Motion Editing Using Combined Partial Motion Clips

By

Nadia Al-Ghreimil

B.S. in Computer Science and Applications, January 1992, The King Saud University

M.S. in Computer Science, May 1996, The George Washington University

A Dissertation submitted to

The Faculty of

The School of Engineering and Applied Science
of The George Washington University in partial satisfaction
of the requirements for the degree of Doctor of Science

December 12, 2002

Dissertation directed by

James Kwangjune Hahn
Professor of Engineering and Applied Science

ABSTRACT

Motion Editing using Combined Partial Motion Clips

By Nadia Al-Ghreimil

Directed by Professor James Kwangjune Hahn

In this dissertation a method is presented for allowing a motion clip to be edited by blending or merging it with another one. Previous methods exist that allow clips to be merged, but they do not allow degrees of freedom that are not defined in both clips to be affected. The work presented here is different. The merged in clip does not necessarily define all degrees of freedom yet the method used for merging allows all degrees of freedom in the original motion clip to be affected. This is achieved through the inclusion of scripts that contain equations for the estimation of the undefined degrees of freedom from the defined ones. These scripts approximate the correlation that exists naturally between joint movements, they are the results of thorough analysis of sample motions. Such a clip is called a Combined Partial Motion Clip.

The method is demonstrated by applying it to the example of throwing. A partial motion clip for throwing is obtained. It contains detailed information only for the degrees of freedom of the throwing arm and trunk. Yet it can be merged with another motion such as walking where even the degrees of freedom of the legs will be affected reasonably to produce a naturally looking final motion of a person throwing while walking. From a comparison of the resulting joint trajectories to originally captured ones, one can conclude that the method produced a realistic motion.

DEDICATION

This work is dedicated to my family; it would not have been possible without them. To my husband who stood beside me every step of the way. To our daughters, Sarah, Norah, and Reem, who had to play without me and came every few minutes to ask whether I was done with my work. To my mother and father, my sisters and brothers, who all took turn in keeping an eye on our children when my husband and I could not.

ACKNOWLEDGEMENT

I would like to thank my advisor Dr. James K. Hahn for his guidance and encouragement. Thanks to Dr. Steven Stanhope for his helpful comments on my early work results and for allowing me to use the equipment in the biomechanics laboratory at the Warren Grant Magnuson Clinical Center at the National Institutes of Health to capture the motion samples that are the core of this dissertation. Thanks to everyone at the lab who helped me setting up the equipment and running the software, especially Tom Kepple. More thanks go to the patient people who were my subjects and never complained even when all motions had to be captured again and again. I also want to thank my friends and family who gave me advice, encouraged me, and helped me out when I needed it most.

TABLE OF CONTENTS

ABSTRACT ii
DEDICATION iii
ACKNOWLEDGEMENT iv
TABLE OF CONTENTS v
LIST OF FIGURES viii
LIST OF TABLES x
LIST OF SYMBOLS xi
CHAPTER 1 INTRODUCTION 1
1.1 Motivation and Goals 1
1.2 Solution and Results 2
1.3 Outline 3
CHAPTER 2 PREVIOUS WORK 4
2.1 Articulated Figure Animation: generating motion clips 4
2.1.1 Keyframing 5
2.1.2 Inverse Kinematics 5
2.1.3 Dynamics 6
2.1.4 Goal-Directed methods 6
2.1.5 Motion Capture 6
2.2 Motion Editing: reusing existing motion clips 8
2.2.1 Frequency Domain 9
2.2.2 Displacement Mapping and Warping 10
2.2.3 Spacetime 11
2.2.4 Blending and Correspondence 12
2.2.5 Motion Spaces 14
2.2.6 Differencing and Emotion 16
2.2.7 Retargetting 18
2.2.8 Synthesizing and Texturing using Collections of Motion Captured Data 19
2.3 Games 20
2.4 Summary 22

CHAPTER 3 A TAXONOMY FOR MOTION EDITING 26
3.1 Main Classification 26
3.2 Classification of 'Mixing Motions' 28
CHAPTER 4 DATA FORMAT AND TERMINOLOGY 31
4.1 Terminology 31
4.1.1 Articulated Figure 31
4.1.2 Pose 32
4.1.3 Partial Pose 32
4.1.4 Trajectory 33
4.1.5 Motion 33
4.1.6 Partial Motion 33
4.1.7 Base Motion 33
4.2 Data Format 33
4.3 Data Conversion 34
CHAPTER 5 COMBINED PARTIAL MOTION CLIPS 40
5.1 Definition 40
5.2 Generating a Combined Partial Motion Clip 41
5.2.1 Data Collection 42
5.2.2 Data Manipulation 42
5.2.3 Partial Action Extraction 42
5.2.4 Data Analysis 47
5.2.5 Clip Combination 48
5.3 Using a Combined Partial Motion Clip 51
CHAPTER 6 EXPERIMENT 53
6.1 Generating a CPMC for 'throwing' 53
6.1.1 Data Collection 53
6.1.2 Data Manipulation 54
6.1.3 Partial Motion Extraction 57
6.1.4 Data Analysis 58
6.1.5 Clip Combination 60
6.1.6 A Three Subject Comparison 62
6.2 Using a CPMC for 'throwing' 68

CHAPTER 7 RESULTS 74
7.1 Verification of Results 74
7.2 Applying the Method to 'Tossing' 80
7.3 Exaggeration 82
7.4 Other Experiments 83
CHAPTER 8 CONCLUSION AND FUTURE WORK 85
8.1 Contribution 85
8.2 Limitations 86
8.3 Future Work 86
REFERENCES 89

APPENDIX A A SCRIPT FOR THE ESTIMATION OF THE NON-THROWING ARM
APPENDIX B A SCRIPT FOR THE ESTIMATION OF THE LEG DATA
APPENDIX C THE POSE FILE PLAYER

LIST OF FIGURES

Figure-1 Displacement Mapping. 11

Figure-2 Parametric Frame Space Interpolation. 15

Figure-3 The joints of an articulated figure and its hierarchy. 32

Figure-4 Adjusting Figure's segment lengths and rest angles. 34

Figure-5 Marker positions relative to articulated figure. 36

Figure-6 Effect of calibration and rest angles after assigning markers for joint orientation computation.. 37

Figure-7 Walking using a calibrated figure with rest angles. 38

Figure-8 Creating a CPMC 41

Figure-9 Difference vs. Transformation. 45

Figure-10 Addition vs. Transformation. 46

Figure-11 Addition vs. Transformation Frame Sequence.. 47

Figure-12 Using a CPMC 50

Figure-13 Algorithm for CPMC usage 52

Figure-14 Data Manipulation and Differencing.. 56

Figure-15 'Walk', 'walk&throw', and their difference 58

Figure-16 Comparison of original difference trajectories with computed ones. 60

Figure-17 Averaging Trajectories 61

Figure-18 Similarities within a base. 61

Figure-19 A generated 'walk&throw' for subject 1. 63

Figure-20 Difference Trajectories of 3 subjects. 64

Figure-21 Original vs. computed difference trajectories of the *non-throwing arm* while *standing* for 3 subjects. 65

Figure-22 Original vs. computed difference trajectories of the *non-throwing arm* while *walking* for 3 subjects. 66

Figure-23 Original vs. computed difference trajectories of the *left leg* while *walking* for 3 subjects. 67

Figure-24 Generated 'sit&throw' motion for two subjects. 69

Figure-25 Generated 'stand&throw' motions for two subjects.. 70

Figure-26 Generated 'walk&throw' motions for two subjects. 72

Figure-27 Comparison of selected generated ‘walk&throw’ trajectories for subject 3.....75
Figure-28 Comparison of generated ‘walk&throw’ for subject 3 76
Figure-29 Comparison of a generated and original ‘stand&throw’ for subject 2.....78
Figure-30 Comparison of selected generated ‘stand&throw’ trajectories for subject 2.....78
Figure-31 Comparison of a generated and original ‘stit&throw’ for subject 4. 79
Figure-32 Comparison of selected generated ‘sit&throw’ trajectories for subject 4.....79
Figure-33 Comparing a generated tossing while walking motion for subject 4 to an original tossing while walking for subject 1 80
Figure-34 Tossing while walking, walking , and their difference of subject 1. 81
Figure-35 Original toss difference trajectories and computed ones using the same scripts as for throwing.....81
Figure-36 Generated ‘walk&toss’ for subject 4 compared to and original ‘walk&toss’ of subject 1.....81
Figure-37 Exaggeration in the form of increased trunk rotation. 82
Figure-38 Exaggeration in the form of varying the resampling rate. 83
Figure-39 Pose File Player.....97

LIST OF TABLES

Table-1 Motion Editing Taxonomy – Part 1 27

Table-2 Motion Editing Taxonomy – Part 2: details of mixing motions 28

Table-3 Marker to Joint Assignments 36

LIST OF SYMBOLS

CPMC	Combined Partial Motion Clip
CycleLength	Length of walk cycle measured in number of frames
DOF	Degree of freedom
HMM	Hidden Markov Model
IK	Inverse Kinematics
LA	Left ankle marker
LF	Left foot marker
LH	Left hand marker
LHU	Left humerus marker
LLE	Lower Left Elbow: Marker just below left elbow
LLK	Lower Left Knee: Marker just below left knee
LP	Left pelvis marker
LS	Left shoulder marker
LUA	Left upper arm marker
LUE	Left Upper Elbow: Marker just above left elbow
LUK	Left Upper Knee: Marker just above left knee
LW	Left wrist marker
NumberOfFrames	The number of frames making up a motion
NumberOfHeelstrikes	The number of heel strikes in a throw-period
NumberOfJoints	The number of joints making up the articulated figure. Here it is = 14
RA	Right ankle marker
RF	Right foot marker
RH	Right hand marker
RHU	Right humerus marker
RLE	Lower Right Elbow: Marker just below right elbow
RLK	Lower Right Knee: Marker just below right knee
RP	Right pelvis marker
RS	Right shoulder marker
RUA	Right upper arm marker

RUE	Right Upper Elbow: Marker just above right elbow
RUK	Right Upper Knee: Marker just above right knee
RW	Right wrist marker
SHMM	Stylistic Hidden Markov Model
VR	Virtual Reality

CHAPTER 1
INTRODUCTION

A new motion editing method is presented in this dissertation. The method depends on the creation of partial motion clips, i.e. they define only some degrees of freedom (DOFs), which contain some additional information about how this partial motion clip may affect other DOFs when it is blended with another motion. Such partial motion clips are extracted from several similar samples that are thoroughly analyzed. The clips are easy to use and are space efficient. The method presented allows for a more natural looking resulting motion.

1.1 Motivation and Goals

Animating articulated figures is a difficult task due to the large number of DOFs that have to be controlled. It gets even more difficult if the figure is a human, because one quickly detects unnatural movements. Motion capture is an effective way to obtain natural looking motions. A drawback of motion capture is that it is hard to maintain specific constraints and therefore often needs editing. Depending on the representation used for the captured motion, different motion editing methods may be applied. There is always a tradeoff between the number of parameters that have to be set or adjusted and the amount of control the animator has over the final animation. Whatever the case, reducing the number of DOFs that have to be dealt with will simplify the task of the animator. One way to do that would be to take advantage of the correlation that exists between joint movements.

Assuming that a reaching motion of an actor while sitting is recorded that will later be incorporated into an animation, it is possible that in the resulting animation the animated character will not reach the desired position exactly or not at the correct instance in time, i.e. there is a difference in target reaching location or timing. This is a case that calls for editing. A simple displacement map or inverse kinematics could be used to solve the problem for the difference in location. Or a timewarp could be used to solve the problem for the difference in timing. But in cases where the differences is due to something else, like for example that the character is standing instead of sitting, the problem is harder to solve because the reaching motion that was recorded while sitting probably does not affect the motion of the legs at all, which might not be the case while standing. It is very likely that there will be some kind of

movement in the lower body as a result of the movement in the upper body. Therefore, if it is desired to use the same recorded 'reaching while sitting' motion and blend it with a standing motion to produce a 'reaching while standing' motion one has to have a way to induce some changes in the legs to make the resulting motion look natural.

Another thing to notice is that 'reaching' is an action mainly involving the upper body. Hence, it would be advantageous to store only the trajectories of the joints that are most active in the action (e.g., the reaching arm and the trunk) because that will reduce the needed storage space. But in that case, when this partial motion clip is merged with sitting, standing, or some other base motion one has to ask how this clip can be merged in properly since it is not desired to simply mask out the DOFs of the arm and trunk and replacing them with the ones in the stored clip (e.g. [Per95], [Ros96]), but to merge it into the base clip and allow this change to affect other DOFs not in the merged clip for a more realistic look.

For those reasons, relationships between body parts that hold for all subjects are being sought. The aim is not for physical correctness, but for a natural look. Additionally the necessary computations should be kept simple, and storage space requirements should be kept low. Hence a new kind of motion clip is introduced: Combined Partial Motion Clips or CPMCs, and a method for their generation and usage.

1.2 Solution and Results

A CPMC is a clip that is a combination of several partial motion data sets that have something in common: an action performed in various base poses. Partial motion means that only some joint trajectories are defined; the ones that are active in the common action. Furthermore, the CPMC contains equations to compute other joint trajectories from the ones included in the partial motion data set. This makes it possible for a CPMC to be used to add an action that primarily involves a specific part of the body to some existing base motion while allowing the uninvolved parts to be affected in a reasonable way and thereby producing a naturally looking motion.

The creation and usage is demonstrated for the specific example of throwing. A partial throwing motion was extracted from throwing while walking, throwing while standing, and

throwing while sitting. The data was analyzed and a CPMC was created, which was then put to the test by using it to edit base motions of several other subjects with successful results.

The measure for success was by judging the playback of the resulting motions and comparing them to originally captured motions of the subject whose base motion was edited as well as the subject whose data was used in the creation of the CPMC. Comparisons were also made between the plotted trajectories of such motions. MATLAB was used for the plotting, and a program written specially for this work in Microsoft Visual C++ was used for the playback of the motions.

The bulk of the computations involves only vector subtraction and vector addition, hence the simplicity goal is met. In addition to averaging samples whenever possible, space efficiency is achieved by storing only partial motion clips and by combining several partial motion clips into one.

1.3 Outline

CHAPTER 2 reviews previous work done in the field of animation and motion editing in particular. CHAPTER 3 provides a taxonomy by which to categorize the available motion editing techniques. CHAPTER 4 defines some terms used in this work, and describes the file format used for the motion samples and how they were generated from the motion captured data. CHAPTER 5 explains how CPMCs are created and how they are used, and CHAPTER 6 demonstrates the method explained in the previous chapter in an example by applying it to throwing. CHAPTER 7 discusses the results of using the CPMC for throwing in addition to the results of some smaller experiments. Finally, CHAPTER 8 concludes the work described and talks about some possible extensions and variations of the work.

The appendices contain MATLAB scripts for the estimation of missing DOFs of the non-throwing arm and the legs. A brief description of the Pose File Player can be found in APPENDIX C.

CHAPTER 2
PREVIOUS WORK

This chapter gives a review of related research, starting with the early methods for producing animations and ending with the latest methods for editing such animations. The use of blending in the computer game industry will be briefly talked about as well.

2.1 Articulated Figure Animation: generating motion clips

Articulated figure motion has been the focus of much research in computer animation for many years. Special attention has been paid to human locomotion: walking and running. "Synthesizing realistic human motion is a challenge: all viewers of an animation are also experienced observers of human locomotion." [MUL99] It is easy to detect erroneous movements simply on the basis that it does not look right. At the same time, it is difficult to isolate the factors causing it to be wrong [AMA96] [BRU93].

There are techniques for generating animations that focus on the physical correctness of a motion, others that focus purely on the appearance, and still others that focus on computational speed while sacrificing physical correctness to some extent. Which technique is to be used depends on the application. When the animation is part of a sports simulation, or for biomechanical analysis, physical correctness and accuracy are a major concern. When the animation is part of a movie, then its appearance is important. But when the animation is part of a virtual reality (VR) system then computational speed is a major concern. The same is true for interactive games. The faster one can compute the basic walking motion of a character the more time is left for 'cosmetics' – making it look good.

In the computer game industry an additional concern is to make the game appealing to the user. Game developers are constantly trying to make characters look and move more naturally. Some important factors are things that affect the appearance of the character, like the movement of their clothes or hair, and the facial expressions or emotions they are capable of showing. But even more basic body movements are still open for improvement. Motions often look jerky and transitions happen too suddenly creating visible discontinuities in the motion.

Therefore, sophisticated methods to blend motions and transition from one action to another are needed.

Methods used to generate motions also vary according to the amount of control they give the animator. Keyframing systems give the animator the most control at very low levels, whereas goal-directed methods give the animator high-level control. Other methods for generating animations fall in between those extrema.

2.1.1 Keyframing

Classical animation is typically done by keyframing. Highly skilled animators sketch the main frames of a desired sequence, which are called *keyframes*. Other animators then "fill in the gaps" between any two keyframes by drawing the *in-between frames*. Keyframing techniques have been computerized and are available commercially. Keyframing systems require only keyframes to be input, and they will then automatically produce the in-between frames using an interpolation technique. The aim of such systems is to simplify the animator's task and to speed up the process of generating a complete animation. Nevertheless, such systems still require a lot of work and skill on the part of the animator. They do, however, give the animator full control over the characters he/she creates.

2.1.2 Inverse Kinematics

When inverse kinematics (IK) is used, the user only needs to position the end-effectors, e.g. hands and feet, and the system will solve for the needed angles in order to reach the desired position. The problem with this approach is that usually there is more than one solution, and there is no way to select a specific one. Some of the solutions may be undesired, because they give unnatural poses. To overcome this problem, additional constraints usually need to be defined, like limiting the joint angles, or minimizing the angular change from frame to frame, etc. Such optimizations are usually solved numerically often involving iterative methods, which tend to be slow and sometimes unstable. An example where inverse kinematics has been used to generate walking is the work of Chung and Hahn who made a walking model that could adjust its steps to be able to walk on uneven terrain [CHU99].

2.1.3 Dynamics

In systems that employ dynamic constraint formulations, the main advantage is that they produce physically correct motions (assuming the given constraints are physically correct). But the drawback is that it is very difficult to specify the necessary constraints in order to achieve a specific motion, and computational complexity is high. Additionally, in animation, it is not always required nor desired to be physically correct.

Hodgins et al develop dynamic controllers to animate athletes focusing on physical correctness [HOD95]. Another method proposed by Hodgins and Pollard [HOD97] adapts an existing controller to a different character. For example, a controller for a man running can be changed to a controller for a woman running. A combination of scaling and iterative searching is used to produce the new controller. Genetic programming has also been successfully used to automatically generate dynamic controllers for some articulated figures [GRI95].

2.1.4 Goal-Directed methods

Goal-directed methods provide high-level control over a character. A goal can be something like throwing a ball into a basket or walking to the door. This method reduces the number of parameters the animator has to specify, but some flexibility and control over the details is lost.

2.1.5 Motion Capture

Another approach to animation, which is different from the previous approaches because it takes its input from real subjects, is motion capture. Actors wear markers or sensors that are tracked using special devices (optical, magnetic, etc.) and thus every movement the actor makes is captured. The data obtained by such a system is usually in the form of a sequence of positions, and sometimes orientations, of each marker in time that can be translated into positions of end-effectors, or orientations of joints. This kind of approach produces realistic motions, but not every motion needed in an animation can be acted out, for example, a character jumping off a cliff. The task of the animator might be reduced through the use of motion capture, but now actors, special equipment, and space are needed.

A capturing session usually consists of two parts: calibration and data acquisition. It is necessary to calibrate the cameras before using them; the way this is done depends on the system used. Then the markers on the actor need to be calibrated to match the animated figure. This is usually done by capturing the actor while standing in a predefined pose called a *zero-pose* or *rest-pose*. The computed joint angles from that data have to match the animated figure in its rest or zero-pose. If they do not match, an offset can be added to the marker readings to achieve a match. That offset remains fixed for this particular capturing session – assuming the markers are not moved. In other words, calibration is done once per session. After that, the desired motions can be captured. The offsets are always added in order to obtain the correct motion on the screen.

The calibration process can be very tedious. Often it requires exact external measurements of marker positions relative to the joints. This is specially the case in biomechanical applications where accuracy is very important. Other applications like games and movies do not require that degree of accuracy. More and more methods are emerging that try to simplify the calibration process while attaining acceptable accuracy.

Bodenheimer et al describe the basic motion capturing process using a magnetic capturing system, including sensor attachment and derivation and inferences of rotational degrees of freedom in real-time. A virtual skeleton is also constructed in real-time, which can be used for immediate feedback. The next step is an offline process involving a robust statistical estimation of the size of the skeleton and an inverse kinematic optimization to produce the desired joint angle trajectories [BOD97]. On a similar note, Molet et al describe an approach that allows real-time conversion of magnetic sensor measurements into human anatomical rotations. They describe in details how the sensor calibration takes place and how the joint rotations are computed [MOL96].

More recently, O'Brien et al proposed a technique to determine the joint parameters of an articulated figure using magnetic motion capture data [OBR00]. During calibration, instead of recording a single reference pose, the subject is asked to move all joints and to try to exert all possible motions. They are then able to determine limb lengths and joint locations without the need for external measurements. The parameters are computed using a linear least squares fit of a rotary joint model to the input data.

A recurring problem with motion capture is that once a motion capturing session is over it might be discovered that the acquired motion is not exactly as was desired. For example, the actor may not have returned to the exact starting posture in order to create a cyclic motion. Or the requirements may have changed. Because it is not feasible to recapture every clip that varies from what is desired, the need for editing existing motion captured data arises.

2.2 Motion Editing: reusing existing motion clips

The notion of motion editing became widely spread with the increased popularity of motion capturing. If a reaching motion to location XYZ should be changed to a reaching motion to location ABC and keyframing was used for the first, then one could simply change the keyframes and use the same technique to achieve the change. If a procedural approach was used to create the original motion, then the same approach could be used for the change. However, if motion capturing was used, there is much more involved in redoing a motion capture than simply rerunning a program. Hire the actor again, get access to the motion capture studio, do the calibration, then repeat the motion exactly the same way as before except for the location to reach to. In the end, the actor might still not be reaching to the correct location or other aspects of the motion might be incorrect now.

For these reasons, it would be better to edit the original motion to achieve the change. The question is then, how can a previously stored motion be changed? In our specific example, how can the reaching motion be changed so that the reach is to location ABC instead of XYZ? Here are a few possibilities:

- Signal Processing: express the motion data in the frequency domain and apply signals processing techniques. Adjust the gains of different frequency bands [BRU95].
- Motion Warping: edit the keyframe and position the hand at location ABC, use this as a constraint, then solve to obtain a smooth deformation [WIT95].
- Spacetime constraints: add a constraint for the hand to be at location ABC at the desired time by dragging the hand to the desired position in the desired frame and solve over the whole motion [COH92] [GLE97] [GLE98b] [LIU94] [WIT88].
- Motion Spaces: record several reach motions to various locations. Interpolate the samples to generate a reach motion to location ABC [GUO96] [ROS98] [WIL97a] [WIL97b].

Independent of the way in which motion data is obtained (i.e., keyframing, procedural techniques, motion capture, etc.) it can be represented in different formats. It could be a collection of curves depicting the change of the joint angles over time, or it could be a collection of poses, and so on. Different representations of the motion data lead to different techniques in editing and combining them. In general, researchers have tried to reduce the number of parameters that have to be handled while still allowing the user a reasonable amount of control over the resulting motion.

2.2.1 Frequency Domain

Bruderlin and Williams considered the curves to be sampled signals [BRU95]. They then applied operations known from signal processing to them. After processing the signals, by creating low-pass and band-pass pyramids, the signals are decomposed depending on the frequencies they contain. The user is then provided with an interface that looks like an equalizer. Changing the sliders the user can change the gains of the different frequency bands resulting new motions and thereby editing the original motion. This technique is based on the realization that low frequencies contain the general motion pattern, whereas high frequencies contain the details and subtleties. The strength of this approach is that it is interactive. The user sees the results of the changes right away. But the drawback is that it is not intuitive, it depends on trial and error to achieve a desired change.

Another approach that deals with the data in the frequency domain is the one proposed by Unuma et al [UNU95]. They captured periodic motions like walking or running. A Fourier series expansion of the joint angles was used as the functional model. Let $\Theta_m(t)$ be the rotational angles of each joint m over the time period t for some motion. The time parameter t is rescaled such that the period of $\Theta_m(t)$ is normalized to be 2π. The expression then looks like this:

$$\Theta_m(t) = A_m0 + \sum_{n\geq 1} A_mn \sin(nt + \phi_mn) \qquad \text{(Equation-1)}$$

If several motions are represented in such a way, the normalization has the effect of aligning the motions in time. A drawback of their representation is that only cyclic motions can be represented with it. On the other hand, their model provides intuitive parameters for controlling kinematic aspects of human locomotion. The parameters include step length, speed,

gait (length of a cycle), and jump height. The user can change them via sliders. The parameters are easily incorporated into the model. For example, the step length corresponds to the value of A_m in (Equation-1). Whereas the speed can be incorporated as the increment used for t in order to generate the animation, where t is the parameter in $\Theta_m(t)$. These parameters allow the user to easily generate variations of a basic periodic motion and thereby creating motions with different characteristics.

2.2.2 Displacement Mapping and Warping

Two other techniques proposed by Bruderlin and Williams are waveshaping and displacement mapping [BRU95]. In waveshaping the user can create a function using control points. That function is then used as a transformation on a selected signal. Typical examples are functions that limit the range of a joint's rotation. Again, this method is not intuitive.

In displacement mapping, an interpolating spline is fitted through the displacement values a user made to a curve, the fitted spline is the displacement map. The parameters for the interpolating spline can be adjusted to make the displacement map more or less smooth (see Figure-1 c and d). The displacement map is then added to the original curve so that the result will reflect the desired changes while maintaining the continuity of the curve (see Figure-1 e and f). This technique allows fine tuning motions. It provides a way to apply local changes while preserving the global appearance of the motion.

Witkin and Popovic hypothesize that high frequency details can survive small transformations [WIT95]. In their technique, the animator changes a few keyframes of the available motion. This is different from pure keyframing since in their technique those frames are used as constraints for a smooth deformation or warp that is to be applied to the motion. The idea is similar to displacement mapping. For example, a normal walking motion can be changed into one where the figure steps over an obstacle. In the same way, a reaching motion could be changed to reach to location ABC instead of XYZ.

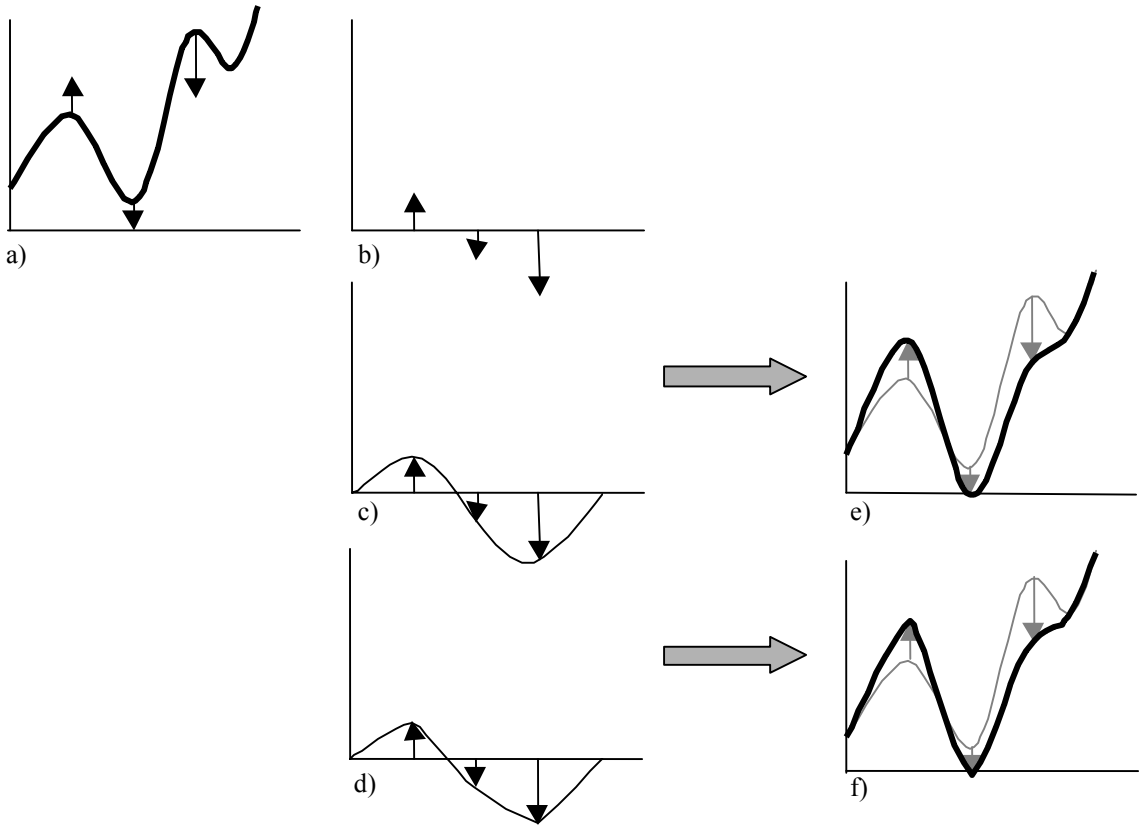

Figure-1 Displacement Mapping.
a) User indicates desired changes
b) Values of displacements
c) Interpolating spline fitted through displacement values to produce a displacement map
d) Changing spline parameters gives a different displacement map
e) Applying map from c) to original curve in a)
f) Applying map from d) to original curve in a)

2.2.3 Spacetime

A new formulation for constraint-based problems, called spacetime constraints, was introduced by Witkin and Kass [WIT88], extended by Cohen [COH92], and extended again by Liu et al [LIU94]. "By specifiying constraints on the motion such as 'jump from here to here, clearing a hurdle in between' and 'don't waste energy' (quotes taken from Reference [WIT88]), the method uses physical laws to produce the motion from first principles. To find the optimal motions, constraints over the entire motion can affect the behavior of a character at the beginning." [GLE98b]

Using a combination of displacement mapping and spacetime constraints, which minimize the change that is introduced to the motion, Gleicher allows editing a motion by dragging the figure to the desired position [GLE97]. In the example mentioned here, the hand could be dragged to location ABC to generate the correct reach. This kind of interface is very intuitive. Since spacetime constraints are used, this technique will make sure the constraints are not violated. This is a clear advantage over the previous methods where there is no guarantee that constraints will not be violated. On the other hand, applying spacetime constraints is usually slow. Nevertheless, Gleicher claims that his method is fast enough allowing the figure to be dragged interactively.

2.2.4 Blending and Correspondence

So far, the examples mentioned are only about editing a single motion. Another use of motion editing, one that involves more than one motion, is to transitions from one clip to another using blending. In addition to using blending for transitions it can be used to generate variations of a motion. Sometimes the blending will affect all DOFs and sometimes it is possible to blend motion data for some specific DOFs. The problem with techniques that support the second kind is that they tend to leave the other DOFs unchanged. In other words, they do not take into account what the affect is of the blended DOFs onto the remaining DOFs.

