VERBS AND ADVERBS:
MULTIDIMENSIONAL MOTION INTERPOLATION USING RADIAL BASIS FUNCTIONS

Charles F. Rose, III

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE

June 1999

in memoriam

Pearl Rose

1910 – 1992

Abstract

This thesis describes a new technology dubbed *Verbs & Adverbs*, the goal of which is to create controllable animation through interpolation of example motions. Use of motion capture or hand-animated source material was a key requirement for this work since artists and capture systems currently produce the most compelling animations. Leveraging the artist's talent for interactive animation is an important goal for 3D human figure animation research.

Interactive, non-scripted animations require control mechanisms. With control, a system can be designed to react to a user's wishes, a simulation's unfolding state, or both. Many control techniques are being pursued by the research community with dynamic, procedural, and interpolated methods the three primary groupings. Dynamic animations require a complex controller and a dynamics simulator. Procedural animations use code-like scripts. Both of these methods are alien to classically-trained animators. Likewise, these techniques do not easily incorporate motion capture. Thus we have developed a technique that supports, rather than supplants, the artist or motion capture system. *Verbs & Adverbs* is a system which seeks to empower the artist, to provide a new way to construct animations for use in the interactive realm.

This thesis describes a process for turning source motions into rich controllable animation segments called "verbs," parameterized along a number of control axes called "adverbs." The core mathematics for this technique uses multi-dimensional function interpolation with radial basis functions. Once created, verbs are placed into a "verb-graph," an object detailing the appropriate

times to transition from one verb to another. The verb-graph is the entity controlled by the interactive system.

Additionally, a number of ancillary topics are detailed in this thesis. Robust motion capture analysis, torque-minimal transitioning between motions, and a standard motion interface, or formalism, are all developed here. These, when combined with the *Verbs & Adverbs* technique, form a powerful animation system.

Acknowledgements

The seashore in New Jersey is exceptionally calming in winter. Despite the January chill, I will go there in a few weeks to savor the fact that my dissertation, *mi diablo en forma de una tesis*, is finished. It has taken me a long time to get to this point. Nearly a decade ago, Mary Wagner first helped me decide that earning a Ph.D. was a good goal. Ortley beach is where I'll go to contemplate finishing this major thing and to think about what to do next.

My lava lamps are bubbling, I have some dreamy music playing in the background, and I'm feeling calm. I think I'm ready to write my acknowledgements. I've been saving them up for a time when I'm feeling particularly connected to my life, because I want to get it right. I want to effectively convey the profound gratitude I feel towards those who have helped me, helped shape me, and to those who helped me hold together through my Ph.D. All the people I'll write about for the next few pages have helped make life to date fulfilling and extremely interesting. I hope that my future is as filled with memorable persons of quality.

Education

Michael Cohen is an exceptional person to work with and for. He helped shepherd me through my Ph.D. and I continue to learn a great deal from him. He's been more patient than I think I deserve. I look forward to the years ahead and to continuing to work with Michael at Microsoft Research.

Jack Gelfand helped me learn about biomechanics and to understand the value of that body of work to computer graphics. This dissertation is better for understanding the connections between modeling human phenomena and modeling human processes. Thanks for that and for shoring up the Princeton side of my bicoastal Ph.D.

Thanks to Adam Finkelstein, my third reader, and to David Dobkin and Perry Cook, the rest of my committee. The last few months have dragged on a little too long. Thanks for not getting too frustrated with delays (or at least for not letting on!)

Melissa Lawson deserves special thanks for keeping track of all the details needed to move through Princeton's graduate program. Further thanks for being an extremely pleasant person to talk to when I'm in town.

Thanks to my office-mates at P'ton: Ramesh Sitaruman, Dan Boneh, James Shaw, and honorary office-mates Colleen Wirth and Steven (Schlomo) Gortler. I can't remember if we ever devised a scheme for the perfect human society, but we certainly used many afternoons trying. Thankfully, the doors at Princeton were thick and our advisors were upstairs.

My time at Princeton was memorable and I will hold the university close to me always. My education did not start there, however, and I'd like to remember my undergraduate mentors who helped me prepare for graduate school. First, I'd like to thank Charles Goldberg, who took a special interest in my education while I was at Trenton State College and who championed me to Princeton's department. I spent a great deal of time in the Princeton area during my youth. Attending the university was always something I wanted to do and he helped me get there. Charlie died recently– tragically early. My visits to TSC will hereafter always be missing an integral component.

Mary Wagner and Ursula Wolz were two professors at TSC who got me

going on the graduate school track. Mary helped me convince myself early on that going to grad school was a laudable goal and Ursula helped me understand that I probably wouldn't be playing golf in the mornings before a leisurely afternoon in the lab. As an undergraduate, I had to invent challenges. Graduate school had them in abundance. Thanks for getting me ready to deal with that shock.

I'd also like to thank the professors in other disciplines who helped make my TSC experience so enlightening. Mort Winston, Robert Anderson, Lynn Waterhouse, Lahna Diskin, and Lee Harrod all deserve thanks for running some of my favorite courses. Special thanks to William DeMerrit for running the college honors program, something I found very fulfilling.

But wait, there's more! With my ten year high school reunion recently past, my thoughts return to Toms River High School South. Thanks to Don Comp and Dennis Pieretti for teaching me the basics of computer science and helping to spark some of my initial interest in the field. Thanks to Janice Gelzer, Sally Howe, Renee Lomell, David Fitzmaurice, and especially Angie Cazolla for helping to foster my broad interests.

Employment

I've had the good fortune to be a part of one of the most exciting groups in the graphics world for the last three years. I'd like to thank Brian Guenter for bringing me to Microsoft Research for my initial internship which was extended through SIGGRAPH deadlines and then to infinity and beyond. While I may occasionally wake up and dread the commute from Capitol Hill to Redmond, I never wake dreading my job. I am honored to be a part of Microsoft Research and the graphics group in particular. Thanks to all of MSRG for being great coworkers.

Thanks to Microsoft's motion capture group for great support and collaboration. Seth Rosenthal, John Pella, Hank Mueret, and Jana Wilcoxen all deserve special thanks for going above and beyond for our project. David Thiel helped us produce videos and helped with sound for our demos. People like these make working at Microsoft an honor as well as a pleasure.

