials. As a result, the cleaning became more fine-tuned and focused, taking care of artifacts having aberrations to a lesser and lesser degree.



(a) (b) (c)

Figure 4.10: Trajectories of markers show artifacts: (a) Jump, (b) Shake, (c) Impact

The cleaning rounds involved a lot of discussions and problem solving. There is no documentation to help in cleaning mocap data that we could go through. The problems included first deciding a threshold for error, such as defining what is an artifact and what is not an artifact. This defines the minimum quality of the data in the Human Motion Database. After that, different artifacts required different solutions using different methods. These methods are not defined anywhere. The motion capture system provides a set of functions or operations, which help in the cleaning process, but we had to select what method to use in each particular case. We reached situations where no apparent clearly defined solution seemed to exist. In those situations, we had to devise a new procedure to solve the problem.



(a)

(b)

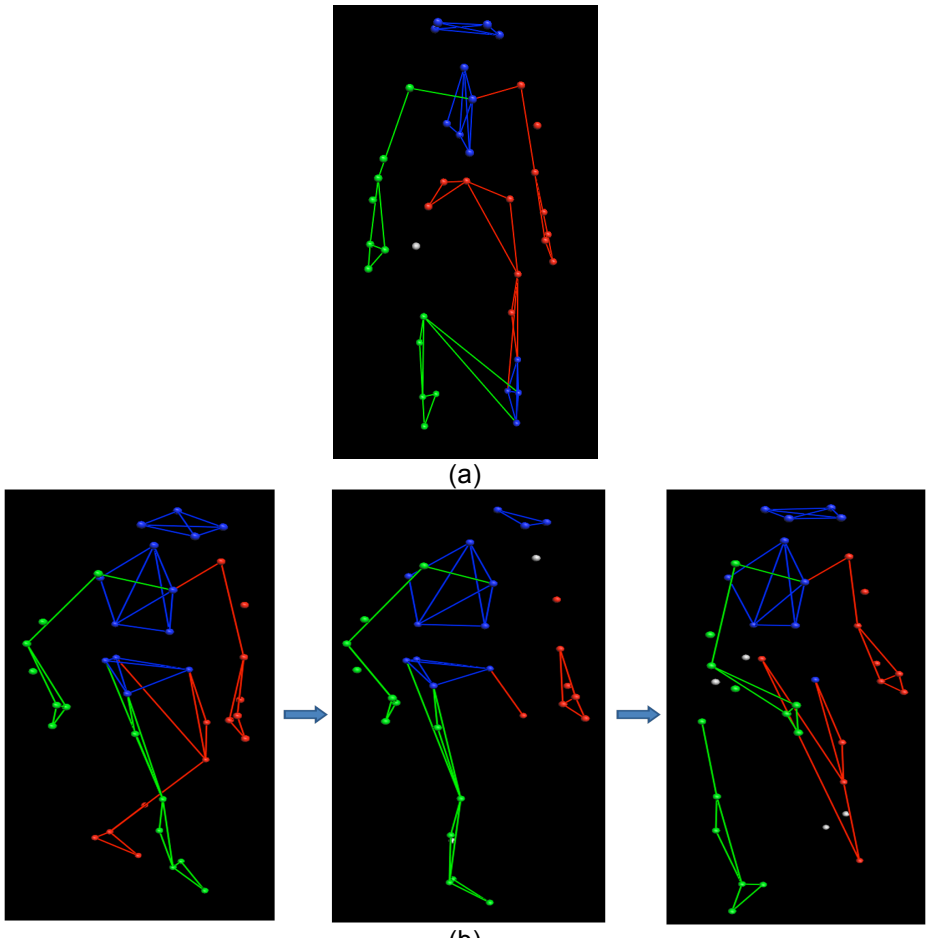
Figure 4.11: Various artifacts such as disappearing markers and falling markers: (a) Two consecutive frames where the STRN marker disappears, (b) Three non consecutive frames when the RTOE marker falls off the subject

The list of errors in order of frequency of occurrence is as follows:

• Jumpy markers: The marker changes its trajectory by a significant margin and distorts the rigid body model of the subject (see Fig. 4.10(a)).

• Shaky markers: The markers have a very inconsistent trajectory with a series of small but sharp jittery movements when a smooth trajectory is expected (see Fig. 4.10(b)).

• Markers changing position or jerky markers due to an impact of motion: The markers show jerky movement as they are on the skin instead of the bones of the subject (see Fig. 4.10(c)).



(a)

(b)

Figure 4.12: Markers being mislabeled: (a) Mislabeling at the beginning, (b) Mislabeling when the subject leaves the volume and reenters: (left) correctly labeled subject before the subject leaves the volume, (center) the subject is partially outside of the volume with some markers missing, (right) subject is completely in the volume again but contains mislabeled markers

• Markers disappears (dropouts): The marker data is not captured for a few frames (see Fig. 4.11(a)).

• Markers falling off the subject: During the capture markers fall from the subject (see Fig. 4.11(b)).

• Mislabeled models (swapping): Markers can be mislabeled in multiple ways. At the beginning, the markers are not labeled correctly (see Fig. 4.12(a)). In the middle of a capture, the marker is not labeled or incorrectly labeled even if the data exists. Subject leaves and reenters the volume. Once the subject reenters, the markers are incorrectly labeled (see Fig. 4.12(b)).

The causes of these errors can be due to various reasons such as occlusion, incorrect reconstruction, and skin movement to name a few. The errors are a result of one or a combination of the following:

- Occlusion: Occlusion occurs when 3D objects are not visible from a certain viewpoint due to other 3D objects obstructing the line of sight. At least two cameras must observe a marker in every frame to allow a 3D reconstruction. The quality of a capture increases as more cameras observe the same marker as it reduces the chance of occlusion. During capture, there are occasions where certain markers could not be captured because the cameras could not observe them in the space. Occlusions result in markers disappearing and, in some cases, jerky trajectories. There are several causes for occlusion. Self-occlusion occurs when a body part of the subject covers the marker. Objects such as a box or a suitcase used during the trials could obstruct the marker. Another subject in the capture space during trials such as shake hands will obstruct the markers. This was a more prominent problem during capture of the interaction dataset as two subjects were involved in all trials.

- Incorrect reconstruction of 3D marker location: This results in a jerky marker trajectory. This occurrence is particularly observed at the edges of the capture volume. The edges of the capture volume have lesser number of cameras covering the space, thus the accuracy and quality of the data captured goes down in these sections of the capture volume.
- Markers loosely attached to the subject: Markers were attached using two-sided tape. However, during capture the tape loses adhesive strength resulting in a very shaky marker and, consequently, a shaky trajectory. Whenever this was observed during the capture phase, the markers were replaced with new ones.
- Lost marker tracking: Once a subject leaves the volume partially or totally and comes back in the same trial, the marker labeling may not be correct as the system could not track the markers through the period when the subject was outside the capture volume. Tracking loss also occurs when the movement of an action is very fast. In these cases, the markers loose the label or are mislabeled. The system can be tuned to capture different speeds of motion. However, the problem occurs when an action involves different degrees of speed in the same trial.
- Skin Movement: The marker configuration model requires markers to be placed on bones as the model considers rigid body parts of the subject. However, the markers placed on the skin do not accurately follow the motion of the bone on which it is placed on. This results in marker trajectories that are very different from what is expected.
- Subjects touching or hitting the marker while doing an action: Subjects may accidently change the position or dislodge the marker while doing an action.

4.2.5 Techniques Used to Correct Errors

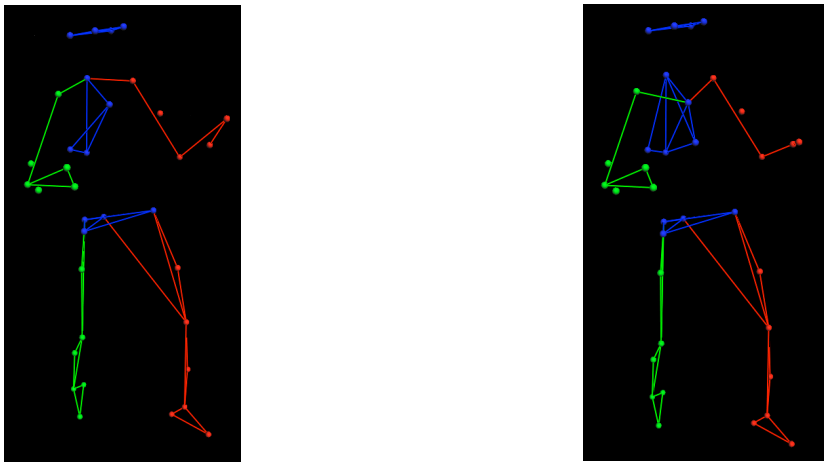
The motion capture system provides tools for post-processing. However, these tools are not capable of solving all the errors mentioned in section 4.2.4. We did not find any

documentation on post-processing and had to set guidelines and use the provided tools in various ways to solve each problem. The techniques used to correct errors are as follows:

- Woltring filter: This is a spline smoothing technique that is provided in the motion capture system. This filter can fill gaps in the marker trajectory by using a spline that tries to fit the missing section. It was observed that this generally provided a good solution for very short gaps. In the Human Motion Database, the use of the Woltiring filter was restricted to gaps whose length is less than 20 frames.
- Rigid body reconstruction macros: The maker configuration model as explained earlier assumes the human figure to consist of rigid body parts to estimate the joint angles. Each rigid body is made of at least three markers. A rigid body such as the head and thorax consists of more than three markers. In a trial, if a marker is missing from one of these rigid bodies, the coordinates of the lost marker can be inferred from the remaining reconstructed markers as long as at least three markers of that rigid body are present in the frame. This function can only work as long as there is one frame in the whole trial where all the markers constituting the rigid body part are reconstructed. For example, the reconstruction macro for the thorax gets all coordinates of markers in the thorax at a particular frame where all five markers in the thorax are reconstructed. Then it uses this information to fill gaps in the whole trial in frames which meet the criteria of at least three markers being present. This function is called “Perform Dynamic Body Language Modeling” and is available under the Workstation Operations in Vicon Nexus. We extended the functionality of this operation by modifying the macro and adapting the code to work for all rigid body parts having more than three markers, namely the head, forearms, the pelvis, and thighs. However, this function did not work very well in all cases. For instance, in an action where the subject bends down, the marker locations of the thorax are significantly different than when the person is standing. Thus, if a marker is missing while the subject is bent down and the macro chooses to calculate the

relative coordinates of the markers in the thorax from a standing position, the estimated marker position will be very different than what is expected.

- Pattern fill: In a trial, multiple markers may have similar trajectories for a few frames. For example, when a subject is walking straight, the markers on the head have very similar trajectories. If one of these markers is noisy, any other marker in the head that has a smooth trajectory can be used to fill or replace the faulty trajectory. The marker from which the pattern or trajectory is taken is called a source marker. However, the change must be evaluated subjectively for quality by checking whether the new trajectory is close to what is expected. Generally, this method involves multiple tries for each gap, using different source markers to fill the gap and find out which one provides the best result.
- Spline fill: The spline fill functionality is similar to the pattern fill as described above. In this case, the gaps are filled by creating a trajectory using cubic spline interpolation. This operation also needs to be subjectively evaluated for quality.



(a) (b)

Figure 4.13: The same frame before and after the cleaning process: (a) Multiple errors such as missing markers and mislabeling, (b) The same frame after post processing with the correct labeling. All the markers could not be reconstructed because there were no ways to generate the data with the degree of accuracy maintained in the database.

- Delete noisy data within files: There are actions where the subject leaves the capture volume and then reenters the volume in the same trial. The motion capture system cannot track and label the subject properly every time the subject reenters. If the subject leaves the volume multiple times, each section of the trial containing the intended motion needs to be disassociated with the previous as well as the next section. This is done by unlabeling all the markers at the beginning and at the end of each section. Then the frames inside each section are labeled. This method creates a gap of data within the motion that consists of unlabeled, noisy, and a partial marker set. These unlabeled markers are deleted. So, in the final output, the trial consists of sections of good data separated by a few frames containing no data.
- Crop data at the beginning and end: While capturing the trial, recording was stopped whenever errors, such as a marker falling off or occlusion for long periods, were detected. These recorded actions are trimmed to get the all the valid clean data while editing out the error prone sections. The motion capture system has the facility to crop only at the beginning and end of a motion trial.

An example of data cleaning can be seen in Figure 4.13 where the same frame is shown before and after the cleaning process. The experience of cleaning the motion data led us to a list of desirable tools and techniques that are not yet available in motion capture systems but will increase the quality of motion data as well as the speed of the cleaning process. These tools are as follows:

- Pattern fill with more options: The motion capture system allows the pattern fill operation using the trajectories from source markers only in the same time frame as the gap that is being filled. However, the motion trial may contain the same motion performed multiple times. In this case, the data of the same marker is not noisy in all the iterations in the same trial or a different trial. If the data from a different time frame

of the same marker could be used for the pattern fill, it would give an additional viable solution to the cleaning process.

- Rigid body macro with more options: The rigid body macro is only effective if there exist at least one frame with all the markers present in the trial. However, there are multiple trials done by the same subject in the same session. There were some trials where one of the markers was missing or noisy throughout the trial. These trials would have benefitted if the tool offered the option to choose a frame from another trial in the same capture session to reconstruct the missing marker. Another issue, as mentioned before, is the change of relative marker positions in a rigid body depending on the nature of the action. The example cited earlier states that the macro operation does not do an accurate job when the subject is bent down. Given the option to choose a specific source frame, such that the source frame is the one used to obtain the relative marker positions in the rigid body, we would be able to select a frame where the subject is bent down to provide a viable and possibly better alternative for this reconstruction.
- Sophisticated spline control: The cubic spline generated in the spline fill operation just gives two ways to control the spline. The spline can be changed by adjusting the first and last frames of the missing trajectory. That means, if the default generated spline with the borders at the beginning and at the end of the gap needs to be changed, we are forced to move the borders over the trajectory which was already there beyond the gap, thus overwriting good data with the spline data. If there were more controls to adjust the spline without changing the first and last frames of the gap, it would preserve the correct data.
- More markers in each rigid body part: If the marker configuration model has more markers per rigid body part then there is a higher chance that at least three markers are captured and reconstructed with good quality. Redundant markers would make the

generation of joint angle data more robust and there would be a higher chance of using the rigid body reconstruction macros to recover missing marker trajectories.

- Markers uniquely identifiable: If the markers on the subject were each uniquely identifiable, then the labeling problem would become much easier. The markers on the subject are passive and all alike. There are motion capture systems that have active markers emitting unique signatures, which could yield much better auto labeling solutions.
- Flexible reconstruction parameters: The reconstruct and label operation has adjustable parameters such as marker movement speed, label model rigidity, quality speed, and minimum number of cameras per marker. These parameters can be set to reconstruct and then label a trial. However, a trial may contain different types of data at different sections within the trial. The more the number of cameras observing the same marker, the more accurate the data is. There are sections in the trial, which can have very good quality data while sections where only two cameras can observe the marker. The current system allows only an optimized setting for the whole trial where we reach a compromise with the quality of overall data in order to get marker trajectories such that only a lesser number of cameras were observing the markers. Similarly an action such as a baseball hit may consist of both slow and high speed sections. However, the setting is not flexible enough to accommodate such actions.

4.2.6 Generation of output data

The outputs generated from the motion capture system are c3d and csv files. After a skeleton is designed, the motion capture files in the bvh format were created. The c3d or Coordinate 3d files is in binary format and contains the 3D marker coordinates for all the frames. The csv or comma separates variable files is in text format and contains both the marker coordinates and the joint angles values for all the frames. The bvh or BioVision

Hierarchy files were generated using a custom made skeleton. The hierarchy of the skeleton is discussed in detail in Chapter 5.