Whenever more than one motion are involved in the editing process, two important aspects have to be taken into consideration: parameter correspondence and time correspondence. *Parameter correspondence* means that models should have compatible structures and the parameters of each model should have similar effects. *Time correspondence* means that the phases of the motions should be aligned, or landmarks should be matched. In a reaching motion, a landmark could be the time of maximum elbow extension, while in walking heel strikes are the usual landmarks. In general, time correspondence is a more difficult problem to solve than parameter correspondence. The latter can be dealt with by simply restricting input motions to be in the same format or provide conversions from one format to another. But establishing time correspondence is more complicated. Researchers have come up with different ways to solve the problem but they all involve some sort of scaling in time. Uniform and non-uniform timewarps have been used, as well as generalizing sequences of events. Warps should be small, because

extreme warps degenerate the quality of the motion captured data and cause it to lose its foremost characteristic: the natural look.

Creating transitions usually involves blending the ending of the first clip and the starting of the second clip over a small time interval. Transitions can also be from one clip to itself thereby generating a cyclic motion. Rose et al were able to generate transitions between motions that minimize the torque needed for the transition while maintaining joint angle constraints. They also described how to make a walking or running motion cyclic, i.e. the first and last frame match exactly. In addition, they proposed a flexible functional expression language. Using this language the user can blend motions and concatenate them easily. It is also possible to single out some DOFs from one motion to be blended with another motion, e.g. salute and walk [ROS96]. However, this kind of blend will not affect the remaining DOFs.

In their method, they used kinematics to control the position of the figure's root. They applied spacetime constraints to non-supporting limbs, and inverse kinematics (IK) to supporting limbs. For motion cyclification, the user marks the approximate beginning and end of a cycle. The program then finds two points, within a small region near the marked locations, such that the difference between the position, velocity, and acceleration is minimized. An offset is then added to the entire time interval to make the endpoints match exactly. All resulting motions satisfy both kinematic and dynamic constraints. The problem with this technique is that it is not interactive.

Similarly, Witkin and Popvic used their motion warping techniques to do transitions between two motions [WIT95]. The two motions are partially overlapped. The overlapped portions are warped to bring them into reasonable alignment according to one or more selected points of correspondence. If the durations of the portions that need to be overlapped are different, they must also be timewarped. Now they can be blended using a weighted sum with a slow-in/slow-out weight function. It would be possible to blend in a partial motion since in their technique all the motion curves (for each joint) can be warped independently, but no specific example is given. Again, the blending would affect only the selected DOFs and not the rest of the body.

Bruderlin and Williams used a non-uniform timewarp to establish time correspondence [BRU95]. Then, every two corresponding parameters (in this case frequency bands) are blended. It is not necessary to blend all the signals making up the motion. For example, it is possible to blend a waving motion with a walking motion. In this case, the blending will be for the signals corresponding to the arm motion only, but the rest of the body remains unaffected.

2.2.5 Motion Spaces

There are several methods that create motion spaces from collections of similar motions and then interpolate between available samples to produce a target motion [GUO96] [WIL97b] [ROS98]. Alignment of the motions is critical in these approaches.

Guo and Robergé solved the problem of time correspondence for human locomotion in a simple way. They proposed a general event sequence for walking and running [GUO96]. Each locomotion cycle consists of the following phases:

- mid-transfer (heel strike in walking, toe-off in running)
- double or non support
- end-transfer (toe-off in walking, heel strike in running)
- single support
- mid-transfer (heel strike in walking, toe-off in running)

Guo and Robergé's method defines a range of movements as a delimited 2D or 3D space using a set of reference frames each called a frame space. The reference frames can be from motion captured data or from previously keyframed motions. Each frame space constitutes one complete cycle of locomotion. It has three dimensions: L) step length, varying from short to long, H) step height, varying from low (for walking) to high (for running), and C) cycle, varying over the phases of a complete locomotion cycle starting and ending at mid-transfer (see Figure-2 a). Variations are then generated by interpolation. That is, to generate a whole sequence several frame spaces are put together and some points (at most one per space frame) are selected. Then an interpolating curve is fitted through these points defining the transitions that happen from one frame to the other (see Figure-2 b). Since all the stored motions have the generalized sequence, they can be interpolated directly.

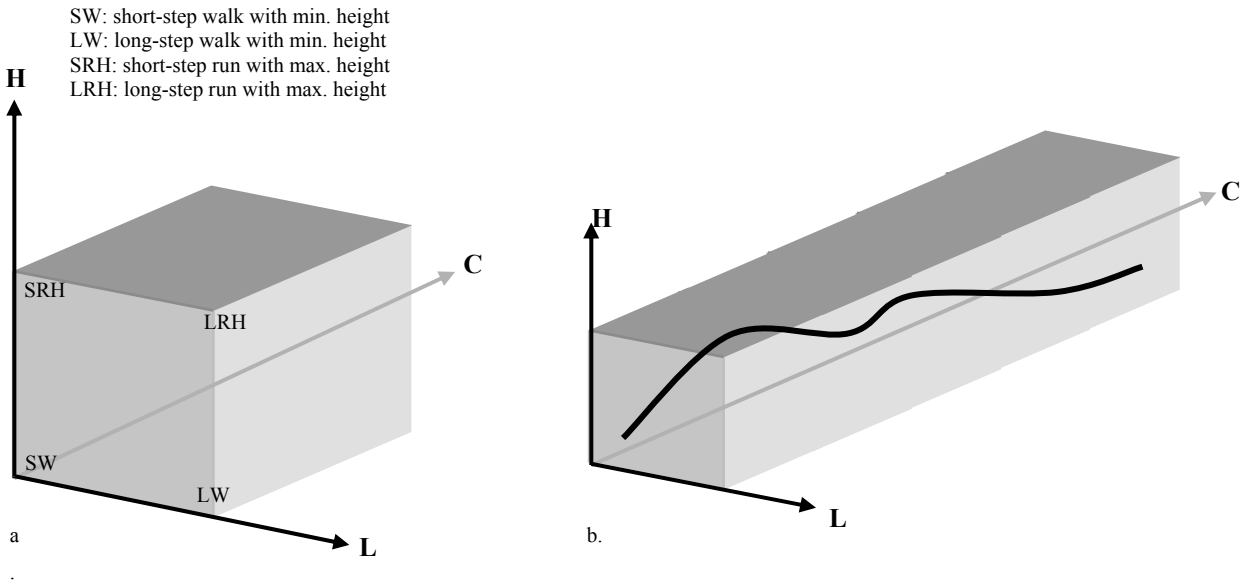

Figure-2 Parametric Frame Space Interpolation.
a) A frame space for the human locomotion cycle
b) A position curve defined in frame space

Wiley and Hahn stored motions as a sequence of poses. They collected similar motions into small data sets [WIL97b]. For example, reaching motions to different points on a plane were stored. From the discrete samples obtained by motion capture a motion space is created. In other words, the sets were parameterized such that each motion corresponds to a particular point in a multidimensional motion space of possible motions. The motion space is then resampled such that the data points lie on a regular grid. This space is now continuous since it is possible to generate any motion corresponding to any target point in the space by interpolating the poses from the neighboring grid points. Limited extrapolation is also possible with this technique.

Wiley and Hahn also used interpolation to generate variations of motions. In this sense, it is similar to Guo and Robergé's approach, but they dealt with time correspondence in a different way. They called it ‘time alignment’ which is just a uniform timewarp. When blending two motions with a different number of poses, the motion with the fewer poses is resampled so that it contains the same number of poses as the other motion. Now the two motions can be blended by interpolating each corresponding pose. The resulting blended motion sequence also needs to be

resampled. This is done such that the number of final poses is equal to the interpolation of the original number of poses of the two sequences.

Rose et al also created multidimensional parameterized motion spaces. They called the spaces verbs parameterized by adverbs [ROS98]. A verb is constructed from a set of example motions that vary only slightly from each other. Each example motion is characterized by an adverb; this places the example motion somewhere in the adverb space. The number of adverbs gives the dimensionality of the motion space. In addition, a set of keytimes needs to be specified for each example motion. Keytimes correspond to instances when important events occur, such as heel strike. Having these keytimes, helps to establish time correspondence by using piecewise linear time scaling so that all example motions will have their time reparameterized to some generic time. The problem of parameter correspondence is solved by restricting the example motions to be similar in structure as well as in their use of joint angles. From this motion space (verb) various motions can be obtained by multivariable interpolation given the desired parameters (adverb). Transitions from one verb to another are achieved by fading one in while fading the other out and merging the two using linear interpolation.

A main problem with motion space techniques is their sizes. To allow better and wider range of interpolation, more example motions have to be collected thereby increasing the amount of storage space needed. This problem is reduced in the method presented here by averaging common data whenever possible.

2.2.6 Differencing and Emotion

Other researchers have proposed differencing methods to edit motions. Given two motion sequences that are equal in terms of the action performed but different in terms of the emotion in which the action is performed, it is possible to extract the emotional content by finding the difference of the two motions. This resulting difference can then be applied to other neutral (emotionless) motions where a different action is performed and the result would be an emotional performance of this action. Here again the used motions need to be aligned.

Aspects other than emotional content may be extracted. All that needs to be done is to take the difference of two motions that vary only in the aspect desired to be extracted. Several

methods have been described that allow to add emotional content to an animation or to give it specific characteristics. Whether it be by using a differencing method, adding noise, or changing some slider that controls different aspects of the motion.

In the method described by Unuma et al, the control of the emotional part is done by the superposition of a Fourier characteristic function onto the original Fourier function [UNU95]. Where the Fourier characteristic function is no more than the difference between two similar motions that differ only in one specific aspect. Since the time parameter of the Fourier series expansion used to represent the motions is normalized to be 2π, the phases of the motions are automatically aligned. For example, if the function describing a brisk walk is subtracted from that describing a normal walk then the result is a characteristic function that describes briskness. This function could be added to a function describing a running motion to obtain a brisk run. Again, briskness is a parameter that can be changed to obtain different levels of briskness. Similarly a function for sadness can be extracted and then applied to existing functions. The difficulty mainly lies in finding appropriate and meaningful parameters.

Similarly, Amaya et al computed emotional transforms using signal processing techniques [AMA96]. The emotional transform is obtained from the difference between a neutral and an emotional motion. They collected data using motion capture. After analyzing the data, they identified two components that vary noticeably over the various emotions: speed (timing) and spatial amplitude (range). The speed transform is implemented as a non-linear timewarp, and the spatial amplification transform is based on signal amplifying methods. These emotional transforms can then be applied to any motion to add to it that specific emotion that is encoded within that transform.

Perlin generated a system in which a puppet responds in real-time to commands from the user. The commands are high-level commands such as “walk”, “dance”, “turn around”, etc. The interface for that is in the form of buttons. As the user pushes the buttons the puppet will move and transitions will occur smoothly. He programmed individual motions for each figure (or puppet) beforehand, these are called actions. Actions can then be assigned weights. And transitions are then obtained by increasing the weight of one action while at the same time decreasing the weight of another. Perlin also allowed overlaying of actions, where a shrugging

with the shoulders action can overlay a standing action, for example. However, no DOFs other than the shoulder are affected by such an overlay. To allow emotional variation, he added noise to the preprogrammed actions. A figure that is standing and not doing anything looks lifeless and stiff. However, if noise is added to the standing action the figure comes alive. If the amount of the noise is increased, the figure can be made to appear to be nervous [PER95].

The signal processing methods proposed by Bruderlin and Williams [BRU95] also allow adding emotion and creating personalized motions including cartoon-like exaggerations. For example, increasing the high frequency band makes the figure look nervous.

2.2.7 Retargetting

Another variation that calls for motion editing is applying a motion to a different character. Gleicher proposed a technique called retargetting that also uses spacetime constraints [GLE98a]. The purpose of it is to apply a motion to a different sized character than the original one. This technique even allows changing a character's size continuously over time producing a morph.

As mentioned in section 2.1, Hodgins and Pollard allowed a controller to be changed in order to accommodate a different character [HOD97]. This can be considered another way of retargetting but it is not motion editing, instead it is controller editing.

Lee and Shin developed an approach that combines a hierarchical curve fitting technique with a hybrid inverse kinematics solver [LEE99]. Hybrid in the sense that it is partially numerical and partially analytical. Using this approach it is possible to generate smooth transitions between motions, to apply a motion to a differently sized character, and even to continuously morph a character while producing a smooth animation. Their method can be viewed as a variation of displacement mapping and spacetime constraints.

Popovic and Witkin transform motions while taking dynamics into consideration [POP99]. Their method allows a motion to be applied to a drastically different character. Gleicher had taken out the dynamics part from the spacetime formulation in order to achieve

faster performance. Now Popovic and Witkin are keeping the dynamics in, but are reducing the complexity of the model.

2.2.8 Synthesizing and Texturing using Collections of Motion Captured Data

In the last couple of years, several methods emerged that deal with the generation of long, continuous motion sequences from a pool of pre-stored motions as well as adding style or texture to existing motions. The pre-stored motions may be short clips or long sequences that are organized in a special way such that transition from one clip to another are built in. Applications for such methods include controlling avatars in virtual environments and computer games, and assisting in animation generation by adding texture or style.

Brand and Hertzmann introduced 'Style Machines', which is a statistical model that can generate new motions with different styles [BRA00]. They used a hidden Markov model (HMM) to which they added a multidimensional style variable resulting a *stylistic* HMM (SHMM). The SHMM is used to analyze motion captured clips thereby extracting a state-space model and separating style from content. The resulting state-space model assigns probabilities to the possible transitions from state to state. One possible way to drive such a style machine is to use video input. A novice person can dance in front of a video camera from which a HMM state sequence can be extracted. Then the style can be changed to that of a professional dancer. An advantage of their method is that it is fully automated, no data segmentation, annotation, or alignment is necessary.

With a similar goal in mind, Pullen and Bregler presented a method for allowing a keyframed partial motion to be edited by adding detail or 'texture' to the DOFs included in the partial motion as well as synthesizing the undefined DOFs from motion captured data [PUL02]. The key to their method is frequency analysis. The motion captured data is decomposed into a Laplacian Pyramid, this allows the separation of details (contained in the higher frequency bands) from the basic motion (contained in the lower frequency bands). The data that identifies the basic motion is segmented so that it can be used to match it to the keyframed input motion that is to be edited. Once a match is found, texture can be added to the keyframed DOFs by using the data in the higher frequency bands, the DOFs that are not defined can be synthesized by simply copying the trajectories of those DOFs from the motion captured data that has been

matched. Which DOFs will be textured or synthesized is under the exclusive control of the user. The amount of change can also be controlled by selecting which frequency bands should be used. Note that there is no interaction between the changes applied to any of the DOFs, which is different from the method that is presented in this dissertation.

Kovar et al presented 'Motion Graphs' which is a directed graph with edges corresponding to motion clips and nodes serving as choice points to transition from one clip to the next [KOV02a]. Such graphs are generated by analyzing motion captured data and identifying segments that are similar enough such that simple blending can produce a reasonable transition. From the analysis the graph is constructed, traversing the graph then allows the synthesis of new motion sequences. Constraints may be violated during transitions as a result of the blending, but the motions are annotated with such constraint positions so that they can be enforced during a post-processing step.

Also using graphs, Lee et al allow avatars to be driven interactively by motion captured data [LEE02]. Their representation consists of two layers: the higher layer is a statistical model and the lower layer is a Markov process. They demonstrated their method by generating new motions using three types of inputs: a list of choices, sketching paths, and vision (i.e. video input). Again, constraints that are violated during transitions are fixed in post-processing using the hierarchical motion fitting algorithm by Lee and Shin [LEE99].

2.3 Games

The market for computer games is very competitive. Therefore, it is essential to create content that grabs and holds the attention of game players. With the advances in 3D rendering technology and the availability of excellent 3D graphics acceleration hardware players find more games with richer graphics and richer environments. To further enhance the gaming experience game developers are trying to improve animation and interactivity [MOT]. For that, they need good authoring tools and good methods to edit motions, blend or merge them, and transition between them.

Currently, a lot of the content of games is simply a playback of prestored linear sequences of animation. In many games the clips are stored in such a way that they start and end

in a specific configuration so that they can be played back one after the other without any need for blending. In others, smooth transitions are achieved by interpolating between the current joint orientations and the orientations in the first frame of the next animation clip. When quaternions are used as the internal representation of the joint orientations interpolation is easier since one does not have to worry about singularities as is the case when an Euler angle representation is used. It also allows interpolating between two poses that are very different yet producing good results. On the other hand, Euler angles are more intuitive to use and have only 3 parameters (as opposed to 4 in quaternions) which makes them more space efficient. For example, in the PC version of Indiana Jones and the Infernal Machine™ by Lucasarts Entertainment, Euler angles are used. Therefore, no interpolation between dissimilar motions happens, e.g. when transitioning from crawling to walking. The Nintendo-64 version of the same game uses quaternions, making it possible to interpolate between all actions.

Motivate is a commercial authoring tool created by the Motion Factory™, Inc. Its main goal is to facilitate the creation of 3D content that allows rich interaction of characters and good story telling using intelligent digital actors. Koga et al describe The Intelligent Digital Actor™ architecture [KOG99]. It combines real-time motion synthesis and dynamic event-based behavior programming. The first part allows the characters motion to be generated on the fly in response to a dynamically changing environment. The second part provides the game's logic through a hierarchical finite state machine that is specifically designed to model the behavior of a complex real-time system.

Using the skill editor of Motivate the animator can describe the basic skills of each character. This is equivalent to creating a motion clip library. For example, to define a 'pick up' skill the animator defines two keyframes: (1) positioning the hand at the handle just before grabbing, and (2) the hand's configuration after grasping the handle. This skill can be used with any object that has an appropriate handle. IK and collision detection are used to generate the final motion. If the object is not within reach the character can walk closer before attempting to grab the object; assuming that the 'walk' skill is already defined. Reversing the sequence of the keyframes allows the character to release an object. In Motivate, it is also possible to blend activities like walking and waving but without any interaction between the upper and lower body.

In the game SWAT™ 3: Close Quarters Battle by Sierra Studios, soldiers have two controllers: one for locomotion (lower body), and one for aiming (upper body). For example, a reload animation is strictly an upper body motion, allowing the characters to reload while walking. Again, there is no relationship between the two controllers, i.e. they do not affect each other. SWAT3 uses quaternions allowing smooth transitions through interpolation [SIE].

RAD Game Tools, Inc. released a product called Granny 3D Animation™ in December 1999 [RAD], another commercial authoring tool. It is a complete system for adding 3D geometry and animations into a game quickly and easily. The main advantage of Granny is its character animation system, through which it is possible to playback and blend multiple animations. For example, one can make a character salute and run at the same time, transition from a walk to a run, etc. Blending is simply done by assigning weights to the animations. The weights can be time varying. The weights and their time varying properties are assigned by the user, they are not assigned automatically. Granny also has an inverse kinematics solver. It can also adapt animations on the fly for characters with different proportions so long as their hierarchy is similar.

These are only a few examples form the game industry, still it is obvious that games use partial motion clips extensively, and there are no good methods to blend them properly. Mostly, the partial motion that is blended in just masks out the original motion of those DOFs and replaces it with the new motion. It will not affect the remaining DOFs.

2.4 Summary

In this chapter, a variety of approaches to accomplish motion reuse have been described. Interactivity, ease of use and control, realism, space requirements, and applicability are several measures for judging editing methods.

Methods that employ dynamics, like the spacetime constraints system of Rose et al [ROS96], are not interactive and hard to control, but are able to produce physically correct motions. Gleicher reduced the spacetime formulation by taking out the dynamics part. The result was an interactive system. He justified this by the assumption that the editing process will create only minor changes to the original (correct) motion thereby maintaining a lot of the physical

characteristics. On the other hand, Popovic and Witkin kept the dynamics in, but simplified the model.

Bruderlin and Williams' signal processing method [BRU95] is interactive, but it is not easily seen what the effect will be from changing the sliders, control therefore depends on trial and error. On the other hand, the methods that provide the user with meaningful parameters are easier to control. This category includes Perlin's system [PER95] and the system of Unuma et al [UNU95].

Some methods have high space requirements, like the methods that use graphs and state-machines [BRA00] [KOV02a] [LEE02]. Therefore, they depend on clustering or pruning of the graphs to keep the size of the graphs under control. They also vary in the methods used for searching the graphs for appropriate clips to make up the new motion as well as in the methods used for analysis of the data to identify transition points and assign probabilities and thereby creating the graph.

Methods that create motion spaces from a collection of sample motions and then use interpolation to generate a specific motion also have high space requirements [GUO96] [ROS98] [WIL97b]. Motion spaces are good because interpolation is relatively easy to do and fast. The difficult part is to put the space together and to parameterize and annotate it correctly. This includes the problem of aligning the motions. An important factor in these methods is the number of samples obtained for the data sets. The more samples obtained the more accurate the interpolations can be, because the generated motion space will be dense. However, the drawback is the additional space requirement. To reduce it, one could combine similar motions by storing the common parts once and then try to compute the other parts procedurally. This is what is being done in the research represented in this dissertation.

For example, if one had a motion space for throwing which included a parameter defining the starting posture e.g. sitting, standing, or walking, a lot of samples would be needed to be able to generate good interpolated motions. If one could assume that the motion of the throwing arm is always the same and the motion of the remaining joints can be induced in the neutral motion (i.e. without throwing) via a relationship to the throwing arm motion, then they would not have

to be stored separately. Just the arm motion needs to be stored once in addition to whatever information is needed to compute the other DOFs depending on posture: sitting, standing, and walking. For that to be possible, one would need to find a relationship between the upper and lower body during throwing. Research done in biomechanics, kinesiology, or other fields that deal with human body movements like sport and exercise sciences may be helpful in understanding the correlations between joint movements.

In the biomechanics community there has been a lot of research devoted to the study of throwing, like pitching in baseball, passing in football, javelin, windmill pitching, and so forth [BAR98] [FLE96]. The focus in those studies is usually on what the kinematic and kinetic parameters are in order to achieve the furthest throw, or the most accurate throw, or the fastest throw, and how to avoid injuries. In addition, the timing and sequencing of actions is studied [ELL88]. There even have been studies that compare such parameters between groups of people with different levels of development [FLE99]. Apparently, there are no studies that specifically address the correlation of the upper and lower body during throwing while in different postures. This is probably because in the field of biomechanics it is not important to find such a relationship. Nevertheless, it could be useful in the field of computer animation, especially in games.

Looking at the applications of the proposed methods, there are only a few methods that consider applying a motion to differently sized characters [GLE98a] [LEE99] [POP99]. Brand and Hertzmann, claim that their Style Machines can be used for retargetting by considering the size of the character as another style variable [BRA00]. Applying a motion to differently sized characters is an important problem to consider. Especially when trying to reuse motion captured data since it is very likely that there is only one actor or actress who will perform the desired motions that will later be applied to many different characters.

Additionally, there are only a couple of methods that dealt with blending individual degrees of freedom of one motion with another and those were by Bruderlin and Williams [BRU95] and Rose et al [ROS96]. However, they did not take into consideration how the DOFs chosen for blending affect the rest of the DOFs. Blending separate DOFs is important in games that store small clips for each part of the body separately, like in martial arts games or sport

games, where each kind of kick or punch is stored by itself. Blending techniques can be improved to produce smoother final motions and to reduce jerkiness if more knowledge about the nature of the movement is incorporated or – as in the method presented here – the correlation between joint movements is taken into consideration.

CHAPTER 3
A TAXONOMY FOR MOTION EDITING

This chapter presents a taxonomy by which one can categorize the existing motion editing techniques described in the previous chapter. It will place the method presented in this dissertation in the correct position relative to the existing editing methods.

A distinction is made between the words 'action' and 'motion' in this dissertation. An *action* is something that is performed, whereas a *motion* is a formal representation of an action. In other words, a motion defines a character's configuration as it changes over time. It can be represented as a function, a sequence of keyframes, a collection of joint curves, or any other representation. This distinction is made because one may not think of 'sitting' as a motion since it does not necessarily involve any movement, yet it is something one does hence it is an action, and once it is converted into a format suitable for animating a character, for instance, it is called a motion.

Furthermore, motions can be defined completely or partially. Completely, meaning that all joint curves are stored, or the keyframes describe the whole character's configuration. Partially, meaning that only the prominent joint curves are stored or the keyframes only describe part of the character's configuration. Similarly, actions may involve the whole body, like walking, standing, sitting, swimming, running, jumping, dancing, or they may involve part of the body, like waving, throwing, punching, scratching the head, pointing, reaching, etc. An important point to note is that a partial action is not necessarily stored as a partial motion; it can be stored as a complete motion.

3.1 Main Classification

Motion editing techniques can be categorized using the following taxonomy, which divides the techniques along two dimensions. The first dimension is the number of motion files used. This dimension can have two values: using a single file only, or using multiple files (the exact number is unimportant). The second dimension specifies the purpose. It can be roughly divided into: transitions, emotion, changing styles or locomotion characteristics, retargetting,

Table-1 Motion Editing Taxonomy – Part 1

Mixing Motions	*Not applicable*	See 0 below
Changing end-effector position	Displacement mapping and waveshaping – [BRU95] e.g. knocking at a different height Warping – [WIT95] e.g. hitting a tennis ball at a different height Spacetime – [GLE97]	Motion Space – [WIL97a] [WIL97b] e.g. reaching to different locations Motion Space – [ROS98] e.g. reaching to different heights
Retarget-ting	Spacetime – [GLE98a] Hierarchical curve fitting and IK- [LEE99] Physically based transforms – [POP99]	Motion spaces could be used although there is no explicit example
Style or Loco-motion Character-istics	Signal Processing – [BRU95] e.g. can exaggerate a walk Fourier Representation – [UNU95] e.g. change slider to control step length Physically based transforms – [POP99] e.g. make a character limp	Differencing – [UNU95] e.g. extract briskness of walk, add to run Motion Space – [GUO96] Motion Space – [WIL97a] [WIL97b] e.g. walking at different slopes Motion Space – [ROS98] e.g. walking and turning left or right Style Machines – [BRA00] e.g. change novice to professional dancing Texturing – [PUL02] e.g. make walk look more life-like.
Emotion	Add noise – [PER95] e.g. to simulate nervousness Signal Processing – [BRU95] e.g. increase high frequency bands to simulate nervousness	Differencing – [AMA96] e.g. extract angriness from a drinking action and applying it to a kicking action
Transition	Overlap ends, find correspondence point with min. difference in position, velocity, and acceleration, distribute error over interval, Fit a least squares cyclic B-spline approximation – [ROS96] e.g. cyclification of walk	Spacetime – [ROS96] e.g. to concatenate several motions Motion Space – [GUO96] e.g. transition from walk to run Fourier Representation – [UNU95] e.g. interpolate to transition from walk to run Warping – [WIT95] Style Machines – [BRA00] Motion Graphs – [KOV02a] Avatar Control – [LEE02]
	Single file	**Multiple files**

Table-2 Motion Editing Taxonomy – Part 2: details of mixing motions

Partial with complete	Signal Processing –[BRU95] e.g. blend walk with drumming Spacetime and masking – [ROS96] e.g. blend a walk with a salute Layering and masking – [PER95] e.g. shrugging the shoulders and standing	
Complete with complete	*Not applicable*, since complete motions define all joints	Signal Processing – [BRU95] e.g. blend two walks
	Affect only some DOFs	**Affect all DOFs**

changing end-effector position without changing the original action, and mixing motions: two or more complete motions, or complete and partial motions. Where retargetting is as defined by Gleicher, i.e. applying a motion to a differently sized character [GLE98a]. And mixing motions can be via blending or merging, overlaying, adding, etc. Table-1 shows where the existing motion editing techniques fit in. It gives the name or category of the technique, its reference, and a published example if possible.

3.2 Classification of 'Mixing Motions'

'Mixing motions' can be further divided depending on what kind of motions are being mixed together, and when partial motions are involved whether the combination has an affect on the whole configuration of the character or only on the included part, see Table-2. It is apparent from Table-2 that there are no methods that mix partial motions with complete motions while taking into account the effect this combination may have on the remaining DOFs. This is where the motion editing method presented in this dissertation fits in.

When mixing motions, one motion has to act as the base to which the other motions are added. Usually the complete motion is the base and the partial motion is the addition. Mixing a partial motion with another partial motion is not in the table because this case can be reduced to any of the other cases as follows:

1. If both partial motions define the same parts it is reduced to the *complete with complete* case. All DOFs in these partial motions will be affected and there are no other DOFs that could be affected.
2. If one partial motion defines a subset of DOFs of the other. It is reduced to the *partial with complete* case, where the partial motion with the larger number of DOFs defined will be considered the complete one and the one with less DOFs may affect some/all DOFs of the other.
3. If the DOFs defined are not subsets of each other, one still has to be chosen as the base and the other one may affect other DOFs not defined in it but defined in the base. DOFs defined in the addition but not in the base are simply ignored. Thus this case is also reduced to the *partial with complete* case.

In the work of Pullen and Bregler a partial motion can be 'textured' from a complete motion [PUL02]. Also undefined DOFs may be synthesized. As described in section 2.2.8 the first step in their method is to find a match between the partial and the complete motion. The partial motion may be the motion of the legs while walking resulting from keyframing. The complete motion may be a result of motion capturing a person walking. A Laplacian pyramid is created dividing the motion into frequency bands. Lower frequency bands can be used to find a match between two motions using selected DOFs of the partial motion. The higher bands are then used to 'texture' the DOFs of the partial motion because they contain the details of the motion. Additionally, DOFs not included in the partial motion, e.g. the arms, may be synthesized by using all frequency bands of those DOFs from the complete motion. This is different from the work presented here, in that the DOFs that will be affected are selected by the user. And the extent to which they are affected is also controlled by the user.

One may still argue that the work of Pullen and Bregler fits into the category of mixing partial with complete motions while affecting all DOFs. But looking closely at their work one realizes that it does not fall into that category. That is because in the work of Pullen and Bregler there is no interaction between the DOFs that are edited. Furthermore, DOFs that are affected in the base existed in the addition, similarly DOFs that are synthesized in the base existed in the addition, but what will be done here is that DOFs that are affected in the base do not exist in the

addition. And that is what is intended in the category of mixing partial with complete motions while affecting all DOFs, that is, affecting DOFs in the base but not in the addition.

From this presentation, it is safe to say that there are no existing methods for editing motions that allow mixing of some DOFs while affecting other DOFs automatically. And that is precisely what will be demonstrated in the next chapters via the creation and usage of Combined Partial Motion Clips.

CHAPTER 4
DATA FORMAT AND TERMINOLOGY

The motion samples used in this research are all results of motion capture. Optical motion capture equipment was used to acquire the motions. The spatial 3D data of the markers was converted to joint orientations and positions of the root so that it could be used to drive an animated character. Therefore, before getting to the heart of the work presented here, a few definitions have to be stated and the conversion routine has to be explained.

4.1 Terminology

The raw data resulting from motion capture is converted into root position and joint orientations. Joint orientations are expressed as XYZ-Euler angles and stored as vectors, $\mathbf{v}_i = [x\ y\ z]^T$ for i = 1 .. **NumberOfJoints**. Therefore, all joints are stored as if they had 3 DOFs even though some have less. The knee joint for example, has only one degree of freedom, a rotation about the x-axis, hence the y and z components of its vector are always zero. This representation was chosen so that all joints could be handled uniformly including the root's position, which is also stored as a vector, $\mathbf{v_0} = [x\ y\ z]^T$. The position of the root is expressed as a relative displacement in the root's local coordinate system. Furthermore, no joint limits are imposed on the joints.