Thanks to Jim Shepherd, Charles Casey, and Mary Kondash at Computer Sciences Corporation, for teaching me how to work in the professional computer science world and thanks to Stella Kern for my initial exposure to gainful employment at the Toms River Library. The lessons you helped teach have proven invaluable as I continue my professional development.

Friends

Michael W. Post has been an exceptionally important friend over the years and has often helped to push me out the local minima I'm so good at finding. I'd be much less happy, and much less close to completion, if our paths had never crossed. Here's to the best human catalyst I know. Let's find another circus maximus.

While Mr. Post convinces me to dye my hair blue (as evidenced at `http://www.research.microsoft.com/~rose`), Matthew Sharkey helps me keep my mental balance. Unchecked anxiety is my most deadly personal foe and Matt has always been my champion in this regard. That and sixteen years of amazing friendship intertwine our lives inseparably.

Michelle Genereux– I still love typing her cool last name after all these years. Michelle has been one of my most trusted mentors since I first met her when I was but a lowly page in Toms River Library's publications department. She is also one of my most closest friends, and I have often benefited from her measured, well-reasoned advice.

Peggy Deaner, fellow New Jersey expatriate, has been an important friend over the years from whom I can always expect sympathy and good advice. I hope I've proven as useful to her. Peggy: best wishes for your own graduate school efforts, which you have recently started. You will make an excellent educator.

To Jen Lutton and Jen Escalante: 3000 miles does not diminish the importance I place in being able to count you as friends. Go grunge and find some jobs in Seattle so we can reform the Jersey Cabal.

To Don Mitchell, Mike Marr, Chris Liles, Briand Sanderson, and the rest of the Crossroads dinner crowd: thanks. The conversations are always stimulating. Soon our plans will be complete and phase two can begin!

I'd also like to thank the Friday night movie crowd, especially Daniel Brown, our ardent movie-night organizer and resident center of the universe. You've all helped put more fun and hope into life than I've had in recent years.

I'd like to especially thank my friends and housemates, Chris Liles and Cindy Grimm. I know I've been testy lately and I'd like to thank you for putting up with me and being great friends. Without you, there would be no groove in our groovy Capitol Hill pad!

Family

I have a very loving, supportive, and (by today's standards) large family. I'd like to thank them all for being there over the years. To my grandparents, aunts, uncles, and nifty cousins, thanks for being a good family.

Donald and John Cullerton deserve a note of thanks for showing me two views of the academic, intellectual community I'd later join. John, I may have dreaded your Christmas-eve quizzing about derivatives and, Donald, I may have taken issue with some of your wilder notions, but I deeply miss both of

you. The holiday's aren't the same now that you are both gone.

The people who deserve the greatest thanks are the members of my immediate family, my sister Pamela and my parents Laurie and Charles. Under stress, my temper can be terrible and my sarcasm vitriolic. I'm afraid that my family has received the brunt of that unpleasantness over the years. For putting up with me and providing a fine home, I thank you. I'm also very grateful for having the lives of these three interesting people so closely interwoven with my own. I couldn't have wished for a better set of parents or a cooler sister.

Finally, thanks to the woman to whom this thesis is dedicated in memoriam, my paternal grandmother, Pearl Rose. The quiet friendship we had helped me to understand the usefulness of quiet introspection. I thoroughly enjoyed visiting you at the beach on the weekends; waking up to your pussycat shaped pancakes, playing alone on the beach during the day, and watching *Love Boat* and *Fantasy Island* together in the evening. This is the stuff of childhood paradise. I remember those times fondly and miss you (and your pride of cats) dearly.

Charles F. Rose, III

Seattle, WA

January 31, 1998

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Conveying emotion in motion has been a paramount goal of human figure animation research. More than simply enabling virtual actors to perform a set of tasks, true believability requires them to act with style and aplomb, daintiness and reserve, wonton assuredness, unidentifiable likability, brazen sex appeal, or with the villain's craven design.

Control of emotion in conjunction with control over the basic form of a motion has been difficult to achieve. *Verbs*, the actions one takes and *adverbs*, the qualifiers which modify those actions are the two key metaphors used in this dissertation. Providing the ability to construct verbs, controllable animations parameterized continuously over a set of adverbs, is the central goal of this work. Leveraging the talents of an animator or the qualities of motion capture without sacrificing controllability, our verbs help forward the goal of providing emotional control in motion.

1.1 The demands of interactivity

To list emotion in motion as something yet to be achieved may appear a miscategorization. *Luxo Jr.*, Pixar's first great animated short, for example, was replete with aesthetically rich motion. It was, however, a static piece, re-

maining the same at each watching. Its designers painstakingly crafted each motion and decided when it was complete. The research community, on the other hand, has largely been concerned with interactive, non-scripted animation. Each run-time instance of an interactive character can yield a novel performance requiring no additional input from an animator. When interactive animations can achieve a degree of quality like the great animated works, then the world of three-dimensional games, virtual actors and avatars, online shared environments, and intelligent agents will be a great deal more convincing.

1.2 Paths to control

Research into controllable human figure animation can be divided into three major groupings: procedural, dynamically simulated, and interpolated. Procedural animation uses code fragments to derive the degree of freedom (DOF) values at a particular time. The procedures can be as sophisticated as needed to provide different motion styles or to react to different conditions of the simulated environment. Writing motion procedures, however, is not how animators choose to work. Dynamically simulated figure animation uses controllers together with a simulated human to generate motion. The degree to which this method succeeds is bounded by how accurately human motion is understood and modeled. Both procedural and dynamic methods have the disadvantage that they cannot leverage the talents of classically trained animators and that they do not easily make use of motion capture technology. This is important since animators and motion capture systems each produce compelling results. To leverage their qualities, a system must use what these resources provide.

Interpolated animation is the third major grouping. This method uses sets of example motions together with an interpolation scheme to synthesize

new motions. The primary problems of this approach are in providing a set of meaningful, high level control knobs to the animator or runtime system, maintaining the aesthetic of the source motions in the interpolated motions, and motion extrapolation. Another consideration is the difficulty of acquiring the examples. Each is precious. Additionally, for an interpolated motion scheme to be used in a run-time environment rather than earlier in the production pipeline, it must be efficient.