CHAPTER 5

SKELETON DESIGN

Motion data can be represented and studied using both the spatial location of the body parts or from joint angles of a skeletal model. The spatial location approach uses a sparse model of the actual body using 3-dimensional points [6]. The movement of these 3-dimensional points over a period of time is used to represent the body motion. Both the number of points and the accuracy of all points determine the quality of the data. There may be some metadata regarding the identity of each point on the model as to which part of the body it actually corresponds to.

The other approach uses the movement of articulated joints in the body to study human motion. This is based on the assumption that the human body is a system of rigid links connected by joints. Although the human body is not actually rigid and the joints are not non-extensible this assumption allows motion to be studied using principles of angular kinematics. This approach also takes advantage of the internal skeleton of a human being, or whatever animal motion that is being studied, and represents human motion in a more anatomically congruent manner. The data obtained using this representation is a set of angles for each joint in the skeleton over a period of time. The joints depend on the skeletal model used to collect the data. Different skeletal models are used in different systems as well in the same system with different objectives. The level of detail may vary from just a basic skeletal model having all the major joints to models including details on individual fingers and facial features.

Motion data captured from our system in its raw format is a just a collection of points which are a representation of the physical markers placed on the subject's body. These points

are spatial data in the world Cartesian coordinate system defined during the calibration phase of the system setup. A skeletal model is created and fitted to the set of markers to obtain angular data for the joints of the skeleton. Skeletal motion capture formats such as bvh and asf/amc are widely used in the animation industry and it also gives a good visual representation of the motion before animators actually rig a character on it.

In this chapter, we will first discuss the challenges and design approach of constructing a skeleton from a marker set. Next, we describe the creation of the hierarchy of the skeletal structure. Finally, we discuss the theory, method, and code used to implement our automatic skeleton building process.

5.1 Skeleton Design Considerations

Designing a skeleton involves the following design considerations: the number of bones in the skeleton, the hierarchy of bones, the position and orientation of the bones, and the markers to drive the motion of each bone. To design a skeleton, we took inspiration from a real human skeleton. During a capture session, the markers are placed on specific points on bones of a subject. The maker configuration model, as described earlier in Chapter 4, determines the marker positions. The design of the skeleton uses the given marker locations and the label information as input. To construct a skeleton, we establish a relationship between subsets of markers and each one of the bones. If we consider each bone to be a rigid body, each of them must be associated with at least three markers to be reconstructed individually. As shown in Figure 5.1, the upper arm can be reconstructed using three markers which can be defined as the origin marker, the orientation marker, and the direction marker. At least three markers are needed to define a 3 dimensional transformation to place and orient the bone. Markers can be reused to build multiple bones. For example, the marker used for direction marker of the upper arm can be used as the origin marker for the lower arm.

Ideally, to reconstruct the human motion perfectly, the skeleton should have exactly the same number of bones as a real human subject. However, this process would require too many

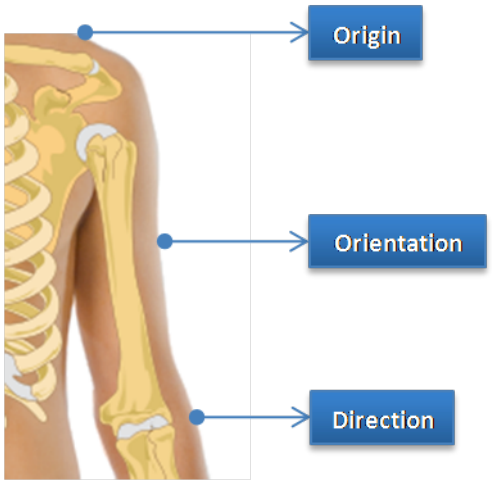


Figure 5.1: Typical markers used to calculate the parameters of each bone.

markers (at least one for each bone and two more shared with others) or some sort of estimation technique to find the relationship between a subset of markers and particular bone.For example, if we consider the human spine, there are 33 bones that compose the vertebral column. Either we need to have three markers assigned to each and every bone in the spine or we need to assign the same markers to multiple bones. In the former case, it is practically impossible to place that many markers on the spine and have a robust automatic labeling algorithm which consistently labels each of the closely placed markers correctly. In the latter case, we must discover the approximate relationship between the movement of each marker subset and the corresponding movement of the bone. To strike a compromise between approximating the actual bone movement and getting real data, we have to take design considerations which will consider the marker set that was used for capturing the data and develop a skeleton as close to a real human skeleton as possible.

We consider each bone to be a rigid body part and a subset of the markers drive each bone. A set of connected bones which are very close to each other can be approximated as one rigid bone. Hence, the spine is divided into upper spine and lower spine. With this setup, we shall obtain accurate data of the joint angle between the lower spine and the hips as well as the lower spine and the upper spine. This setup is necessary to estimate the real data from the

given marker set. The set of markers C7 (cervical vertebrae 7 in the lower neck), CLAV (clavicle), STRN (sternum), T10 (thoracic vertebrae 10), RBAK (right back), LPSI and RPSI (posterior hips) were used to calculate the lower and upper spine. If we were to divide the spine into 33 bones, we would have had to use the same set of seven markers to estimate the angles of these 33 bones. This way, only the lowest bone in the spine will contain real captured data and all the other bones connected to it will be synthetic data as approximated or estimated from that real data. We avoided such estimations.

5.2 Skeleton Hierarchy and Bones

The skeleton created consists of twenty one bones including a dummy root. A dummy root is needed to conform to the hierarchical tree structure of the bvh and asf/amc file formats. The dummy root does not have any dimension of its own. It does have a coordinate in the global coordinate system. This provides the initial location to find the transformation of all succeeding branches. All the bones in the hierarchical structure have a local coordinate system,

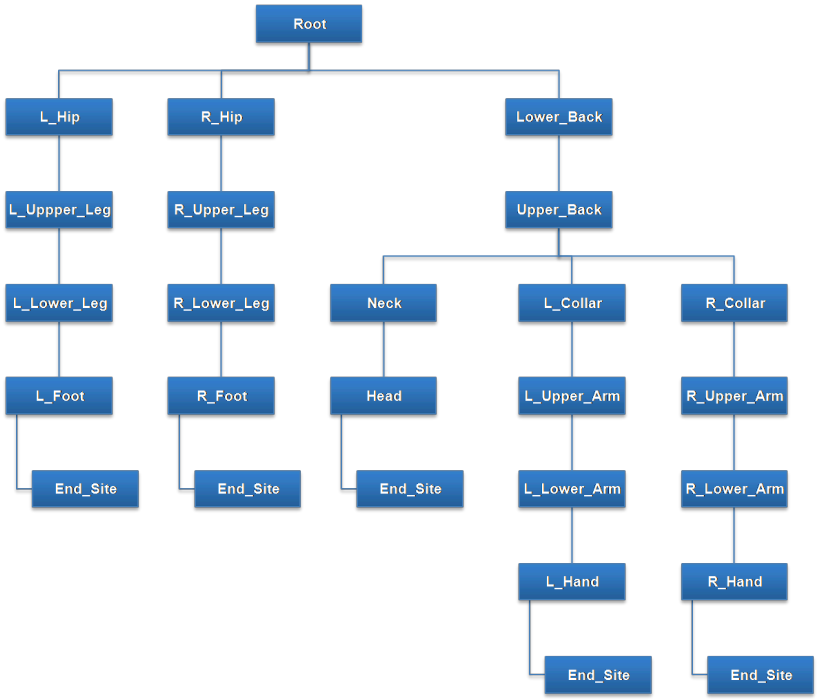


Figure 5.2: The hierarchy used to build our custom skeleton.

which is calculated with respect to its immediate parent. We have followed the convention of placing the root at the posterior hip location of the skeleton providing the base for the hip bones as wells as the lower spine.

Bones that did not have markers representing them are not added to the skeleton. Markers were placed on the base of the fingers and toes, so dummy bones like fingers and toes are not included in the skeleton. Pairs of bones doing very similar motions like the radius and ulna or the tibia and fibula are represented as one bone. Movements of bones such as the wrist bones residing between the forearm and the palm are impossible to capture using external markers and, consequently, avoided. The only exception was to include the neck bone joining the end of the spine to the base of the head. Although there were no markers to differentiate the movement between the neck and the head of the subject, the result of fusing the head and neck together produced a skeleton which generated a very unnatural animation. The hierarchy of the skeleton is shown in Figure 5.2. The end sites, as mentioned in the figure, are not actually bones but they are specified to indicate the end of a branch.

Table 5.1. The set of markers associated with each bone to set up the location, direction, and orientation of each bone.

Bone	Set of Markers
Root	LASI, RASI, LPSI, RPSI
L_Hip	LASI, LPSI, RPSI
L_Upper_Leg	LASI, LTHI, LKNE
L_Lower_Leg	LKNE, LTIB, LANK
L_Foot	LANK, LHEE, LTOE
R_Hip	RASI, LPSI, RPSI
R_Upper_Leg	RASI, RTHI, RKNE
R_Lower_Leg	RKNE, RTIB, RANK
R_Foot	RANK, RHEE, RTOE
Lower_Back	STRN, T10, LPSI, RPSI
Upper_Back	CLAV, C7, STRN, T10, RBAK
L_Collar	CLAV, C7, LSHO
L_Upper_Arm	LSHO, LUPA, LELB
L_Lower_Arm	LELB, LFRM, LWRA, LWRB
L_Hand	LWRA, LWRB, LFIN
R_Collar	CLAV, C7, RSHO
R_Upper_Arm	RSHO, RUPA, RELB
R_Lower_Arm	RELB, RFRM, RWRA, RWRB
R_Hand	RWRA, RWRB, RFIN
Neck	LBHD, RBHD, C7, CLAV, LSHO, RSHO
Head	LFHD, RFHD, LBHD, RBHD, C7, CLAV

5.3 Automatic Construction of Skeleton

Once the set of bones are decided, each bone must be placed and oriented in the three dimensional space. As the skeleton has a hierarchical structure, any transformation in a parent bone affects all its children. The position and orientation of the bones are controlled by a subset of markers as described in Section 5.1. The set of markers associated with each bone bone is indicated in Table 5.1. Each bone has its own local coordinate system and the axes of this

system must be aligned using the markers associated with the bone. Instead of visually aligning the bones, we implemented a script to calculate vectors from three markers and orient the bones using these vectors. Table 5.1 shows the markers associated with each bone. This enables automation and consistency in the generation of skeletons across all the fifty subjects in our database. As the orientation of one bone is not constrained by the orientation of its parent bone, all bones other than the root have three degrees of freedom. Although this violates the actual bone structure of real human joints, we assume this will not cause any problems because the marker data is gathered from real humans. The joint data derived from these markers should not break the natural laws.

The markers that are used to determine the position and orientation of a bone are also used to drive the rigid body bones during the motion. The hierarchical structure of the bones ensures that the joint angles of all the joints can be calculated with no gaps.

A script was written to create the skeleton automatically for every subject using the scripting language Vicon HSL that is provided with Vicon Blade. Using this script, we were able to generate a consistent skeleton for all the 50 different subjects. The script uses the 3D Cartesian coordinates of individual markers and calculates the position and orientation of each bone. The set of markers are used to create three parameters, the origin position, the aim vector, and the up vector. The bones are defined by its point of origin, the direction in which the bone points to from its parent to its child, and the direction in which the bone's Y axis will face. Other parameters of the bone such as the length of the bone are obtained from the origin of the bone to the origin of its child bones in the skeleton hierarchy. The end sites in the hierarchy define the length of the leaf nodes of the tree. This offset, called pre-translation, sets the length of the parent bone. The markers associated with each bone are used to calculate the origin position, the aim vector, and the up vector for each bone. The markers were always placed on bones of the human subjects. As the reconstructed skeleton consists of joints, the location of the joints need to be close to the real joints of a real human skeleton to result in realistic data.

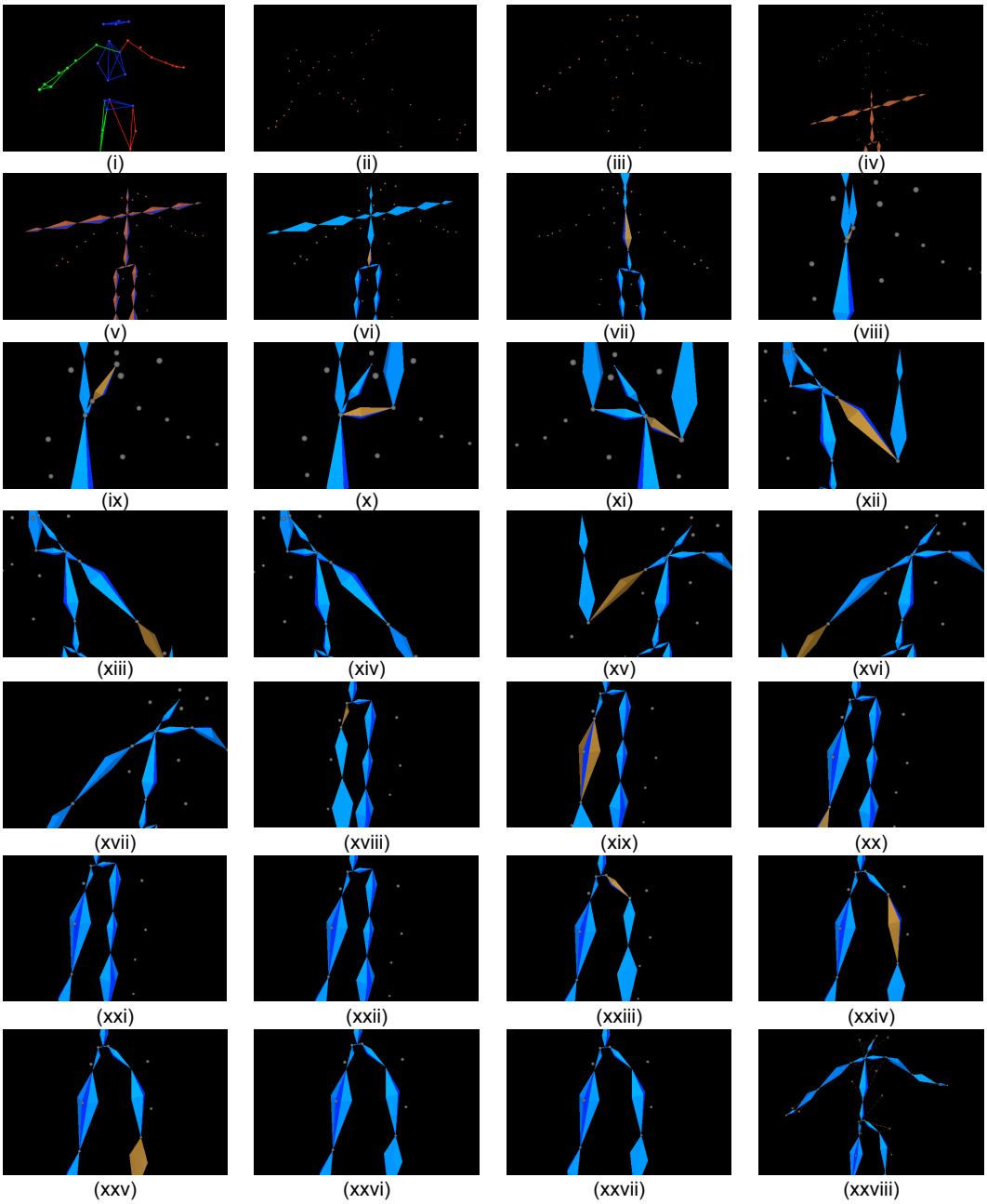


Figure 5.3: (i) Markers in Vicon Nexus, (ii) Markers in Vicon Blade in Z up format, (iii) Markers in Vicon Blade in Y up format, (iv) Actor_1.vst loaded, (v) Root is placed, (vi) Lower_Back is placed, (vii) Upper_Back is placed, (viii) Neck is placed, (ix) Head is placed, (x) L_Collar is placed, (xi) R_Collar is placed, (xii) L_Upper_Arm is placed, (xiii)) L_Lower_Arm is placed, (xiv) L_Hand is placed, (xv) R_Upper_Arm is placed, (xvi) R_Lower_Arm is placed, (xvii) R_Hand is placed, (xviii) R_Hip is placed, (xix) R_Upper_Leg is placed, (xx) R_Lower_Leg is placed, (xxi) R_Foot is placed, (xxii) L_Hip is placed, (xxiii) L_Upper_Leg is placed, (xxiv) L_Lower_Leg is placed, (xxv) L_Foot is placed, (xxvi) Whole figure is constructed with constraints.