4.1.1 Articulated Figure

An articulated figure is a hierarchical representation that defines a figure as a sequence of joints and segments (see Figure-3). Segments can be of variable length, but here fixed length is assumed. The DOFs of the figure depend on the number of joints. In general, the root has 6 DOFs 3 for the position and 3 for the orientation, and all the other joints have up to 3 DOFs for their orientation.

The articulated figure shown in Figure-3 is in its **zero-pose**, where all joint rotations are zero. A right-handed system is used as the local coordinate system of each joint. The y-axis usually runs along the segment except for the collar bones. When the figure is in its zero pose, the x-axis points to the left, the y-axis points upward, and the z-axis points to the front. This is true for all joints.

In this research, the neck, head, collars, and toes were not used. Therefore, the total number of joints used was **NumberOfJoints** =14. Since the root possesses 6 DOFs (represented by $\mathbf{v_0}$ and $\mathbf{v_1}$) , the back, shoulders, and hips possess 3 DOFs, all other joints only 1 DOF, the total number of DOFs was 1*6+5*3+8*1= 29.

4.1.2 Pose

A pose defines the values of all DOFs of the articulated figure at an instance in time:
$\mathbf{P} = [\mathbf{v_0} \ldots. \mathbf{v_k}]^T$, where k = **NumberOfJoints**.

4.1.3 Partial Pose

A partial pose defines the values of some DOFs of the articulated figure at an instance in time, e.g. the pose of the trunk and arm during a waving motion:
$\mathbf{PP} = [\{\mathbf{v_i} : i \in \{0,1,\ldots, \mathbf{NumberOfJoints}\}\}]^T$, if $\mathbf{v_0}$ (position) is included then $\mathbf{v_1}$ (root) has to be included too.

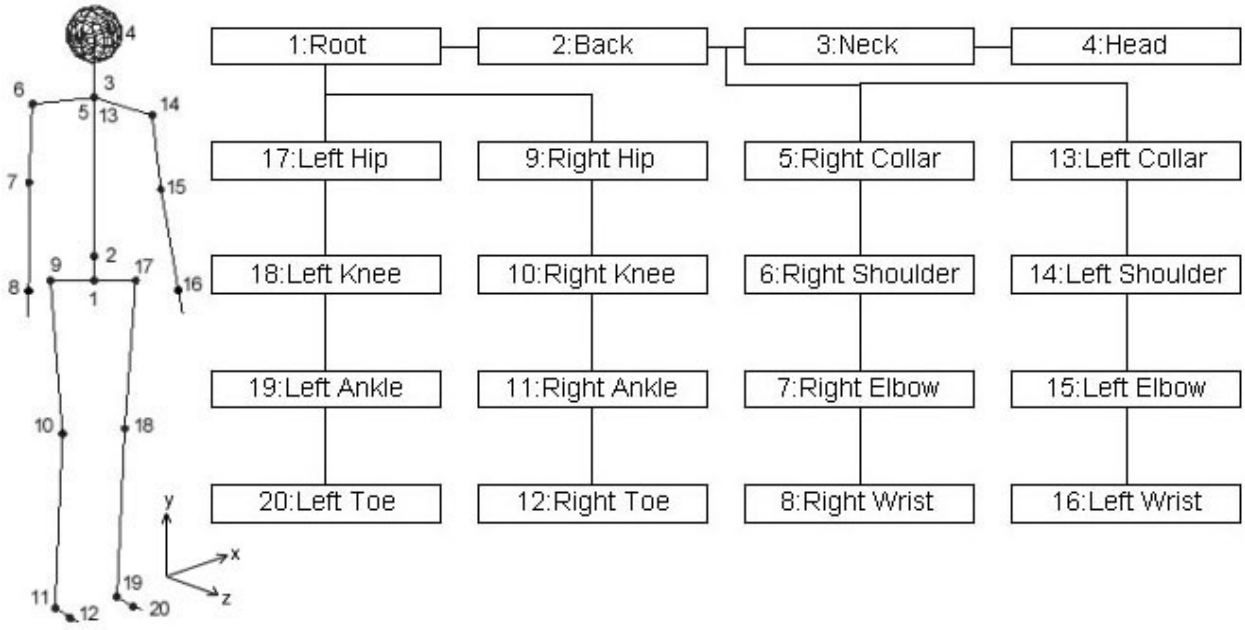

Figure-3 The joints of an articulated figure and its hierarchy.

4.1.4 Trajectory

A trajectory $q_i(t)$ is a function that defines the values of $\mathbf{v_i}$ as it changes over time. The maximum number of trajectories used in this research is **NumberOfJoints** +1 = 15, that is, the trajectory for position, plus the trajectories of all joints.

4.1.5 Motion

A motion is a collection of poses taken at contiguous instances in time:

$\mathbf{M}=[\mathbf{P_1},\mathbf{P_2},\ldots\mathbf{P_n}]$, where n = **NumberOfFrames**.

Or the concatenation of all trajectories:

$\mathbf{m}(t) = \{\mathbf{q_i}(t)$: i = 0,1,…, **NumberOfJoints** $\}$, where t = 0.. **NumberOfFrames**-1.

For a specific t, **m**(t) corresponds to a pose.

4.1.6 Partial Motion

A partial motion is a collection of partial poses taken at contiguous instances in time:

$\mathbf{PM}=[\mathbf{PP_1},\mathbf{PP_2},\ldots\mathbf{PP_n}]$, where n = **NumberOfFrames**.

Or the concatenation of some joint trajectories:

$\mathbf{pm}(t) = \{\mathbf{q_i}(t)$: i $\in \{0,1,\ldots,$ **NumberOfJoints** $\}\}$, where t = 0.. **NumberOfFrames** -1.

For a specific t, **pm**(t) corresponds to a partial pose.

4.1.7 Base Motion

A base motion is a motion, partial or complete, to which another partial motion is added.

4.2 Data Format

The motion captured data is stored after conversion in ASCII files, called **Pose Files**. Pose files have the following format:

Number of frames on the first line: **NumberOfFrames**

Rest and calibration angles of all joints on the second line: $\mathbf{r_1}\ \mathbf{c_1}\ \mathbf{r_2}\ \mathbf{c_2}\ \ldots.\ \mathbf{r_k}\ \mathbf{c_k}$

For each frame, one frame per line, starting at the third line: **frameNumber** $\mathbf{v_0}\ \ldots\ \mathbf{v_k}$

Where calibration ($\mathbf{c_i}$) and rest ($\mathbf{r_i}$) angles are also vectors, all vectors are displayed in this format: (x,y,z), and k = **NumberOfJoints**. Converting the data to binary format can help reduce

the space requirements. But since the aim was to be able to retrieve the data easily as well as to be able to read the data using a regular text editor, the ASCII format was chosen.

4.3 Data Conversion

The motion samples used in this research were collected at the biomechanics laboratory at the Warren Grant Magnuson Clinical Center at the National Institutes of Health (NIH). An optical motion capture system by Vicon was used. It outputs data in C3D format which is a binary file consisting of a header section, followed by a parameter section, and finally a data section. The data section contains the 3D point coordinate data written frame-sequentially, that is, the 3D positions of all markers are written frame after frame. No orientations are provided. Code for reading C3D files was graciously provided by NIH.

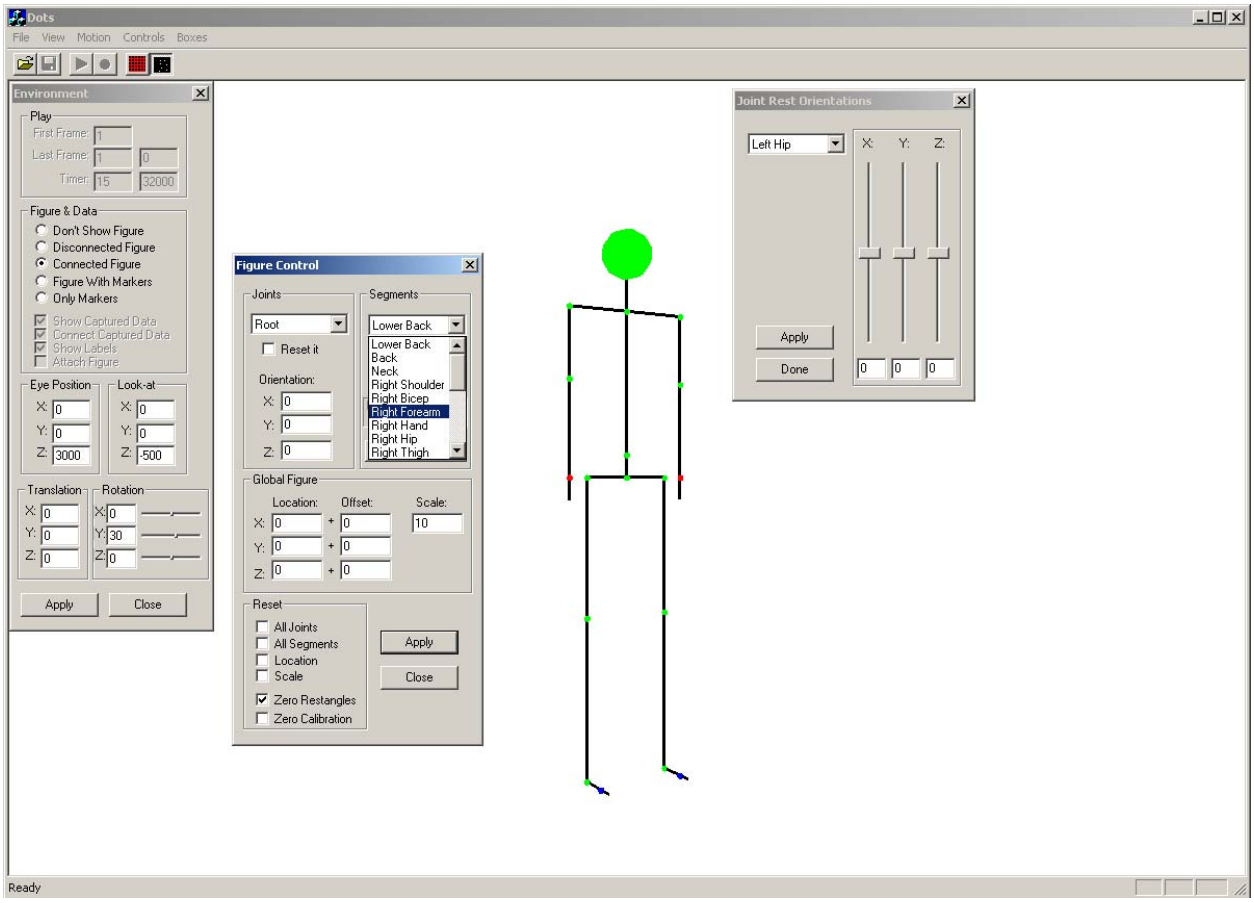

Figure-4 Adjusting Figure's segment lengths and rest angles.

The positional data had to be converted to joint orientations and root position to make it suitable for driving the animation of an articulated figure such as the one shown in Figure-3 above. A program for that purpose was written using Microsoft Visual C++, the Pose File Generator. The program starts by asking the user to adjust the figures segment lengths, and rest angles to match the subject. See Figure-4 where the articulated figure's rest angles are still at zero. When rest angles are adjusted the articulated figure should look similar to the one in Figure-5.

To obtain an orientation for a 3 DOFs joint one has to have at least 3 markers at the corresponding segment in order to be able to create a local coordinate system and then derive the orientation. This was the case for the root, which provides the overall orientation of the figure, where 2 markers were placed on the hips and one marker at the lower back, approximately at waist height. For the shoulder 3 markers were placed on the upper arm. And for the hip joints a little compromise was made, since there is not much y-rotation, and only 2 markers were placed on the upper leg. During computation of orientations the z-axis for the hips would be approximated by the help of the x-axis of the root. The back joint was handled in a similar manner. For other joints, two markers would be used at a time, which mostly would be placed on the corresponding segment and thereby define the y-axis of the local coordinate system.

Consequently, the next step in the program is for the user to identify which markers will be used for the computation of which joint. It allows 3 markers to be assigned for the root, the shoulders, and the hips. All other joints are only allowed 2 markers. Figure-5 shows the location of the markers relative to the body, as they were used in the samples collected for this dissertation. And Table-3 shows which markers were assigned to which joint, some joint orientations were not computed (i.e., not used) because they were either irrelevant, or they were ignored to keep the problem at hand at a manageable complexity.

Once the markers are assigned, the orientations can be computed, starting with the root then moving down the hierarchy. Each joint's local coordinate system is computed using the markers assigned to it, in cases where only 2 markers are chosen the missing axes are approximated by those of the parent's joint. Then each joint's orientation is computed as the

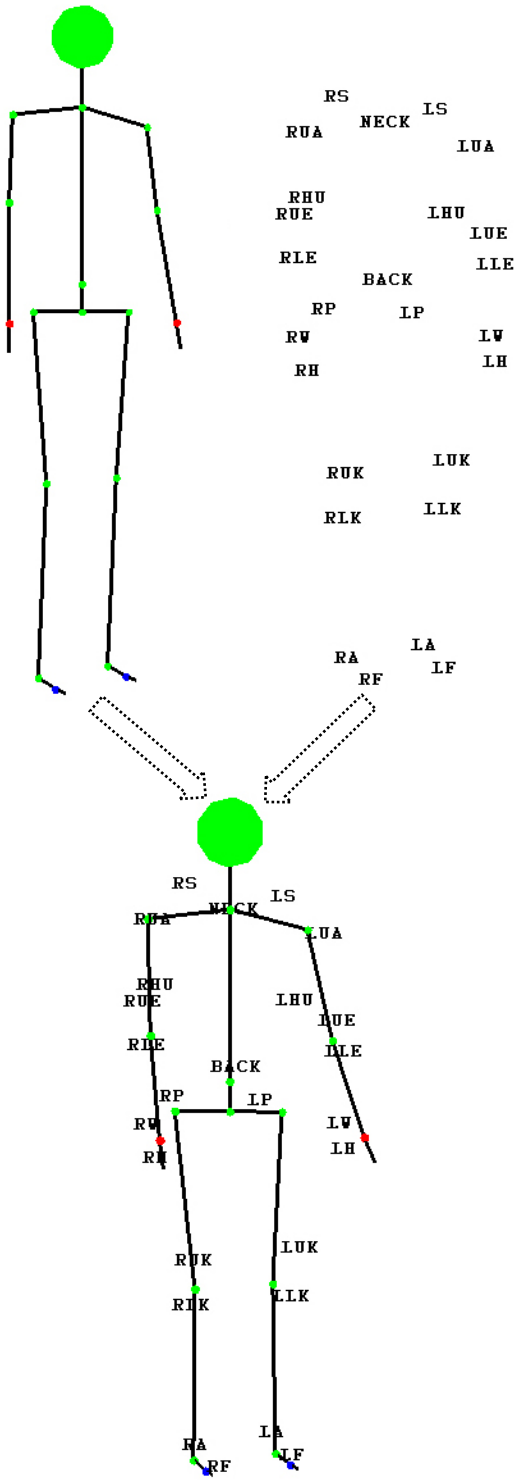

Figure-5 Marker positions relative to articulated figure.

Table-3 Marker to Joint Assignments

	JOINT	MARKERS	USED
1	Root	BACK, RP, LP	Yes
2	Back	BACK, NECK	Yes
3	Neck	not available	No
4	Head	not available	No
5	Right Collar	NECK, RS	No
6	Right Shoulder	RUA, RHU, RUE	Yes
7	Right Elbow	RLE, RW	Yes
8	Right Wrist	RW, RH	Yes
9	Right Hip	RP, RUK	Yes
10	Right Knee	RLK, RA	Yes
11	Right Ankle	RA, RF	Yes
12	Right Toe	not available	No
13	Left Collar	NECK, LS	No
14	Left Shoulder	LUA, LHU, LUE	Yes
15	Left Elbow	LLE, LW	Yes
16	Left Wrist	LW, LH	Yes
17	Left Hip	LP, LUK	Yes
18	Left Knee	LLK, LA	Yes
19	Left Ankle	LA, LF	Yes
20	Left Toe	not available	No

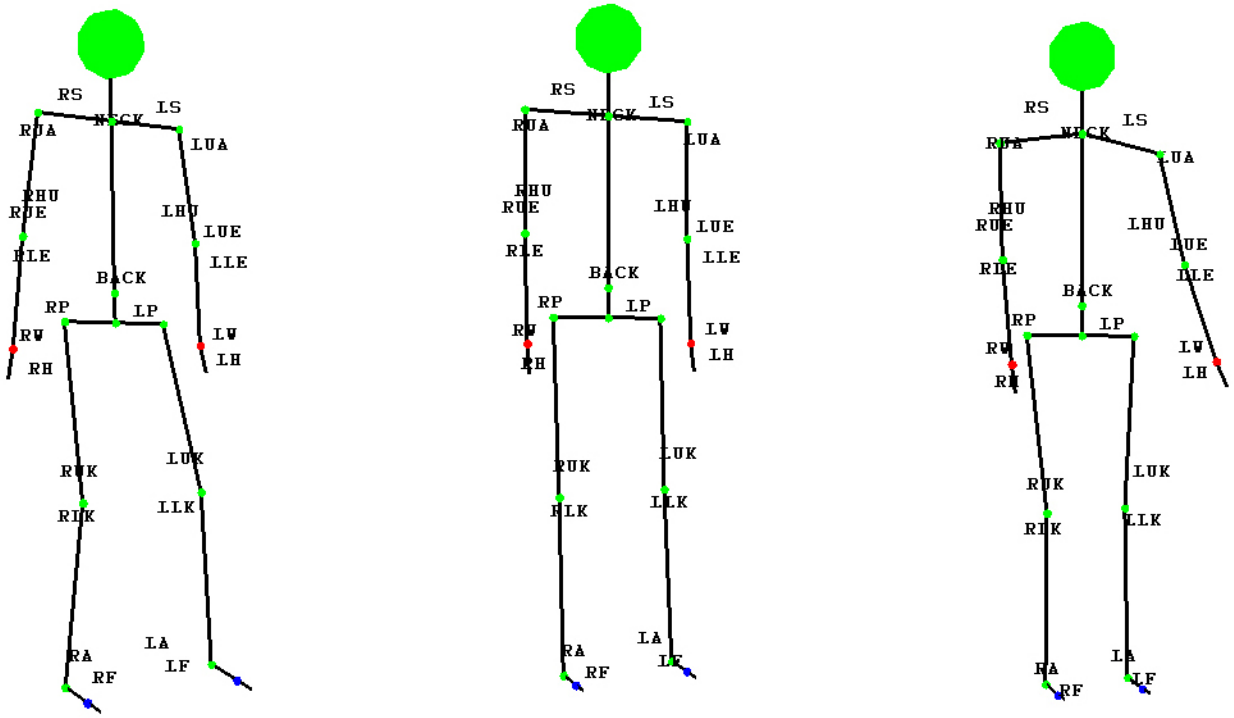

Figure-6 Effect of calibration and rest angles after assigning markers for joint orientation computation. **Left:** not calibrated and rest angles are set to zero. **Middle:** calibrated, but rest angles are set to zero. **Right:** calibrated and rest angles are set to match subject.

frame transformation between the local coordinate system of the joint and that of its parent. The root's parent is considered to be the world coordinate system. For 1 DOF joints, the computed transformation is applied to the world's y-axis, and then the pure x-rotation is computed using a trigonometric function call to 'atan2' with the y and z components of the transformed y-axis as the parameters.

One thing that has to be taken into consideration is the calibration of the markers. When the subject is recorded at zero-pose the readings of all joint angles should be zero. But since marker placement is not always accurate and the markers are often not aligned with the segments, the values of the computed joints orientations may not be all zero. Therefore, a calibration angle for each joint has to be added which is equal to the inverse of the angle measured at zero-pose (as in the left image of Figure-6). These calibration angles will remain the same as long as the markers are not moved. This means that they only need to be computed once

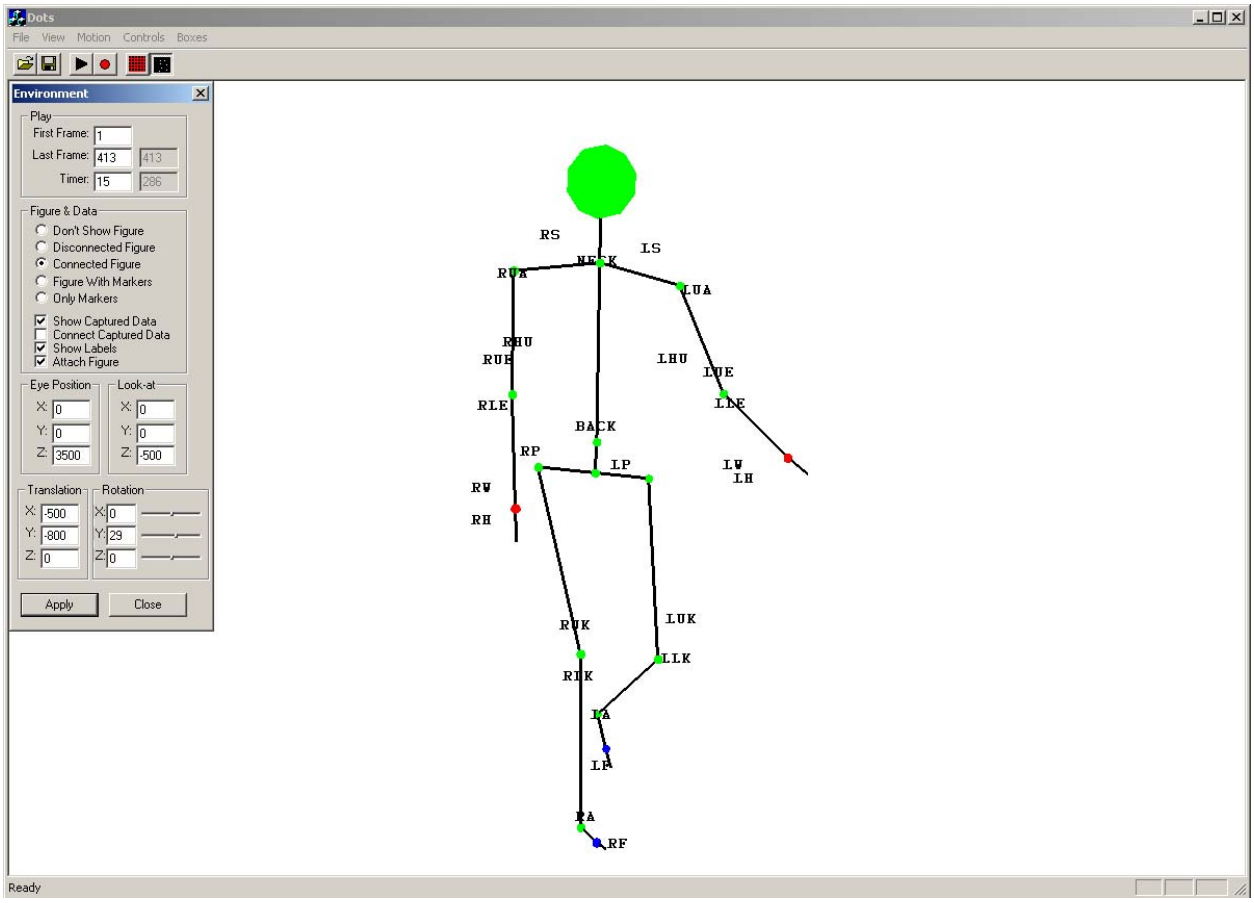

Figure-7 Walking using a calibrated figure with rest angles.

for each subject. Subsequent motions will use the computed calibration angles to compute the actual orientations of the joints that compose the recorded action, see Figure-7.

The program then allows writing out the computed orientations as a Pose file, which is given the extension .pos. Its format is described in section 4.2. A Pose file can be played back using another program: the Pose File Player, which has limited IK capabilities, see APPENDIX C. Data for the figure can also be written to a file given the extension .fig. It will include the segment lengths, rest angles, and calibration angles. The program allows other operations such as taking snapshots of the display area, or even recording a sequence of frames. Such images are written in TGA format. It allows writing out selected joint locations in world space for selected frames, which can be helpful in data verification. And segments may be turned on or off for better display.

This chapter presented some key definitions. It then went on to explain the conversion of the raw data resulting from motion capture to a format suitable for use to drive an articulated figure. The conversion started with segment and rest angle specification, then marker assignment, calibration, and finally writing out pose files.

CHAPTER 5
COMBINED PARTIAL MOTION CLIPS

The solution to the problem of mixing partial motions will be described here. The objective is to be able to edit a base motion to generate a variation of it by mixing it with a partial motion. It should be possible to use the same partial motion clip to generate the same kind of variation in a multitude of base motions. For that, the partial motion clip has to be equipped with some 'knowledge' about the different base motions and how other DOFs will be affected in each case. The partial clip should be as compact as possible, yet allow as large a number of DOFs to be affected as necessary to make the resulting motion look natural.

For one kind of action, multiple samples of that action performed during different base motions are analyzed to find the effect the partial motion has on the rest of the body. The samples are then combined into one clip called a Combined Partial Motion Clip (CPMC). What is contained in a CPMC, how it is created, and how it is used is described next.

5.1 Definition

A *Combined Partial Motion Clip* is a combination of several similar partial motion clips. Combined, in that the included samples are not all stored individually, but are averaged whenever possible. Partial, in that the included samples define only some DOFs of the articulated figure. Similar, in that they describe a specific action.

The difference between the samples is that they are performed during different base actions. Furthermore, a CPMC contains scripts that allow the computation of other DOFs. The scripts contain equations that represent an approximation to the correlation between joint movements. This makes it possible to use a CPMC to edit a base motion by generating a variation of it while affecting all DOFs in a reasonable way. The same CPMC can be used for as many different base motions as were included in the samples, each time generating the same kind of variation.

5.2 Generating a Combined Partial Motion Clip

To create a CPMC two main tasks need to be accomplished: 1) extract the partial common action, 2) find the effect of the DOFs in the partial action on other DOFs. Therefore, several steps are taken in order to produce a CPMC (see Figure-8 below). First, motion samples are collected. Then, the samples are preprocessed to ready them for partial action extraction. Then, the partial action is extracted and analyzed. Finally, the samples are combined into a CPMC.

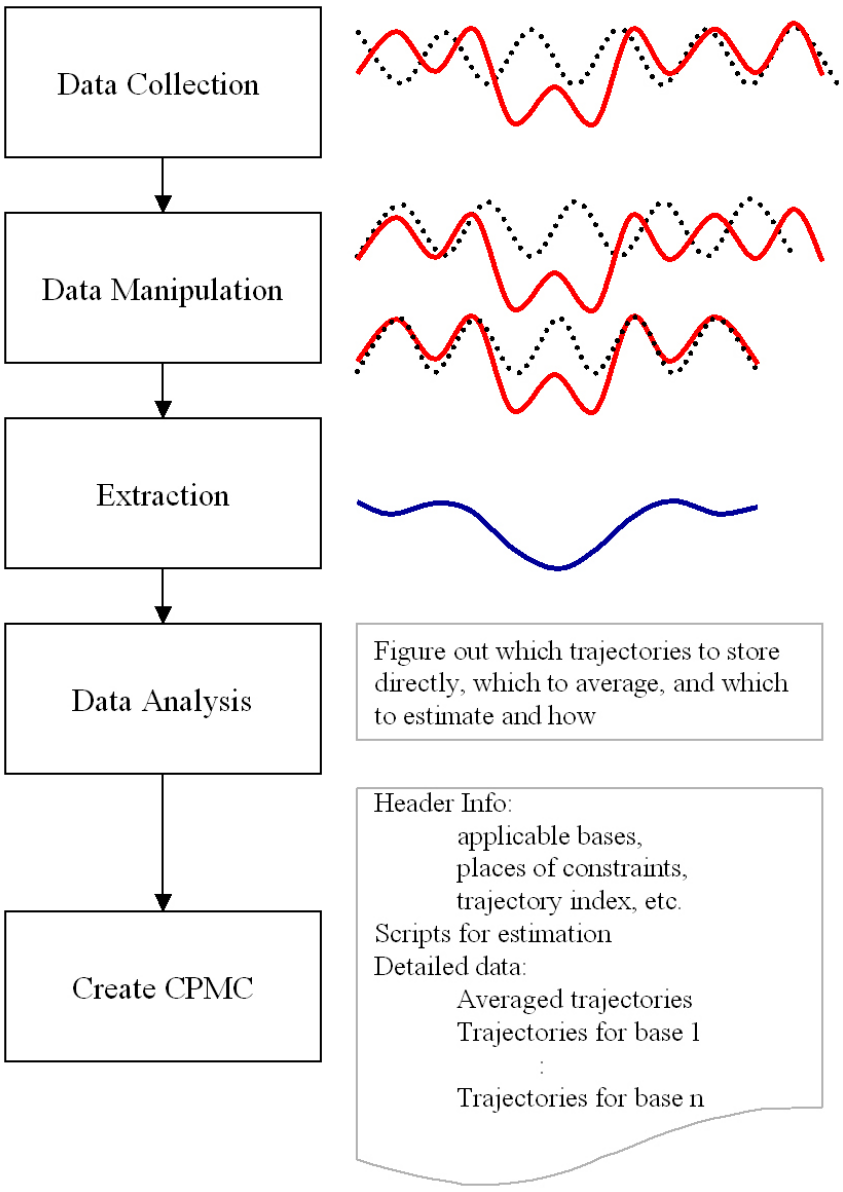

Figure-8 Creating a CPMC

5.2.1 Data Collection

Motion data for the desired action (i.e., the one that causes the variation) needs to be collected. The data should include multiple samples where the action is performed during different base actions, e.g., reach while walking, reach while jumping, and reach while squatting. The base actions without the desired action also need to be collected, e.g. walking only, jumping only, and squatting only. Samples should be taken from several subjects to account for the variations in personal style for a more complete analysis.

5.2.2 Data Manipulation

Since multiple samples are being dealt with simultaneously, this method bears similarity to motion space techniques where parameter and time correspondence is of major concern. But not all samples will be stored, that is where this method differs from motion spaces, and that is why no interpolation will be needed.

The collected motions need to undergo some preprocessing to make them ready for partial action extraction and to establish time correspondence. The main task is to align the motions based on some landmarks, e.g. heel-strikes, maximum arm extension, etc., in order to achieve time correspondence. Shifting, resampling, and trimming usually accomplish the task. There is no need to worry about parameter correspondence because all samples are from the same source so they are in a uniform format.