1.3 Verbs & adverbs

Verbs & Adverbs addresses these issues while providing controllable motion usable in a runtime environment. Through the creation of parameterized motions, which we call "verbs" parameterized by "adverbs", a single authored verb produces a continuous range of subtle variations of a given motion at real-time rates. As a result, simulated figures alter their actions based on their momentary mood or in response to changes in their goals or environmental stimuli. For example, we demonstrate a "walk" verb that is able to show emotions such as happiness and sadness, and demonstrate subtle variations due to walking uphill or downhill while turning to the left and right. Emotional and structural control are handled using the same mechanism. Aesthetics of the source material are maintained and used in the synthesis of new motions as well.

1.4 Higher-level control

Interactive systems need one other major mechanism. Verbs, by their nature are succinct motions, like walk-cycle, reach, wave, etc. We do not seek to encapsulate all of a character's motion for an entire script in a single verb. Be-

ing able to control an actor's happiness, for example, while he played through an hour's worth of pre-planned actions would not be particularly interactive. Interactive systems, therefore, make use of smaller units of animation.

The goal, however, is to yield an overall motion which looks like it *could* have been planned in advance and carefully crafted offline. One of the primary telltales that a motion is composed of pieces are awkward transitions. Smooth transitioning, explained in Section 5.5 and Appendix B, address this problem.

Verbs, nodes in a graph, and transitions, arcs in a directed graph form a new unit: the *verb-graph*. Combined with algorithms to enforce reasonable root motion and enforce continuous adverb change, it is possible to make smooth controllable animations composed of pieces with seamless transitions. The verb-graph becomes the controlled object rather than the verbs in a runtime system.

This dissertation will detail the preparation of example motions, the constructions of verbs, a multi-resolution radial basis function interpolation scheme, artist refinement of verbs, transitioning between verbs, and the verb-graph. Together, these pieces can be used to create an interactive animation system.

1.5 Organization

There are many terms and functions to be defined as this thesis progresses. Each will be described in turn, but the reader will find Appendix A contains the complete motion formalism, which is defined piecemeal in Chapter 3 as new concepts are added.

This dissertation began with an introduction describing the *Verbs & Adverbs* technique with some of its results. The following chapter is an overview of human figure animation, placing the *Verbs & Adverbs* system in context. Fol-

lowing the overview chapter, Chapter 3 discusses the acquisition of examples, the process of taking raw motion capture or hand animated data and putting it into a canonical form from which verbs can be constructed. This chapter also introduces the motion formalism, a key abstraction upon which all animation objects depend. Chapter 4 is a discussion of constructing verbs from sets of examples, multi-resolution radial B-spline approximation, and time-warping. Verb-graphs, the topic of Chapter 5, provide the glue with which interactive animation systems can be constructed from verbs and transitions. Chapter 6 summarizes results and provides an analysis of a small user-study. Finally, conclusions and a discussion of future directions appears in Chapter 7. Appendix A, as mentioned previously, summarizes the motion formalism and Appendix B describes the side-topic of torque-minimal transitioning.

Chapter 2

Human figure animation overview

When most people hear the word "animation," they think of classic two-dimensional animation such as great Disney masterpieces like *Snow White* or *Cinderella*. These works are truly 2D creations. That the rolling magical landscapes beyond the castle walls looked far away was due to the artist's illusory skills. Any change in the position of the virtual camera in the scene would require the artist to paint new images.

Three-dimensional computer graphics generates a virtual environment, which can be viewed from any angle, even cinematically uninteresting ones. In this sense, it is a very free and unrestricted medium. Virtual camera movement can be provided by the computer so as to not destroy the illusion of solidity. Characters in 3D environments can more easily interact with one another or with objects in the environment while still allowing unrestricted camera motion. A character picking up a cup in a 3D environment, for example, has a well-defined problem to solve. One possible solution is found in [123]. The 2D problem may seem simpler at first. The problems of registering (aligning) the hand of the 2D character, or sprite, projecting the correct view of the cup for an arbitrary camera angle, and placing the cup back on the table so that it

actually appears to be on the table show that 2D animation is in fact harder in some regards.

This chapter will overview 3D figure animation research, the issues and the different paths to resolving these issues developed by researchers in the field. *Verbs & Adverbs* is an interpolated animation technique, but understanding of competing approaches will be valuable to the reader in gauging the qualities and effectiveness of the technique.

Three-dimensional animated characters are known by many names. *Character* is one often used here. *Actor*, *virtual actor*, or *synthetic actor* are other terms used in this thesis, each made popular by the work of the Thalmann and Magnenat-Thalmann who seek to make synthetic actors as lifelike and compelling as living actors [137]. *Avatar*, a term made popular in Neal Stephenson's influential novel *Snow Crash* [134], denotes the virtual manifestation of a living user. Two other terms, *figure* and *articulated figure* are terms I'll use heavily in this thesis.

A useful way to view many of the notions presented in this chapter is a set of levels of abstraction. The chapter will progress from a low level of abstraction to higher levels. *Verbs & Adverbs* is a mid-level system. Figure 2.1 shows the levels from low abstraction at the bottom to high at the top. This is a chart for a single character. Crowd behavior and multi-character interaction are also important and will be covered in this overview briefly. *Verbs & Adverbs* is an example of interpolated animation. Badler [5] treats these topics much more extensively in his latest virtual humans work. Here I am concerned solely with abstractness as it relates to the motion of one character.