As it not possible to obtain the actual measurement of the location of the real joints, we had to get the closest estimation of the respective joints from the marker subset provided. Joints such as elbows, shoulders, and knees had only one marker. The markers are placed at an offset to the actual joint position. However, for these joints it is not possible to calculate a point closer to the joint with consistency without more information. Joints that had two markers, such as the wrists, can be approximated and placed in the middle of the two markers. However, all these approximations were considered not to be highly erroneous except for the hip joints. It is advisable to put at least three markers on each joint to correctly estimate the actual joint location. A step by step diagram of each bone being placed automatically with the script can be seen in Figure 5.3 while the full script code can be seen in Appendix B.

The markers for the hip joints are placed on the anterior superior and posterior superior spines, which are prominent bone structures of the hip that can be measured from the outside. The actual hip joints are however at a greater offset with respect to other bones than the measurable marker positions captured during the sessions. If the joints are placed like other joints on the markers themselves it will bring a huge amount of error due to the distance from the actual location. However, there is a lot of material and research trying to solve this problem as this is an important problem in human gait analysis where precise location of joints are needed. Using the method described in [16], the markers RASI, LASI, RPSI, and LSI were used to calculate the locations of the hip joint centers. First, pelvic width is calculated as the distance between RASI and LASI. The hip joints have been calculated from the middle of the pelvic width. It is estimated to be offset 36% laterally, 22% posteriorly, and 30% caudally.

The other joint that does not lie on a marker is the joint between the neck and the head. As mentioned in Section 5.2, the neck bone was included in the skeleton design even though there were no markers to differentiate between the head and the neck. The joint has been placed at a location that is the weighted geometric center of the following markers: C7, CLAV, LSHO, RSHO, RFHD, LFHD, RBHD, and LBHD with only C7 being weighted by 2 while every

other marker is weighted by 1. The end site for the bone representing the head is placed at the center of RFHD, LFHD, RBHD, and LBHD. The location of the markers can be seen in Figure 4.1.

CHAPTER 6

METHODOLOGY

This chapter describes the structure of the Human Motion Database and the methodology developed to collect several aspects of human motion. First, we discuss and justify the need for a database like the Human Motion Database. Then, we introduce the goals and unique features in the database. Finally, we enumerate which areas, problems, and applications the database is targeted.

The construction of a benchmark database with attributes relevant to a particular problem creates an even measuring ground for all researchers. The advancement of motion capture technology in recent years has paved the way for good quality motion data which can be used to study human motion. There are several motion capture databases publicly available for the research community. The CMU Motion Capture Database [1] is the most widely used and contains a wide variety of actions organized in a very loose structure. On the other hand, databases like the HDM05 [12] have structure, but their range and quantity of motions are very limited compared to the CMU database. We have created a motion capture database by using novel techniques and sampling methods. The database has structure and can be directly applied to current research problems.

6.1 Understanding Human Motion

As mentioned in Chapter 2, the design of a structured database should be guided by specific goals. The main goals of Human Motion Database are: create a corpus of human motion, create data targeted towards specific research areas, provide ground truth data to quantitative evaluation of the performance of techniques, and provide data in a standard format to be used by the research community.

The ultimate purpose of the HMD is to provide data for the better understanding of human motion. Human motion consists of a range of different aspects. The same action can be preformed in various ways. Furthermore, human beings are equally adept at performing individual actions as well as well as combining them in various ways to produce complex actions. Interactive motions are also different, as they require coordination among two or more people. We believe to understand human motion we must study all these various aspects of human motion.

Actions can be classified as simple and complex from the perspective of the granularity of motions. Some simple actions, which semantically are considered to be well-defined, can be broken down into smaller sets of actions. For example, the action jump can be defined as a motion where a subject leaves the ground with both legs and then lands again. However, the actual jump motion is preceded by a bending from the knees and ankles without which performing a jump motion is practically impossible. If we consider the motion squat, the start of the action is performed in the same way as that of a jump. Almost all actions which have a semantically clear meaning can be divided and sub-divided into multiple actions which can again be defined individually. We have considered actions which are semantically defined as one action to be simple actions. Thus we consider jump to be a simple motion since it is defined as consisting of all sub-actions necessary to perform the motion. The identification of these subdivisions or segments is a research problem. The corpus of human motion in our database, named a praxicon, contains simple motions.

Human motion also varies and adjusts over age, gender, skeletal structure, and personal style of the person performing the actions. It has been shown that an observer, if shown the motion data without giving any background about the subject that was captured, can identify features such as gender and emotion [11]. We can also observe a distinct change in the performance of the same action over the skeletal structure of the subject. The HMD is built to

provide data to understand all these variations by achieving a distribution over these parameters.

Actions having the same meaning can be performed in many different ways. For example, the reach action can be defined as extending an arm to touch a point in space. However, the action may vary considerably depending on the target point of space the subject is trying to reach. The study of this aspect of the human motion is known as generalization.

Motions which are a combination of two simple actions are called complex actions. These complex actions can be composed by joining two motions sequentially one after another (transitioning), by performing two motions in parallel (splicing), and by performing motion in coordination with multiple subjects (interaction). These forms of compositionality are specific research areas and have been discussed in detail in Section 6.3. Our database contains both simple actions and complex actions.

The study of human motion must encompass the full repertoire of these variations identified above and the performance of any algorithm and technique can be properly evaluated when they are tested with data providing this variation.

The unique features of the Human Motion Database are: a wide range of motions to create a dictionary of commonly performed actions by humans, known as praxicon, to aid classification and recognition problems; a varied range of skeleton structures distributed over height and weight of the subject; the subjects are also distributed in terms of gender and age; data targeted towards the study of complex motions; and a set of interactions between two subjects. These features are explained in detail in the next sections.

6.2 Database Structure

6.2.1 Praxicon

A praxicon, a lexicon of human motion, is a novel concept that we are introducing in this database. Although existing databases contain a range of motions, there is no underlying methodology or well-defined rules in the creation of those databases to attempt a complete set

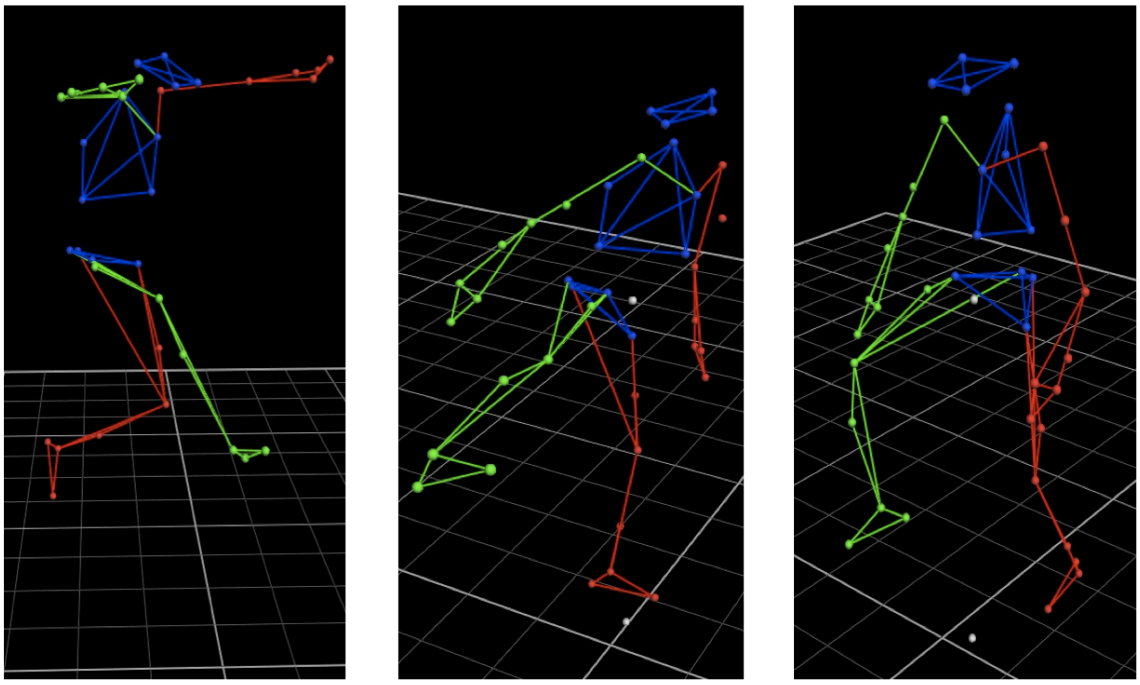


Figure 6.1: Meaningful, observable, and voluntary actions: jump, kick, and step up.

of human motions. The aim of our praxicon is to learn the primitives or the building blocks of human motion. Our praxicon contains over 350 actions performed by a single subject (Figure 6.1 shows three such actions). The list of all possible actions was constructed considering only meaningful, observable, and voluntary movement. To obtain a set of meaningful actions, we used the hierarchical structure of the lexical database WordNet [4] to select a set of meaningful "concrete" verbs that denote an action. Then the set is filtered by picking out only verbs corresponding to observable actions. Examples of non-observable actions are thinking and snoring which are actions but do not involve any perceivable motion. This set of meaningful, observable verbs is further filtered to find actions which are voluntary and are not initiated by someone other than the subject. The actions are a subset from a much larger set of all actions that a human can perform. The ultimate goal is to enable the learning of the primitives of motion. Hence, a wide range of different actions is chosen to capture all possible primitives

Other constraints during short listing of the actions to be included in the praxicon considered limitations of capturing actions which created too many occlusions like rolling on the

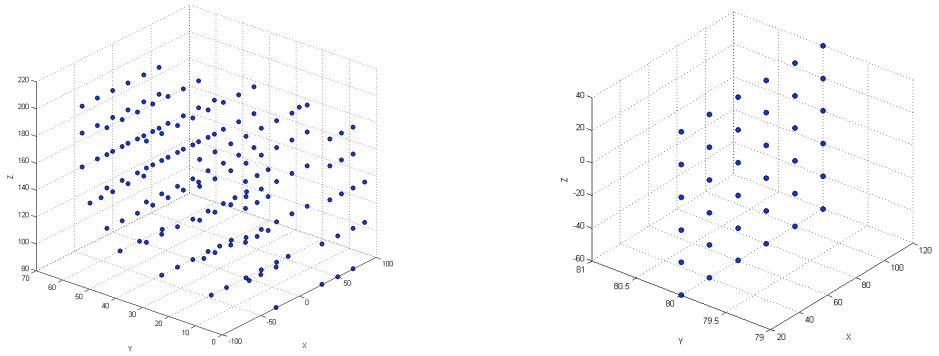
ground, limitations of the capturing environment thus not allowing actions such as swim or horse riding, and actions which cannot be consistently replicated in a natural manner such as tripping or slipping.

6.2.2 Range of skeletal structures

The range of skeletal structures is distributed over 50 different subjects. The distribution was constructed with the help of a Voronoi diagram as described in Section 3.3. This enabled the selection of candidates such that an even distribution and coverage is obtained. All 50 subjects perform the same set of 70 actions. This results in a set of 3500 motion trials distributed according to gender, height, weight, and age. This data can also be used to study retargeting, classification, and identification of motion as explained in Section 6.3.

6.2.3 Multiple ways to perform each action

When a motion is described semantically, the verb by itself does not define the exact way the action is performed. There must be additional adverbs or quantified values to describe a motion more accurately. For example, a person sits on a chair may make perfect sense semantically but there are external parameters such as the height of the chair, the reclining angle at the back which may alter the posture of the subject. Similarly the action of a person walking while carrying a box may depend significantly on the dimensions and weight of the box. The Human Motion Database contains data to study such phenomena.



(a) (b)

Figure 6.2: Target points in 3D space for actions in the generalization dataset: (a) Reach action, (b) Kick action

There are 270 motion trials captured from one subject which are divided into 163 reach motions, 7 sit-stand up motions, 36 kick motions, 32 step motions, 8 hop motions, 8 jog motions, 8 jump motions, and 8 walk motions. Each of the six sets provides data for the same action but performed differently with respect to a discrete volume of space. For example, the reach motion can be described as the subject reaching into a 3D discrete volume at various uniformly distributed points in the volume. The volume is defined as a right-handed Cartesian coordinate system with the origin at the projected center of mass on the floor when the subject is standing straight. There are 163 trials in this dataset where the subject does the reach motion and touches 163 different points in the volume (See Figure 6.2).

Similarly the sit-stand up set consists of the subject sitting and then standing up for 7 different heights of the chair. The kick motion has the subject kicking at various points in a fronto-parallel vertical plane. The step action consists of the subject stepping in different directions. The first motion is stepping directly forward which is termed 0 degrees. The next motion is to the right front of the subject at an angle of 45 degrees. These sets of 8 step actions in different directions are repeated at four varying step sizes. This yields a dataset of 32 motion trials. Each of the hop, jog, jump, and walk motions follow the same direction scheme as described in the step action. The motions are in different directions starting at 0 degrees and taking 45 degree increments. However, these motions are not repeated for varying step sizes and thus have only 8 trials each.

Additionally, a different subject has a set of 60 trials that are divided into 16 kick motions, 9 lift motions, and 35 reach motions. Each of these three actions provides data for the same action but performed with different kicking speed, lifting load weight, and reaching speed. The speed of the kick and reach motions are uniformly varied keeping the point where the subject reaches or kicks constant. The weights being lifted were also uniformly increased for the lift motions.

6.2.4 Complex Motion

Complex motion as described before is a combination of two or more actions. Complex motion can also include when two or more subjects are interacting with each other. The three datasets of complex motion in the Human Motion Database are the sequential composition dataset, the parallel composition dataset, and the interaction dataset.

The parallel composition dataset contains 99 motion files corresponding to sets of 41 different parallel motions. For example, each of the 41 sets will consist of the individual motions such as walk and wave separately as well as the motion performing walk and wave simultaneously.

The sequential composition dataset contains 53 motion files corresponding to sets of 39 different complex motions where the simple motions are sequentially concatenated one after the other. Each of the 39 sets includes individual motion files such as jog and jump as well as the jog and jump performed consecutively. There is also a co-articulation dataset where motions such as step and reach are first performed in a single template manner and, after that, concatenated with variations of the same motion. For example, all the step co-articulation motions have a common section which is stepping at a 45 degree angle. After this section, this single motion template is concatenated with a step motion ranging from 0 degrees to 315 degrees in 45 degree steps. Similarly, all the 8 reach motions have a 45 degree reach as the common control motion which is concatenated with a second reach towards directions ranging from 0 degrees to 315 degrees in 45 degree steps.