5.2.3 Partial Action Extraction

Initially, in order to extract the desired action a transformation was computed that would transform the plain base motion into the base motion with the desired action at each frame. Since joint orientations are stored as Euler angles and the position as a relative displacement, computing the transformations for them is done in different ways. To compute a transformation from one orientation to another the Euler angles must be converted to rotation matrices first which involves trigonometric function calls, multiplications, additions and subtractions. Then the dot product of each corresponding unit row vector of the two rotation matrices has to be taken to compute the transformation as another rotation matrix. Finally, the resulting rotation matrix has to be converted back to Euler angles which involves square roots, more trigonometric function calls, and divisions. To compute a transformation of one positional vector to another, only their

vector difference has to be taken. A clip extracted this way, could then be used by applying the transformations contained in it to another base motion. Again, this involves the costly conversion of the Euler angles to rotation matrices, multiplying the matrices – another costly operation, and then converting them back to Euler angles. It is evident that this whole process is computationally expensive.

But experimenting with the data at hand, led to the conclusion that one could merely use a vector difference of the Euler angles to extract the change, and later use a vector addition to generate the new motion with equally good results thus simplifying the process dramatically. This also meant that the joint vectors $\mathbf{v_i}$ could be handled the same way the positional vector $\mathbf{v_0}$ was handled. See Figure-9 and Figure-10 below for a comparison. Note there is a small difference in the trajectories produced by addition and the ones produced by transformation in the right image of Figure-10, but looking at the actual playback of such a motion the difference is negligible (see Figure-11). Furthermore, such differences only occur in joints with more than 1 DOF because the x, y, and z components in the Euler angle representation are not independent of each other. For 1 DOF joints computing the transformation or taking a difference produced identical results. The advantage of using differencing and addition outweighs the disadvantage. Hence, differencing was chosen as the means for extracting the partial action in this work, and addition is then used for editing a base motion with a CPMC.

Taking the difference of two motions is not a new idea. The purpose for which it is used in the work presented here is what is new. For example, in the work of Amaya et al, the purpose was the extraction of *emotion*, a difference was taken between a motion performed showing some emotion (e.g., sad or angry) and the same motion performed in a neutral state [AMA96]. Therefore, the extracted difference was a measure of emotion that could be used to transform other neutral motions to emotional motions. Unuma et al extracted *characteristics of a walk*, e.g., briskness, by computing the difference between a brisk walk and a neutral walk [UNU95]. This characteristic could then be added to a running motion for example. Or it could be used to transition from a slow walk to a brisk walk using interpolation.

An analogy to image based rendering (IBR) methods can be made. Blending partial motions is like inserting synthetic objects (equivalent to a partial motion) into real scenes

(equivalent to a base motion). The insertion if done simply by copying the synthetic object into the scene will not look realistic since there is no interaction with the scene. The inserted object should be able to cast shadows and reflect other objects if it is shiny and be itself reflected by other objects already in the scene and so on. In other words lighting effects have to be considered (equivalent to affecting other DOFs). Debevec computed the color difference between two synthetic images, one without any objects (just a plane lit by the lighting model of the real scene), and one with the objects that will be placed into the real scene. The extracted difference was then added to the real image, thereby allowing shadows to be cast and light to be refracted [DEB98].

Liu et al computed ratio images, instead of differences, between images of faces with and without expression. The ratio image would therefore encrypt the changes in lighting due to the changes in expression which would include wrinkles. Together with geometric warping, a ratio image could then be used to change other person's neutral faces and make expressions more expressive [LIU01].

Similarly, Noh and Neumann were able to clone expressions by computing a vector difference between the vertices of two meshes of a facial model with two different expressions. The obtained difference (motion vectors) could then be applied to other facial models after establishing correspondences between this model and the one used in the differencing. The correspondence points helped in warping the model and adjusting the direction and magnitude of the motion vectors [NOH01].

In the work presented here, the computed difference is *an additional movement of part of the body* that is the result of a simple vector difference of each corresponding trajectory at each frame between the motion with the desired action and the plain base motion.

Let $\mathbf{m_{b1}(t)}$ define a base motion, and $\mathbf{m_{d1}(t)}$ define a motion of the same subject performing the desired action during the same base motion. The desired motion $\mathbf{m_1(t)}$ is extracted by taking their difference. The same is repeated for all samples of the desired action and their corresponding base motions:

$$\mathbf{m_i(t) = m_{di}(t) - m_{bi}(t)} \qquad \text{for i=1..NumberOfBases} \qquad \text{(Equation-2)}$$

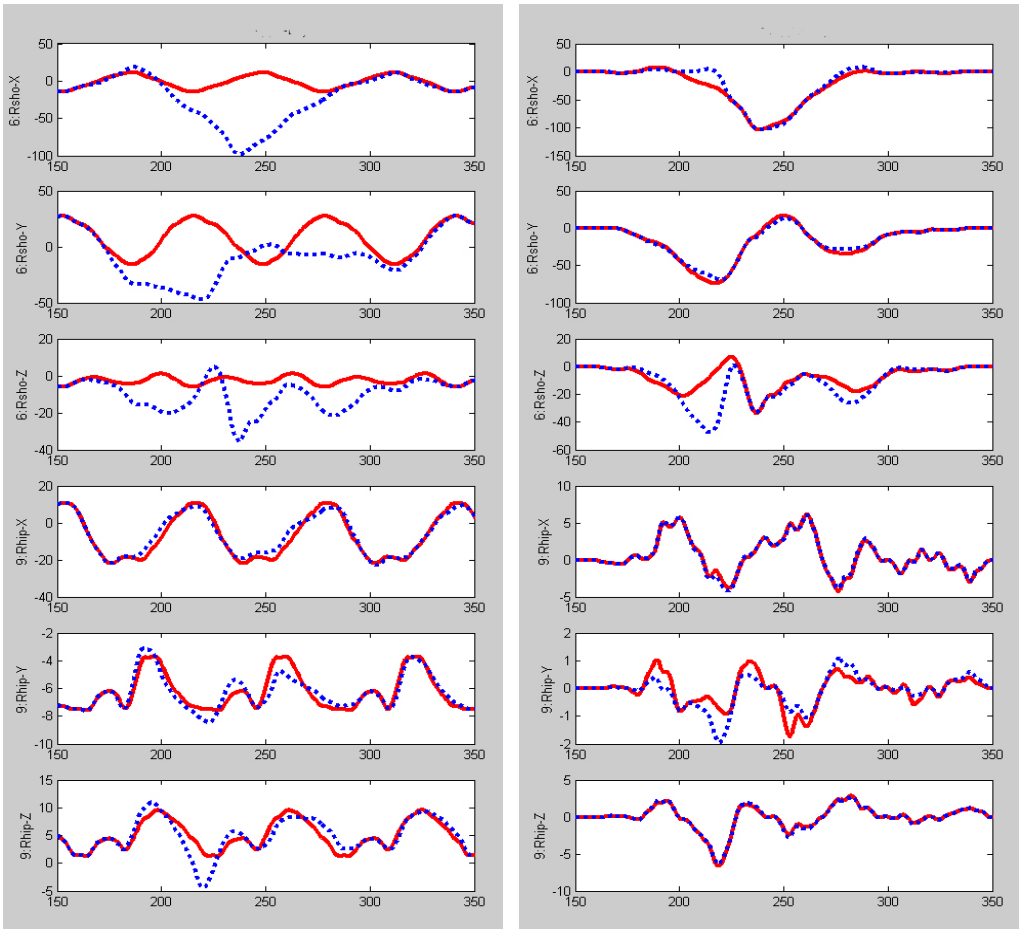

Figure-9 Difference vs. Transformation. Trajectories of the xyz-rotational components of the right shoulder and right hip. Vertical axis denotes degrees, and horizontal axis denotes frame numbers. **Left:** a walking motion – the base (solid) and the same subject's 'walk&throw' motion – the base with a variation (dotted). **Right:** the extracted partial action – the throwing motion – computed as a difference (solid) and as a transformation (dotted).

This will result in a collection of extracted desired actions, one for each base at least. The process is repeated for all subjects leading to more collections, one for each subject. At this stage all actions are still stored as complete motions.

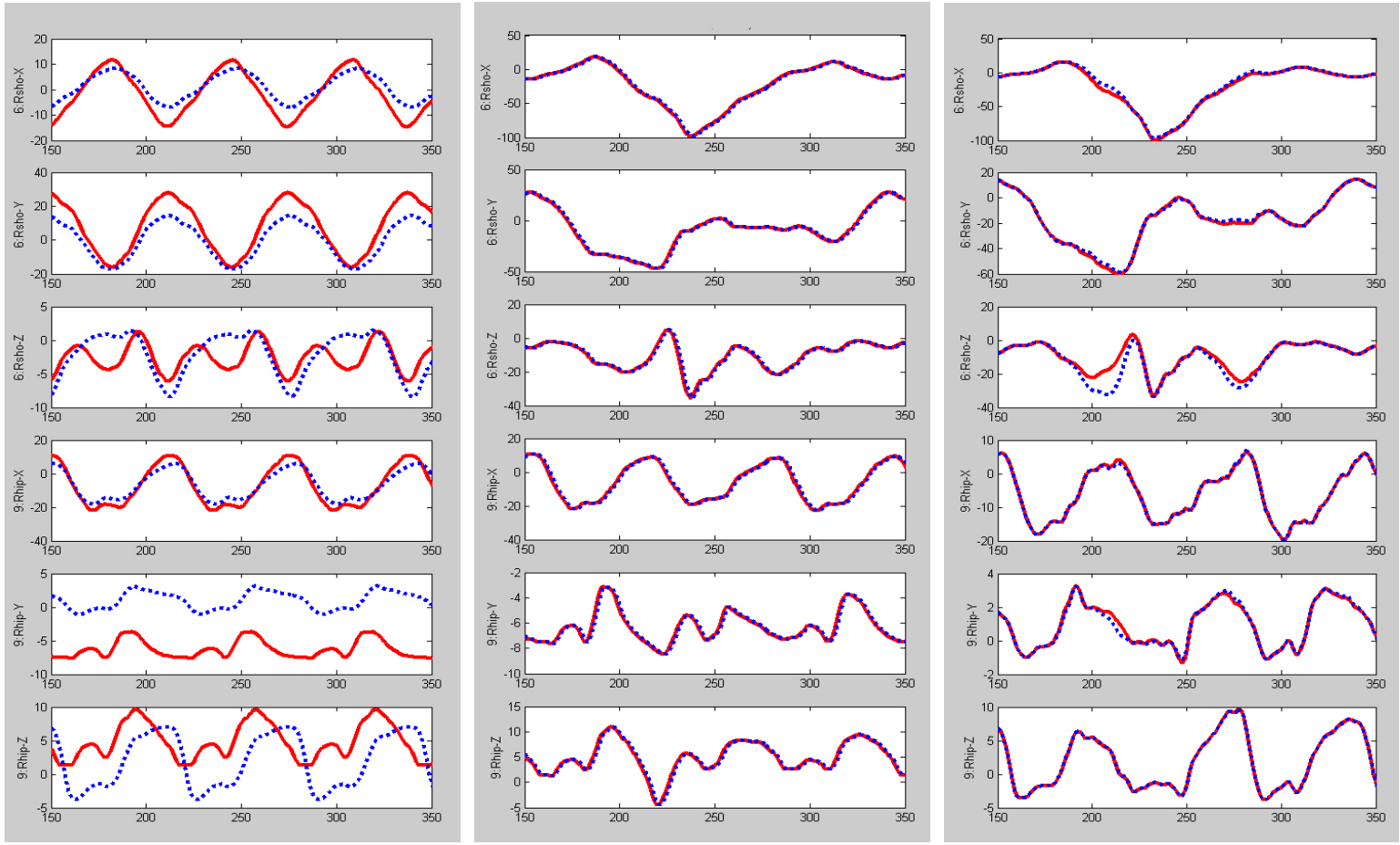

Figure-10 Addition vs. Transformation. Trajectories of the x, y, and z rotational components of the right shoulder and right hip. Vertical axis denotes degrees, and horizontal axis denotes frame numbers. **Left:** two walking motions – two bases, one of the same subject whose motions were used in the extraction of the partial action (solid), and another subject's walking motion (dotted). **Middle:** resulting motion of combining walk base and extracted throw of original subject using addition (solid) and using transformation (dotted), note the trajectories are identical and the original 'walk&throw' motion has been recovered. **Right:** resulting motion of combining another subject's walk base and extracted throw of original subject using addition (solid) and using transformation (dotted).

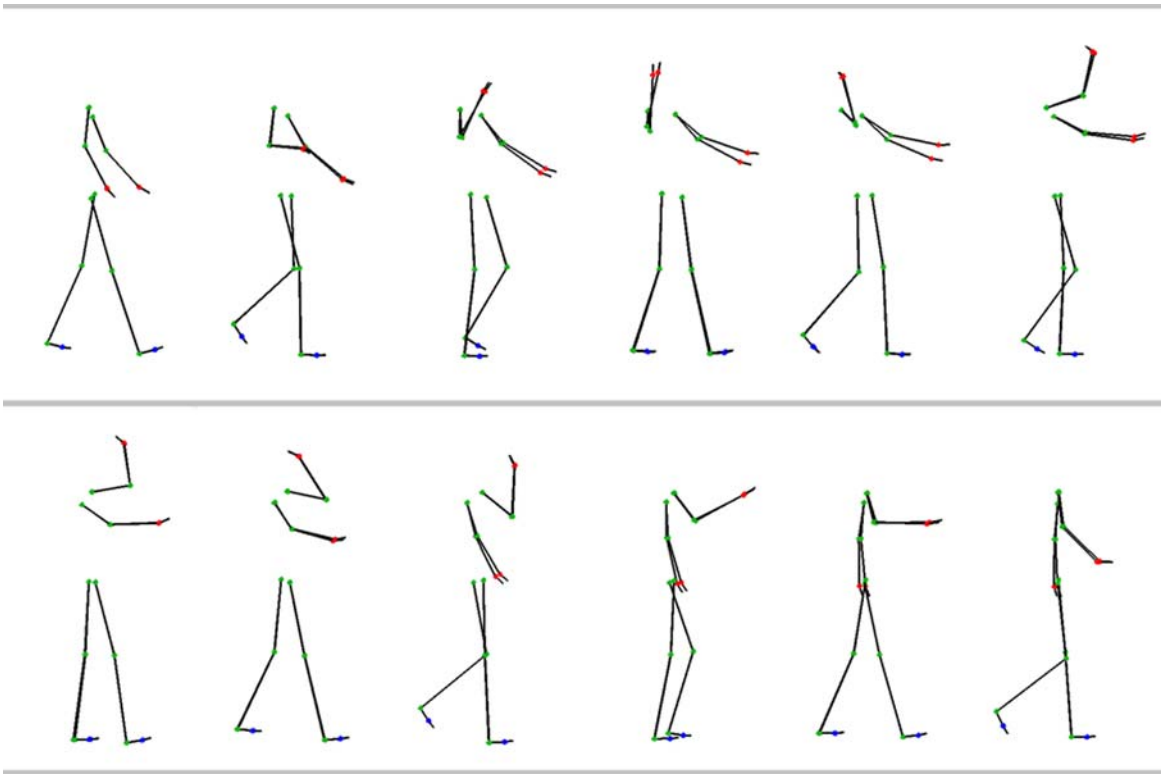

Figure-11 Addition vs. Transformation Frame Sequence. Two superimposed frame sequences of two generated 'walk&throw' motions, one by addition, and one by translation.

5.2.4 Data Analysis

It is desired to store a minimal amount of data without losing the details of the motion that characterize it, i.e. to store the extracted action as a partial motion. For that, the extracted difference motions ($\mathbf{m_i}$) are carefully analyzed in order to find out which trajectories need to be stored as they are, which can be averaged (across the bases and/or subjects) and which can be recovered through a relationship with the stored ones. In other words, it is desired to store the difference motion as a partial motion **pm** which includes detailed trajectories of the joints that are the most active ones in the extracted action. For the remaining joints not included in **pm**, a function has to be computed that relates the changes that occur there to the changes in other joints. That is, each joint i not included in **pm** should have a function $\mathbf{f_i}(\mathbf{pm}(t))$ defined through which the joint curve $\mathbf{q_i}(t)$ can be estimated. In some cases, the functions may differ for the same joint depending on the base motion, in other cases they may be the same for all bases, which is preferred.

The question asked is: What are the similarities and differences

1. between all $\mathbf{m_i}$s of one subject?
2. between subjects for a specific $\mathbf{m_i}$, i.e. the ones extracted from a specific base motion?

To answer those questions one has to compare the joint trajectories and their derivatives. Trajectories that are nearly identical in all $\mathbf{m_i}$s of one subject are usually the ones that are the main constituents of the desired action being extracted, like for example, the trajectories of the arm joints in a reaching motion. These are the main candidates for inclusion in the partial motion clip.

When analyzing the original motion data it is often hard to see how similar the affects of a particular DOF are on another for the different base motions. Since the main interest is not in the correlation between the joints in the base motion itself, but in the correlation between the DOFs involved in the partial motion and other DOFs independent of the base, the changes are analyzed. That is, the difference trajectories taken between the detailed motion and the bases are analyzed, because this takes out the effect of the base motion and enables one to find more accurate relationships between the DOFs of the partial motion and the DOFs of the base motion. These relationships can then be used to approximate the trajectories that will not be stored in the partial motion clip.

Furthermore, depending on the action on hand, one may turn to the research done in the field of biomechanics or kinesiology for help in analyzing the data and obtaining relationships between the movements of the joints. Or one may look at the data in an abstract form and use statistical methods for analysis, e.g. as a multivariate time series. This may require some cooperation with someone specialized in those fields.

5.2.5 Clip Combination

In light of the fact that several motions are used at the same time this technique bears similarity to motion space techniques. Wiley and Hahn [WIL97b], Rose et al [ROS98], and Guo and Robergé [GUO96] collected similar motions and parameterized them. New motions are then computed by interpolation. Wiley gives examples of reaching to different locations, or walking

on different slopes. He uses (bi)linear interpolation. Guo and Robergé focus on walking and running parameterized by step length and jump height. They also use linear interpolation. Rose et al use radial basis functions and multivariable interpolation. Samples in the motion space (verbs) are assigned parameter values (adverbs) by hand. A collection of walks can have an adverb specifying how sad or happy the walk is, another adverb is speed, another one could be step length. In all these methods, each sample motion is stored in the space. This leads to high space requirements. But in the method presented here, instead of storing all motions in the space they are combined into one motion clip that is suitable for different initial base actions thereby eliminating the need for interpolation and reducing the storage space requirements.

The results of the analysis step above dictate the way the difference motions are combined. Trajectories that are nearly identical across the bases can be averaged and stored. Whether one uses the data from one subject or all subjects depends on how much variation there is between them. For trajectories that can be estimated from their relationship with stored trajectories, the scripts for their computation are stored. The remaining trajectories making up the partial motion need to be stored individually for each base. This combination leads to a reduction in storage space requirements.

Let the number of bases be **b**, the number of trajectories that constitute the extracted partial action be **n**, where **n**<(**NumberOfJoints**+1), the number of averaged trajectories be **a** (i.e., the number of trajectories stored separately for each base = **n-a**), and the estimated ones be **e**. With the currently available editing methods, one would have to store the trajectories of all DOFs for all bases to be able to add the desired motion to another one and affect all DOFs, that is (**NumberOfJoints**+1)***b** trajectories. If only a partial motion is stored then only **n*b** trajectories need to be stored. Since some trajectories are averaged, a further reduction is possible, because the averaged trajectories are stored only once for all bases collectively. Therefore, the total number of trajectories stored is reduced to:

a+(**n-a**)***b** (Equation-3)

Consequently, the total reduction is equal to:

e*b + a*(b-1) (Equation-4)

Current methods would only affect **n*b** trajectories (equal to the number of trajectories stored); the method presented here allows the editing to affect (**n**+**e**)***b** trajectories (equal to the number of trajectories stored plus the estimated ones).

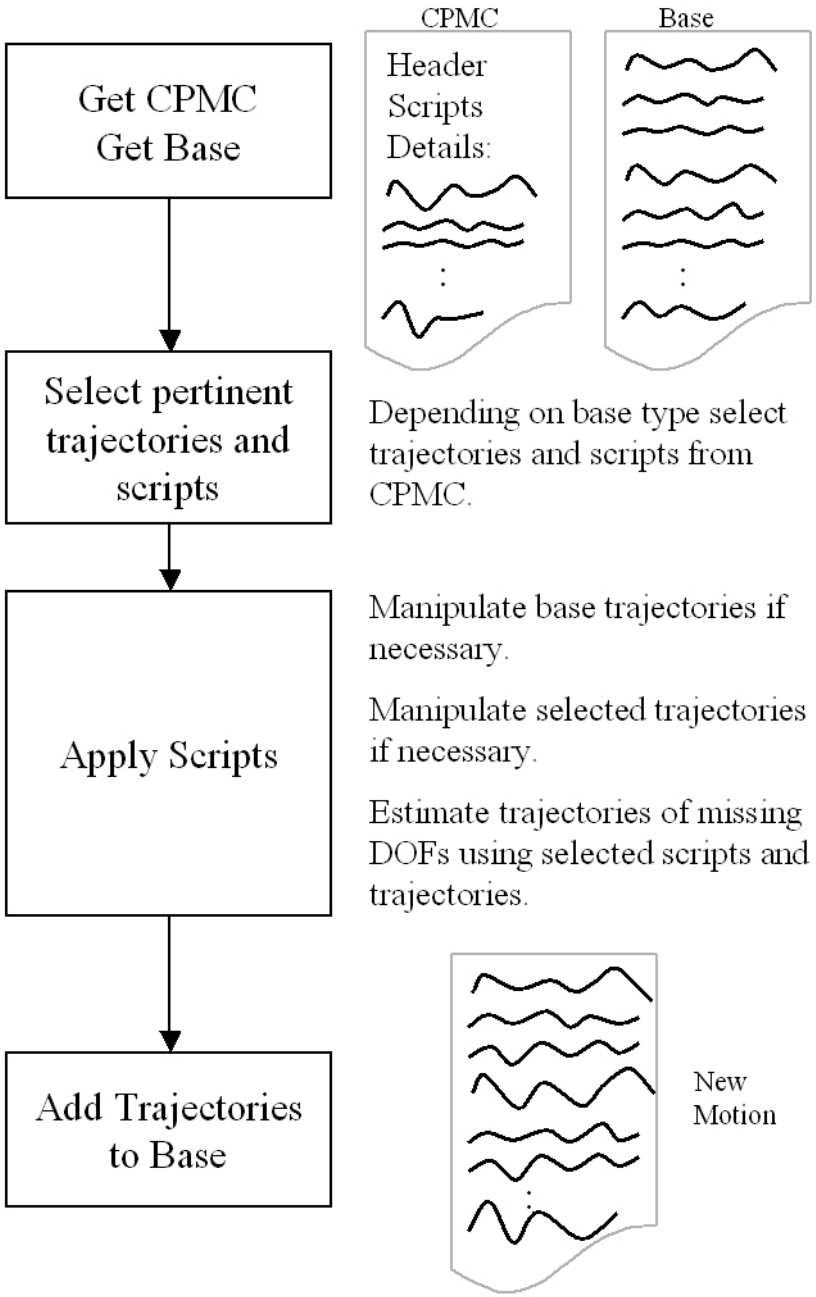

Figure-12 Using a CPMC

All the trajectories selected for storage and the scripts need to be preceded by an appropriate header. The header should contain information about constraint positions if necessary, like heel strike locations, or time of maximum arm extension, etc. It also should state which bases this CPMC applies to, and contain an index of the trajectories and scripts, and pointers to them or some other means for accessing and retrieving individual scripts and trajectories.

5.3 Using a Combined Partial Motion Clip

A CPMC can then be used by extracting the necessary trajectories and scripts from it according to the type of the base motion, computing additional trajectories, and adding them to the base motion. See Figure-12 above.

The main information needed for a successful addition of a base motion and a CPMC is the type of the base motion. Some base motions may need additional information, e.g., the cycle length and the time of the first right heel strike in a running base motion. Which data is needed exactly is identifiable from the header information of the CPMC and the scripts for that particular base.

Once the type of base motion is known, the appropriate trajectories $\mathbf{q}_i$ and scripts $\mathbf{f}_j$ can be selected from the CPMC with the help of the indexes and pointers contained in the header. Some base motions may require some preprocessing, for example resampling. Also, the selected trajectories need to be matched to the base motion since the CPMC is created for base motions with specific parameters, in the case of running it may be step length, jump height, and cycle length. After that, additional trajectories can be computed according to the scripts. Since a CPMC is a result of a difference, the last step is to add all the computed and selected trajectories to their corresponding trajectories in the base motion using a simple vector addition at each frame to get the final result. An algorithm is given in Figure-13 below.

```
Get type of base b
Get header information from CPMC
Get all q_i from CPMC for base b, where i∈{0,1,…, NumberOfJoints}
Get all f_j from CPMC for base b, where j∈{0,1,…, NumberOfJoints} and j≠i
Get all q_bk from base b, where k ={0,1,…, NumberOfJoints}
If necessary according to header information
      Get additional information about base b
      Resample the q_bk (or manipulate otherwise)
If necessary to match some aspect of base b
      Resample the q_i (or manipulate otherwise)
Compute q_i=f_j(pm)  where i=j, and pm = {selected and manipulated q_i(t)}
Compute new trajectories q_newk = q_bk + q_i , where k=i
```

Figure-13 Algorithm for CPMC usage

After this presentation it is clear that the creation of the CPMC depends on the simple operation of differencing for extraction of the partial action. After that the role of analysis comes into play. It is needed to find relationships between joint data in the difference trajectories so that they can be put into scripts and stored in the CPMC. Analysis is also needed to identify which trajectories can be averaged and which can not. Once the CPMC is created, addition is the main operation needed when using a CPMC to edit a base motion. Since differencing and addition are the main operations needed, the goal of simplicity has been met. And since trajectories are combined and averaged if possible, the goal of reducing space requirements has also been met. The next chapter will give a demonstration.

CHAPTER 6
EXPERIMENT

The method described in CHAPTER 5 was put to the test using a specific example. A throwing motion was extracted from several clips that involved throwing while walking, throwing while standing, and throwing while sitting. A CPMC was created and used to edit base motions from several other subjects. The resulting motions were believable and looked natural. Their quality was assessed by comparing them to original motions (in CHAPTER 7).

6.1 Generating a CPMC for 'throwing'

The steps explained in the previous chapter in section 5.2 and depicted in Figure-8 for generating a CPMC will be discussed next with regard to the specific example of throwing.

6.1.1 Data Collection

The overhand throwing motion of six subjects (three males and three females, with different heights) was obtained via optical motion capture at the biomechanics laboratory at the Warren Grant Magnuson Clinical Center at the National Institutes of Health (NIH). A Vicon system was used with seven cameras that operated at 60Hz. The subjects had 26 light reflecting markers attached to their body: five on each leg, six on each arm, one on top of each shoulder, one on the neck, and one on the back. See Figure-5 in section 4.3.

The subjects were simply asked to throw a ball to another person standing approximately 10 feet away. The action was performed using a lightweight sponge ball during walking ($\mathbf{m_{d1}}$), during standing ($\mathbf{m_{d2}}$), and during sitting ($\mathbf{m_{d3}}$). For the walking case, the subjects were asked to walk a few steps, throw the ball, and continue to walk. For the standing case, the subjects were asked to remain standing and not to take a step for the throw. For both standing and sitting, the subjects were asked to return their hands after the throw to the positions they started from, that would be straight down the sides while standing, and on their laps while sitting. The subjects also performed just walking ($\mathbf{m_{b1}}$), just standing ($\mathbf{m_{b2}}$), and just sitting ($\mathbf{m_{b3}}$). Each action was captured several times until at least 5 acceptable recordings were obtained.

Some problems were encountered in the throwing while walking case due to the limited space available for motion capture. As mentioned above, the lab used was the biomechanics laboratory at NIH, which is a gait lab where usually just the leg motions are captured over a range of 2 or 3 steps. Therefore, capturing the whole body was a challenge. And capturing it while walking and throwing was an even bigger challenge especially since it was desired to capture at least 7 steps.

The data was cleaned up, which is a very time consuming and tedious task. Clean up mainly involved filling in missing or mixed up marker data by hand whenever possible, otherwise the motion had to be discarded. Marker data may have been missing due to occlusions during the capturing process or because a marker had fallen off. Marker data may have been mixed up when two markers were place too close together. Their captured trajectories may have been misidentified by the user and assigned to the wrong markers. Additionally, the data was smoothed by computing a weighted moving average to reduce the effects of noise contained in it. Finally, the data was converted to pose files as explained in 4.3.

6.1.2 Data Manipulation

Some of the converted data still needed further refinement in part as a result of the problem encountered while capturing as explained above. The refinement mainly consisted of extending the throwing while walking motions with a few steps at the beginning and end. This was done by finding a closest match in a walking motion of the same subject for the beginning of the 'walk&throw' motion and its ending. Since there are several 'walk' motions for each subject, the one with a cycle length equal to that at the beginning and end of the 'walk&throw' was considered for the extension.

The closest match is identified as the frame with the minimum value for the distance between two corresponding partial poses in the two motions. Where the distance between two partial poses is denoted by: $||\mathbf{PP_2}-\mathbf{PP_1}||$ and is computed as the summation of the norms of the difference between each corresponding joint vector included in the partial pose and between position vectors if included:

$$||\mathbf{PP_2}-\mathbf{PP_1}|| = \sum_{\mathbf{v_i} \in \mathbf{PP_1}} ||\mathbf{v_{i1}}-\mathbf{v_{i2}}|| \qquad \text{(Equation-5)}$$

Partial poses were used because some joints were unused and should not be included in the distance measure, and because some have less significance than others, e.g. differences in ankle orientation are less important than differences in hip orientations. Another possibility would have been to use a weighted sum of the norms. When a match was found in the 'walk' motion the frames for one or two walk cycles were copied and attached to the front of the 'walk&throw' motion. Similarly, a match at the ending of the 'walk&throw' was found in the 'walk' motion and one or two cycles were copied and appended to the 'walk&throw' motion.

Since matching just one frame can lead to visible discontinuities the actual process was to find a sequence of matching frames (approximately 9-15 frames) in the 'walk' motion which were overlapped with the corresponding sequence of frames in the 'walk&throw' motion and blended linearly. For concatenations at the beginning of the 'walk&throw' motion, the 'walk' motion would be faded out while fading in the 'walk&throw' motion over the duration of the overlapped sequence. The opposite was done for concatenations at the end of the 'walk&throw' motion.

Another reason for refinement was to make the walking motions cyclic. The same matching procedure was used only that the two parts matched would both be in the same motion. For this to work properly a small offset after which to start searching for a match had to be specified otherwise each frame would have been matched with itself. The offset chosen was 10 frames. Only the hip, knee, and ankle joints were included in the partial poses that were considered for their distance. The frames between the two matched parts made up one cycle plus an overlap region. This sequence was then used to create a walk with any number of cycles by overlapping its ends and linearly interpolating several times.

After that, the motions of each subject were prepared for differencing. 'Walk' $\mathbf{m_{b1}(t)}$ and 'walk&throw' $\mathbf{m_{d1}(t)}$ motions had to be aligned properly. The cyclic walking motions and the refined 'walk&throw' motions were used now. There was no need to match step lengths because the change in step length will be computed implicitly in view of the fact that $\mathbf{v_0}$ was included in the differencing process. Neither does one need to match the motions' translational component explicitly because the position of the articulated figure is expressed as a displacement in the local coordinate system of the root. Only initial orientations of the roots have to be aligned. This

applies to all bases. For the motion samples used here, subjects were always facing the same direction at the time of motion capture, thus the initial root orientations were equal and no further manipulation in that regard was needed.