Some useful overviews have appeared recently which may prove useful to the reader. Allbeck and Badler [1] frame the problem of 3D articulated figure animation providing a healthy snapshot of the field with extensive bibliog-

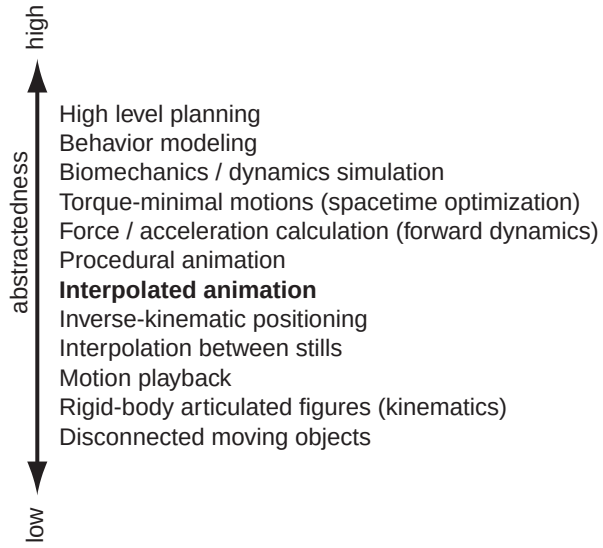

Figure 2.1: Levels of abstraction in figure motion

raphy. They look to answer how close animation research is to fulfilling the vision in *Snow Crash*. Likewise, Earnshaw synopsizes some of the primary virtual humans work currently underway in [43]. Other traditions may prove useful in the understanding of animation. Dance and human motion analysis, for example, are tied deeply to human figure animation. Bartenieff and Lewis [17] describe the field of motion analysis. Of particular note is Laban analysis, used in the dance community, which seeks to describe in a systematic qualitative way the subtleties of human motion.

2.1 Forward kinematics

Disconnected objects moving in space is the lowest reasonable level of abstraction for figure animation. It is a useful level for detailing the motion, or *kinematics*, of simple rigid body objects like tables, chairs, and, of course, teapots. Simplicity here refers not to the geometric complexity of the objects, but to the fact that they have no moving pieces. Objects composed

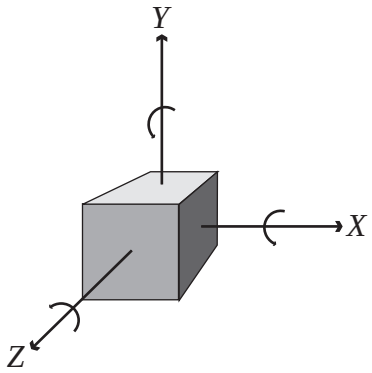

Figure 2.2: Degrees of freedom

of sub-objects, which move in relation to one another, are known as *articulated* objects. The position and orientation of any simple object or part of a more complicated object can be described using six numbers: three to translate along the **X**, **Y**, and **Z** axes, and three to rotate around them each in turn. The last three values are called Euler angles and they together with the translational components are shown in Figure 2.2. Each of these values is called a *degree of freedom* and represents one manner in which an object can be positioned or oriented. These six degrees of freedom (DOFs) together can position and orient an object arbitrarily in space. Euler angles are the most widely used formulation, despite having inherent problems for certain operations such as motion blending, something done here extensively. Quaternions, described by Shoemake [129] [130], Duff [42], and Barr [16], provide a more stable representation. They are used here for transitioning (Section 5.5), though not blending as our motion capture analysis method described in Chapter 3 alleviates the problems normally associated with Euler angles with respect to motion blending.

A 3D character like a person is traditionally constructed from smaller units. A human would likely be made up of a torso, upper-arms, forearms, hand, head, thigh, lower-leg, etc, depending upon the desired level of com-

plexity of the figure. Chadwick [31] provides a good overview of linked figure animation. Most of the results shown later in this thesis use a 44 DOF model with 16 major body segments. This has been enough to yield believable human motion without too many extraneous DOFs. More accurate models, for example those that include fingers and toes, can certainly be designed. Nedel and Thalmann [109] use a 62 DOF model with 31 separate body segments. Maurel, et. al. [97] describe an even more accurate model of the human shoulder complex, the part of the body which causes the most trouble since it is not well approximated by the articulated rigid body abstraction. At eighteen DOFs per shoulder, however, it may not be practical for use in many circumstances and may be cumbersome for animators. Badler describes skeletons with multiple levels of detail in [7] and [5], which can make practicality and correctness less at odds.

Articulated figure movement is described hierarchically. Motion of one part of the body, like the knee, is expressed in terms of its parent body part. Likewise, movement of one part of the body effects all the subordinate parts of the body. For the knee, the ankle and foot would probably be subordinate and the center of the hip would likely be the root point. The root joint is special in that it is used to position and orient the entire body in the global coordinate frame (or any frame to which the body is placed subordinate). Likely roots are between the hips, the feet, and hands. Typically the root is chosen based upon the current action. For example, a character hanging from a handhold would likely be rooted at that hand.

The basic unit of the articulated figure is the *joint* or *node*. There are many ways to define joints, some more general than others. Denavit-Hartenberg notation is a widely used general formulation and is detailed in [41].

In this thesis, a simple formulation of joint will be used. Joints each

have one associated motion (DOF), a translation or rotation along or about one of the principle axes. Translational joints are referred to as *prismatic* and rotational as *rotary* or *revolute*. Joints are offset from and parented to another joint save for the root which has no parent. Joints have a potentially empty set of child joints. A joint, therefore, can be defined as

$$J = \{\mathbf{o}, D, C\} \tag{2.1}$$

where $\mathbf{o}$ is the vector offset relating the joint to its parent joint, D is the type of motion, to be shortly defined, and C the set of child joints parented in the joint J. D can be a translation about the $\mathbf{X}$, $\mathbf{Y}$, or $\mathbf{Z}$, or a rotation about one of those axes. Additionally, the joint can have no motion, indicating that it is an *end-effector*. The end-effector joint has no motion or children, just an offset. It is a place onto which objects are grafted or where interaction with the environment likely occurs. Typical effectors are the fingers or palm. Multiple DOF joints, such as the shoulder, are created by using multiple one-DOF joints with zero-length offsets.

The position a character takes when all DOFs are set to zero is alternately called the *birth*, *home*, or *rest* position. This configuration is due entirely to the effect of the sums of the joint offsets from the root out to the parts of the body. When DOFs are not zero, the position of the body parts can be described by matrix multiplication. Figure 2.3 shows a two link-arm. Global DOFs are shown in green and hierarchical DOFs shown in red.