The interaction dataset contains actions that require mutual or reciprocal motions between multiple subjects. The actions may require synchronization between the subjects and the motion of one subject must always depend on the motion of the other. Examples of synchronized movements are dancing waltz and handshake where both subjects must perform mutually with respect to each other in order to execute the action. Non-synchronized actions are, for instance, push or slap where one subject initiates and executes the action and the other

subject exhibits the resultant involuntary action. There are 143 trials containing data involving two subjects performing actions conforming to the definition of interaction as explained above.

6.3 Research Problems

The different datasets of the HMD are designed to be used in various research problems pertaining to the study of different aspects of human motion. Problems such as classification and identification concern the cognitive challenge of motion. Retargeting is the problem of adapting the motion of one skeleton to that of another skeleton having different structure. Generalization is the challenge of converting all variations of a semantically unique action from a discretized space to a continuous space. Transitioning is the problem of concatenating two different actions such that the synthesized motion seems realistic. Splicing refers to the problem of merging actions performed by two or more different body parts into one single whole body motion. These problems are explained in detail in the following sub-sections and a relation is made on how the HMD will provide the ideal data to solve such problems.

6.3.1 Classification and Identification

Given that all actions can be performed in so many ways and so many styles, algorithms must approach the action recognition problem by discovering the fundamental features of each action. The corpus of motions and the range of skeletons will enable robust methods for motion cognition problems. The Human Motion Database contains enough data to train robust classifiers able to classify actions performed in various manners according to all aspects covered in our datasets. The motion cognition problems (*i.e.*, classification and identification) can be approached in the following different ways with the HMD:

- The 350 actions from the praxicon can be used to find a set of essential features for human movement. This way, each motion is represented by a subset of the essential features.

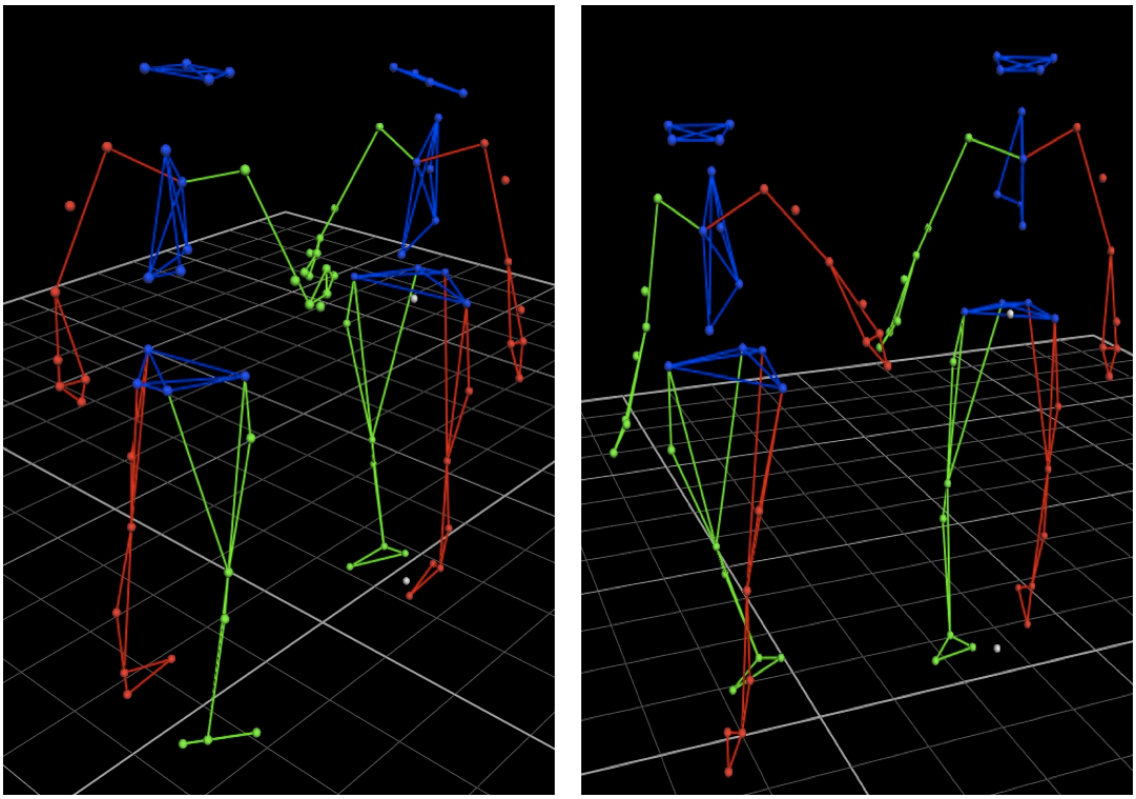


Figure 6.3: Interaction database with actions handshake (left) and passing a suitcase (right).

- The 50 different subjects perform the same subset of 70 actions in the praxicon. This can bring considerable skeletal structure variability to improve classifiers robustness.
- Actions are varied according to the attributes of the performing subject. The distribution of the database according to gender and age allow the study of methods for the identification of these attributes from motion data.
- Once the essential features of a particular action are identified, other features such as personal style may be discovered to enable the identification of an individual subject from motion data.
- The interaction dataset consisting of 143 actions should also be used to understand human motion in a more complete sense as a significant part of human activity is based on interaction (see Fig. 6.3).

6.3.2 Retargeting

Retargeting is the problem of adapting the motion performed by one subject to consider the articulated figure of another target subject that is structurally different from the source subject. The structural difference between articulated figures can be the length of each segment.

One of the main goals of retargeting is to keep as much of the original features of the motion as possible while adapting it to the target figure. For example, when the motion of a biped subject is retargeted to a quadruped subject, the retargeted quadruped motion is supposed to be similar to the bipedal motion. In animation, retargeting refers to using the motion captured from one subject to another character. The skeleton of the subjects must be different in at least one of the structural aspects. Animation retargeting can be extended to retarget human motion to humanoid robots that can be represented as an articulated figure.

The Human Motion Database provides motion data from 50 subjects with different skeletal structures distributed in terms of height and weight. Figure 6.4 shows four such

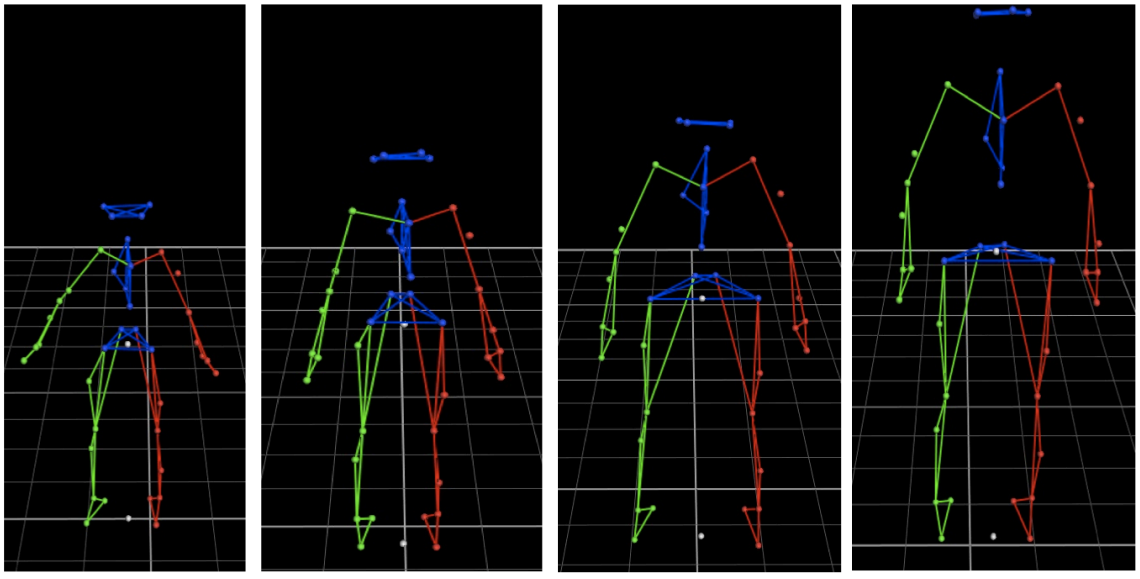


Figure 6.4: Different skeletal structures.

skeletons. The subjects perform the same subset of 70 actions in the praxicon. This provides ample data to divide into training and testing sets. For example, while 40 subjects can be used to train and learn how motion varies with skeletal structure, 10 subjects can be used to test. This provides the additional advantage of measuring the performance quantitatively with ground truth data. Another challenge of retargeting is to identify features which are independent of the structure of the skeleton and dependent only on external factors. For example, this dataset will help in identifying features such as “at least one foot should be on the floor while walking” while that feature is not necessarily true during a jump motion. Artifacts such as foot skating where the foot of the subject slides along the floor unnaturally are also undesirable. Current approaches aim to apply motion specific constraints to make the motion realistic. These constraints are however identified by humans and heuristics are developed to enforce these constraints. On the other hand, these constraints are unique to each particular type of motion. The Human Motion Database allows the study of these unique constraints for each action and consequently, the discovery of more robust methods for motion retargeting that consider different constraints for each action.

6.3.3 Generalization

Generalization is the problem where all the variations of a semantically defined action are parameterized from a discrete space to a continuous space. For example, a reach motion can vary according to the point where it reaches or the speed at which the reach action is performed. One approach to such a problem is to learn from the sample data where parameters such as speed and reaching point are known. Then the reach space or the speed variable can be parameterized to synthesize motion at any reaching point or at any speed within the range of the training data.

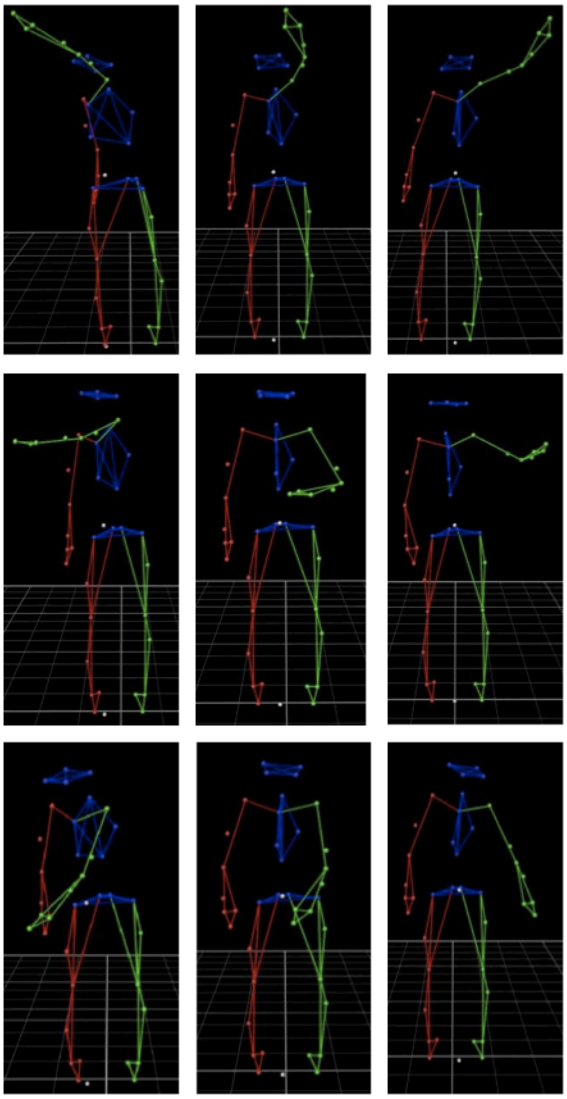


Figure 6.5: Reach motion data with nine target points of the whole dataset.

The Human Motion Database provides discrete data for 11 such actions as described in Section 6.2.3. For all actions, only one parameter such as speed or reach point or direction of movement is changed while everything else is kept constant. Figure 6.5 shows nine instances of the reach dataset.

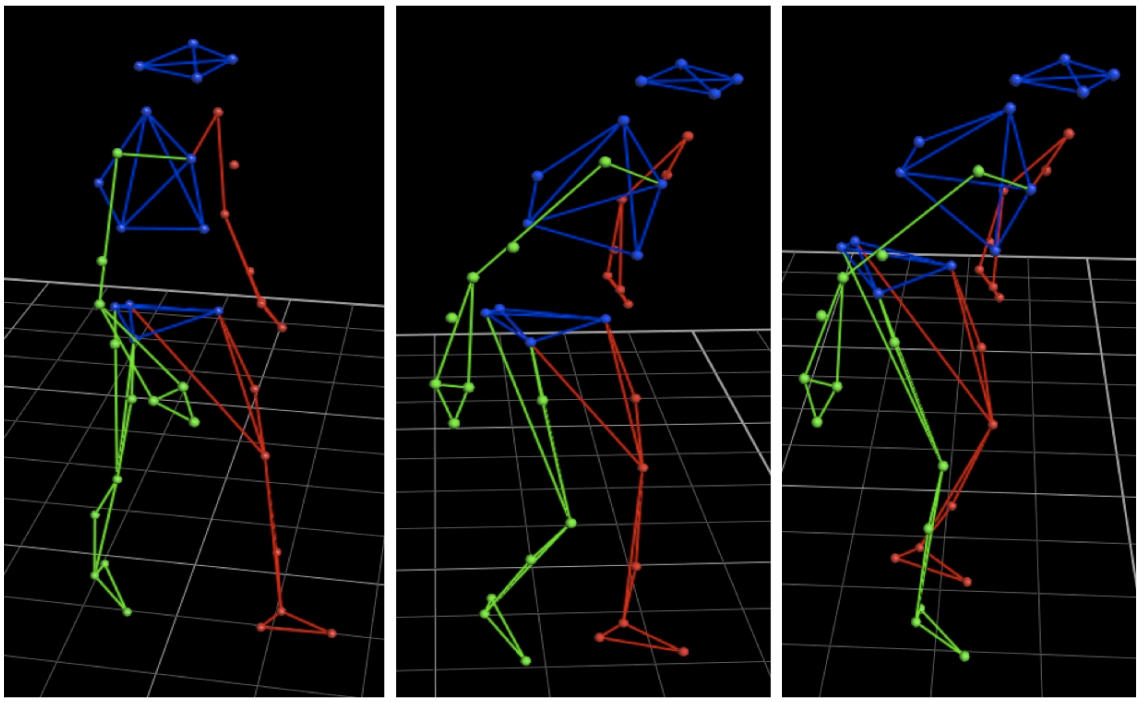


Figure 6.6: Transitioning data. A walk action (left), a jump action (center), and the ground truth data for a concatenated walk and jump action (right).

6.3.4 Transitioning

Transitioning can be defined as the sequential concatenation of different actions. When two actions are to be joined one after another, transitioning techniques verify whether their concatenation is feasible and then find the motion transition that takes a virtual character from one action to the next action. For example, a walk motion and a jump motion can be sequenced together to produce a walk and then a jump (See Figure 6.6). The Human Motion Database provides 39 sets of sequential motion data as part of the complex motion dataset, each of which consists of two individual trials of the subject performing the simple actions and one trial of the subject performing the transitioned action.