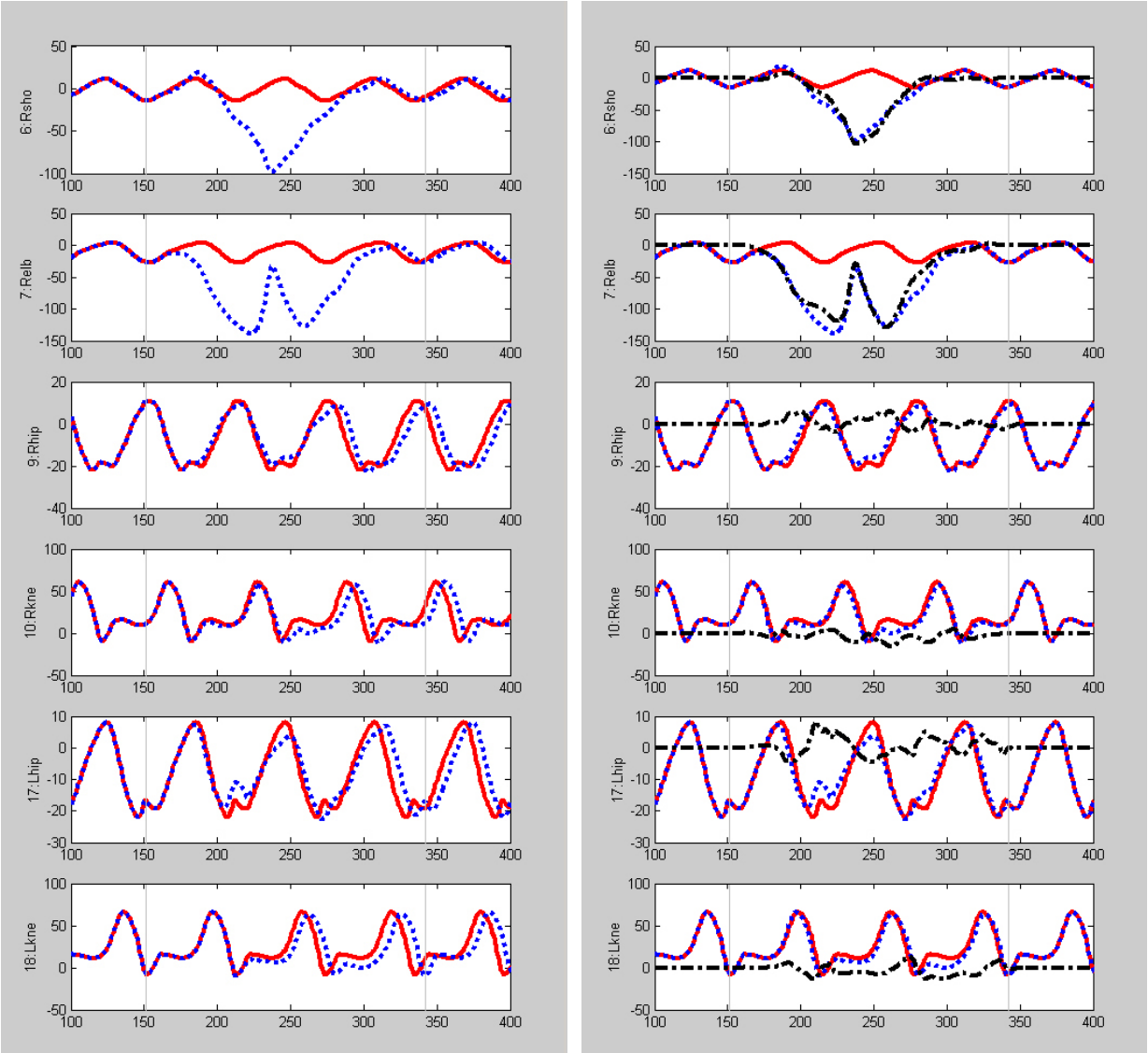

Figure-14 Data Manipulation and Differencing. X-rotation of the right shoulder, elbow, hip, and knee, and the left hip and knee. Vertical axis denotes degrees, and horizontal axis denotes frame numbers. Vertical lines indicate the throw-period beginning and end. Solid line indicates a 'walk', dotted line a 'walk&throw' of the same subject, and dashed line their difference after alignment. **Left:** original walk is only shifted to match left heel strike at the beginning of throw-period. **Right:** original walk is shifted and resampled to match heel strikes at the beginning and end of the throw-period.

Since during walking it is usually the case that one slows down in preparation for the throw, part of the walking motion $\mathbf{m_{b1}(t)}$ was resampled. To be more specific, part of the walking motion starting at a heel strike that matched the one just before the initiation of throwing in $\mathbf{m_{d1}(t)}$ and ending at a heel strike just after the throw when normal walking was resumed (the *throw-period*) was resampled so that its duration would be matched with that in $\mathbf{m_{d1}(t)}$. The throw-period may extend over 3 to 7 heel strikes, differing from one person to another depending on their personal throwing style. In a right-handed throw the throw-period may start as early as a left heel strike before the right arm starts the wind-up phase. It ends a few steps later at a right or left heel strike, after the arm is back to its normal position, i.e. after the follow-through phase, when normal walking can be resumed. The period is marked with two vertical lines in Figure-14 above.

Which heel strike is to be used is initially approximately identified from the playback of the motion. To find the exact time, i.e. exact frame number the knee joint data is examined. Heel strikes are taken to be at the time when the knee joint is at its maximum extension, i.e. the x-rotation is very close to zero. Based on that, the heel strikes marking the throw-period can be identified.

Let the starting heel strike be at time t_1 in $\mathbf{m_{d1}(t)}$ and at time t_2 in $\mathbf{m_{b1}(t)}$. And let the ending heel strike be at time t_3 in $\mathbf{m_{d1}(t)}$ and at time t_4 in $\mathbf{m_{b1}(t)}$. The amount by which $\mathbf{m_{b1}(t)}$ is shifted to align the beginning of the throw-period is equal to t_2-t_1, see the left image of Figure-14. The rate at which $\mathbf{m_{b1}(t)}$ is resampled to align the end of the throw-period is equal to $(t_3-t_1)/(t_4-t_2)$, see the right image of Figure-14. This rate is applied to all joints and to the position. But for the position, additional treatment is required; it has to be scaled by the inverse of the resampling rate, because the position is stored as a relative displacement. This process was done for each pair of walks and walk&throws for each subject.

6.1.3 Partial Motion Extraction

After alignment differences were taken by applying (Equation-2) from section 5.2.3. Since there are three base motions several samples of $\mathbf{m_1(t)}$, $\mathbf{m_2(t)}$, and $\mathbf{m_3(t)}$ for each subject resulted from the differencing process. These samples were then analyzed comprehensively. The difference of one sample is shown in the right image of Figure-14 and in Figure-15.

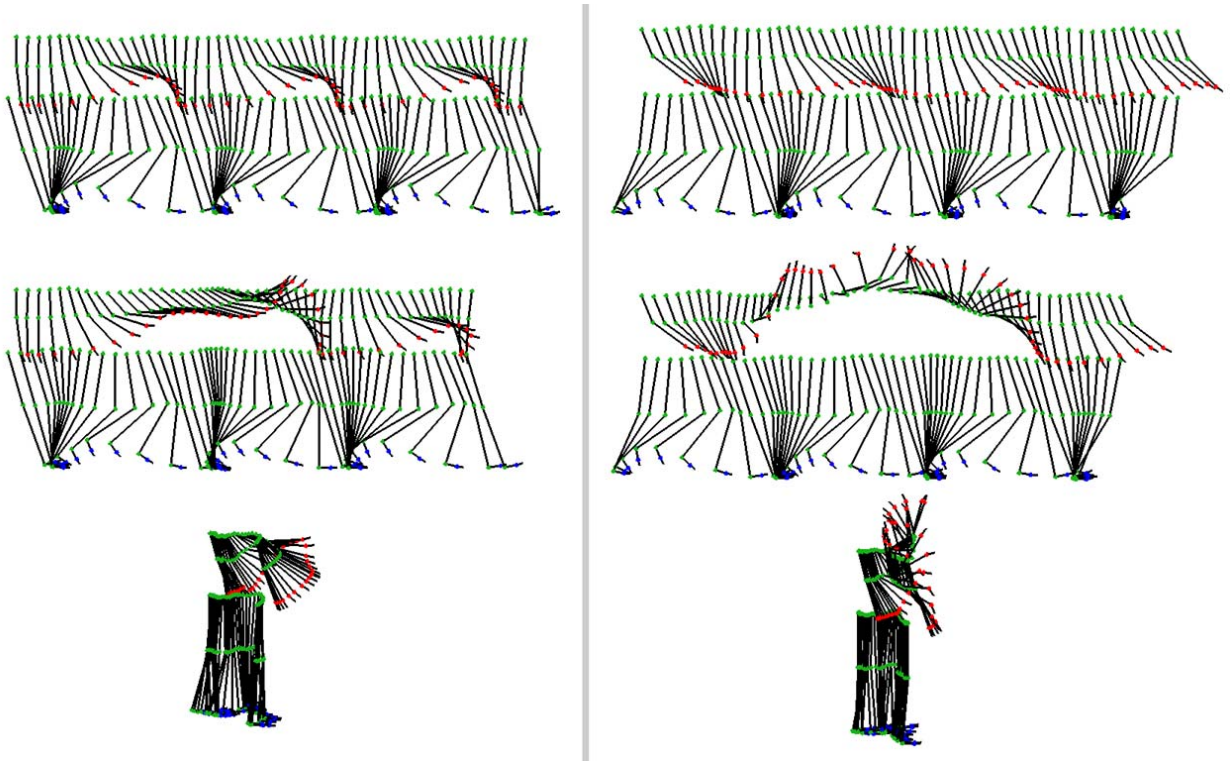

Figure-15 'Walk' (top), 'walk&throw' (middle), and their difference (bottom) depicted as multiple frame exposures with every 4th frame shown. **Left:** left arm and leg. **Right:** right arm and leg.

6.1.4 Data Analysis

In the field of biomechanics, a lot of research is devoted to throwing. The various kinds of throwing are explored, e.g. baseball pitching, football passing, javelin, etc. The main purpose of the research though is to explain the mechanics of throwing and prevention of injuries [FLE96]. Some research compared throwing mechanisms in persons with different levels of professionalism [FLE99], other research compared free-throws of athletes in wheelchairs with those of non-disabled players [DOW97]. Yet some other research, looked into the changes in throwing patterns as a result of change in ball size [BUR92]. And others studied timing relationships in throwing [ELL88]. This shows the diversity of research concerning throwing and finding relationships between some variables. But apparently there is no research specific to the coordination of the upper body and the lower body during throwing, which is what is being sought in this example.

Putting the collected motion data and their differences into a format suitable for statistical analysis as a multi-variate time series, which seemed to be appropriate for the objective at hand, was unfeasible at the time this research was conducted. Therefore, simpler analysis methods were reverted to. That is, comparing the plots of the trajectories and their derivatives.

From such comparison, it was noticeable that the throwing-arm motion was almost identical in $\mathbf{m_1}$, $\mathbf{m_2}$, and $\mathbf{m_3}$ for a specific subject, but would differ across subjects because of their distinct throwing style (see Figure-20). Nevertheless, the changes induced in the non-throwing arm by the change of the motion of the throwing arm for example would be similar across subjects and even across bases, and could therefore be computed using the same equations. (For clarification examples are shown in section 6.1.6.) More specifically, the y-rotation of the trunk affected the swinging of the non-throwing arm, e.g. the x-rotation of the shoulder. The non-throwing arm was also affected by the motion of the shoulder of the throwing arm. And the changes in the elbow and wrist of the non-throwing arm could be approximated as scaled versions of the changes in the shoulder (see Figure-16). A MATLAB script for the computation of the non-throwing arm is given in APPENDIX A.

As for the legs, the changes only needed to be computed for the case of walking. For sitting and standing, one could simply apply inverse kinematics to keep the feet steady on the ground since the root position and orientation and therefore the hip positions were known. From the comparison of the leg joint trajectories in the original motions $\mathbf{m_{b1}(t)}$ and $\mathbf{m_{d1}(t)}$ – as opposed to the differences – it became apparent that there was only a slight increase/decrease in the hip rotation at the peaks/valleys, which coincide with right and left heel strikes (refer to Figure-14, in section 6.1.2). The changes in the hips can be explained as a way to compensate for the changes in the body's heading direction. Hence, the amount of change was related to the change in the position of the root and a simple displacement map could be constructed. The change in the x-direction (sideways) affected the z-rotation of the hips and vice versa, i.e. the change in the z-direction (forwards and backwards) affected the x-rotation of the hips. APPENDIX B gives a script for computing such a displacement map. Changes in knees and ankles were similar (see Figure-16).

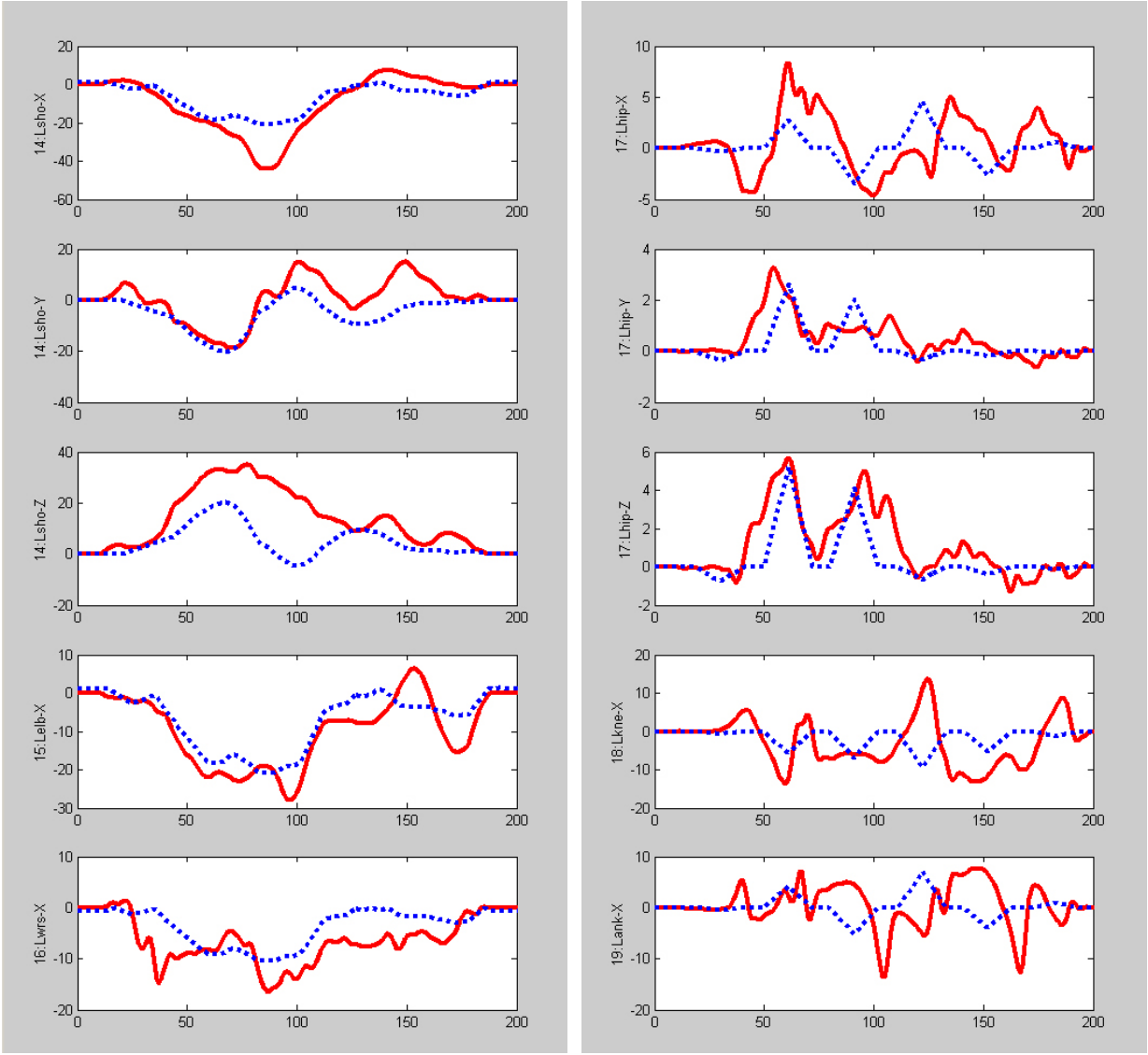

Figure-16 Comparison of original difference trajectories (solid) with computed ones (dotted) of the left arm and left leg for the walking case. Vertical axis denotes degrees, horizontal axis denotes frames.

6.1.5 Clip Combination

From this analysis it was concluded that in addition to cycle length and the equations or scripts for computing the non-throwing arm and the legs, only the trajectories of the trunk (root, and back), the position, and the throwing arm (shoulder, elbow, and wrist) needed to be stored, i.e., **n**=6.

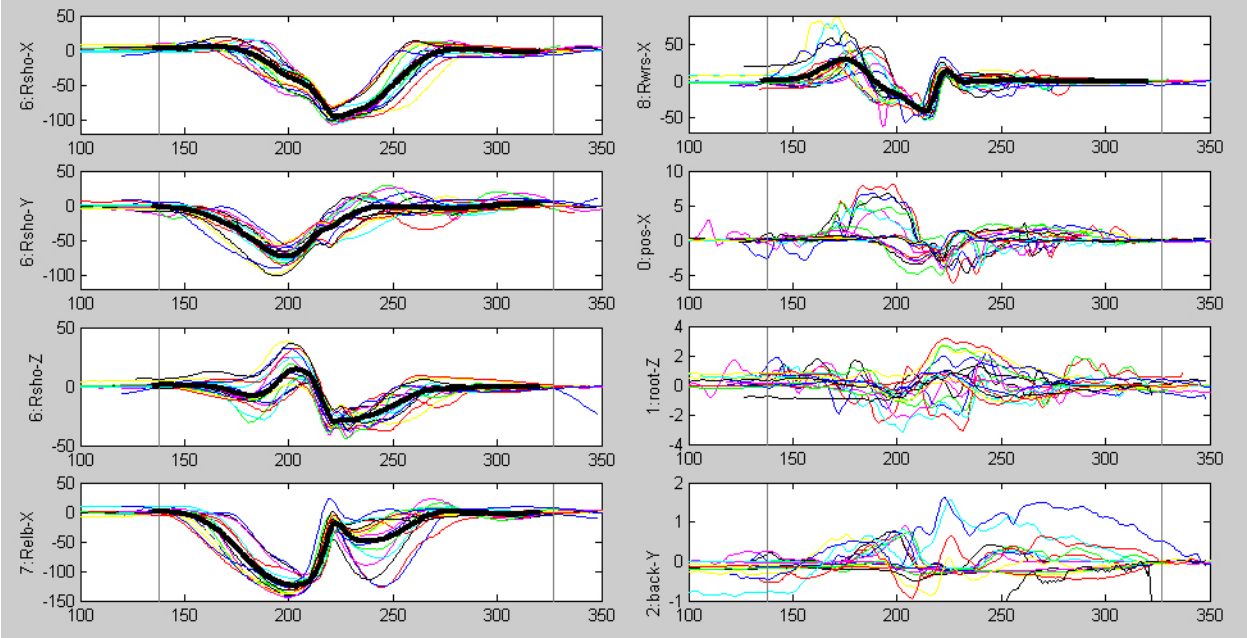

Figure-17 Averaging Trajectories. Difference trajectories of one subject are shown for all three bases. Some trajectories can be averaged across bases (thick line) when they are similar enough, the ones shown here are right shoulder (xyz), right elbow (x), and right wrist (x). Others may be too different, the ones shown here are position (x), root (z), and back (y). Vertical axis denotes degrees, horizontal axis denotes frames. The two vertical lines mark the beginning and ending of the throw-period.

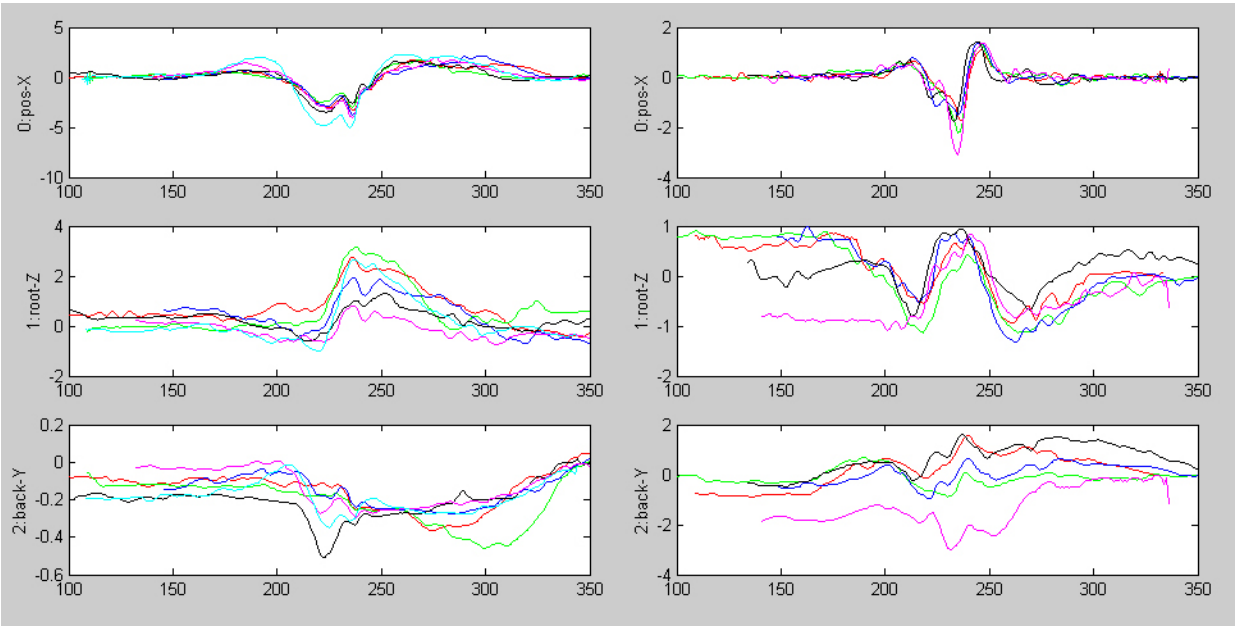

Figure-18 Similarities within a base. Difference Trajectories of position (x), root (z), and back (y) for one subject are shown. **Left:** for standing base. **Right:** for sitting base.

Since the changes of the trajectories of the throwing arm were very similar during all three base motions (**b**=3) only their average was stored (**a**=3) (this is true for the data of one subject, different subjects have different styles and their data can not be averaged, see the next section). On the other hand, the trajectories of the root, back, and position were too different and needed to be stored individually for walking, standing, and sitting (see Figure-17). Note that for each base on its own the trajectories are similar as shown in Figure-18. It would be possible to average each base's samples separately for inclusion in the CPMC, but it was chosen to select one sample from each base as a representative for it and include only that sample in the CPMC.

If a complete motion had been used then 15*3=45 trajectories would have been needed to be stored. But using the CPMC the total number of trajectories stored was only 3+(6-3)*3=12 according to (Equation-3) in section 5.2.5, leading to a total reduction of 33 trajectories according to (Equation-4) also in section 5.2.5. Furthermore, only 6 trajectories were used at a time, but the total number of trajectories that were affected in the resulting motion was 15, since 9 trajectories were estimated: the non-throwing arm, and both legs (**e**=9). In terms of DOFs, the number of DOFs used at a time was 14, the number of estimated DOFs was 15, and hence the number of affected DOFs was 29.

6.1.6 A Three Subject Comparison

Figure-20 depicts some difference trajectories while walking and while standing of the right arm (throwing arm) of three subjects. From there, it is visible that the difference trajectories are similar across bases for the same subject but different between subjects. That is most apparent in the right image of the figure, where it is very clear that the trajectories drawn with the same line style are similar, which indicates similarity within a subject. This is due to the fact that every person has their distinct throwing style, which is even more apparent in the samples used in this work since the only instructions specific to the throw where that it had to be an overhand throw. For more specific throwing motions like pitching a baseball, the differences between subjects are probably less obvious because to pitch a baseball correctly one has to follow very precise movement sequences otherwise it is not going to be a correct pitch. Due to the variation between subjects, only the data of one subject was used in the final CPMC, that of subject 1. If a different style of throwing is desired, another CPMC can be created by using the motion data from another subject with that specific style. For CPMCs for professional throwing

styles it probably would be possible to average all persons' data or the data may be so similar that one single sample could be used as a representative for the whole collection.

An important thing to notice though is that the same equations for the computation of other DOFs can be used since the data of all subjects was used during analysis. Initially relationships were found by focusing on the data of one subject and then they were checked to make sure they hold for other subjects as well. This way the equations were generalized to all bases and to all subjects.

Figure-21, and Figure-22, compare original difference trajectories to computed ones of the non-throwing arm for the three subjects for standing and walking respectively. All are computed using the same script, see APPENDIX A. And Figure-23 compares original difference trajectories to computed ones of the legs for the three subjects while walking. All are computed using the same script, see APPENDIX B. Even though, some trajectories look somewhat different the resulting motions when using the computed trajectories and adding them to other persons' bases look believable. The explanation for this occurrence is that each component of the orientation is displayed separately therefore, one can not know the difference in the actual rotation when all three components are combined into an Euler angle. And therefore, the differences in the plots are often not visible in the playback, or not identified as unnatural movements. This has been tested by extracting throws as partial motions – without creating a CPMC – from all subjects and editing other subjects' base motions. The same procedures for editing as if a CPMC had been used were used. See Figure-19 for one example.

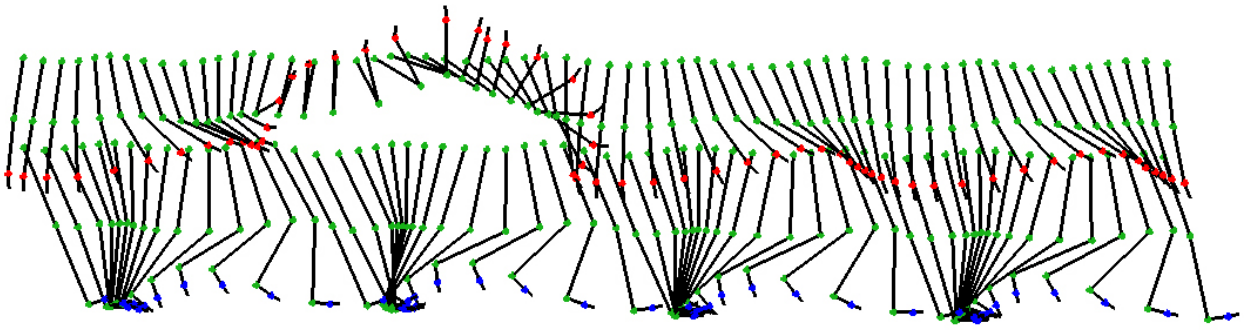

Figure-19 A generated 'walk&throw' for subject 1 using the partial throwing motion extracted from subject 2 and estimating the left arm and legs using the scripts.

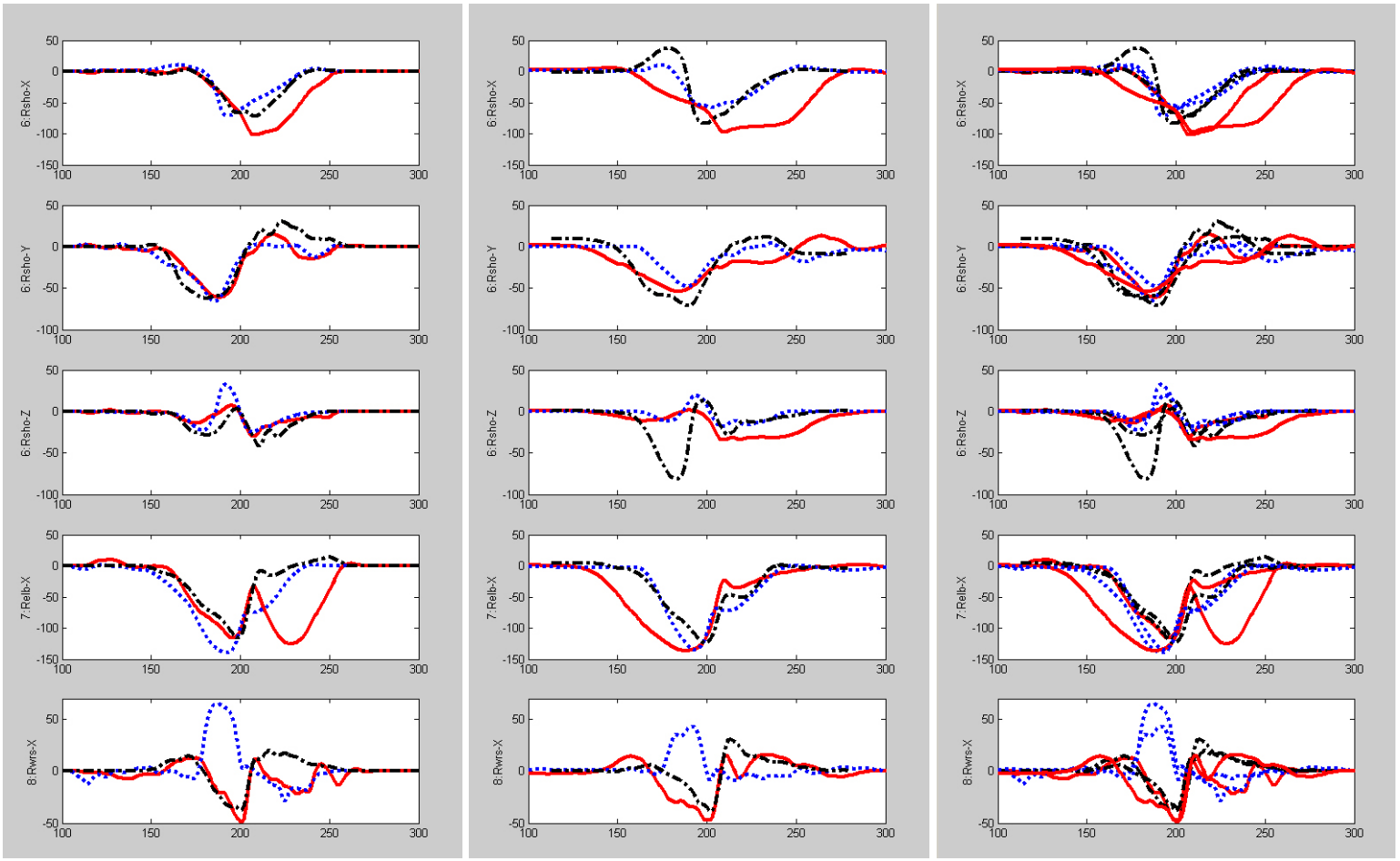

Figure-20 Difference Trajectories of 3 subjects. Vertical axis denotes degrees, horizontal axis denotes frames. Solid line represents the data of subject 1, dotted line data of subject 2, and dashed line data of subject 3. **Left:** Differences while walking. **Middle:** Differences while standing. **Right:** Both differences.