The global position and orientation of a character can be determined for the hierarchical figure using the recursive function Algorithm 2.1.

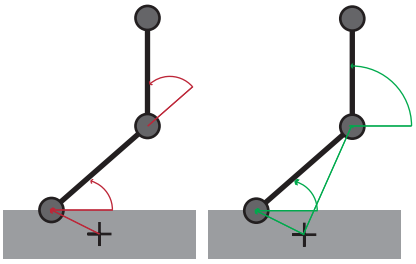

Figure 2.3: Hierarchical vs. global motion for a two-link arm

Algorithm 2.1 Positioning the character globally

`GlobalPosition (`J`, `M_p`)`

{

$$J.M = M_p \bullet \overbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x & y & z & 1 \end{bmatrix}}^{\text{offset from parent}} \bullet \left\{ \overbrace{\begin{array}{cc} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{X}-\text{translation}} & \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos d & \sin d & 0 \\ 0 & -\sin d & \cos d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{X}-\text{rotation}} \\ \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & d & 0 & 1 \end{bmatrix}}_{\mathbf{Y}-\text{translation}} & \underbrace{\begin{bmatrix} \cos d & 0 & -\sin d & 0 \\ 0 & 1 & 0 & 0 \\ \sin d & 0 & \cos d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Y}-\text{rotation}} \\ \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & d & 1 \end{bmatrix}}_{\mathbf{Z}-\text{translation}} & \underbrace{\begin{bmatrix} \cos d & \sin d & 0 & 0 \\ -\sin d & \cos d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{Z}-\text{rotation}} \end{array}}^{\text{DOF motion - use one}} \right.$$

`for all children` $J.C_i$

`    GlobalPosition (`$J.C_i$ `, `$J.M$`)`

}

J is the joint for which the global position is being calculated. M_p is the global matrix of J's parent. The x, y, and z values are the offset of J from the parent. The joint's DOF value is d. To position the entire character a top level call is made with the character's root for J and the identity matrix I for M_p. The joint is very reminiscent of the coordinate frame and it is not surprising that 3D articulated figures are very naturally created in a retained mode graphics architecture.

2.2 Keyframing

As Lasseter pointed out in his SIGGRAPH'87 paper *Principles of Traditional Animation Applied to 3D Computer Animation* [85], 3D animation is a creative process much like its 2D predecessor. Similar techniques are used by animators in order to create stunning work. Thomas and Johnston's landmark book *Disney Animation– The Illusion of Life* [138] describes these principles, like anticipation and ease-in/ease-out, in detail.

Hand animated 3D pieces are typically constructed by the animator using a process known as *keyframing*. Important poses of the character are hand designed and placed in time. The in-between frames are generated by an animation system, such as *SoftImage*$^{(TM)}$ or *3D-Studio/Max*$^{(TM)}$. The animator refines the keys and adds new ones as necessary until satisfied with the result. Maestri [94] provides a good overview of this process. Steketee and Badler describe keyframing techniques and some motion transitioning in [133].

The resulting motion data is a set of DOF curves, which when applied to the appropriate skeleton and played back at the correct rate, reproduce the original motion. A DOF curve is a curve parameterized by time and which yields values for one DOF. A DOF curve is one way to implement a DOF function and is the way typically used in this dissertation. If the motion data

is sufficiently continuous, storing every frame is wasteful. Human motions are usually well represented using a smooth representation since (in general) we make smooth motions. Piecewise-linear, B-spline, hierarchical B-spline, and wavelet representation would all be useful ways to approximate the frame-by-frame data. In this thesis, we typically use B-splines.

While all the basis function types are useful, some are more desirable than others. Compression and fidelity, as always, are the two competing goals when choosing the correct basis function for approximating data. This is an especially important concern for motion capture data since it is copious in quantity. For example, a soccer animation data-set used in [124] takes a great deal of space in its raw form: 215,650 coefficients for roughly 2.7 minutes of animation.

What becomes apparent from an inspection of the data, however, is that the curves are relatively free of high-frequency information. A small set of coefficients of smooth basis functions will maintain high fidelity. A number of candidate encodings could be chosen. As stated previously, this thesis primarily used a B-spline representation. Furthermore, not all the DOF curves have the same curviness (frequency). Many of them are very smooth, requiring far fewer coefficients. Coefficient requirements are shown in the table for different quality ratings. Capin, et. al. [28] explore the bandwidth requirements of human motion data over a networked environment.

A wavelet representation is also likely to be useful for motion data, though no analysis of it has been performed here. Wavelets would be particularly useful for progressive transmission and refinement over a high latency network like the internet. The effectiveness of wavelet representations has been well established by [58] [91] [48].

2.3 Inverse kinematics

Forward kinematics positions a character globally given a set of DOF values. Inverse kinematics determines a set of DOF values which will yield a desired global configuration. Two standard methods involve linear and non-linear minimization of error between desired and current global configuration. Each of these methods is iterative and highly dependent upon starting guess.

If the initial guess is sufficiently close to the desired configuration, a simple linearization can be used. While kinematic positioning is non-linear (Algorithm 2.1), it is sufficiently linear for small changes of the DOFs. Iterative improvement, therefore, converges to a solution. The DOF values needed to satisfy a kinematic constraint can be found by solving the linear system

$$\mathbf{J}\,\boldsymbol{\Delta\theta} = \boldsymbol{\Delta\mathbf{x}} \tag{2.2}$$

where $\mathbf{J}$ is the Jacobian of the DOFs with respect to the desired global constraint, $\boldsymbol{\Delta\mathbf{x}}$ the error vector between goal and current, and $\boldsymbol{\Delta\theta}$ the solution. By iterating through each of the constraints, the figure's configuration will likely converge to a solution satisfying each. This cannot, however, be ensured since articulated figure positioning is non-linear, as was seen in Algorithm 2.1 and since two or more constraints may vie with one another if they are incompatible, such as might occur with a knee and foot constraint further apart than those joints actually are. Typically, this technique is useful for one constraint or sets of constraints which do not interfere with one another, such as might occur if each of the extremities of the body were constrained within the workspace of the figure. An overview of this technique can be found in Girard and Maciejewski[54]. Another good source is Watt and Watt [143] or [22]. Maciejewski [92] describes parallel network approach for solving IK problems using this technique.