6.3.5 Splicing

The motion splicing problem consists in merging two different motions performed by different body parts into a single concurrent whole body motion. For example, a subject performing a walk motion and a wave motion separately can be spliced together to produce a

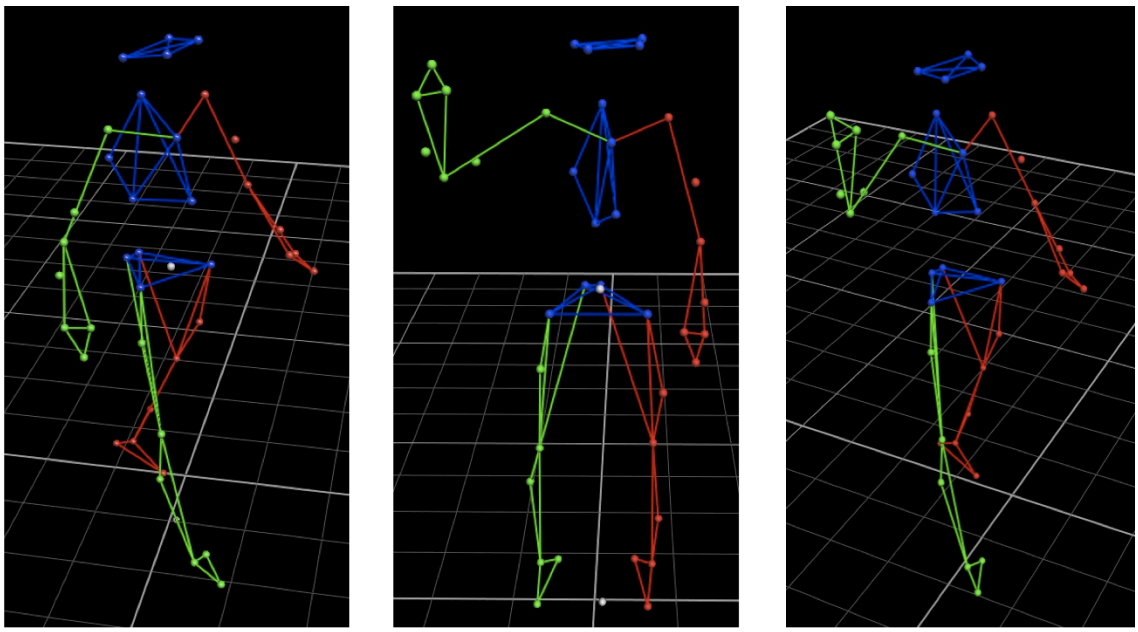


Figure 6.7: Splicing dataset. A walk action (left), a wave action (center), and the ground truth of a simultaneous walk and wave action (right).

single walk and wave motion being performed in parallel (see Fig. 6.7). The literature has various methods to solve automatic motion splicing but performance evaluation is subjective. The results of such algorithms are evaluated visually for a degree of realism. There are no databases that provide ground truth data that can be used to get a quantitative measurement by comparing the synthesized spliced data to the real motion data.

The Human Motion Database provides 41 sets of parallel motion data as part of the complex motion dataset, each of which consists of individual trials of the subject performing the two simple actions separately and one trial of the subject performing the spliced action.

CHAPTER 7

CONCLUSION

Motion databases have a strong potential to guide progress in the field of machine recognition and motion-based animation. Existing databases either have a very loose structure that do not sample the domain according to any controlled methodology or too few action samples which limits their potential to quantitatively evaluate the performance of motion-based techniques. The controlled sampling of the motor domain in the database may lead investigators to identify the fundamental difficulties of motion cognition problems and allow the addressing of these issues in a more objective way. In this thesis, we have described the construction of our Human Motion Database using controlled sampling methods (parametric and cognitive sampling) to obtain the structure necessary to the quantitative evaluation of several motion-based research problems.

A number of applications ranging from automatic surveillance, motion-based animation, humanoid robotics, smart environments, and human-machine interfaces can benefit from the structured data in our Human Motion Database. For example, the detection of suspicious activities can be studied to aid in automatic surveillance of possible dangerous situations. Another salient feature of our Human Motion Database is the presence of ground truth data for the compositionality dataset. This has the scope to introduce quantitative evaluation of synthesized data as opposed to subjective evaluation. The large collection of data also facilitates the study of human motion in a more complete sense by classifying the human repertoire of motion as a composition of different types of motion such as individual actions, iteractions, compositional or compund actions as well as generalized actions. We intend to provide the basis by which any algorithm can be judged or evaluated by their performance in

the different aspects as mentioned above. The novel sampling methodologies followed aims to give an insight into how motion varies parametrically with height, weight, age and gender of the subject. The sampling also extends to provide an understanding of how certain actions, even though they are performed differently, are perceived cognitively to be the same. For example, a person picking up different weights where the motion varies considerably depending on the amount of weight being picked.

The Human Motion Database is organized into several components: the praxicon dataset, the cross-validation dataset, the generalization dataset, the compositionality dataset, and the interaction dataset. The main contributions of this thesis include:

- a survey of human motion databases describing data sources related to motion synthesis and analysis problems
- a sampling methodology that takes advantage of a systematic controlled capture, denoted as cognitive sampling and parametric sampling
- a novel structured motion database organized into several datasets addressing a number of aspects in the motion domain
- a study of the design decisions needed to build a custom skeleton to generate joint angle data from marker data
- a study of the motion capture technologies and the general optical motion capture workflow including capturing and post processing data

APPENDIX A

WORLKFLOW TO GENERATE BVH FILES IN VICON BLADE

The workflow in Blade to create the Pipeline to generate bvh files is as follows:

- Click on Import file and load the clean and labeled subject calibration trial c3d file.
- If Vicon Blade Script Editor is not already open, go to the Editor tab and Click on Script Editor.
- Click on Open Script in Script Editor and load Snap_Skeleton.hsl.
- The first two lines convert the c3d data from Z up to Y up format and then load the custom skeleton .vst. Check to see if the path for the custom Actor_1.vst is correct.
- Click on Run Script.
- Visually check the result without changing the frame of the trial. Go to the Post Processing tab and click on Solve Motion. This should solve the motion for all the frames.
- If Vicon Blade Skeleton Editor is not already open, go to the Editor tab and click on Skeleton Editor.
- In the character box, make sure that Actor_1 is chosen such that the skeleton hierarchy is shown in a hierarchical representation below it.
- Click the Create Script button. Make sure that "Root Node", "To File" and "Generate Solver Info" are selected in the new box. Give a name and click OK to generate the .hsl solver script.
- If Vicon Blade Pipelines is not already open, go to the Editor tab and Click on Pipelines.
- Click on the circular Main button on the top left of Vicon Blade and click on Preferences.
- Click on Directories and make sure the path where the solver script was saved as well as the path to the script ZuptoYup (this was a script specially supplied by Vicon Support to change the coordinate system of the markers from Z up mode to Y up mode) is added in the script directories. Click on Reparse and then click Ok.

- In the Pipelines tab make a New Pipeline and add the following scripts: ZuptoYup, Solver script, and SolveSkeleton located in C:\ProgramFiles (x86) \Vicon \Blade1.6 \Scripts \DefaultOps\) in that order. Save the pipeline with an appropriate name.

The batch processing workflow in Blade to generate bvh is as follows:

- The Pipelines for Subject 1 to 50 as well as Subject A and B are already created and stored at C:\Program Files (x86)\Vicon\Blade1.6\Scripts\HMD_Pipelines and can just be loaded in the Pipelines tab. To solve a trial with an action, import the c3d and then just run the Pipeline for the Subject. Then click on Export at the top to get the required format including bvh.
- For batch processing, click on the Main Button on the top left of Vicon Blade and click on Batch Process.
- Select the Script/Pipeline, the output directory, the export type as bvh, and click on Add Files to add the required files. Click on Start. Trials of only one subject can be batch processed at a time.