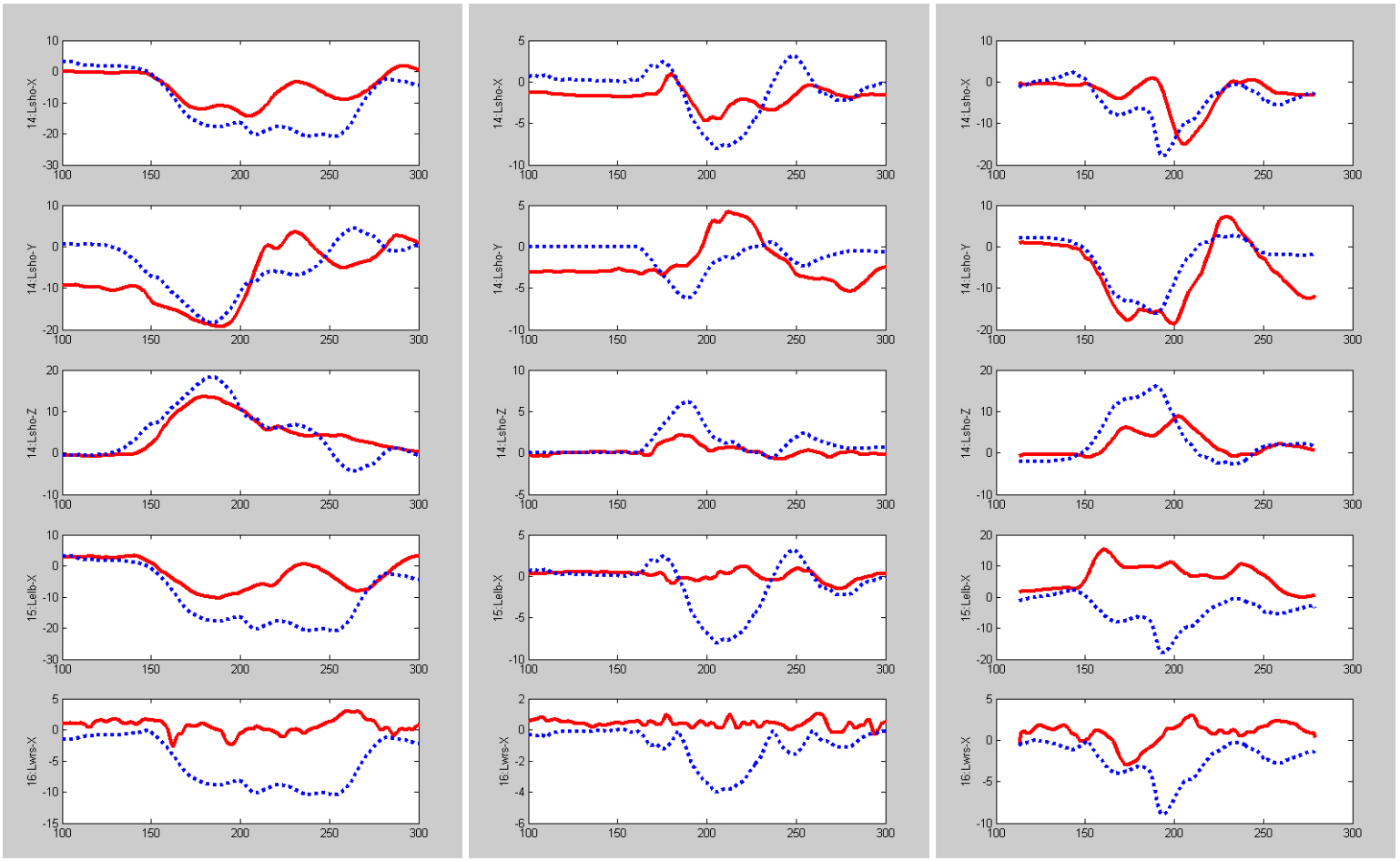

Figure-21 Original vs. computed difference trajectories of the *non-throwing arm* while *standing* for 3 subjects. Originals are in solid lines and computed ones are in dotted lines. Vertical axis denotes degrees, horizontal axis denotes frames. **Left:** first subject. **Middle:** second subject. **Right:** third subject.

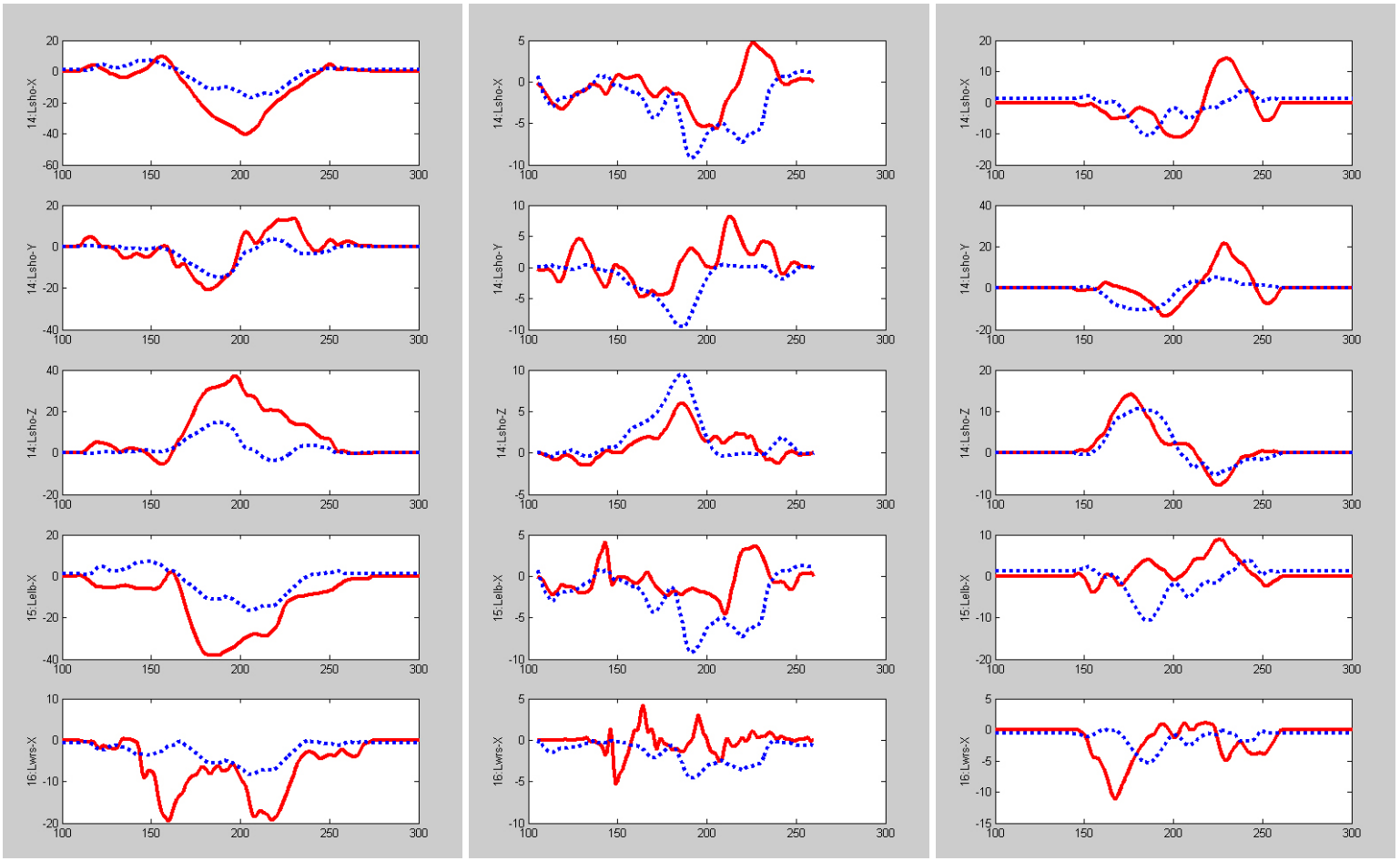

Figure-22 Original vs. computed difference trajectories of the *non-throwing arm* while *walking* for 3 subjects. Originals are in solid lines and computed ones are in dotted lines. Vertical axis denotes degrees, horizontal axis denotes frames. **Left:** first subject. **Middle:** second subject. **Right:** third subject.

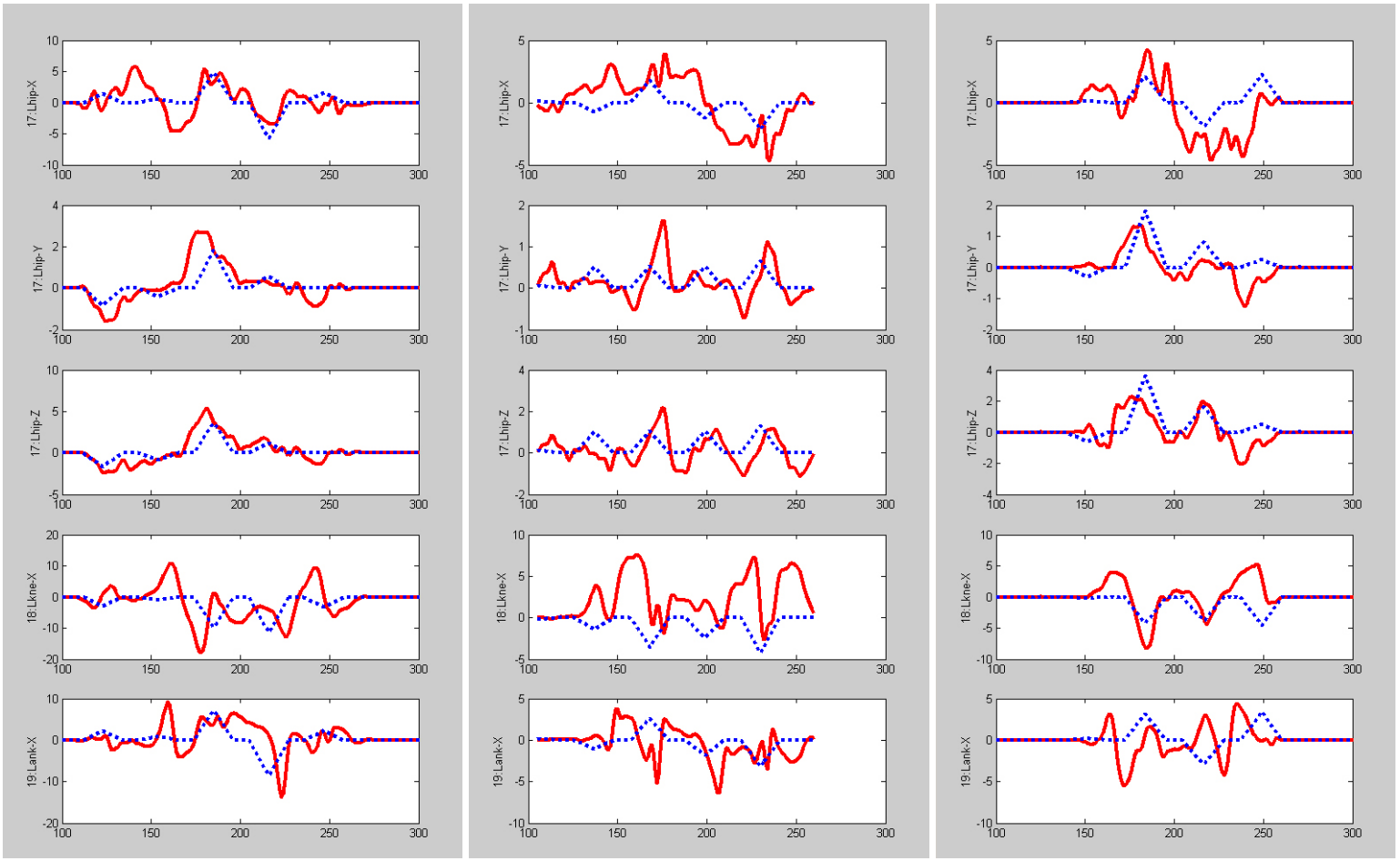

Figure-23 Original vs. computed difference trajectories of the *left leg* while *walking* for 3 subjects. Originals are in solid lines and computed ones are in dotted lines. Vertical axis denotes degrees, horizontal axis denotes frames. **Left:** first subject. **Middle:** second subject. **Right:** third subject.

6.2 Using a CPMC for 'throwing'

The CPMC for 'throwing' that was created as explained in the previous section was then used to incorporate a throwing action to other subject's walking, standing, and sitting base motions. The base walking motions differed in their cycle length and style. To do the editing the algorithm shown in Figure-13 given in section 5.3 was used as follows.

When the base motion represented a sitting action, only the averaged throwing arm trajectories and the root, back, and position trajectories specific to sitting were retrieved from the CPMC. No manipulations were necessary, nor were there any additional trajectories to be computed. Therefore, the retrieved trajectories were added as they were, using vector addition, to their corresponding trajectories in the sitting base motion. The non-throwing arm and the legs were fixed in place using inverse kinematics at absolute locations in world space. In reality, the non-throwing arm may actually be fixed in world space as if holding on to the chair's arm, or it may be fixed to a position relative to the hip joint in the case where the hand rests on the thigh. In the latter case the absolute position of the hand will change when the thighs move, which may be the case with extreme forward trunk motion, when the subject may lift from his/her chair. The results for two generated 'sit&throw' motions of two subjects are shown in Figure-24. The results of applying IK to fix the feet are very clear and good. The non-throwing arm though, still seems to be floating around a little bit in the lower right image but is fixed in the upper right image.

When the base motion represented a standing action the non-throwing arm would be computed using the equations that resulted from the analysis and that were stored within the CPMC. Hence, the appropriate script was retrieved as well as the averaged throwing arm trajectories and the root, back, and position trajectories specific to standing. No extra manipulation was necessary. Then the script could be applied to estimate the non-throwing arm trajectories. The resulting trajectories and the retrieved trajectories were added to their corresponding trajectories in the standing base motion. The legs, like in sitting, could be fixed with inverse kinematics. The results for two subjects are shown in Figure-25.

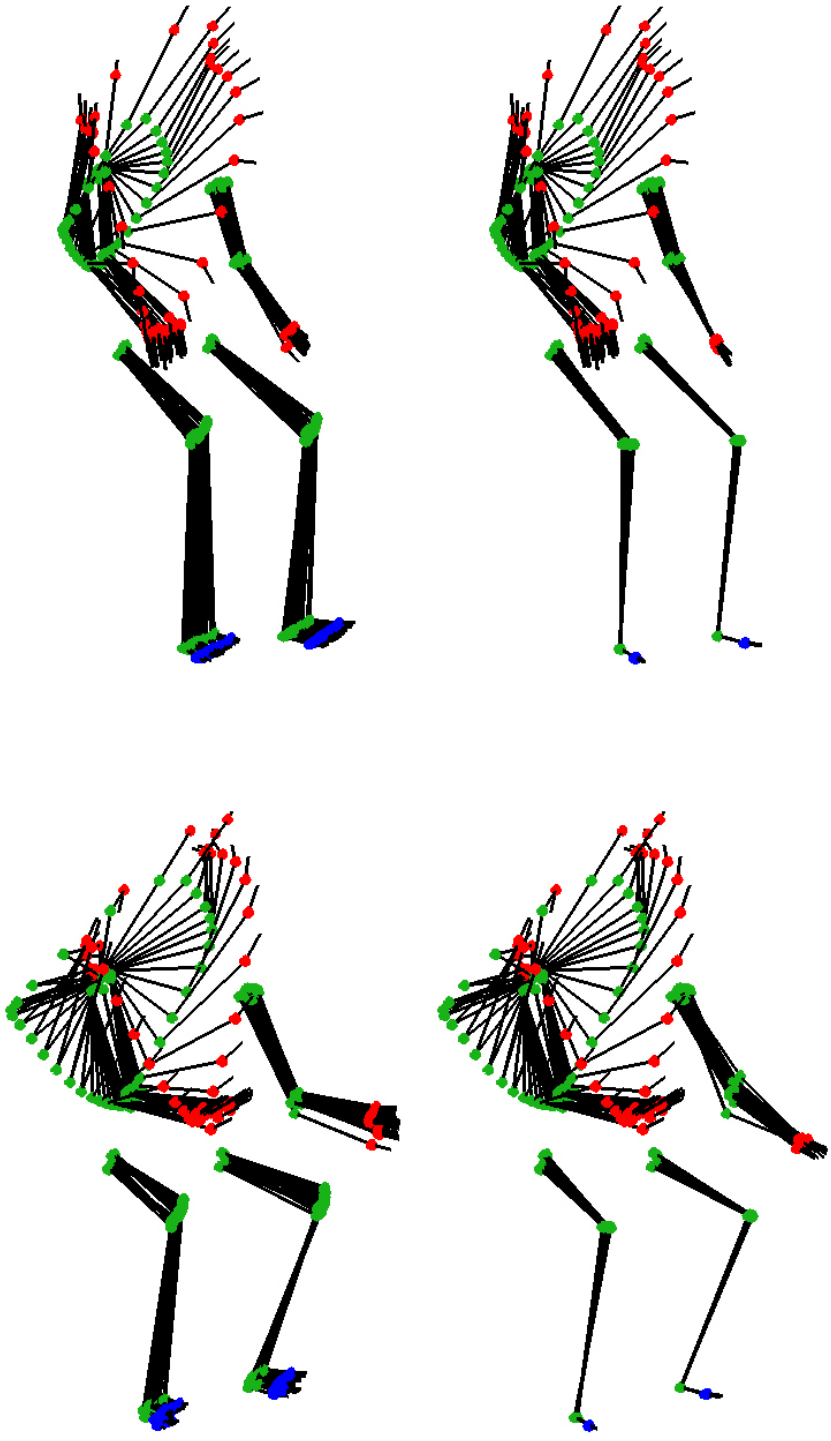

Figure-24 Generated 'sit&throw' motion for two subjects. Multiple exposures with every 4th frame shown. **Left:** result without applying IK to fix feet and non-throwing arm. **Right:** result after applying IK.

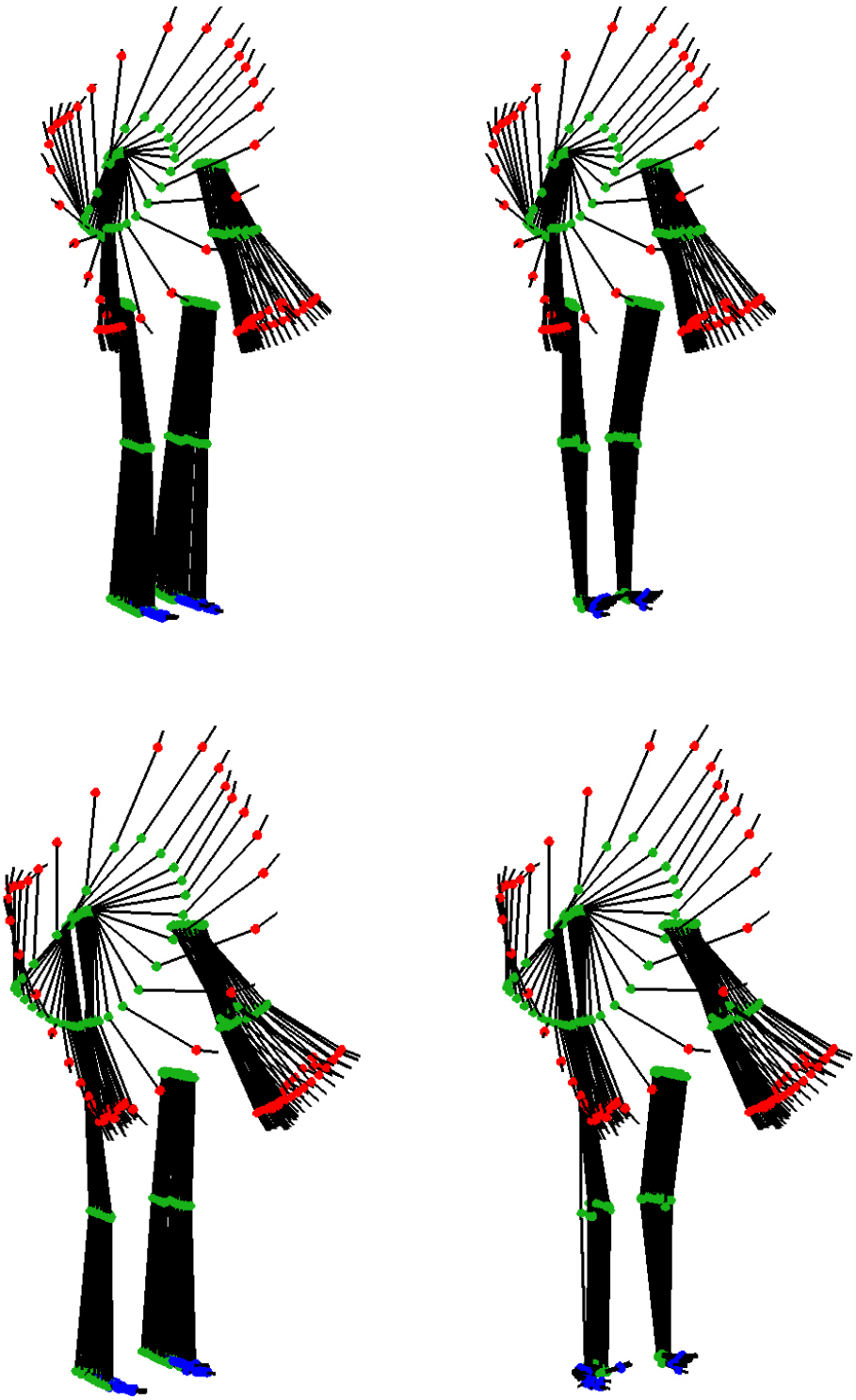

Figure-25 Generated ‘stand&throw’ motions for two subjects. Multiple exposures with every 4th frame shown. **Left:** result without applying IK to fix feet. **Right:** result after applying IK, which did not work as expected.

The inverse kinematics routine used was IKAN: Inverse Kinematics using ANalytical methods. It was developed at the Center for Human Modeling and Simulation at the University of Pennsylvania [TOL00]. It did not work as expected for the standing case. The main cause for this is probably that in the case of standing, the leg is almost straight, and the routine depends on the reference triangle formed by the hip, knee, and ankle joints. A better understanding of the IKAN routines may be needed for improved results. Another cause is that in the work presented here, no joint limits were imposed. Therefore, the knee may bend a bit backward because of inaccuracies in the calibration of the markers and inaccuracies in matching the articulated figure's rest angles to the actual subject. But this problem can be overcome by using other IK routines. It would also be more advantageous to use a routine where one can fix the ball of the foot instead of the heel.

The third and last possibility was that the base motion represented a walking action. The first step was to read the header and find out what was needed. In this case, the cycle length of the base (**CycleLength**$_{\mathbf{Base}}$) and the location of a left heel strike in the base were needed. The header should contain information about the length of the throw-period in terms of number of heel strikes (**NumHeelStrikes**) and with which heel strike it starts (left or right) as well as the cycle length that it applies to (**CycleLength**$_{\mathbf{CPMC}}$). Then the scripts for estimating the non-throwing arm as well as for the legs were retrieved. Also the averaged throwing arm trajectories and the root, back, and position trajectories specific to walking were retrieved.

Using the information extracted from the header, a throw-period with the appropriate number of heel strikes and starting with the correct one was identified in the base. This period was resampled to slow it down to about 80-90% of the original speed. Heel strike locations were taken to be at the point where the knee is extended the most, which corresponds to a minimum in the x-rotation of the knee joint. Assuming that an appropriate heel strike occured at frame $\mathbf{f_b}$, this would mark the beginning of the throw-period. The ending of the throw period would be $\mathbf{f_e}$, which would simply be taken as the closest heel strike to frame $\mathbf{f_b}$ + [(**NumHeelStrikes**-1) * **CycleLength**$_{\mathbf{Base}}$ / 2]. Hence, the sequence of frames from $\mathbf{f_b}$ to $\mathbf{f_e}$ is the throw-period which was resampled.

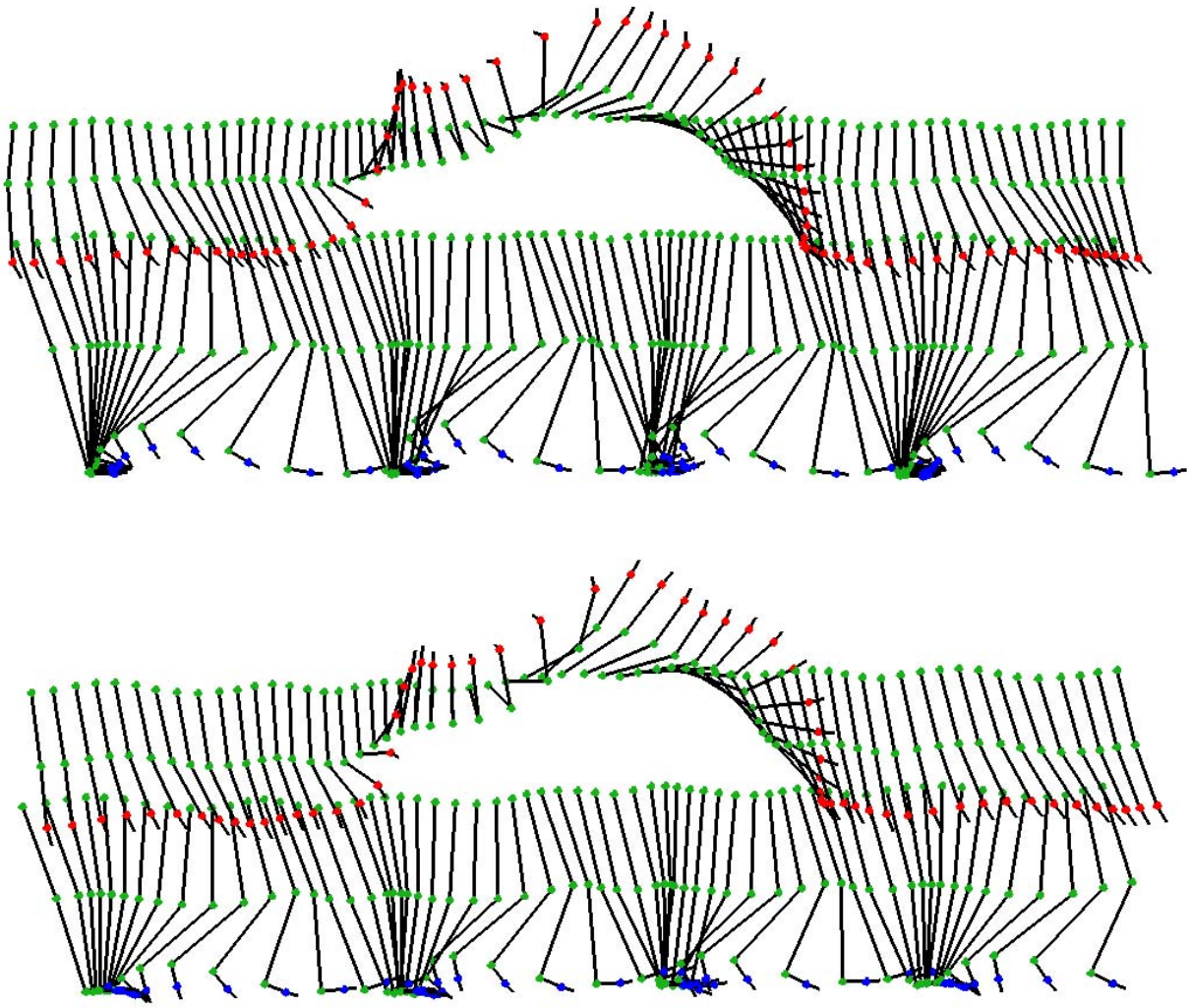

Figure-26 Generated 'walk&throw' motions for two subjects. Multiple exposures of the right arm and leg with every 4th frame shown.

Furthermore, the amount of slow-down was related to the peak velocity of the elbow under the assumption that it is related to how vigorous the throw is. This is the only time when a derivative had to be computed. As mentioned earlier, the derivatives of the difference trajectories were taken into consideration during analysis, but they ended up not being used in any of the extracted relationships. This is good, in the sense that it eliminated the need for computing derivatives.

After resampling the throw-period in the base, the averaged throwing-arm data and the root, back, and position data specific to walking was resampled to match the length of the

slowed-down throw-period in the base if necessary. Matching $\mathbf{CycleLength_{CPMC}}$ to $\mathbf{CycleLength_{Base}}$ is another way of describing it. Then, the trajectories for the non-throwing arm were computed as well as for the legs. Finally, all the resampled trajectories and the computed ones were added to the corresponding trajectories in the base motion.

The resulting 'walk&throw' motions for two subjects are shown in Figure-26. The subject in the top row had a walking cycle of 64 frames hence only the throw-period in the base motion had to be resampled, but the CPMC data did not have to be resampled, it was already matching. On the other hand, the subject in the bottom row had a walking cycle of only 59 frames and hence the CPMC data had to be resampled as well as the throw-period in the base. In reality, one may speed up during the throw-period, as was the case for one of the subjects (subject 2). It turned out that the relation to peak velocity of the elbow as used in the scripts included in the CPMC worked for that case as well, it can be seen in Figure-19.

The creation and usage of a CPMC has been demonstrated in detail for the special case of overhand throwing. The relationships found during analysis applied to all bases and all subjects. Furthermore, the resulting motions looked very natural and were often mistaken for originally captured motions as will be shown in the following chapter.

CHAPTER 7
RESULTS

In this chapter, the motions obtained from editing base motions using the CPMC for throwing as explained in the previous chapter will be examined. Also the results of applying the method to a different kind of throwing – tossing – will be shown. Other approaches that have been tried to find relationships between the upper and lower body motion will also be stated.

7.1 Verification of Results

The motions resulting from the editing method described in CHAPTER 5, as tested in the experiment in CHAPTER 6, are very encouraging. It is to be noted that no limits were put on the joints; which may have improved the results further.

To assess the quality of the generated motions they were compared to originally captured motions in two formats: plots of trajectories, and playback of motions. A dilemma that is faced is that since the throw is in a specific style, the style of the person it is captured from, comparing an edited motion of another person in terms of its trajectories to an originally captured motion of that person may not be the best measure, because the changes induced in other parts of the body depend exclusively on the motion stored in the CPMC. But still they could be used to see if the trajectories were reasonably smooth, did not have any jumps, and that they adhered to the overall shape of the original curves. To get a better picture, comparisons were also made between the resulting edited motions and the equivalent motions of the subject whose data was used in the creation of the CPMC (subject 1). Furthermore, since the motions that involve walking differed in their cycle length, during comparison the original motions were uniformly resampled to match the cycle length of the generated motion. Figure-27 shows such a comparison for a generated ‘walk&throw’ motion for subject 3. The trajectories of the right arm and leg are compared to original but resampled trajectories of subject 3, as well as original but resampled trajectories of subject 1 whose data was used in the creation of the CPMC.