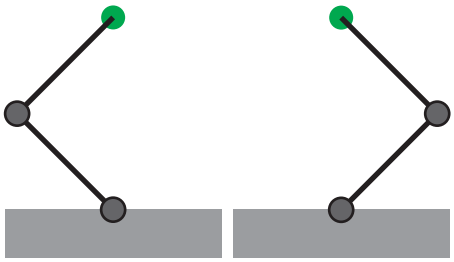

Figure 2.4: Redundant IK solutions

A much more robust, though slower, method that can handle multiple position and orientation constraints simultaneously was developed by Zhao and Badler[150]. The goal is to minimize the non-linear cost function F,

$$F(\Theta) = \sum_{j} \left((P_{j \in \mathbf{J}}(\Theta) - \hat{P}_j)^2 + (O_{0,j}(\Theta) - \hat{O}_{0,j})^2 + (O_{1,j}(\Theta) - \hat{O}_{1,j})^2 \right) \quad (2.3)$$

where $P_j(\Theta)$, $O_{0,j}(\Theta)$, and $O_{1,j}(\Theta)$ are the position and orientation vectors for the jth joint and $\hat{P}_j$, $\hat{O}_{0,j}$, and $\hat{O}_{1,j}$ the desired position and orientation vectors. The paper details the differentiation of this cost function F which enables minimization using a robust non-linear solver such as BFGS [53].

Each of these techniques is sensitive to its initial guess. This causes problems since 3D manipulators are (in general) redundant. Figure 2.4 shows two possible solutions a two-link manipulator might take to position the end-effector on the green dot. In general, such as with a three-link arm positioning to the green dot, there are an infinite number of solutions.

Furthermore, there is no guarantee that a similar inverse kinematics problem yield a similar DOF values. An epsilon change in the constraints may trigger a large change in resulting DOF values. Rose et al [124] dealt with this problem by solving for IK constraints over the whole of the motion rather than on a frame to frame basis, dubbed spatio-temporal IK. Integration of error coupled with a continuous representation of motion (in that case B-splines) yielded smooth IK solutions. Michael Gleicher [55] [56] uses a similar, more

efficient, technique to achieve impressive results in retargetting motion from one articulated figure to another. Inconsistent IK causes particular trouble to motion capture analysis, as will be explored in Section 3.7. That section will detail ways to combat this problem in that light.

2.4 Motion capture

Hand designed animations are time consuming to produce and can lack the highly realistic look desired for sports games or for animation composited into feature-length (non-animated) motion pictures. Motion capture is a popular way to acquire motions that convey gritty realism to the viewer. Originally this technique grew out of the motion analysis needs of the biomechanics community, but now is firmly in the service of computer games and the motion picture industry [116].

Motion capture is more like puppetry or physical acting than animation. A human actor, combined with a sensing technology is the way pose information is extracted. An analysis phase fits this pose information to a hierarchical articulated figure. Over time, changes in pose produce an animation. Motion capture systems are commercially available. The three main types are optical, magnetic, and exoskeletal. At the present time, no type is clearly superior; each has useful qualities. Figure 2.5 shows an example of each type of motion capture system.

Optical systems use markers placed on the body together with multiple calibrated cameras to extract global 3D position information. Occlusion is this technique's primary flaw, though it produces some of the most accurate data and allows the greatest range of motion for the motion capture actor. Occlusion occurs when one or more of the cameras cannot sense a marker due to the actor's body being in the way. Cost, however, is a practical limitation,