APPENDIX B

SCRIPT TO GENERATE CUSTOM SKELETON AUTOMATICALLY

```
ZupToYup;
loadFile -importType "selCreateNew" "C:/Users/Arnab/Desktop/Skeletons/Actor_1.vst";

create Marker Origin;
create Marker Aim ;
create Marker Up;
vector $vKeys, $vPreTrans, $vPreRot;
float $KeyX, $KeyY, $KeyZ, $PreX, $PreY, $PreZ;

vector $RFHDPos       = `getPosition RFHD -ws`;
vector $LFHDPos       = `getPosition LFHD -ws`;
vector $RBHDPos       = `getPosition RBHD -ws`;
vector $LBHDPos       = `getPosition LBHD -ws`;
vector $C7Pos         = `getPosition C7   -ws`;
vector $T10Pos        = `getPosition T10  -ws`;
vector $RBAKPos       = `getPosition RBAK -ws`;
vector $CLAVPos       = `getPosition CLAV -ws`;
vector $STRNPos       = `getPosition STRN -ws`;
vector $RSHOPos       = `getPosition RSHO -ws`;
vector $RUPAPos       = `getPosition RUPA -ws`;
vector $RELBPos       = `getPosition RELB -ws`;
vector $RFRMPos       = `getPosition RFRM -ws`;
vector $RWRAPos       = `getPosition RWRA -ws`;
vector $RWRBPos       = `getPosition RWRB -ws`;
vector $RFINPos       = `getPosition RFIN -ws`;
vector $LSHOPos       = `getPosition LSHO -ws`;
vector $LUPAPos       = `getPosition LUPA -ws`;
vector $LELBPos       = `getPosition LELB -ws`;
vector $LFRMPos       = `getPosition LFRM -ws`;
vector $LWRAPos       = `getPosition LWRA -ws`;
vector $LWRBPos       = `getPosition LWRB -ws`;
vector $LFINPos       = `getPosition LFIN -ws`;
vector $RASIPos       = `getPosition RASI -ws`;
vector $LASIPos       = `getPosition LASI -ws`;
vector $RPSIPos       = `getPosition RPSI -ws`;
vector $LPSIPos       = `getPosition LPSI -ws`;
vector $RTHIPos       = `getPosition RTHI -ws`;
vector $RKNEPos       = `getPosition RKNE -ws`;
vector $RTIBPos       = `getPosition RTIB -ws`;
vector $RANKPos       = `getPosition RANK -ws`;
vector $RHEEPos       = `getPosition RHEE -ws`;
vector $RTOEPos       = `getPosition RTOE -ws`;
vector $LTHIPos       = `getPosition LTHI -ws`;
vector $LKNEPos       = `getPosition LKNE -ws`;
vector $LTIBPos       = `getPosition LTIB -ws`;
vector $LANKPos       = `getPosition LANK -ws`;
vector $LHEEPos       = `getPosition LHEE -ws`;
vector $LTOEPos       = `getPosition LTOE -ws`;

//Root
vector $Root_Origin      = << float( $RPSIPos.X + $LPSIPos.X )/2 , float( $RPSIPos.Y +
$LPSIPos.Y)/2, float( $RPSIPos.Z + $LPSIPos.Z)/2 >>;
```

```
setPosition Origin $Root_Origin;
vector $Root_Aim = <<float( (($RASIPos.X+$LASIPos.X)/2) ), float(
(($RASIPos.Y+$LASIPos.Y)/2)  ), float( (($RASIPos.Z+$LASIPos.Z)/2) ) >>;
setPosition Aim $Root_Aim;

vector $a = << float($Root_Aim.X-$Root_Origin.X), float($Root_Aim.Y-$Root_Origin.Y),
float($Root_Aim.Z-$Root_Origin.Z) >>;
vector $b = << float($Root_Origin.X-$Root_Origin.X), float(($Root_Origin.Y+1)-$Root_Origin.Y),
float($Root_Origin.Z-$Root_Origin.Z) >>;
vector $c = cross($a, $b);
vector $d = cross($c, $a);
setPosition Up << $Root_Origin.X+$d.X, $Root_Origin.Y+$d.Y, $Root_Origin.Z+$d.Z   >>;

select Origin Aim Up;
snapToSystemAlign Root 0 0 0 "Z" "Y" -allTime;

//Lower_Back
vector $Lower_Back_Origin = $Root_Origin;
setPosition Origin $Lower_Back_Origin;
vector $Lower_Back_Aim = $T10Pos;
setPosition Aim $Lower_Back_Aim;
vector $a = << float($Lower_Back_Aim.X-$Lower_Back_Origin.X), float($Lower_Back_Aim.Y-
$Lower_Back_Origin.Y), float($Lower_Back_Aim.Z-$Lower_Back_Origin.Z) >>;
vector $b = << float($STRNPos.X-$Lower_Back_Origin.X), float($STRNPos.Y-
$Lower_Back_Origin.Y), float($STRNPos.Z-$Lower_Back_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($Lower_Back_Origin.X+$c.X), float($Lower_Back_Origin.Y+$c.Y),
float($Lower_Back_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign Lower_Back 0 0 0 "Y" "X" -allTime;
select Aim Upper_Back;
snapTo -allTime;

$vPreTrans = `getVectorProperty Lower_Back "Pre_Translation"`;
select Lower_Back;
$KeyY = `getFloatProperty Lower_Back "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod Lower_Back;
select Lower_Back; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty Lower_Back "Pre_Rotation"`;
select Lower_Back;
$KeyX = `getFloatProperty Lower_Back "Rotation.x"`;
$KeyY = `getFloatProperty Lower_Back "Rotation.y"`;
$KeyZ = `getFloatProperty Lower_Back "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
```

```
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod Lower_Back;
select Lower_Back; selectProperty Rotation; selectKeys -all; cutKeys;

//Upper_Back
vector $Upper_Back_Origin = $T10Pos;
setPosition Origin $Upper_Back_Origin;
vector $Upper_Back_Aim = $C7Pos;
setPosition Aim $Upper_Back_Aim;
vector $a = << float($Upper_Back_Aim.X-$Upper_Back_Origin.X), float($Upper_Back_Aim.Y-
$Upper_Back_Origin.Y), float($Upper_Back_Aim.Z-$Upper_Back_Origin.Z) >>;
vector $b = << float($STRNPos.X-$Upper_Back_Origin.X), float($STRNPos.Y-
$Upper_Back_Origin.Y), float($STRNPos.Z-$Upper_Back_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($Upper_Back_Origin.X+$c.X), float($Upper_Back_Origin.Y+$c.Y),
float($Upper_Back_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign Upper_Back 0 0 0 "Y" "X" -allTime;
select Aim Neck;
snapTo -allTime;
select Aim L_Collar;
snapTo -allTime;
select Aim R_Collar;
snapTo -allTime;

$vPreTrans = `getVectorProperty Upper_Back "Pre_Translation"`;
select Upper_Back;
$KeyY = `getFloatProperty Upper_Back "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod Upper_Back;
select Upper_Back; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty Upper_Back "Pre_Rotation"`;
select Upper_Back;
$KeyX = `getFloatProperty Upper_Back "Rotation.x"`;
$KeyY = `getFloatProperty Upper_Back "Rotation.y"`;
$KeyZ = `getFloatProperty Upper_Back "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod Upper_Back;
select Upper_Back; selectProperty Rotation; selectKeys -all; cutKeys;

//Neck
vector $Neck_Origin = $C7Pos;
```

```
setPosition Origin $Neck_Origin;
vector $Neck_Aim =
<<float(((2*$C7Pos.X+$RSHOPos.X+$LSHOPos.X+$RBHDPos.X+$LBHDPos.X)/6) ),
float(((2*$C7Pos.Y+$RSHOPos.Y+$LSHOPos.Y+$RBHDPos.Y+$LBHDPos.Y)/6) ),
float(((2*$C7Pos.Z+$RSHOPos.Z+$LSHOPos.Z+$RBHDPos.Z+$LBHDPos.Z)/6) ) >>;
setPosition Aim $Neck_Aim;
vector $a = << float($Neck_Aim.X-$Neck_Origin.X), float($Neck_Aim.Y-$Neck_Origin.Y),
float($Neck_Aim.Z-$Neck_Origin.Z) >>;
vector $b = << float($CLAVPos.X-$Neck_Origin.X), float($CLAVPos.Y-$Neck_Origin.Y),
float($CLAVPos.Z-$Neck_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($Neck_Origin.X+$c.X), float($Neck_Origin.Y+$c.Y),
float($Neck_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign Neck 0 0 0 "Y" "X" -allTime;
select Aim Head;
snapTo -allTime;

$vPreTrans = `getVectorProperty Neck "Pre_Translation"`;
select Neck;
$KeyY = `getFloatProperty Neck "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod Neck;
select Neck; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty Neck "Pre_Rotation"`;
select Neck;
$KeyX = `getFloatProperty Neck "Rotation.x"`;
$KeyY = `getFloatProperty Neck "Rotation.y"`;
$KeyZ = `getFloatProperty Neck "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod Neck;
select Neck; selectProperty Rotation; selectKeys -all; cutKeys;

//Head
vector $Head_Origin = $Neck_Aim;
setPosition Origin $Head_Origin;
vector $Head_Aim = <<float((($LFHDPos.X+$RFHDPos.X+$RBHDPos.X+$LBHDPos.X)/4) ),
float((($LFHDPos.Y+$RFHDPos.Y+$RBHDPos.Y+$LBHDPos.Y)/4) ),
float((($LFHDPos.Z+$RFHDPos.Z+$RBHDPos.Z+$LBHDPos.Z)/4) ) >>;
setPosition Aim $Head_Aim;
vector $a = << float($Head_Aim.X-$Head_Origin.X), float($Head_Aim.Y-$Head_Origin.Y),
float($Head_Aim.Z-$Head_Origin.Z) >>;
```

```
vector $b = << float($CLAVPos.X-$Head_Origin.X), float($CLAVPos.Y-$Head_Origin.Y),
float($CLAVPos.Z-$Head_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($Head_Origin.X+$c.X), float($Head_Origin.Y+$c.Y),
float($Head_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign Head 0 0 0 "Y" "X" -allTime;
select Aim Head_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty Head "Pre_Translation"`;
select Head;
$KeyY = `getFloatProperty Head "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod Head;
select Head; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty Head "Pre_Rotation"`;
select Head;
$KeyX = `getFloatProperty Head "Rotation.x"`;
$KeyY = `getFloatProperty Head "Rotation.y"`;
$KeyZ = `getFloatProperty Head "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod Head;
select Head; selectProperty Rotation; selectKeys -all; cutKeys;

//Head_end
select Aim Head_End;
snapTo -allTime;
$vPreTrans = `getVectorProperty Head_End "Pre_Translation"`;
select Head_End;
$KeyX = `getFloatProperty Head_End "Translation.x"`;
$KeyY = `getFloatProperty Head_End "Translation.y"`;
$KeyZ = `getFloatProperty Head_End "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod Head_End;
select Head_End; selectProperty Translation; selectKeys -all; cutKeys;
```

```
//L_Collar
vector $L_Collar_Origin = $Neck_Origin;
setPosition Origin $L_Collar_Origin;
vector $L_Collar_Aim = $LSHOPos;
setPosition Aim $L_Collar_Aim;
vector $a = << float($L_Collar_Aim.X-$L_Collar_Origin.X), float($L_Collar_Aim.Y-
$L_Collar_Origin.Y), float($L_Collar_Aim.Z-$L_Collar_Origin.Z) >>;
vector $b = << float($CLAVPos.X-$L_Collar_Origin.X), float($CLAVPos.Y-$L_Collar_Origin.Y),
float($CLAVPos.Z-$L_Collar_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Collar_Origin.X+$c.X), float($L_Collar_Origin.Y+$c.Y),
float($L_Collar_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Collar 0 0 0 "Y" "X" -allTime;
select Aim L_Upper_Arm;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Collar "Pre_Translation"`;
select L_Collar;
$KeyY = `getFloatProperty L_Collar "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod L_Collar;
select L_Collar; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty L_Collar "Pre_Rotation"`;
select L_Collar;
$KeyX = `getFloatProperty L_Collar "Rotation.x"`;
$KeyY = `getFloatProperty L_Collar "Rotation.y"`;
$KeyZ = `getFloatProperty L_Collar "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Collar;
select L_Collar; selectProperty Rotation; selectKeys -all; cutKeys;

//R_Collar
vector $R_Collar_Origin = $Neck_Origin;
setPosition Origin $R_Collar_Origin;
vector $R_Collar_Aim = $RSHOPos;
setPosition Aim $R_Collar_Aim;
vector $a = << float($R_Collar_Aim.X-$R_Collar_Origin.X), float($R_Collar_Aim.Y-
$R_Collar_Origin.Y), float($R_Collar_Aim.Z-$R_Collar_Origin.Z) >>;
vector $b = << float($CLAVPos.X-$R_Collar_Origin.X), float($CLAVPos.Y-$R_Collar_Origin.Y),
float($CLAVPos.Z-$R_Collar_Origin.Z) >>;
vector $c = cross($a, $b);
```

```
setPosition Up << float($R_Collar_Origin.X+$c.X), float($R_Collar_Origin.Y+$c.Y),
float($R_Collar_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Collar 0 0 0 "Y" "X" -allTime;
select Aim R_Upper_Arm;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Collar "Pre_Translation"`;
select R_Collar;
$KeyY = `getFloatProperty R_Collar "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod R_Collar;
select R_Collar; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Collar "Pre_Rotation"`;
select R_Collar;
$KeyX = `getFloatProperty R_Collar "Rotation.x"`;
$KeyY = `getFloatProperty R_Collar "Rotation.y"`;
$KeyZ = `getFloatProperty R_Collar "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Collar;
select R_Collar; selectProperty Rotation; selectKeys -all; cutKeys;

//L_Upper_Arm
vector $L_Upper_Arm_Origin = $LSHOPos;
setPosition Origin $L_Upper_Arm_Origin;
vector $L_Upper_Arm_Aim        = $LELBPos;
setPosition Aim $L_Upper_Arm_Aim;
vector $a = << float($L_Upper_Arm_Aim.X-$L_Upper_Arm_Origin.X),
float($L_Upper_Arm_Aim.Y-$L_Upper_Arm_Origin.Y), float($L_Upper_Arm_Aim.Z-
$L_Upper_Arm_Origin.Z) >>;
vector $b = << float($LUPAPos.X-$L_Upper_Arm_Origin.X), float($LUPAPos.Y-
$L_Upper_Arm_Origin.Y), float($LUPAPos.Z-$L_Upper_Arm_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Upper_Arm_Origin.X+$c.X), float($L_Upper_Arm_Origin.Y+$c.Y),
float($L_Upper_Arm_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Upper_Arm 0 0 0 "Y" "X" -allTime;
select Aim L_Lower_Arm;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Upper_Arm "Pre_Translation"`;
select L_Upper_Arm;
$KeyY = `getFloatProperty L_Upper_Arm "Translation.y"`;
```

```
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod L_Upper_Arm;
select L_Upper_Arm; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty L_Upper_Arm "Pre_Rotation"`;
select L_Upper_Arm;
$KeyX = `getFloatProperty L_Upper_Arm "Rotation.x"`;
$KeyY = `getFloatProperty L_Upper_Arm "Rotation.y"`;
$KeyZ = `getFloatProperty L_Upper_Arm "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Upper_Arm;
select L_Upper_Arm; selectProperty Rotation; selectKeys -all; cutKeys;

//L_Lower_Arm
vector $L_Lower_Arm_Origin = $LELBPos;
setPosition Origin $L_Lower_Arm_Origin;
vector $L_Lower_Arm_Aim = <<float( (($LWRAPos.X+$LWRBPos.X)/2) ), float(
((($LWRAPos.Y+$LWRBPos.Y)/2)  ), float( (($LWRAPos.Z+$LWRBPos.Z)/2) ) >>;
setPosition Aim $L_Lower_Arm_Aim;
vector $a = << float($L_Lower_Arm_Aim.X-$L_Lower_Arm_Origin.X),
float($L_Lower_Arm_Aim.Y-$L_Lower_Arm_Origin.Y), float($L_Lower_Arm_Aim.Z-
$L_Lower_Arm_Origin.Z) >>;
vector $b = << float($LFRMPos.X-$L_Lower_Arm_Origin.X), float($LFRMPos.Y-
$L_Lower_Arm_Origin.Y), float($LFRMPos.Z-$L_Lower_Arm_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Lower_Arm_Origin.X+$c.X), float($L_Lower_Arm_Origin.Y+$c.Y),
float($L_Lower_Arm_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Lower_Arm 0 0 0 "Y" "X" -allTime;
select Aim L_Hand;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Lower_Arm "Pre_Translation"`;
select L_Lower_Arm;
$KeyY = `getFloatProperty L_Lower_Arm "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod L_Lower_Arm;
select L_Lower_Arm; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty L_Lower_Arm "Pre_Rotation"`;
select L_Lower_Arm;
$KeyX = `getFloatProperty L_Lower_Arm "Rotation.x"`;
```

```
$KeyY = `getFloatProperty L_Lower_Arm "Rotation.y"`;
$KeyZ = `getFloatProperty L_Lower_Arm "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Lower_Arm;
select L_Lower_Arm; selectProperty Rotation; selectKeys -all; cutKeys;

//L_Hand
vector $L_Hand_Origin = $L_Lower_Arm_Aim;
setPosition Origin $L_Hand_Origin;
vector $L_Hand_Aim = $LFINPos;
setPosition Aim $L_Hand_Aim;
vector $a = << float($L_Hand_Aim.X-$L_Hand_Origin.X), float($L_Hand_Aim.Y-
$L_Hand_Origin.Y), float($L_Hand_Aim.Z-$L_Hand_Origin.Z) >>;
vector $b = << float($LWRAPos.X-$L_Hand_Origin.X), float($LWRAPos.Y-$L_Hand_Origin.Y),
float($LWRAPos.Z-$L_Hand_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Hand_Origin.X+$c.X), float($L_Hand_Origin.Y+$c.Y),
float($L_Hand_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Hand 0 0 0 "Y" "X" -allTime;
select Aim L_Hand_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Hand "Pre_Translation"`;
select L_Hand;
$KeyY = `getFloatProperty L_Hand "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod L_Hand;
select L_Hand; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty L_Hand "Pre_Rotation"`;
select L_Hand;
$KeyX = `getFloatProperty L_Hand "Rotation.x"`;
$KeyY = `getFloatProperty L_Hand "Rotation.y"`;
$KeyZ = `getFloatProperty L_Hand "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Hand;
select L_Hand; selectProperty Rotation; selectKeys -all; cutKeys;
```

```
//L_Hand_end
select LFIN L_Hand_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Hand_End "Pre_Translation"`;
select L_Hand_End;
$KeyX = `getFloatProperty L_Hand_End "Translation.x"`;
$KeyY = `getFloatProperty L_Hand_End "Translation.y"`;
$KeyZ = `getFloatProperty L_Hand_End "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod L_Hand_End;
select L_Hand_End; selectProperty Translation; selectKeys -all; cutKeys;

//R_Upper_Arm
vector $R_Upper_Arm_Origin = $RSHOPos;
setPosition Origin $R_Upper_Arm_Origin;
vector $R_Upper_Arm_Aim       = $RELBPos;
setPosition Aim $R_Upper_Arm_Aim;
vector $a = << float($R_Upper_Arm_Aim.X-$R_Upper_Arm_Origin.X),
float($R_Upper_Arm_Aim.Y-$R_Upper_Arm_Origin.Y), float($R_Upper_Arm_Aim.Z-
$R_Upper_Arm_Origin.Z) >>;
vector $b = << float($RUPAPos.X-$R_Upper_Arm_Origin.X), float($RUPAPos.Y-
$R_Upper_Arm_Origin.Y), float($RUPAPos.Z-$R_Upper_Arm_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($R_Upper_Arm_Origin.X+$c.X), float($R_Upper_Arm_Origin.Y+$c.Y),
float($R_Upper_Arm_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Upper_Arm 0 0 0 "Y" "X" -allTime;
select Aim R_Lower_Arm;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Upper_Arm "Pre_Translation"`;
select R_Upper_Arm;
$KeyY = `getFloatProperty R_Upper_Arm "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod R_Upper_Arm;
select R_Upper_Arm; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Upper_Arm "Pre_Rotation"`;
select R_Upper_Arm;
$KeyX = `getFloatProperty R_Upper_Arm "Rotation.x"`;
$KeyY = `getFloatProperty R_Upper_Arm "Rotation.y"`;
$KeyZ = `getFloatProperty R_Upper_Arm "Rotation.z"`;
```

```
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Upper_Arm;
select R_Upper_Arm; selectProperty Rotation; selectKeys -all; cutKeys;

//R_Lower_Arm
vector $R_Lower_Arm_Origin = $RELBPos;
setPosition Origin $R_Lower_Arm_Origin;
vector $R_Lower_Arm_Aim = <<float( (($RWRAPos.X+$RWRBPos.X)/2) ), float(
(($RWRAPos.Y+$RWRBPos.Y)/2)  ), float( (($RWRAPos.Z+$RWRBPos.Z)/2) ) >>;
setPosition Aim $R_Lower_Arm_Aim;
vector $a = << float($R_Lower_Arm_Aim.X-$R_Lower_Arm_Origin.X),
float($R_Lower_Arm_Aim.Y-$R_Lower_Arm_Origin.Y), float($R_Lower_Arm_Aim.Z-
$R_Lower_Arm_Origin.Z) >>;
vector $b = << float($RFRMPos.X-$R_Lower_Arm_Origin.X), float($RFRMPos.Y-
$R_Lower_Arm_Origin.Y), float($RFRMPos.Z-$R_Lower_Arm_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($R_Lower_Arm_Origin.X+$c.X), float($R_Lower_Arm_Origin.Y+$c.Y),