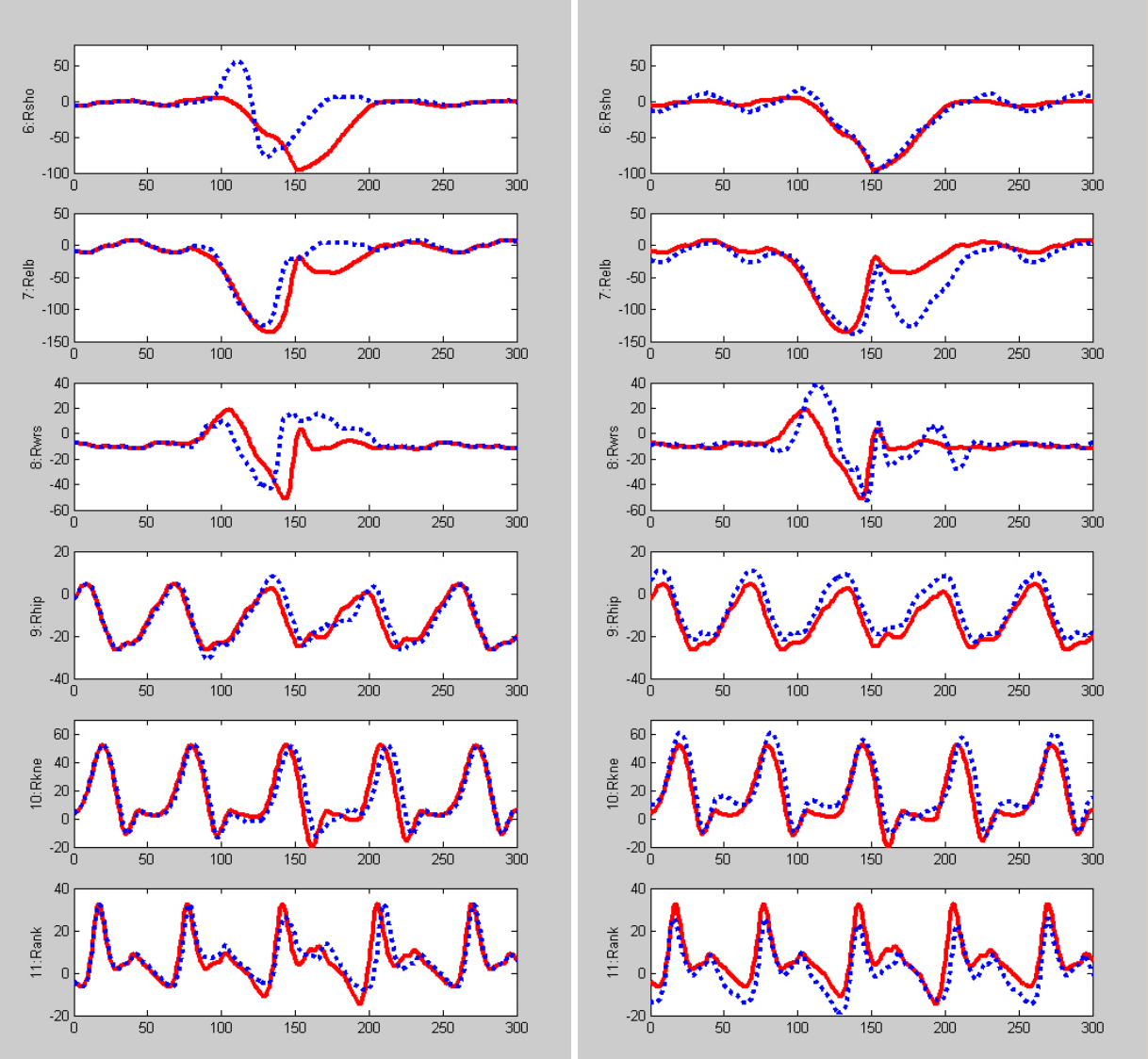

Figure-27 Comparison of selected generated 'walk&throw' trajectories for subject 3 (solid) **Left:** to an original 'walk&throw' of subject 3 (dotted). **Right:** to an original 'walk&throw' of subject 1 (dotted).

As mentioned previously, sometimes the trajectories may look very different, but since each component of the orientation is displayed separately it is difficult to estimate the difference in the actual rotation when all three components are combined into an Euler angle. Also, often the differences in the plots are not visible in the playback, or are not identified as unnatural movements. Therefore, the motions were also compared by observing their playback using the Pose File Player.

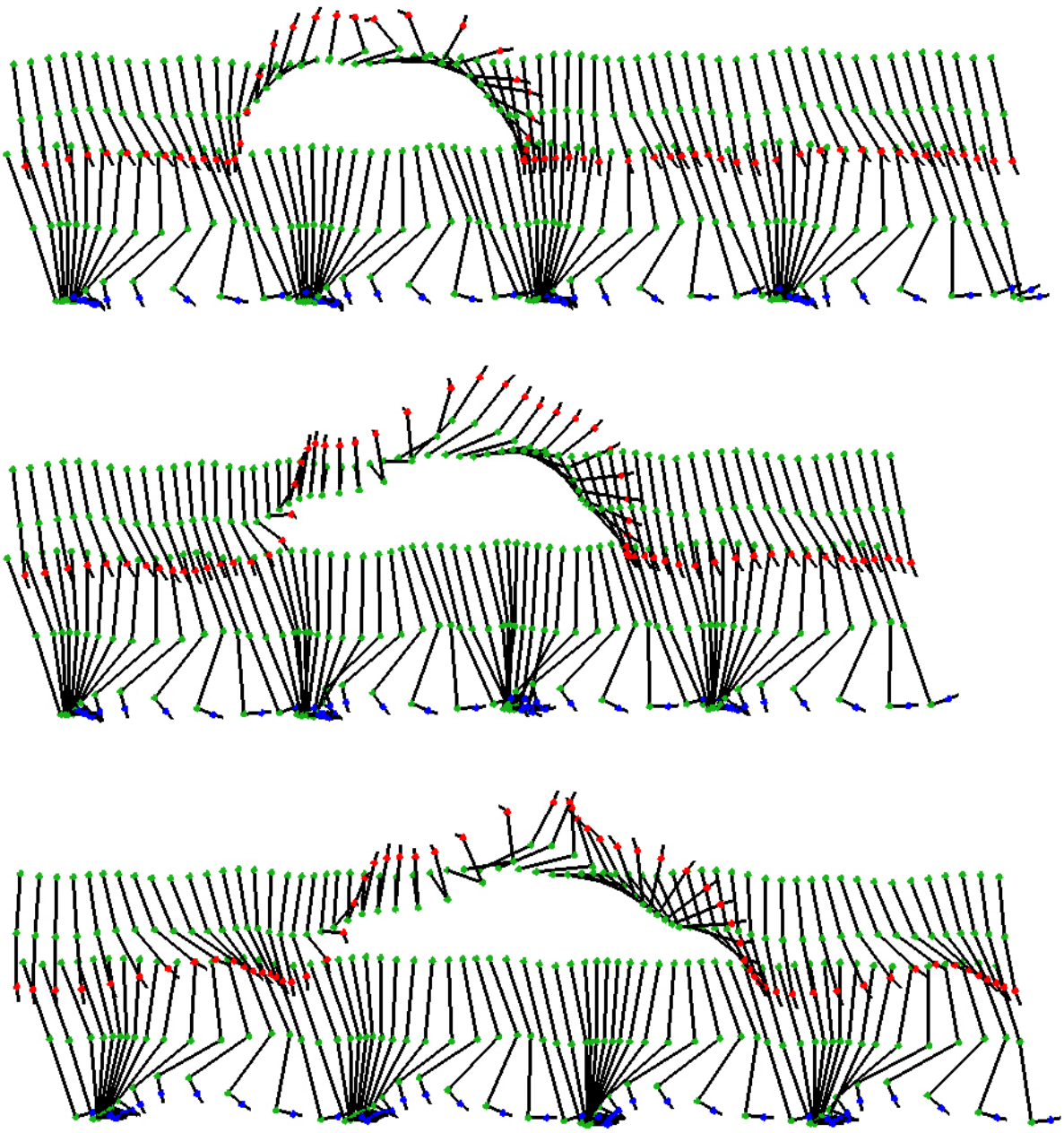

Figure-28 Comparison of generated 'walk&throw' for subject 3 to two original motions. Multiple exposures of the right arm and leg with every 4th frame shown over a range of 250 frames. **Top:** An original 'walk&throw' of subject 3. **Middle**: Generated 'walk&throw' for subject 3. **Bottom:** An original 'walk&throw' of subject 1.

A problem here is that judging the naturalness of a motion by looking at the playback may be subjective. To overcome this problem, original and edited motions were played simultaneously. Several observers were asked to pinpoint the original motion from the edited ones and they could not. No formal user study was conducted. Figure-28 shows how such a

comparison for subject 3 would look like if every 4th frame were displayed. It depicts the same motions as in Figure-27, except that the original motions are unchanged, i.e. not resampled.

A problem that was encountered during editing was the slipping of the feet, but this is a minor problem that all motion editing methods suffer from with the exception of methods that explicitly put constraints on the foot locations. The problem is usually corrected during post-processing. One could use such a method as that of Kovar et al [Kov02b] for the walking case. In the work presented here, only during standing and sitting were inverse kinematics methods used. In particular, IKAN was used to fix the feet on the floor as show previously in section 6.2, in Figure-24 and Figure-25.

Actually, the top portion of Figure-25 represents the generated 'stand&throw' motion of subject 2. This data is compared to an original 'stand&throw' of subject 2 and to an original 'stand&throw' of subject 1 in Figure-29 and Figure-30. It is very clear that the personal style of subject 2 is different from that of subject 1. The throwing action extended over a shorter period of time. Also there was not much rotation of the trunk; therefore, the left arm did not swing as much as in the throwing style of subject 1 (from the CPMC). The left two columns of Figure-29 show a frame sequence which illustrates the generated 'stand&throw' of subject 2 without applying IK to fix the feet on the floor. That is visible in the last few frames where the trunk bends forward and the legs slide to the back since without IK no orientations for the legs are computed. As mentioned earlier, the IK routine used did not work as expected. For some reason only the x- and z-components of the hips' rotation are accurately computed, but the y-component is not. This is visible in Figure-30 where the trajectories shown are after applying IK.

In Figure-31 and Figure-32 the generated 'sit&throw' motion of subject 4 is compared to original motions. The same motion has been depicted in the lower portion of Figure-24. As mentioned previously the non-throwing arm seems to be floating a bit. Looking now at some of the trajectories in Figure-32 one can see visible spikes in the shoulder and elbow trajectories drawn in solid line style as a result of the inverse kinematics routine used. It seems to be a gimbal lock problem.

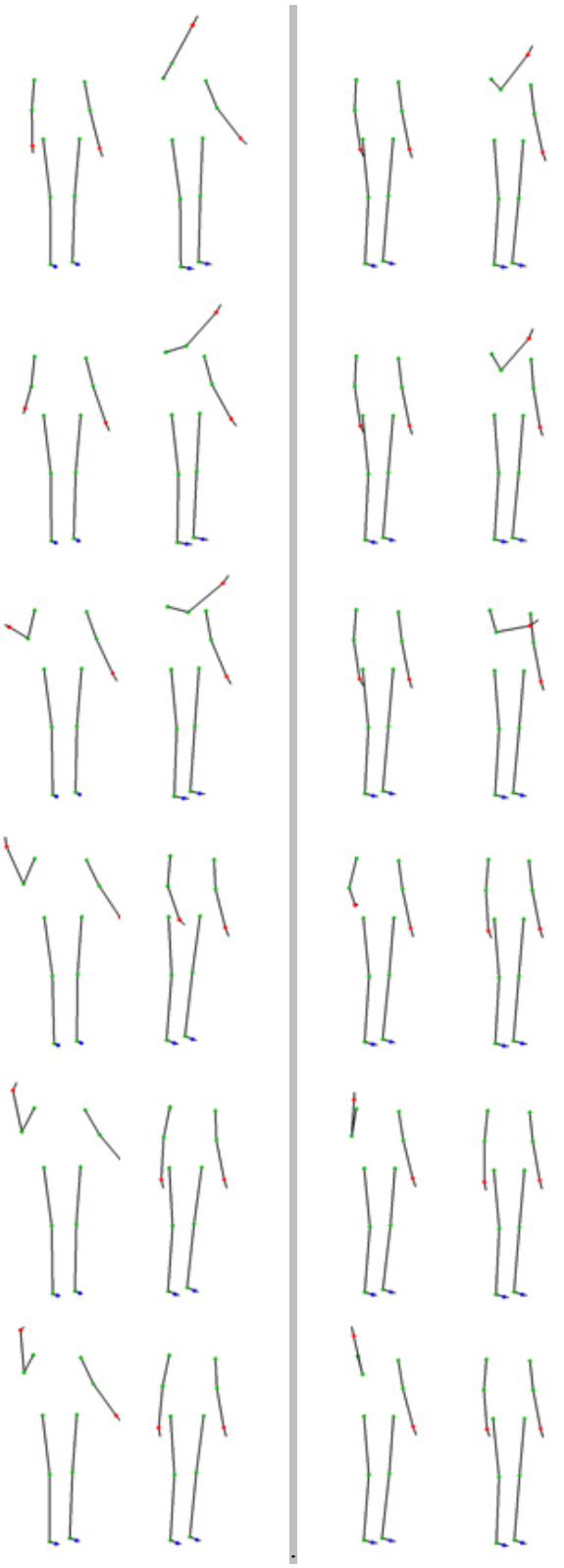

Figure-29 Comparison of a generated (left 2 columns) and original (right 2 columns) 'stand&throw' for subject 2. Without IK. Every 10th frame is shown.

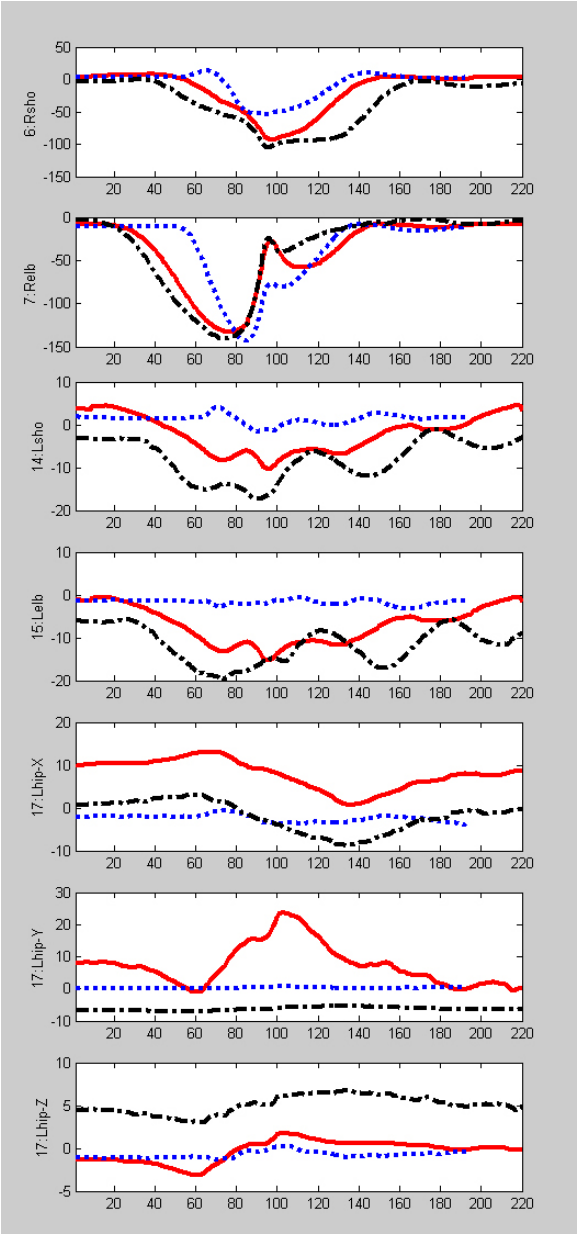

Figure-30 Comparison of selected generated 'stand&throw' trajectories for subject 2 after applying IK (solid) to an original 'stand&throw' of subject 2 (dotted) and to an original 'stand&throw' of subject 1 (dashed). The trajectories shown are: the x-components of the right shoulder and elbow, the x-components of the left shoulder and elbow, and the xyz-components of the left hip.

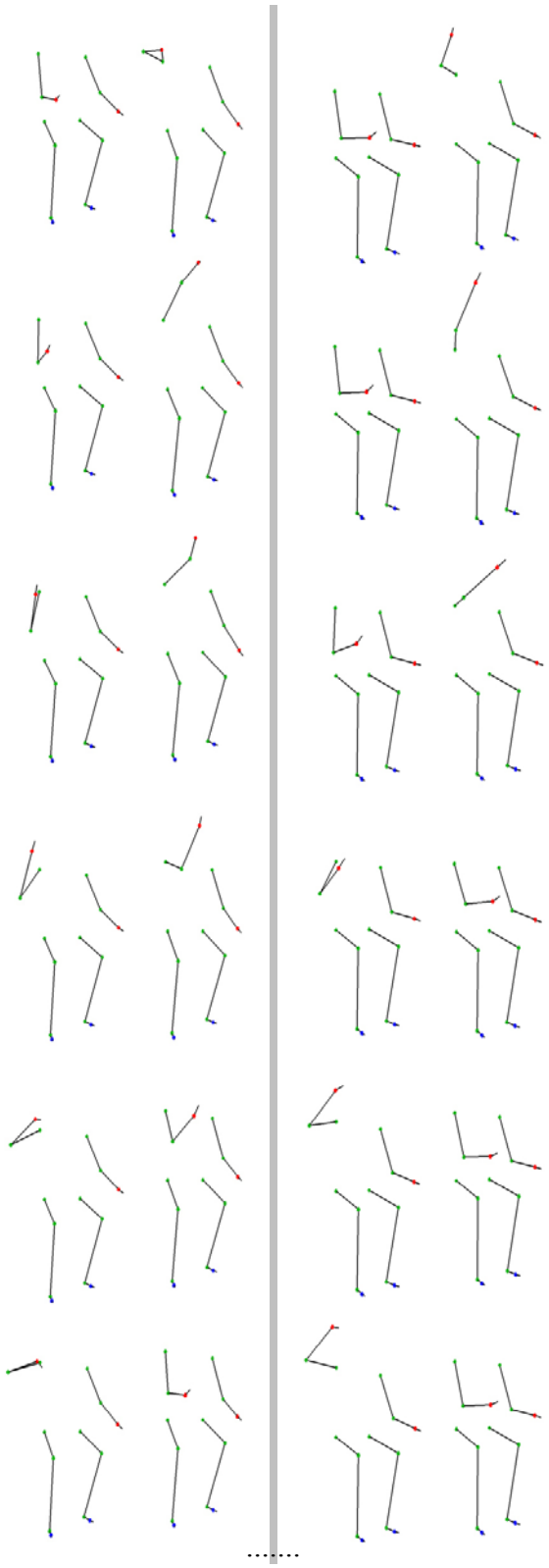

Figure-31 Comparison of a generated (left 2 columns) and original (right 2 columns) 'stit&throw' for subject 4. With IK. Every 10th frame is shown.

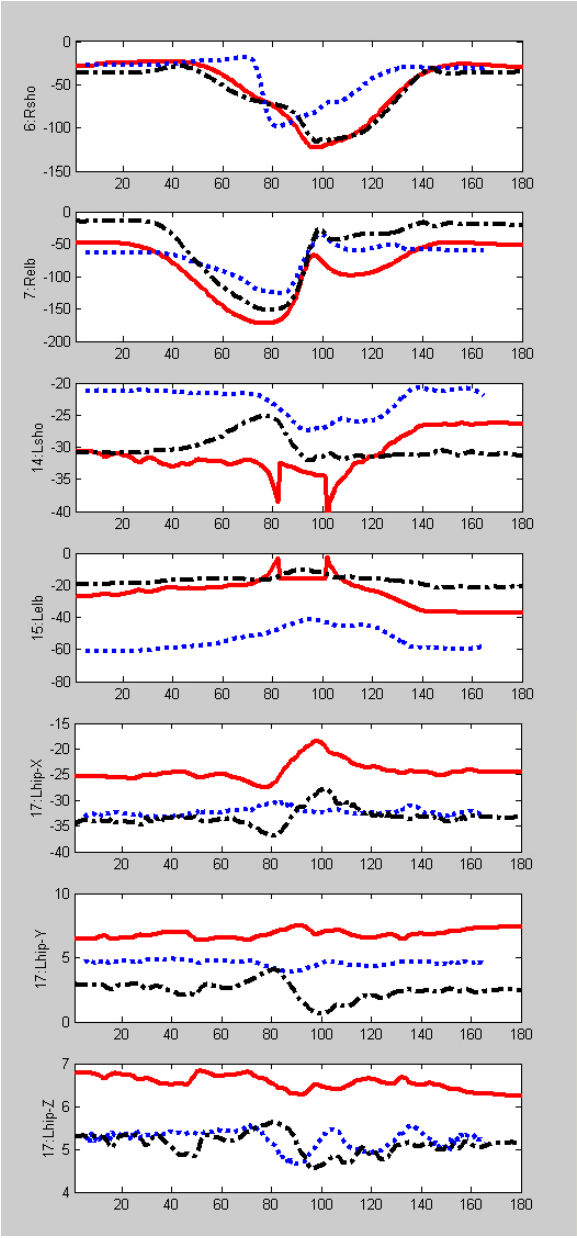

Figure-32 Comparison of selected generated 'sit&throw' trajectories for subject 4 after applying IK (solid) to an original 'sit&throw' of subject 4 (dotted) and to an original 'sit&throw' of subject 1 (dashed). The trajectories shown are: the x-components of the right shoulder and elbow, the x-components of the left shoulder and elbow, and the xyz-components of the left hip.

Having said that, overall the motions played back smoothly, except for the left arm in some 'sit&throw' motions cause by spikes like the ones just mentioned. But these spikes are due to the IK routine, not due to the CPMC editing method, and can probably be eliminated by using a different IK routine, or by investigating IKAN more and fine tuning the code. Similar jumps were visible in the legs during standing for the same reason.

7.2 Applying the Method to 'Tossing'

The same 6 subjects were asked to toss the ball as if they were tossing and empty soda can into a trash can. They were asked to perform the tossing while walking, while standing, and while sitting. The captured data was cleaned up, converted to root location and joint orientations, and manipulated the same way the throwing data was handled. Then the tossing motion was extracted by differencing. Instead of analyzing the data to obtain new relationships, the same equations as for throwing were put to the test. This decision was made based on the fact that tossing is another form of throwing. This decision was also made with the hope that the relationships can be generalized even more so that it works for all kinds of throws.

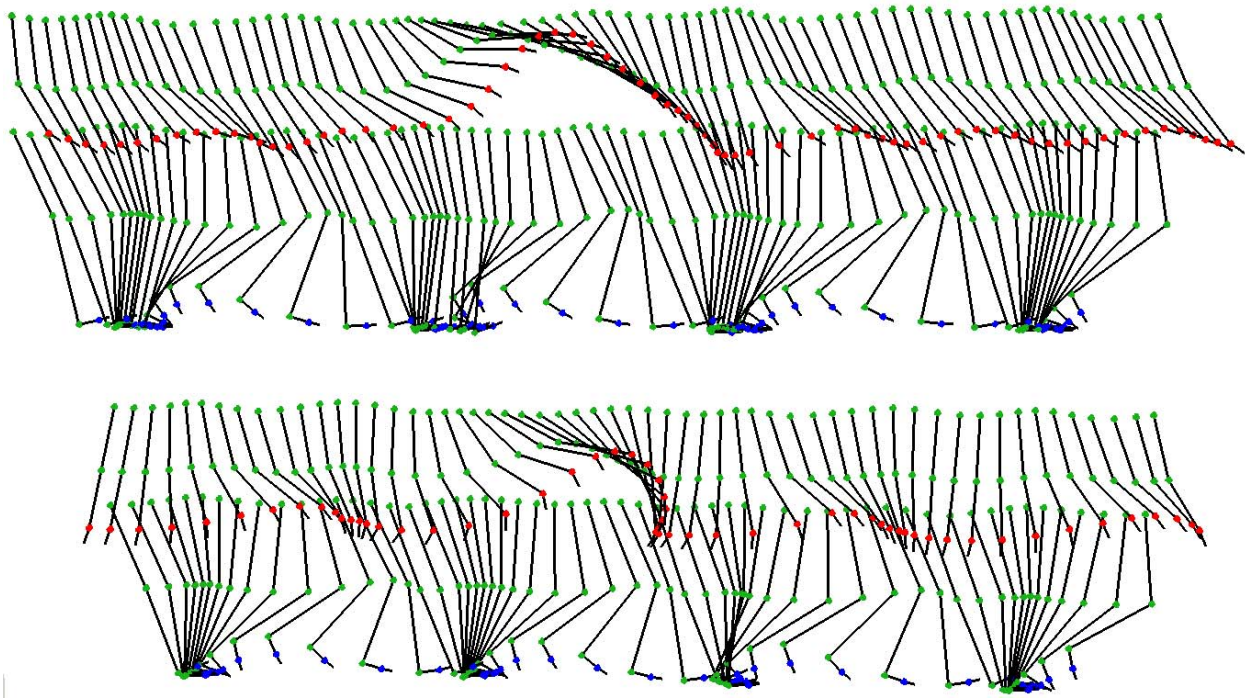

Figure-33 Comparing a generated tossing while walking motion for subject 4 (**top**) to an original tossing while walking for subject 1 (**bottom**)

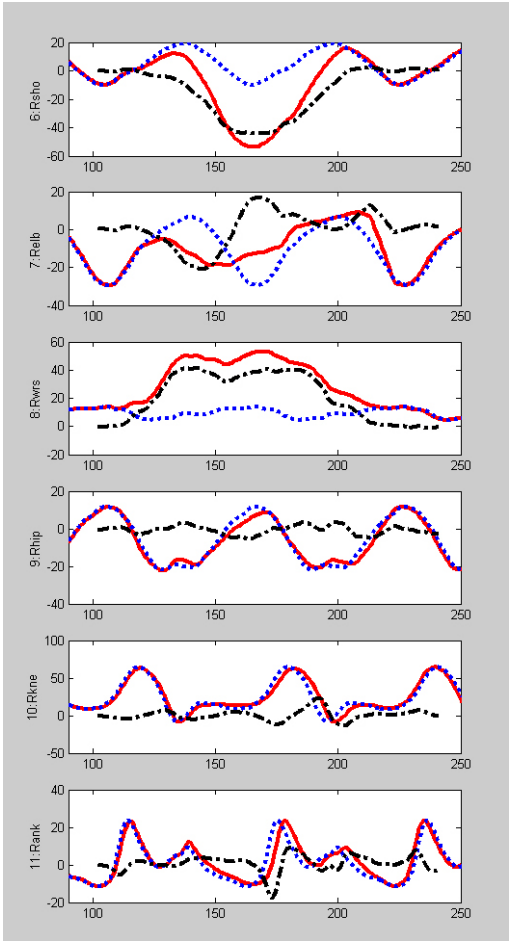

Figure-34 Tossing while walking (solid), walking (dotted), and their difference (dashed) of subject 1.

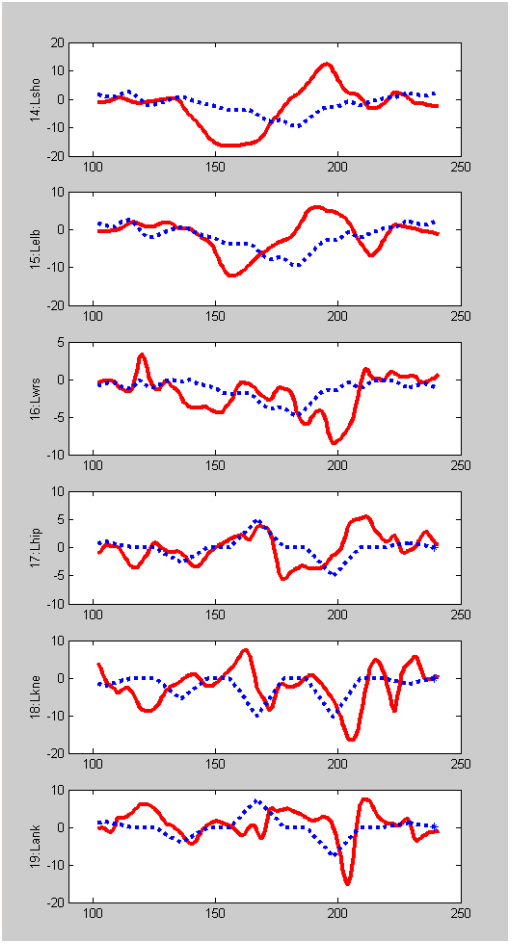

Figure-35 Original toss difference trajectories (solid) and computed (dotted) ones using the same scripts as for throwing.

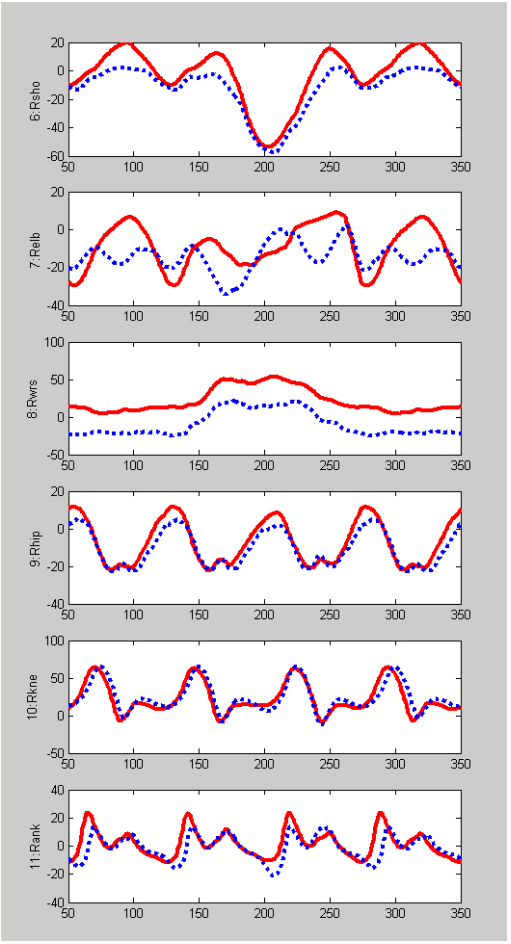

Figure-36 Generated 'walk&toss' for subject 4 (solid) compared to and original 'walk&toss' of subject 1 (dotted).

From the result, depicted in Figure-33 through Figure-36, it is seen that the scripts did indeed work for the tossing motions. This indicates that for similar partial motions the relationship to other body parts is similar. But it is surmised that for radically different actions, e.g., for boxing, the relationships are probably different. Still, since human motion is highly correlated one may be able to find very general relationships between the different DOFs of the articulated figure. This is especially true since one can think of the changes that occur in other DOFs merely as reactions to changes in some specific DOFs. It may not be important to know what action is being performed when some joint changes from one orientation to another in order to be able to estimate what will happen in other joints that are not directly involved in the action being performed. It may be more important to know how fast such a change occurred instead.