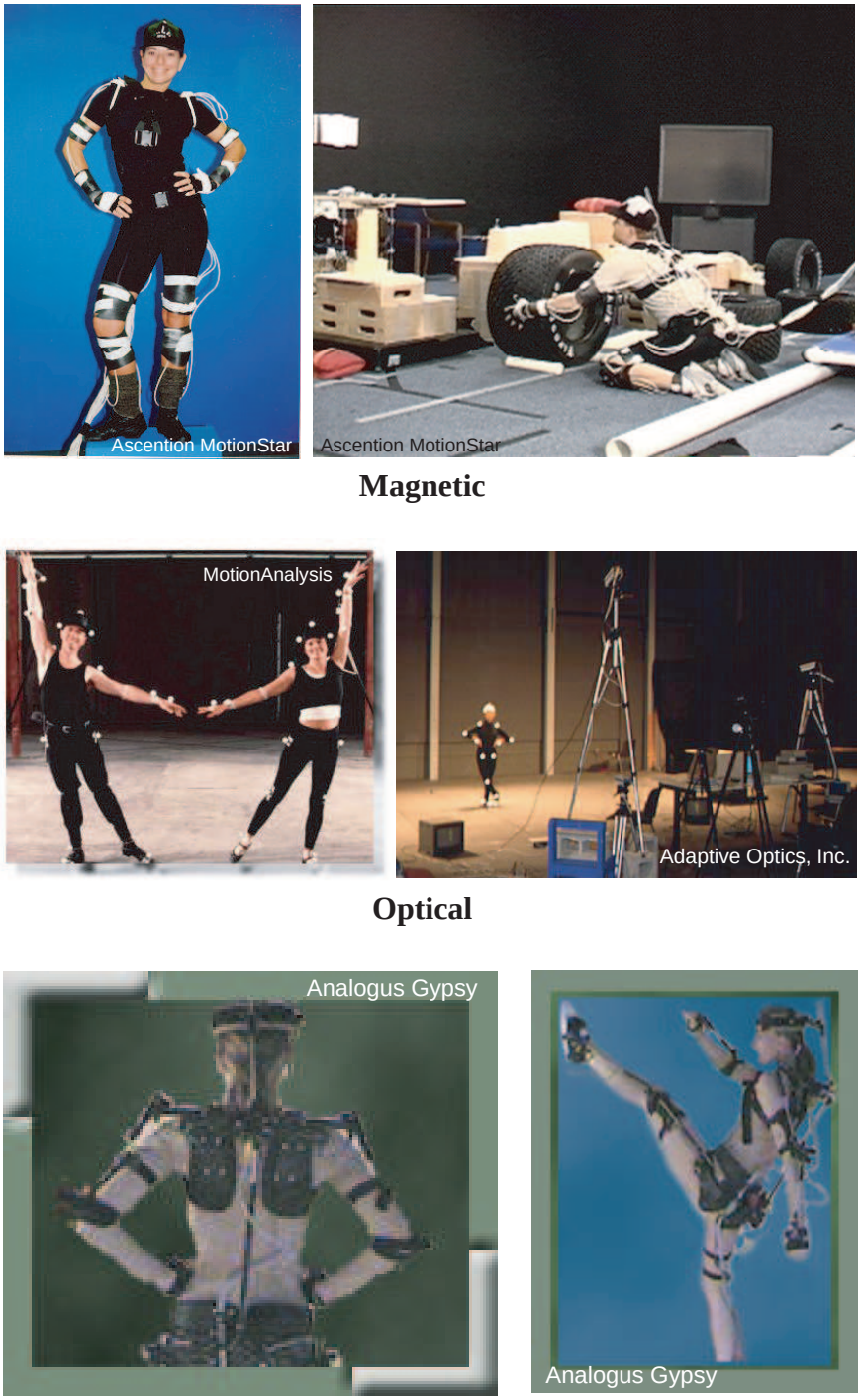

Figure 2.5: Motion capture systems

as an optical system is typically three to four times as expensive as a magnetic system.

Magnetic systems rely on a magnetic field and a set of sensors attached to the body. Occlusion is not an issue and orientation data can be read along with position, which is a plus. Magnetic systems pose numerous challenges, however. Ferrous metals will disrupt the field, so metallic props cause problems. Wooden props, put together with brass nails and screws, are often built, as in the wooden racecar built as a prop for the motion captured pit crew in *Microsoft Precision Racing*(TM), as shown in Figure 2.6. Another problem is the cabling. Even non-tethered systems seriously restrict the range of motion available to the motion capture actor. The extent of cabling can be seen in Figure 2.5. Magnetic motion capture is constraining. Hiring actors able to overcome the awkwardness of the apparatus is key to getting quality motion.

Exoskeletal systems fit the actor with a metal exoskeleton that moves to conform to the motion of the actor. This motion is recorded through various angle and length sensors built into the exoskeleton. This method is the most restrictive one to the actor's range of motion. Falling would likely prove a costly and painful mistake. Also, as none of the sensors is global, error builds up from the root of the exoskeleton, typically between the hips, to the extremities. This can cause unacceptable drift in position and orientation of the hands and feet.

An interesting twist on the exoskeleton is to remove it from the actor. Then the exoskeleton becomes an input device for stop-motion animations like the great movies of Ray Harryhousen. *Digital Image Design* sells one such system, the *Monkey*(TM), shown in Figure 2.7.

Considerable literature exists on using and editing motion capture data in animation, (e.g.,[6] [26] [112] [124] [148]). daSilva, et. al., propose an anima-

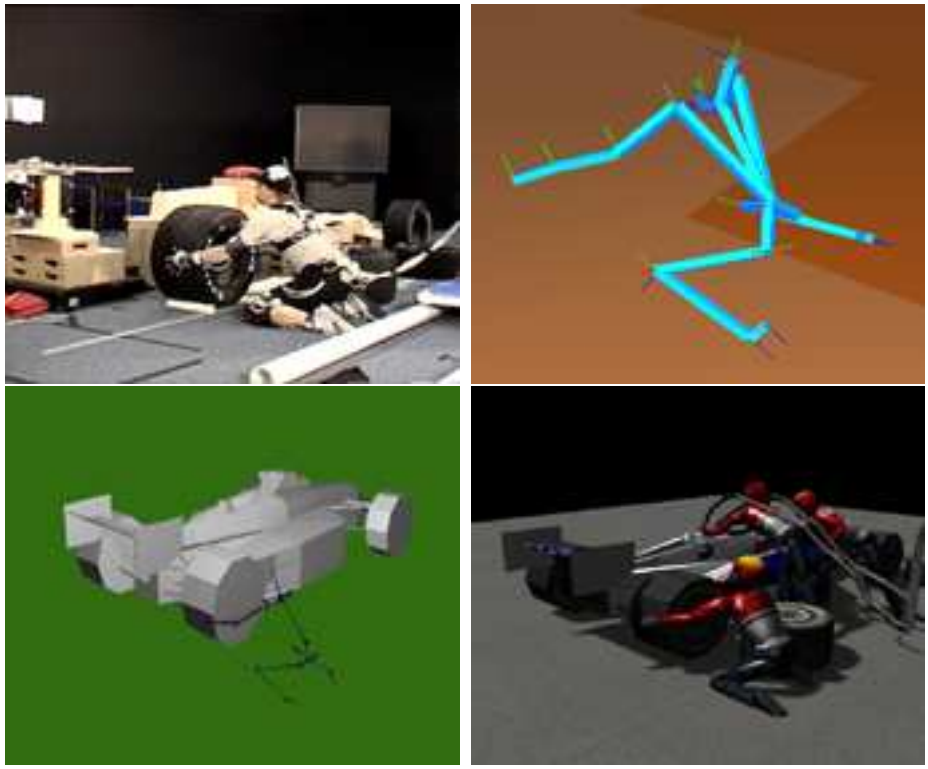

Figure 2.6: Motion capture phases for *Microsoft Precision Racing*

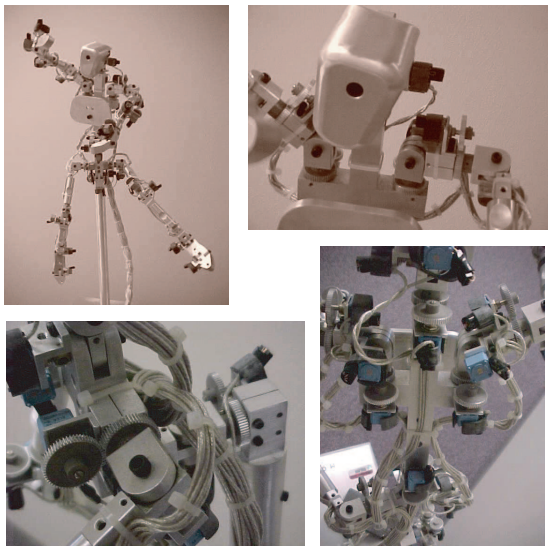

Figure 2.7: *Digital Image Design*'s "Monkey"

tion system based on motion capture data incorporating methods for pulling motions apart, concatenation, transitioning, etc. in [37]. Editing suites were also described in many of the previously mentioned papers and are embodied in successful commercial products like *Kinetix Character Studio 2.0*(TM) and the kinematics software from *Nichimen Graphics*. Lamouret and van de Panne cut, paste, and modify motions from an example motion database to fulfil animation tasks in [83].