float($R_Lower_Arm_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Lower_Arm 0 0 0 "Y" "X" -allTime;
select Aim R_Hand;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Lower_Arm "Pre_Translation"`;
select R_Lower_Arm;
$KeyY = `getFloatProperty R_Lower_Arm "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod R_Lower_Arm;
select R_Lower_Arm; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Lower_Arm "Pre_Rotation"`;
select R_Lower_Arm;
$KeyX = `getFloatProperty R_Lower_Arm "Rotation.x"`;
$KeyY = `getFloatProperty R_Lower_Arm "Rotation.y"`;
$KeyZ = `getFloatProperty R_Lower_Arm "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Lower_Arm;
select R_Lower_Arm; selectProperty Rotation; selectKeys -all; cutKeys;
```

```
//R_Hand
vector $R_Hand_Origin = $R_Lower_Arm_Aim;
setPosition Origin $R_Hand_Origin;
vector $R_Hand_Aim = $RFINPos;
setPosition Aim $R_Hand_Aim;
vector $a = << float($R_Hand_Aim.X-$R_Hand_Origin.X), float($R_Hand_Aim.Y-
$R_Hand_Origin.Y), float($R_Hand_Aim.Z-$R_Hand_Origin.Z) >>;
vector $b = << float($RWRAPos.X-$R_Hand_Origin.X), float($RWRAPos.Y-
$R_Hand_Origin.Y), float($RWRAPos.Z-$R_Hand_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($R_Hand_Origin.X+$c.X), float($R_Hand_Origin.Y+$c.Y),
float($R_Hand_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Hand 0 0 0 "Y" "X" -allTime;
select Aim R_Hand_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Hand "Pre_Translation"`;
select R_Hand;
$KeyY = `getFloatProperty R_Hand "Translation.y"`;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$vPreTrans = <<$vPreTrans.x, $PreY, $vPreTrans.z>>;
setProperty Pre_Translation $vPreTrans -onMod R_Hand;
select R_Hand; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Hand "Pre_Rotation"`;
select R_Hand;
$KeyX = `getFloatProperty R_Hand "Rotation.x"`;
$KeyY = `getFloatProperty R_Hand "Rotation.y"`;
$KeyZ = `getFloatProperty R_Hand "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Hand;
select R_Hand; selectProperty Rotation; selectKeys -all; cutKeys;

//R_Hand_end
select RFIN R_Hand_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Hand_End "Pre_Translation"`;
select R_Hand_End;
$KeyX = `getFloatProperty R_Hand_End "Translation.x"`;
$KeyY = `getFloatProperty R_Hand_End "Translation.y"`;
$KeyZ = `getFloatProperty R_Hand_End "Translation.z"`;
$PreX = $vPreTrans.x;
```

```
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod R_Hand_End;
select R_Hand_End; selectProperty Translation; selectKeys -all; cutKeys;

//R_Hip
vector $R_Hip_Origin = $RPSIPos;
setPosition Origin $R_Hip_Origin;

vector $Hip_Origin = <<float( (($RASIPos.X+$LASIPos.X)/2) ), float(
(($RASIPos.Y+$LASIPos.Y)/2)  ), float( (($RASIPos.Z+$LASIPos.Z)/2) ) >>;
vector $Hip_Vx = <<float( ($RASIPos.X-$LASIPos.X) ), float( ($RASIPos.Y-$LASIPos.Y)  ),
float( ($RASIPos.Z-$LASIPos.Z) ) >>;
float $PW = getLength($Hip_Vx);
$Hip_Vx = `normalize $Hip_Vx`;
vector $Hip_Vz = cross($Hip_Vx, << 0, 1, 0 >>); // Potential error: check order
vector $Hip_Vy = cross($Hip_Vx, $Hip_Vz);       // Potential error: check order
vector $R_Hip_Aim = <<float(
($Hip_Origin.X+0.36*$PW*$Hip_Vx.X+0.22*$PW*$Hip_Vy.X+0.30*$PW*$Hip_Vz.X) ), float(
($Hip_Origin.Y+0.36*$PW*$Hip_Vx.Y+0.22*$PW*$Hip_Vy.Y+0.30*$PW*$Hip_Vz.Y)  ), float(
($Hip_Origin.Z+0.36*$PW*$Hip_Vx.Z+0.22*$PW*$Hip_Vy.Z+0.30*$PW*$Hip_Vz.Z) ) >>;

setPosition Aim $R_Hip_Aim;
vector $a = << float($R_Hip_Aim.X-$R_Hip_Origin.X), float($R_Hip_Aim.Y-$R_Hip_Origin.Y),
float($R_Hip_Aim.Z-$R_Hip_Origin.Z) >>;
vector $b = << float($LPSIPos.X-$R_Hip_Origin.X), float($LPSIPos.Y-$R_Hip_Origin.Y),
float($LPSIPos.Z-$R_Hip_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($R_Hip_Origin.X+$c.X), float($R_Hip_Origin.Y+$c.Y),
float($R_Hip_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Hip 0 0 0 "Y" "X" -allTime;
select Aim R_Upper_Leg;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Hip "Pre_Translation"`;
select R_Hip;
$KeyX = `getFloatProperty R_Hip "Translation.x"`;
$KeyY = `getFloatProperty R_Hip "Translation.y"`;
$KeyZ = `getFloatProperty R_Hip "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod R_Hip;
```

```
select R_Hip; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Hip "Pre_Rotation"`;
select R_Hip;
$KeyX = `getFloatProperty R_Hip "Rotation.x"`;
$KeyY = `getFloatProperty R_Hip "Rotation.y"`;
$KeyZ = `getFloatProperty R_Hip "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Hip;
select R_Hip; selectProperty Rotation; selectKeys -all; cutKeys;

//R_Upper_Leg
vector $R_Upper_Leg_Origin = $R_Hip_Aim;
setPosition Origin $R_Upper_Leg_Origin;
vector $R_Upper_Leg_Aim = $RKNEPos;
setPosition Aim $R_Upper_Leg_Aim;
vector $a = << float($R_Upper_Leg_Aim.X-$R_Upper_Leg_Origin.X),
float($R_Upper_Leg_Aim.Y-$R_Upper_Leg_Origin.Y), float($R_Upper_Leg_Aim.Z-
$R_Upper_Leg_Origin.Z) >>;
vector $b = << float($RTHIPos.X-$R_Upper_Leg_Origin.X), float($RTHIPos.Y-
$R_Upper_Leg_Origin.Y), float($RTHIPos.Z-$R_Upper_Leg_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($R_Upper_Leg_Origin.X+$c.X), float($R_Upper_Leg_Origin.Y+$c.Y),
float($R_Upper_Leg_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Upper_Leg 0 0 0 "Y" "X" -allTime;
select Aim R_Lower_Leg;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Upper_Leg "Pre_Translation"`;
select R_Upper_Leg;
$KeyX = `getFloatProperty R_Upper_Leg "Translation.x"`;
$KeyY = `getFloatProperty R_Upper_Leg "Translation.y"`;
$KeyZ = `getFloatProperty R_Upper_Leg "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod R_Upper_Leg;
select R_Upper_Leg; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Upper_Leg "Pre_Rotation"`;
select R_Upper_Leg;
```

```
$KeyX = `getFloatProperty R_Upper_Leg "Rotation.x"`;
$KeyY = `getFloatProperty R_Upper_Leg "Rotation.y"`;
$KeyZ = `getFloatProperty R_Upper_Leg "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Upper_Leg;
select R_Upper_Leg; selectProperty Rotation; selectKeys -all; cutKeys;

//R_Lower_Leg
vector $R_Lower_Leg_Origin = $RKNEPos;
setPosition Origin $R_Lower_Leg_Origin;
vector $R_Lower_Leg_Aim = $RANKPos;
setPosition Aim $R_Lower_Leg_Aim;
vector $a = << float($R_Lower_Leg_Aim.X-$R_Lower_Leg_Origin.X),
float($R_Lower_Leg_Aim.Y-$R_Lower_Leg_Origin.Y), float($R_Lower_Leg_Aim.Z-
$R_Lower_Leg_Origin.Z) >>;
vector $b = << float($RTIBPos.X-$R_Lower_Leg_Origin.X), float($RTIBPos.Y-
$R_Lower_Leg_Origin.Y), float($RTIBPos.Z-$R_Lower_Leg_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($R_Lower_Leg_Origin.X+$c.X), float($R_Lower_Leg_Origin.Y+$c.Y),
float($R_Lower_Leg_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Lower_Leg 0 0 0 "Y" "X" -allTime;
select Aim R_Foot;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Lower_Leg "Pre_Translation"`;
select R_Lower_Leg;
$KeyX = `getFloatProperty R_Lower_Leg "Translation.x"`;
$KeyY = `getFloatProperty R_Lower_Leg "Translation.y"`;
$KeyZ = `getFloatProperty R_Lower_Leg "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod R_Lower_Leg;
select R_Lower_Leg; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Lower_Leg "Pre_Rotation"`;
select R_Lower_Leg;
$KeyX = `getFloatProperty R_Lower_Leg "Rotation.x"`;
$KeyY = `getFloatProperty R_Lower_Leg "Rotation.y"`;
$KeyZ = `getFloatProperty R_Lower_Leg "Rotation.z"`;
$PreX = $vPreRot.x;
```

```
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Lower_Leg;
select R_Lower_Leg; selectProperty Rotation; selectKeys -all; cutKeys;

//R_Foot
vector $R_Foot_Origin = $RANKPos;
setPosition Origin $R_Foot_Origin;
vector $R_Foot_Aim = $RTOEPos;
setPosition Aim $R_Foot_Aim;
vector $a = << float($R_Foot_Aim.X-$R_Foot_Origin.X), float($R_Foot_Aim.Y-
$R_Foot_Origin.Y), float($R_Foot_Aim.Z-$R_Foot_Origin.Z) >>;
vector $b = << float($RHEEPos.X-$R_Foot_Origin.X), float($RHEEPos.Y-$R_Foot_Origin.Y),
float($RHEEPos.Z-$R_Foot_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($R_Foot_Origin.X+$c.X), float($R_Foot_Origin.Y+$c.Y),
float($R_Foot_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign R_Foot 0 0 0 "Y" "X" -allTime;
select Aim R_Foot_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Foot "Pre_Translation"`;
select R_Foot;
$KeyX = `getFloatProperty R_Foot "Translation.x"`;
$KeyY = `getFloatProperty R_Foot "Translation.y"`;
$KeyZ = `getFloatProperty R_Foot "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod R_Foot;
select R_Foot; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty R_Foot "Pre_Rotation"`;
select R_Foot;
$KeyX = `getFloatProperty R_Foot "Rotation.x"`;
$KeyY = `getFloatProperty R_Foot "Rotation.y"`;
$KeyZ = `getFloatProperty R_Foot "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
```

```
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod R_Foot;
select R_Foot; selectProperty Rotation; selectKeys -all; cutKeys;

//R_Foot_end
select RTOE R_Foot_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty R_Foot_End "Pre_Translation"`;
select R_Foot_End;
$KeyX = `getFloatProperty R_Foot_End "Translation.x"`;
$KeyY = `getFloatProperty R_Foot_End "Translation.y"`;
$KeyZ = `getFloatProperty R_Foot_End "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod R_Foot_End;
select R_Foot_End; selectProperty Translation; selectKeys -all; cutKeys;

//L_Hip
vector $L_Hip_Origin = $LPSIPos;
setPosition Origin $L_Hip_Origin;
float $zero= 0.0;
$Hip_Vx = <<float($zero-$Hip_Vx.X), float($zero-$Hip_Vx.Y), float($zero-$Hip_Vx.Z)>>;
vector $L_Hip_Aim = <<float(
($Hip_Origin.X+0.36*$PW*$Hip_Vx.X+0.22*$PW*$Hip_Vy.X+0.30*$PW*$Hip_Vz.X) ), float(
($Hip_Origin.Y+0.36*$PW*$Hip_Vx.Y+0.22*$PW*$Hip_Vy.Y+0.30*$PW*$Hip_Vz.Y)  ), float(
($Hip_Origin.Z+0.36*$PW*$Hip_Vx.Z+0.22*$PW*$Hip_Vy.Z+0.30*$PW*$Hip_Vz.Z) ) >>;

setPosition Aim $L_Hip_Aim;
vector $a = << float($L_Hip_Aim.X-$L_Hip_Origin.X), float($L_Hip_Aim.Y-$L_Hip_Origin.Y),
float($L_Hip_Aim.Z-$L_Hip_Origin.Z) >>;
vector $b = << float($RPSIPos.X-$L_Hip_Origin.X), float($RPSIPos.Y-$L_Hip_Origin.Y),
float($RPSIPos.Z-$L_Hip_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Hip_Origin.X+$c.X), float($L_Hip_Origin.Y+$c.Y),
float($L_Hip_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Hip 0 0 0 "Y" "X" -allTime;
select Aim L_Upper_Leg;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Hip "Pre_Translation"`;
select L_Hip;
$KeyX = `getFloatProperty L_Hip "Translation.x"`;
$KeyY = `getFloatProperty L_Hip "Translation.y"`;
$KeyZ = `getFloatProperty L_Hip "Translation.z"`;
$PreX = $vPreTrans.x;
```

```
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod L_Hip;
select L_Hip; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty L_Hip "Pre_Rotation"`;
select R_Hip;
$KeyX = `getFloatProperty L_Hip "Rotation.x"`;
$KeyY = `getFloatProperty L_Hip "Rotation.y"`;
$KeyZ = `getFloatProperty L_Hip "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Hip;
select L_Hip; selectProperty Rotation; selectKeys -all; cutKeys;

//L_Upper_Leg
vector $L_Upper_Leg_Origin = $L_Hip_Aim;
setPosition Origin $L_Upper_Leg_Origin;
vector $L_Upper_Leg_Aim = $LKNEPos;
setPosition Aim $L_Upper_Leg_Aim;
vector $a = << float($L_Upper_Leg_Aim.X-$L_Upper_Leg_Origin.X),
float($L_Upper_Leg_Aim.Y-$L_Upper_Leg_Origin.Y), float($L_Upper_Leg_Aim.Z-
$L_Upper_Leg_Origin.Z) >>;
vector $b = << float($LTHIPos.X-$L_Upper_Leg_Origin.X), float($LTHIPos.Y-
$L_Upper_Leg_Origin.Y), float($LTHIPos.Z-$L_Upper_Leg_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Upper_Leg_Origin.X+$c.X), float($L_Upper_Leg_Origin.Y+$c.Y),
float($L_Upper_Leg_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Upper_Leg 0 0 0 "Y" "X" -allTime;
select Aim L_Lower_Leg;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Upper_Leg "Pre_Translation"`;
select L_Upper_Leg;
$KeyX = `getFloatProperty L_Upper_Leg "Translation.x"`;
$KeyY = `getFloatProperty L_Upper_Leg "Translation.y"`;
$KeyZ = `getFloatProperty L_Upper_Leg "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
```

```
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod L_Upper_Leg;
select L_Upper_Leg; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty L_Upper_Leg "Pre_Rotation"`;
select L_Upper_Leg;
$KeyX = `getFloatProperty L_Upper_Leg "Rotation.x"`;
$KeyY = `getFloatProperty L_Upper_Leg "Rotation.y"`;
$KeyZ = `getFloatProperty L_Upper_Leg "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Upper_Leg;
select L_Upper_Leg; selectProperty Rotation; selectKeys -all; cutKeys;

//L_Lower_Leg
vector $L_Lower_Leg_Origin = $LKNEPos;
setPosition Origin $L_Lower_Leg_Origin;
vector $L_Lower_Leg_Aim = $LANKPos;
setPosition Aim $L_Lower_Leg_Aim;
vector $a = << float($L_Lower_Leg_Aim.X-$L_Lower_Leg_Origin.X),
float($L_Lower_Leg_Aim.Y-$L_Lower_Leg_Origin.Y), float($L_Lower_Leg_Aim.Z-
$L_Lower_Leg_Origin.Z) >>;
vector $b = << float($LTIBPos.X-$L_Lower_Leg_Origin.X), float($LTIBPos.Y-
$L_Lower_Leg_Origin.Y), float($LTIBPos.Z-$L_Lower_Leg_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Lower_Leg_Origin.X+$c.X), float($L_Lower_Leg_Origin.Y+$c.Y),
float($L_Lower_Leg_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Lower_Leg 0 0 0 "Y" "X" -allTime;
select Aim L_Foot;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Lower_Leg "Pre_Translation"`;
select L_Lower_Leg;
$KeyX = `getFloatProperty L_Lower_Leg "Translation.x"`;
$KeyY = `getFloatProperty L_Lower_Leg "Translation.y"`;
$KeyZ = `getFloatProperty L_Lower_Leg "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod L_Lower_Leg;
select L_Lower_Leg; selectProperty Translation; selectKeys -all; cutKeys;
```

```
$vPreRot = `getVectorProperty L_Lower_Leg "Pre_Rotation"`;
select L_Lower_Leg;
$KeyX = `getFloatProperty L_Lower_Leg "Rotation.x"`;
$KeyY = `getFloatProperty L_Lower_Leg "Rotation.y"`;
$KeyZ = `getFloatProperty L_Lower_Leg "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Lower_Leg;
select L_Lower_Leg; selectProperty Rotation; selectKeys -all; cutKeys;

//L_Foot
vector $L_Foot_Origin = $LANKPos;
setPosition Origin $L_Foot_Origin;
vector $L_Foot_Aim = $LTOEPos;
setPosition Aim $L_Foot_Aim;
vector $a = << float($L_Foot_Aim.X-$L_Foot_Origin.X), float($L_Foot_Aim.Y-
$L_Foot_Origin.Y), float($L_Foot_Aim.Z-$L_Foot_Origin.Z) >>;
vector $b = << float($LHEEPos.X-$L_Foot_Origin.X), float($LHEEPos.Y-$L_Foot_Origin.Y),
float($LHEEPos.Z-$L_Foot_Origin.Z) >>;
vector $c = cross($a, $b);
setPosition Up << float($L_Foot_Origin.X+$c.X), float($L_Foot_Origin.Y+$c.Y),
float($L_Foot_Origin.Z+$c.Z) >>;
select Origin Aim Up;
snapToSystemAlign L_Foot 0 0 0 "Y" "X" -allTime;
select Aim L_Foot_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Foot "Pre_Translation"`;
select L_Foot;
$KeyX = `getFloatProperty L_Foot "Translation.x"`;
$KeyY = `getFloatProperty L_Foot "Translation.y"`;
$KeyZ = `getFloatProperty L_Foot "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod L_Foot;
select L_Foot; selectProperty Translation; selectKeys -all; cutKeys;

$vPreRot = `getVectorProperty L_Foot "Pre_Rotation"`;
select L_Foot;
$KeyX = `getFloatProperty L_Foot "Rotation.x"`;
$KeyY = `getFloatProperty L_Foot "Rotation.y"`;
```

```
$KeyZ = `getFloatProperty L_Foot "Rotation.z"`;
$PreX = $vPreRot.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreRot.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreRot.z;
$PreZ = $PreZ + $KeyZ;
$vPreRot = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Rotation $vPreRot -onMod L_Foot;
select L_Foot; selectProperty Rotation; selectKeys -all; cutKeys;

//L_Foot_end
select LTOE L_Foot_End;
snapTo -allTime;

$vPreTrans = `getVectorProperty L_Foot_End "Pre_Translation"`;
select L_Foot_End;
$KeyX = `getFloatProperty L_Foot_End "Translation.x"`;
$KeyY = `getFloatProperty L_Foot_End "Translation.y"`;
$KeyZ = `getFloatProperty L_Foot_End "Translation.z"`;
$PreX = $vPreTrans.x;
$PreX = $PreX + $KeyX;
$PreY = $vPreTrans.y;
$PreY = $PreY + $KeyY;
$PreZ = $vPreTrans.z;
$PreZ = $PreZ + $KeyZ;
$vPreTrans = <<$PreX, $PreY, $PreZ>>;
setProperty Pre_Translation $vPreTrans -onMod L_Foot_End;
select L_Foot_End; selectProperty Translation; selectKeys -all; cutKeys;

select Origin Aim Up;
delete;

//Adding Constraints
attach Actor_1\\Head Actor_1\\LFHD Actor_1\\LBHD Actor_1\\RFHD Actor_1\\RBHD;
attach Actor_1\\Neck Actor_1\\C7 Actor_1\\CLAV Actor_1\\LBHD Actor_1\\RBHD;
attach Actor_1\\L_Collar Actor_1\\C7 Actor_1\\CLAV Actor_1\\LSHO;
attach Actor_1\\L_Upper_Arm Actor_1\\LSHO Actor_1\\LUPA Actor_1\\LELB Actor_1\\CLAV;
attach Actor_1\\L_Lower_Arm Actor_1\\LWRA Actor_1\\LWRB Actor_1\\LFRM Actor_1\\LELB;
attach Actor_1\\L_Hand Actor_1\\LWRB Actor_1\\LWRA Actor_1\\LFIN Actor_1\\LFRM;
attach Actor_1\\Upper_Back Actor_1\\C7 Actor_1\\CLAV Actor_1\\RBAK;
attach Actor_1\\Lower_Back Actor_1\\T10 Actor_1\\STRN Actor_1\\RPSI Actor_1\\LPSI;
attach Actor_1\\Root Actor_1\\LPSI Actor_1\\RPSI Actor_1\\LASI Actor_1\\RASI;
attach Actor_1\\R_Hip Actor_1\\LPSI Actor_1\\RPSI Actor_1\\RASI;
attach Actor_1\\L_Hip Actor_1\\LPSI Actor_1\\RPSI Actor_1\\LASI;
attach Actor_1\\R_Upper_Leg Actor_1\\RASI Actor_1\\RTHI Actor_1\\RKNE Actor_1\\RPSI;
attach Actor_1\\R_Lower_Leg Actor_1\\RKNE Actor_1\\RTIB Actor_1\\RANK Actor_1\\RHEE;
attach Actor_1\\R_Foot Actor_1\\RANK Actor_1\\RHEE Actor_1\\RTOE Actor_1\\RTIB;
attach Actor_1\\L_Upper_Leg Actor_1\\LKNE Actor_1\\LTHI Actor_1\\LASI Actor_1\\LPSI;
attach Actor_1\\L_Lower_Leg Actor_1\\LKNE Actor_1\\LTIB Actor_1\\LANK Actor_1\\LHEE;
attach Actor_1\\L_Foot Actor_1\\LANK Actor_1\\LHEE Actor_1\\LTOE Actor_1\\LTIB;
attach Actor_1\\R_Collar Actor_1\\CLAV Actor_1\\C7 Actor_1\\RSHO;
```

```
attach Actor_1\\R_Upper_Arm Actor_1\\RSHO Actor_1\\RUPA Actor_1\\RELB Actor_1\\CLAV;
attach Actor_1\\R_Lower_Arm Actor_1\\RFRM Actor_1\\RELB Actor_1\\RWRA Actor_1\\RWRB;
attach Actor_1\\R_Hand Actor_1\\RFIN Actor_1\\RWRA Actor_1\\RWRB Actor_1\\RFRM;
```