7.3 Exaggeration

It is believed that increasing the amount of trunk rotation would make the throwing motion look more energetic. To test this, the y-component of the difference trajectory of the back joint was scaled. It was increased 25%. Then the non-throwing arm and the leg trajectories were computed and added to the base. The result looked somewhat believable (see Figure-37), except that the left arm did bend backwards. This could be prevented if proper joint limits were enforced. Increasing the rotation more than 25% led to an abrupt spinning of the left arm about the shoulder joint.

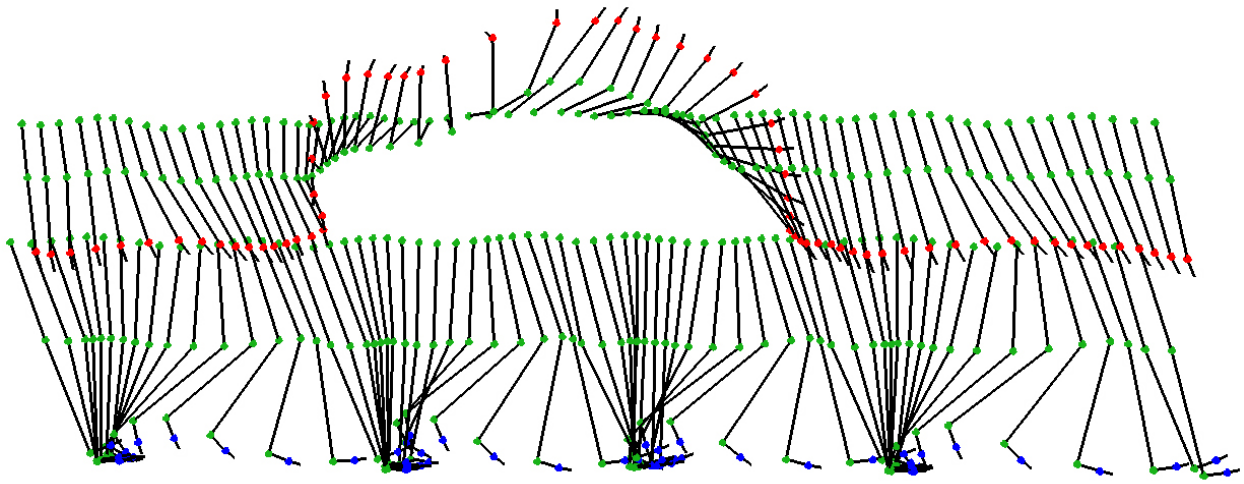

Figure-37 Exaggeration in the form of increased trunk rotation.

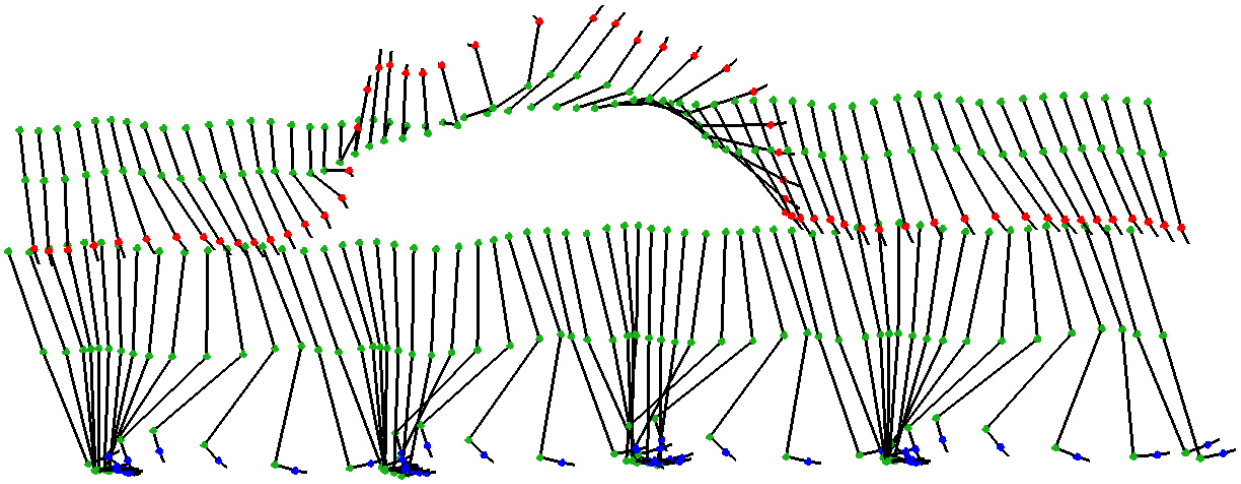

Figure-38 Exaggeration in the form of varying the resampling rate.

Another variable that can be changed to allow a wider variety of generated throws is the resampling factor for the throw-period. Figure-38 shows the result of speeding up the throw period instead of slowing it down for subject 3.

7.4 Other Experiments

It was attempted to compare the joint trajectories in the spatial domain. That is, by computing the joint locations in world space as they change over time. This conversion, though simple, since only the figure's hierarchy has to be traversed and the corresponding translations and rotations have to be applied, is computationally expensive due to the many needed matrix multiplications. Furthermore, any relationships that would be discovered in the spatial domain would be transferable to the rotational domain, i.e. can be found in the rotational domain. The spatial data was studied briefly but nothing noteworthy stood out. Consequently, this conversion was deemed not worth the overhead.

Another attempt was made to extract relationships between the shoulder line (the line connecting the right and left shoulder joints) and the hip line (the line connecting the right and left hip joints). It was thought that an examination of the relative rotations of those lines about the y-axis and the z-axis as well as with reference to the world coordinate system would be fruitful, but it was not. More importantly, it was not clear how to convert a relationship, if found, to the rotational domain of the joints. If such a conversion was not possible or too complicated it

would affect the whole editing process by making it more complicated in turn, defying the goal of simplicity.

Finally, different representations for the joint orientations were considered. Mainly due to the fact that Euler angles suffer from the well known gimbal lock problem. Quaternions were disregarded because of their storage overhead, even though differencing and addition can be done with quaternions almost as easily as with vectors. Additionally, it was not tested if taking the difference of two quaternions is somewhat equivalent to computing a transformation from one to the other. If it were not, a method other than differencing would have been needed to extract the partial action.

Spherical coordinates were considered, but since there are no standard conversions to and from rotation matrices which are needed for animating the articulated figure on the computer, this option was disregarded as well. Though it was considered to use spherical coordinates during analysis only, but then the same problem as with shoulder and hip line comparison would be faced, i.e. how to convert relationships to the rotational domain expressed in Euler angles.

In summary, the resulting motions looked natural. When compared in trajectory form, it was very clear that the generated motions would lie within an acceptable motion range of the subject. Furthermore, examples for tossing and exaggeration have been presented through which it has been shown that the same scripts that are used for throwing can be used for tossing. Other attempts have been briefly discussed.

CHAPTER 8
CONCLUSION AND FUTURE WORK

This chapter will conclude this work, by summarizing the main contribution of this research and by touching on some possible extensions.

8.1 Contribution

A new kind of motion clip is introduced which is space efficient and easy to use. It facilitates the editing of different base motions by adding a partial motion clip to them while allowing more DOFs than the ones stored in the partial motion clip to be affected, resulting in better looking final motions. This kind of motion clip is called Combined Partial Motion Clip or CPMC for short. Previous editing methods that allow mixing of partial motions with other motions do not take into consideration the affects such a mixing may have on other DOFs not included in the partial motion used for editing.

The advantage of CPMCs is that they are space efficient. Data for editing several base motions are combined into one clip. Only the main contributing joints are included in the detailed part of the CPMC which are averaged across the bases whenever possible. Other DOFs can be estimated by using the scripts that are stored in the CPMC. Another advantage of CPMCs is their ease of use. The main operation needed is a simple vector addition. Similarly, during the creation of CPMC the simple operation of differencing is used to extract the partial action. These advantages have been discussed in CHAPTER 5 and in CHAPTER 6.

The processes performed when editing a base motion with a CPMC can be run in parallel. That is, when resampling of the base is needed it can be started along with the resampling of the retrtieved trajectories (which need to be matched to the resampled base trajectories). Since resampling is frame-sequential, as soon as the first resampled frames are available the process of generating additional trajectories using the embedded scripts can be started. This is also a frame-sequential process, and as soon as the first output of this process is available it can be added to the corresponding trajectories in the base.

CPMCs may be useful in the gaming industry which largely depends on motion clip libraries. CPMCs would provide better blending techniques due to the fact that they take into account the effects that some DOFs have on others. They enhance the way characters move. Therefore, they would go hand in hand with methods that enhance the appearance of characters to make more realistic looking characters for a better gaming experience.

8.2 Limitations

The success of CPMCs is dependent on finding equations that govern the relationship between joint movements. The quality and amount of motion samples available for analysis are major factors in providing a better basis for a more comprehensive analysis. Domain knowledge and expert advice can become handy in analyzing the data. The better the analysis the more likely correct relationships are extracted and better approximations can be achieved. Therefore, one can truly say that a CPMC is only as good as the scripts contained in it.

The method was mainly tested on overhand throwing, which is an upper body movement. The motions generated using the method look believable, and – as shown in section 7.1 – fall within the normal motion range of the subject. The method will work just as well for other upper body motions, like other kinds of throwing, boxing, reaching, waving, pointing, etc. For the method to be extended to lower body motions, such as kicking, it is expected to be more difficult to find relationships since keeping balance will become a major factor.

8.3 Future Work

As said above, good scripts are the key ingredient in making a good CPMC. The ideal solution would be to separate the scripts from the CPMC and have the editing method take care of them. This means that more general relationships have to be found that are independent of the action being performed. If that were possible, such a relationship would have to be true in all cases, then it could be coded in the editing method, and the CPMC would only need to contain the detailed partial motion since its effects on other DOFs would be taken care of in the editing procedure.

It is hard to imagine that such a universal relationship could be obtained. If it could be obtained, it probably would be very complicated and would need to be broken down into smaller

parts, each applying to specific cases or specific groups of DOFs thereby complicating the editing method. But what is imaginable, what is indeed possible, is to group similar types of motions for which the same relationships hold, just as has been shown for overhand throwing and tossing. For that, more partial motions have to be analyzed. Better analysis methods would be helpful. Also collaboration with experts in fields that deal with human body movement would be advantageous.

The incorporation of simple dynamics could help in estimating the motion of 'inactive' limbs. For example, in the throwing while standing case, the major cause for the swinging of the non-throwing arm is a reaction to the rotation of the trunk. Since, one of the goals was simplicity, and physical correctness was not mandatory, but the aim was for a natural looking motion, approximations to dynamic equations could be used. Dynamics may have the advantage that one may be able to take balance into account. Also with dynamics, the generated arm motion when exaggerating by increasing the trunk rotation would be modeled better.

A natural extension to the work presented here, would be to make clips that not only work for different bases but have other variables as well. Something one may call a multidimensional CPMC or MCPMC. For example, the throwing CPMC which applies to three base motions could be extended to apply to different masses of objects being thrown. Another dimension could be the distance thrown. For that, one would need to collect samples along each dimension. Then for each dimension eliminate the effect of the variable by subtracting it from a reference motion, for example the one with the smallest mass and shortest distance.

There is room for improvements in several other aspects:

1. In the marker data conversion to root position and joint orientations.
2. A better inverse kinematics routine used to fix the legs could be used.
3. Alignment of motions may be automated.
4. The quality of the results could be verified by conducting a formal user study.
5. Joint limits could be enforced.
6. Frequency analysis and hierarchical decompositions of the trajectories could be used to enhance analysis.

In conclusion, the new method presented in this dissertation of using Combined Partial Motion Clips has been discussed in detail and tested thoroughly. Based on the resulting generated motions and their comparison to original motions the method has been proven successful. CPMCs are simple to use and are space efficient. They could be incorporated into games to provide better motion blending for better character movement, which in turn enhances the gaming experience. A lot of analysis will be needed since there are many kinds of partial motions that may come up in a game.

REFERENCES

[AMA96] Kenji Amaya, Armin Bruderlin, and Tom Calvert. “Emotion from Motion.” In Graphics interface 1996, pp.222-229.

[ARI02] Okan Arikan and D. A. Forsyth. “ Interactive Motion Generation from Examples.” In Proceedings of SIGGRAPH 2002, Aug. 2002, pp.483-490.

[BAD94] Norman Badler, Ramamani Bindiganavale, John Granieri, Susanna Wei, and Xinmin Zhao. “Posture Interpolation with Collision Avoidance.” In Proceedings of Computer Animation, May 1994, pp.13-20.

[BAR98] Steven W. Barrentine, Glenn S. Fleisig, James A. Whiteside, Rafael F. Escamilla, and James R. Andrews. “Biomechanics of windmill softball pitching with implications about injury mechanisms at the shoulder and elbow.” Journal of Orthopaedics and Sports Physical Therapy, 28(6), Dec 1998, pp.405-415.

[BIA98] L. Bianchi, D. Angelini, G.P. Orani, F. Lacquaniti. “Kinematic coordination in human gait: relation to mechanical energy cost.” Journal of Neurophysiology. 79(4), April 1998, pp.2155-2170.

[BOD97] Bobby Bodenheimer, Charles Rose, Seth Rosenthal, and John Pella. “The process of motion capture: Dealing with the data.” In Computer Animation and Simulation, Eurographics Workshop, Springer-Verlag, 1997, pp.3-18.

[BRA00] Matthew Brand and Aaron Hertzmann. “Style Machines.” In Proceedings of SIGGRAPH 2000, Aug. 2000, pp.183-182.

[BRU89] Armin Bruderlin and Thomas Calvert. “Goal-directed, dynamic animation of human walking.” In Computer Graphics, July 1989, pp.233-242. Proceedings of SIGGRAPH 1989.

[BRU93] Armin Bruderlin and Thomas Calvert. “Interactive Animation of Personalized Human Locomotion.” In Graphics Interface 1993, pp.17-23.

[BRU96] Armin Bruderlin and Thomas Calvert. “Knowledge-Driven, Interactive Animation of Human Running.” In Graphics Interface, [vol] ([no]), 1996, pp.213-221.

[BRU95] Armin Bruderlin and Lance Williams. “Motion Signal Processing.” In Proceedings of SIGGRAPH 1995, Aug. 1995, pp.97-104.

[BUR92] Allen W. Burton, Nancy L. Greer, and Diane M. Wiese. “Changes in Overhand Throwing Patterns as a Function of Ball Size.” Pediatric Exercise Science, 4(1), Feb. 1992, pp.50-67.

[CHU99] Shih-kai Chung and James K. Hahn. “Animation of Human Walking in Virtual Environments.” In Proceedings of Computer Animation '99, May 1999, pp.4-15.

[COH92] Michael Cohen. “Interactive Spacetime Control for Animation.” In Proceedings of SIGGRAPH 1992, 1992, pp. 293-302.

[DEB98] Paul Debevec. “Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography.” In Proceedings of SIGGRAPH 1998, July 1998, pp.189 –198.

[DOW97] John H. Downing. “A Comparison of Wheelchair, Varsity, and Novice Basketball Players on Selected Free Throw Shooting Variables.” Palaestra, 13(2), Spring 1997, pp. 12-13.

[ELL88] Bruce Elliot, J. Robert Grove, and Barry Gibson. “Timing of the Lower Limb Drive and Throwing Limb Movement in Baseball Pitching.” International Journal of Sport Biomechanics, 4, 1988, pp.59-67.

[FLE96] Glenn S. Fleisig, Steven W. Barrentine, Rafael F. Escamilla, James R. Andrews. “Biomechanics of overhand throwing with implications for injuries.” Sports Medicine Review. 21(6), June 1996, pp.421-437.

[FLE99] Glenn S. Fleisig, Steven W. Barrentine, Nigel Zheng, Rafael F. Escamilla, and James R. Andrews. “Kinematic and kinetic comparison of baseball pitching among various levels of development.” Journal of Biomechanics, 32(12), Dec 1999, pp.1371-1375.

[GIR85] Michael Girard and A.A. Maciejewski. “Computational Modeling for the Computer Animation of Legged Figures.” In Proceedings of SIGGRAPH 1985, 1985, pp.263-270.

[GLE97] Michael Gleicher. “Motion Editing with Spacetime Constraints.” In Interactive 3D Graphics 1997, pp.139-148.

[GLE98a] Michael Gleicher. “Retargetting Motion to New Characters.” In Proceedings of SIGGRAPH 1998, Aug. 1998, pp.33-42.

[GLE98b] Michael Gleicher and Peter Litwinowicz. “Constraint-based motion adaptation.” In The Journal of Visualization and Computer Animation, 9(2), April/June 1998, pp.65–94.

[GRI95] Larry Gritz and James Hahn. “Genetic Programming for Articulated Figure Motion.” In The Journal of Visualization and Computer Animation, Vol.6, 1995, pp.129-142.

[GUO96] Shang Guo and James Robergé. “A High-Level Control Mechanism for Human Locomotion Based on Parametric Frame Space Interpolation.” In Computer Animation and Simulation, Eurographics Workshop, Springer-Verlag, 1996, pp.95-107.

[HOD97] Jessica Hodgins and Nancy Pollard. “Adapting Simulated Behaviors for New Characters.” In Proceedings of SIGGRAPH 1997, Aug. 1997, pp.153-162.

[HOD95] Jessica Hodgins, Wayne Wooten, David Brogan, and James O’Brien. “Animating Human Athletics.” In Proceedings of SIGGRAPH 1995, Aug. 1995, pp.71-78.

[KOG99] Yotto Koga, Geoff Annesley, Craig Becker, Mike Svihura, and David Zhu. “On Intelligent Digital Actors.” White Paper, http://www.motion-factory.com/products/whppr_imagina.htm, 1999.

[KOV02a] Lucas Kovar, Michael Gleicher, and Frederic Pighin. “Motion Graphs.” In Proceedings of SIGGRAPH 2002, Aug. 2002, pp.473-482.

[KOV02b] Kovar, L., M.Gleicher, and J. Schreiner. “Footskate Cleanup for Motion Capture Editing.” In proceedings of the ACM SIGGRAPH symposium on Computer animation, July 2002, pp.97-104.

[LEE02] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. “Interactive Control of Avatars Animated with Human Motion Data.” In Proceedings of SIGGRAPH 2002, Aug. 2002, pp.491-500.

[LEE99] Jehee Lee and Sung Young Shin. “A Hierarchical Approach to Interactive Motion Editing for Human-like Figures.” In Proceedings of SIGGRAPH 1999, 1999, pp.39-48.

[LEE01] Jehee Lee and Sung Young Shin. “A Coordinate-invariant Approach to Motion Analysis.” In Graphical Models, 63(2) 2001, pp.87-105.

[LIU94] Zicheng Liu, Steven Gortler, and Michael Cohen. “Hierarchical Spacetime Control.” In Proceedings of SIGGRAPH 1994, 1994, pp.35-42.

[LIU01] Zicheng Liu, Ying Shan, and Zhengyou Zhang. “Expressive Expression Mapping with Ration Images.” In Proceedings of SIGGRAPH 2001, Aug. 2001, pp 271-276.

[MOL96] Tom Molet, Ronan Boulic, and Daniel Thalmann. “A Real Time Anatomical Converter for Human Motion Capture.” In Computer Animation and Simulation, Eurographics Workshop, Springer-Verlag, 1996, pp.79-94.

[MUL99] Franck Multon, Laure France, Marie-Paule Cani-Gascuel, and Giles Debunne. “Computer Animation of Human Walking: a Survey.” In The Journal of Visualization and Computer Animation, 10 (1), Jan/Mar 1999, pp.39-54.

[NOH01] Jun-yong Noh, and Ulrich Neumann. “Expression Cloning.” In Proceedings of SIGGRAPH 2001, Aug. 2001, pp.277-288.

[OBR00] James F. O'Brien, Robert E. Bodenheimer, Gabriel J. Brostow, and Jessica K. Hodgins. “Automatic Joint Parameter Estimation from Magnetic Motion Capture Data.” Proceedings of Graphics Interface 2000, Montreal, Quebec, Canada, May 15-17, pp. 53-60.

[PER95] Ken Perlin. “Real-time Responsive Animation with Personality.” In IEEE Transactions on Visualization and Computer Graphics, 1(1), March 1995, pp.5-15.

[POP99] Zoran Popovic and Andrew Witkin. “Physically Based Motion Transformation.” In Proceedings of SIGGRAPH 1999, Aug. 1999, pp.11-20.

[PUL02] Katherine Pullen and Christoph Bergler. “Motion Capture Assisted Animation: Texturing and Synthesis.” In Proceedings of SIGGRAPH 2002, Aug. 2002, pp.501-508.

[ROS96] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael Cohen. “Efficient Generation of Motion Transitions using Spacetime Constraints.” In Proceedings of SIGGRAPH 1996, Aug. 1996, pp.147-154.

[ROS98] Charles Rose, Michael Cohen, Bobby Bodenheimer. ”Verbs and Adverbs: Multidimensional Motion Interpolation.” In IEEE Computer Graphics and Applications, 18(5), Sept/Oct 1998, pp.32-40.

[TOL00] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. “Real-time Inverse Kinematics Techniques for Anthropomorphic Limbs.” In Graphical Models, 62 (5), Sept. 2000, pp. 353-388.

[UNU95] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. “Fourier Principles for Emotion-based Human Figure Animation.” In Proceedings of SIGGRAPH 1995, Aug. 1995, pp.91-96.

[WIL97a] Douglas J. Wiley. Parameterized Motion Models for Computer Graphics Characters. PhD Dissertation, School of Engineering and Applied Science, Department of Electrical Engineering and Computer Science, The George Washington University, 1997.

[WIL97b] Douglas J. Wiley and James Hahn. “Interpolation Synthesis of Articulated Figure Motion.” In IEEE Computer Graphics and Applications, 17(6), Nov/Dec 1997, pp.156-160

[WIT88] Andrew Witkin and Michael Kass. “Spacetime Constraints.” In Proceedings of SIGGRAPH 1988, Aug. 1988, pp.159-168.

[WIT95] Andrew Witkin and Zoran Popovic. “Motion Warping.” In Proceedings of SIGGRAPH 1995, Aug. 1995, pp.105-108.

[MOT] The Motion Factory, http://www.motion-factory.com

[RAD] RAD Game Tools, Inc., http://www.radgametools.com

[SIE] Sierra Studios, http://www.sierrastudios.com - Personal conversation with Jim Napier, Lead Engineer for SWAT Close-Quarters Battle, responsible for the graphics engine and game architecture.

APPENDIX A
A SCRIPT FOR THE ESTIMATION OF THE NON-THROWING ARM

For a base of standing or walking, the changes in the trajectories of the non-throwing arm can be computed as given in the following MATLAB script:

```
function [s,e,w]=computeNonThrowingArm(root,back,ts)
% [s,e,w]=computeNonThrowingArm(root,back,ts)
% given the changes in the orientation of the root,
% back and throwing-arm shoulder (ts) this function
% will compute the changes in the orientation of
% the shoulder (s), elbow (e), and wrist (w) of
% the non-throwing arm

x=1; y=2; z=3;
torso=root+back;

temp1=torso(y,:)+torso(z,:)-torso(x,:);
f1=10-(abs(min(ts(3,:)+ts(2,:)+ts(1,:)))/20);
temp2=temp1+ts(x,:)/f1;
f2=10;
g=1.25;
n=-temp2/f2;

s(x,:)=temp2+f3*2.^n;
s(z,:)=-ts(y,:)/4;
s(y,:)=-s(z,:);

e=zeros(size(s));
e(x,:)=s(x,:);

w=zeros(size(s));
w(x,:)=-abs(s(x,:));

%end computeNonThrowingArm
```

Note that for a base of sitting, the non-throwing arm's orientation is computed by using IK to fix the position of the wrist in space or relative to the thigh if the hand was resting on it.

APPENDIX B

A SCRIPT FOR THE ESTIMATION OF THE LEG DATA

For a base of walking, the change induced in the legs can be computed as given in the following MATLAB script:

```
function [nh,nk,na,th,tk,ta] =
          computeWalkingLegsNew(pos,lhs,cycle)
% [nh,nk,na,th,tk,ta]=computeWalkingLegs(pos,lhs,cycle)
% given the change in postition this function will
% compute the changes in the orientation of the hip, knee
% and ankle of the non-throwing side (nh,nk,na)
% as well as the throwing side (th,tk,ta)
% for the diffrerence of throwing while walking

x=1; y=2; z=3;

% set all to zeros
nh=zeros(size(pos));     nk=nh;    na=nh;
th=nh;                   tk=nh;    ta=nh;

n=ceil(cycle/6);    % a window for linear interpolation
numFrames=length(nh);
rhs=lhs-ceil(cycle/2);

for j=1:4 % for four cycles

     if rhs>=1 & rhs<=numFrames
          xval_at_rhs = - pos(x,rhs)/1.5;
          zval_at_rhs =   pos(z,rhs);
          nh(z,rhs)=xval_at_rhs;
          nh(x,rhs)=zval_at_rhs;
          th(x,rhs)=-zval_at_rhs;
          th(z,rhs)=-xval_at_rhs;
          for i=1:n
               xval = xval_at_rhs * (n-i)/n;
               zval = zval_at_rhs * (n-i)/n;
               if rhs+i <= numFrames
                    nh(z,rhs+i)=xval;
                    nh(x,rhs+i)=zval;
                    th(x,rhs+i)=-zval;
                    th(z,rhs+i)=-xval;
               end
               if rhs-i >= 1
                    nh(z,rhs-i)=xval;
```

```
                    nh(x,rhs-i)=zval;
                    th(x,rhs-i)=-zval;
                    th(z,rhs-i)=-xval;
               end
          end
     end

     if lhs>=1 & lhs<=numFrames
          xval_at_lhs = pos(x,lhs)/1.5;
          zval_at_lhs = pos(z,lhs);
          nh(z,lhs)=xval_at_lhs;
          nh(x,lhs)=-zval_at_lhs;
          th(x,lhs)= zval_at_lhs;
          th(z,lhs)=-xval_at_lhs;

          for i=1:n
               xval = xval_at_lhs * (n-i)/n;
               zval = zval_at_lhs * (n-i)/n;
               if lhs+i <= numFrames
                    nh(z,lhs+i)=xval;
                    nh(x,lhs+i)=-zval;
                    th(x,lhs+i)=zval;
                    th(z,lhs+i)=-xval;
               end
               if lhs-i >= 1
                    nh(z,lhs-i)=xval;
                    nh(x,lhs-i)=-zval;
                    th(x,lhs-i)=zval;
                    th(z,lhs-i)=-xval;
               end
          end
     end
     rhs=rhs+cycle;
     lhs=lhs+cycle;
end % for

nh(y,:)=nh(z,:)*.5;           th(y,:)=th(z,:)*.5;
nk(x,:)=-abs(nh(x,:)*2);      tk(x,:)=-abs(th(x,:)*2);
na(x,:)=nh(x,:)*1.5;          ta(x,:)=th(x,:)*1.5;

%end computeWalkingLegs
```

Note that for a base of sitting, or standing the legs' orientation is computed by using IK to fix the position of the feet on the floor.

APPENDIX C
THE POSE FILE PLAYER

The Pose File Player is a program written in Visual C++. It reads a pose file (.pos) and plays it back. Multiple files can be opened simultaneously which is good for making comparisons. Inverse kinematics can be applied to the legs and/or arms then new motion can be written as a new .pos file.

Similar to the POSE GENERATOR, figure segments and rest angles can be adjusted, or read in from .fig files. And snapshots may be taken or a sequence may be recorded. Images are written in TGA format. Segments may be turned on or off for display, also multiple exposures may be displayed. The frame spacing is adjustable by the user, default is 10.

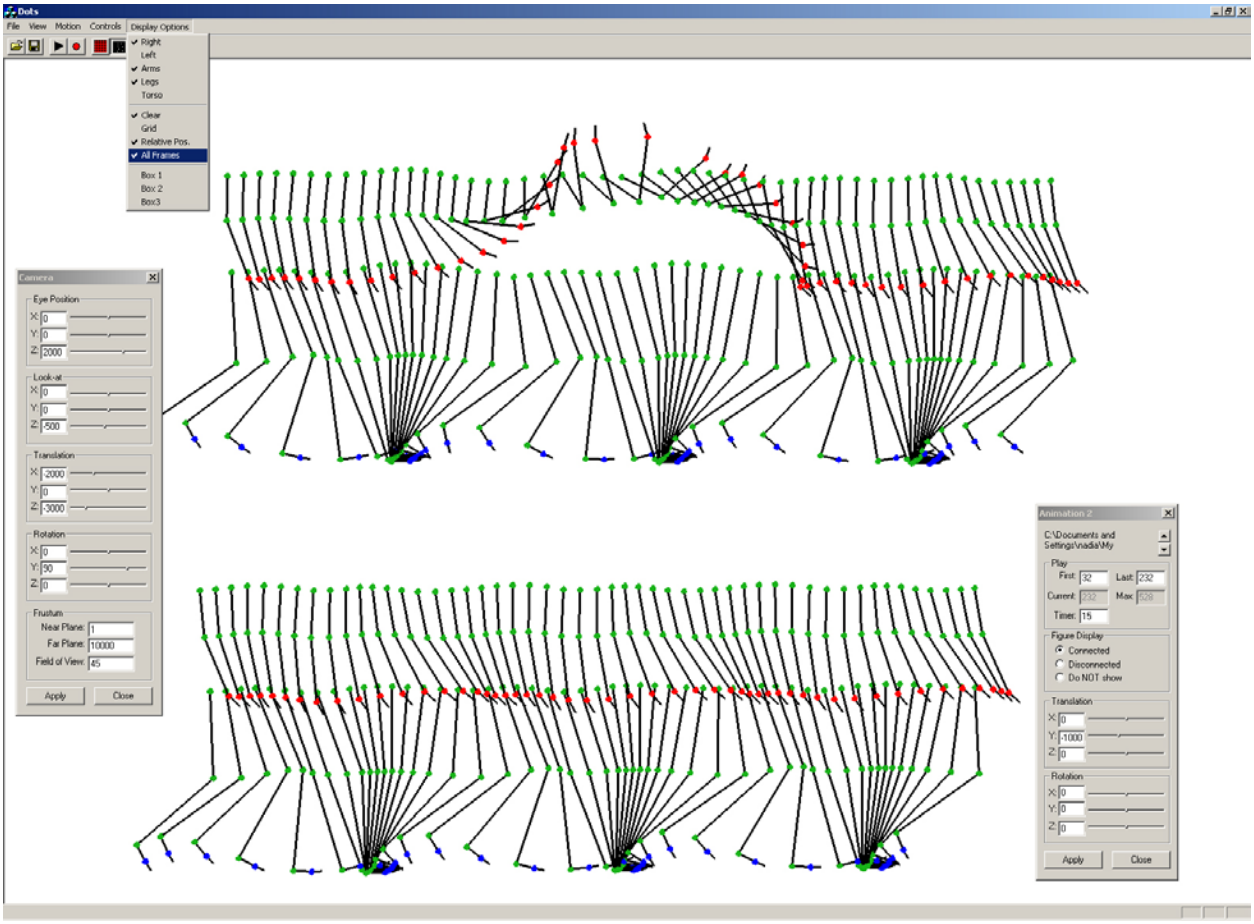

Figure-39 Pose File Player