APPENDIX C

LIST OF ACTIONS IN HMD

List of actions in the Praxicon :

adjust hair
answer phone
answer phone sitting
answer wall phone
arm wrestle
balance on one leg
bang door
beat
bend knee
bend over
bob horizontal
bob vertical
bounce
bow
braid
brush
brush teeth horizontal
brush teeth inside
brush teeth vertical
bump
bump in place
butterfly
call
carry both hands heavy
carry both hands light
carry both hands medium
carry one hand heavy
carry one hand light
carry suitcase
carry together heavy
cast spell
catch one hand
chair spin
chair walk backwards
chair walk backwards both
chair walk forward
chair walk forward both
chair walk sideways
check
chin in hand
chop
chop hor
chop ver
clap
clap cross
clap mute
climb ladder
cock head
coil rope
comb hair
come sign
crab backwards
crab forward
crab sideways
cradle
crawl
crawl backwards
crawl sideways
cross legs
crouch
cut with knife
dangle legs
dart
dial phone
dial phone circle
dig hole
dodge
drink bottle
drink water
drum
duck
dust off
elbow
embrace
fan
fan vertical
fishing
flail
flap
flex ankle
flex ankle B
flex bicep
flex elbow
flex tricep
flex wrist
flick
fling
flip coin
fluff up
get by
hail
halt
handover
hang clothes
head butt
heave
heft
high five
hike up shorts
hike up socks
hit
hit the dirt
hoist
hold hands
hop
hunch
hurdle
hurl
incline head
incline head hand
jerk open
jog
jump backwards
jump down
jump forward
jump inplace
jump inplace 180
jump inplace 90
jump sideways
jump up
jump walk
jumping jack
kick full
kick high
kick low
kick sideway
kick soccer pass
kneel
kneel both
knock door
lasso
lean object forward
lean object sideways
lift box
lift heavy
lift light
lift medium
limp
look around
look around 180
look back
look sidways
march
mop floor
nod maybe
nod no
nod yes
nose picking
nudge
open box

open-close bottle

pace step	pace step	pace step
paint wall	paint wall	paint wall
pass over	pass over	pass over
pass over suitcase	pass over suitcase	pass over suitcase
pass sign	pass sign	pass sign
pat shoulder	pat shoulder	pat shoulder
pick up	pick up	pick up
pick up floor	pick up floor	pick up floor
pick up suitcase	pick up suitcase	pick up suitcase
pick up together heavy	pick up together heavy	pick up together heavy
play guitar	play guitar	play guitar
play piano	play piano	play piano
play violin	play violin	play violin
poise	poise	poise
poke ground	poke ground	poke ground
pour	pour	pour
pronate	pronate	pronate
pronk	pronk	pronk
pull	pull	pull
pull chair	pull chair	pull chair
pull gun	pull gun	pull gun
pull heavy chair	pull heavy chair	pull heavy chair
pull nail	pull nail	pull nail
pull rope hor	pull rope hor	pull rope hor
pull rope hor cross	pull rope hor cross	pull rope hor cross
punch	punch	punch
punch curved	punch curved	punch curved
punch hook	punch hook	punch hook
punch uppercut	punch uppercut	punch uppercut
push	push	push
push chair	push chair	push chair
push heavy chair	push heavy chair	push heavy chair
push swing	push swing	push swing
pushup	pushup	pushup
put on cap	put on cap	put on cap
put on hat	put on hat	put on hat
put on headphones	put on headphones	put on headphones
reach	reach	reach

step on	
step on brakes	step on brakes
step over	step over
step up	step up
step up high	step up high
step up medium	step up medium
step up over	step up over
step up over medium	step up over medium
stir	stir
stir big	stir big
stomp	stomp
stoop	stoop
stop sign	stop sign
stretch back	stretch back
stretch head	stretch head
stretch kick sideways	stretch kick sideways
stretch sideways	stretch sideways
stretch thigh	stretch thigh
stretch thigh back	stretch thigh back
stretch thigh sideways	stretch thigh sideways
stretch triceps	stretch triceps
stretch triceps B	stretch triceps B
stride	stride
stroll	stroll
swagger	swagger
swat	swat
sweep broom	sweep broom
swing bat	swing bat
swipe	swipe
swipe reverse	swipe reverse
tap	tap
tap foot	tap foot
throw catch	throw catch
throw catch random	throw catch random
throw pitch	throw pitch
thrust	thrust
thump	thump
tip toe	tip toe
tip toe backwards	tip toe backwards

List of Interactions:

andoleta
arm wrestling
back pet
basketball ground pass
basketball
beat
boxing
brush hair
brush teeth
bump while walking
caress arms
caress face
carry on arms
carry together box
carry together flag
carry together luggage
carry together sofa chair
carry together table
chuck
ciranda
collar grab
comb hair
cut nails
dance forro
dance waltz
dodgeball
elbow
embrace
eskimo kiss
excercise arms
excercise legs
feel pulse
feel
fist pound
fold sheets
football
forehead slap
frogleap
give drink
give injection
hand kiss
handover keys
handshake
high five
le petit petat
lean on hand
lean on
left hand to upper body touch
mace
make out
march
massage back hit
massage back
massage feet
mutual face contact
mutual face kiss
mutual lip kiss
nudge
palpate stomach
pass both hands suitcase back
pass both hands suitcase front
pass both hands suitcase sideways
pass one hand box back horizontal
pass one hand box back
pass one hand box front vertical
pass one hand box front
pass one hand box sideways vertical
pass one hand box sideways
pass one hand suitcase back static
pass one hand suitcase back
pass one hand suitcase front static
pass one hand suitcase front
pass one hand suitcase sideways exaggerated
pass one hand suitcase sideways static
pass one hand suitcase sideways
pick up together box
pick up together luggage
pick up together sofa chair
pick up together table
piggyback ride
poke
pour water into glass from bottle
pour water into glass from jar
pressing noses
pull back
pull chair to sit
pull front one hand
pull front
pull hair
pull side
push back
push front
push side
push wheelchair
raise put chair
right hand to upper body touch
rock paper scissors
shake
slap on back
slap
slip keys
soccer
spin around chair
spin rope
sponge bath
spoon food
stab both hand down up front
stab both hand sideways front
stab both hand up down front
stab both hands down up back
stab both hands down up defence
stab both hands sideways back
stab both hands sideways defence
stab both hands up down back
stab both hands up down defence
stab one hand down up back
stab one hand down up front defence
stab one hand down up front
stab one hand sideways back left

stab one hand sideways back
stab one hand sideways defence
stab one hand sideways front
stab one hand up down back
stab one hand up down front defence
stab one hand up down front
stand on push back
stand on push side
stand on push
sword fight
throw catch ball one hand up
throw catch ball one hand
throw catch ball
throw catch box one hand
throw catch box
tickle
tug war
turn on push
volleyball
walk hand in hand
walk hug
wheelbarrow
whip far
whip short

REFERENCES

[1] CMU Mocap Database, 2001. http://mocap.cs.cmu.edu

[2] ICS Action Database, The University of Tokyo, 2003-2009. http://www.ics.t.u-tokyo.ac.jp/action/

[3] Georgia Tech Human Identification at Distance Database. http://www.cc.gatech.edu/cpl/projects/hid/

[4] Princeton University "About WordNet." WordNet. Princeton University. 2010. http://wordnet.princeton.edu

[5] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. Chang, S. Lee, and S. Narayanan, “IEMOCAP: Interactive emotional dyadic motion capture database,” Language Resources and Evaluation, vol. 42, no. 4, pp. 335-359, 2008.

[6] M. Dekeyser , K. Verfaillie and J. Vanrie, “Creating stimuli for the study of biological-motion perception,” *Behav. Res. Meth. Instrum. Comput.*, 34, 375–382, 2002.

[7] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, “Real Time Motion Capture Using a Single Time-Of-Flight Camera,” To appear in *CVPR 2010.*

[8] R. Gross and J. Shi, “The CMU motion of body MOBO database,” Technical report, Carnegie Mellon Univ., 2001.

[9] G. Guerra-Filho, “Optical Motion Capture: Theory and Implementation,” Brazilian Computing Society, Revista de Informática Teórica e Aplicada, 12(2), 61-89, 2005.

[10] B.-W. Hwang, S. Kim, and S.-W. Lee, “A full-body gesture database for automatic gesture recognition,” Proc. of Int. Conf. on Automatic Face and Gesture Recognition, pp. 243-248, 2006.

[11] Y. Ma, H. Paterson, and F. Pollick, "A motion capture library for the study of identity, gender, and emotion perception from biological motion," Behavior Research Methods, vol. 38, pp. 134–141, 2006.

[12] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, A. Weber. "Documentation: Mocap database HDM05," Technical report CG-2007-2, Universität Bonn, 2007.

[13] D. Roetenberg, “Inertial and magnetic sensing of human motion,” Ph.D. thesis, Twente University. 2006.

[14] L. Sigal and M. Black, “HumanEva: Synchronized video and motion capture dataset for evaluation of articulated human motion,” Technical Report CS-06-08, Brown Univ., 2006.

[15] D. Vlasic , R. Adelsberger , G. Vannucci , J. Barnwell , M. Gross , W. Matusik , J. Popović, “Practical motion capture in everyday surroundings,” ACM Transactions on Graphics (TOG), v.26 n.3, July 2007.

[16] K. Venesky , C. L. Docherty , J. Dapena , and J. Schrader, “Prophylatic ankle braces and knee varus-valgus and internal-external rotation torque,” Journal of Athletic Training, 41, 3, 2006.

BIOGRAPHICAL INFORMATION

Arnab Biswas did his undergraduate in College of Engineering & Management, Kolaghat where earned his Bachelor of Technology degree in Information Technology. After that he graduated from the University of Texas at Arlington with a Masters in Computer Science. For the grduate degree he developed a Human Motion Databse, which is a structured repository for human motion. He is interested in motion based animation problems and would like to apply his skills in the industry.