Motion capture for use in animation has been surveyed in [96] and various descriptions of the end product of its use have appeared (see, for example, [93] [66]). The work by Molet *et al.* [105] gives an alternative technique to inverse kinematics for going from sensors on an actor to an animated articulated figure.

As will become clear later, motion interpolation is much simplified if a consistent treatment of DOF angles is generated by the motion capture analysis for a set of similar motions, such as a repertoire of walks. As motion capture data is noisy, robust motion capture processing can be the difference between a process improvement or process nightmare. Chapter 3 will detail the process developed in support of the *Verbs & Adverbs* system. Additionally, the motion capture system was used for the commercial products *Microsoft Baseball3D*(TM) and *Microsoft Precision Racing*(TM).

2.5 Parameterized motion and interactivity

As mentioned in the introduction, control over motion is an essential requirement for interactive animation. Parameters, control variables, are the way an animation is directed from one moment to the next. Motions can either maintain state or be stateless. Motions with state "remember" and can use that knowledge to make judgements regarding future character positions. State-

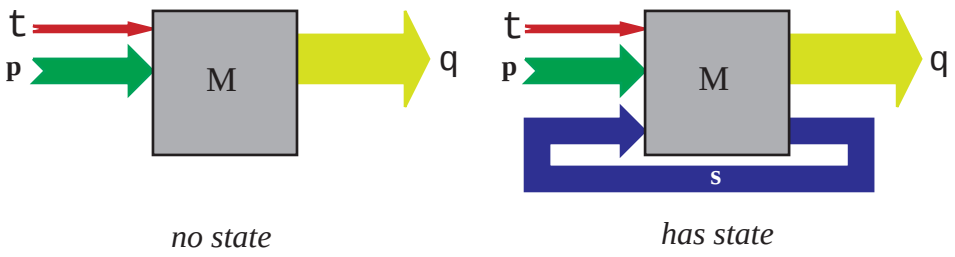

Figure 2.8: Motion **M** produces DOF values Θ given time τ and control parameters **p**. State information **s** may be kept from one iteration to the next.

	Pro	Con
Has-state	Decisions based on history	Hysterisis – history dependent
	Non-repetitive behavior	Unidirectional time (in general)
	Dynamic simulation	Sensitivity to time step
No-state	Rewindable	No dynamic simulation
	Not sensitive to time step	No reasoning about past

Table 2.1: Trade-offs in stateless vs. stated motions

less motions take as input only the time variable and the control parameters. Figure 2.8 depicts these two motion types. There are reasons for each kind of motion. Some trade-offs are listed in Table 2.1. While hysterisis can be used to form more complex actions based on past actions, keeping state can be restriction. It is listed in the con category for that reason. Likewise, the inability for stateless systems to reason about the past is listed as one of their cons.

Parts of the *Verbs & Adverbs* system are stateless (verbs - Chapter 4) and some have modest state (verb-graphs - Chapter 5). One important aspect of the system is that no part is sensitive to the size of time-steps, thus time-steps can grow arbitrary large as system resources are balanced at runtime. Each of these issues will be addressed in the appropriate sections.

2.6 Dynamics-based motion

Forward dynamic simulation generates the behavior of objects in a virtual space given a set initial conditions, forces and torques acting upon the bodies, and descriptions of the dynamic properties of the objects. No one has done more to acquaint the graphics community with the uses of dynamic simulation than David Baraff. His papers detail rigid body dynamics [9] [10], friction [11], non-rigid dynamics [12], efficiency concerns [13] [14], and, most recently, cloth dynamics [15]. Efficiency is a major problem with dynamic simulation and limits its usefulness in the general case. Much of the work in this area has involved finding efficiency gains for special case problems, as Milenkovic [104] did for systems of large numbers of tightly packed objects. Using modal dynamics yields Pentland and Williams [111] some interesting effects and efficiency gains. Some of their objections to other techniques are alleviated by [51]. Metaxas and Terzopoulos [102] describe how to simulate dynamically deformable objects. Witkin, Welch, and Gleicher describe inter-active-time modification to dynamical simulations in [146] and [149].

For dynamics to be used with articulated figures, a dynamics formulation which supports hierarchical linked figures must be used. The formulation of the dynamics equations directly effects the efficiency of the algorithms using to compute it. Le Grange's formulation has been used, but suffers from being exponential in the number of DOFs [147]. Well understood compilation techniques like common subexpression elimination can be used to reduce the size of the resulting expression trees considerably [91]. Complicated, linear-recursive formulations were developed by Featherstone [47], Hollerbach [76], and Balafoutis and Patel [8]. Balafoutis' forms the most efficient formulation to date and uses a tensor formulation with clever identities that reduce the equations considerably. Schröder and Zeltzer [128] and McKenna and Zeltzer [100] used

Featherstone, Liu and Cohen [89] used Hollerbach, and we used Balafoutis and Patel in [124]. That formulation is detailed in Appendix B.

Dynamic simulation, however, presents a serious problem to the animator: uncontrollability. To make his compelling dynamically simulated sequences, for example, Baraff had to iteratively tweak the initial conditions in order to have it achieve a desired result, not an interface useful to your average animator. Interactive animation requires that a system or user be able to direct the outcome of the animation. Barzel, Hughes, and Wood [18] discuss this problem and make a key observation: there is enough uncertainty in the understanding of dynamics properties and simulation such that one can "fudge" the results of a dynamic simulation to achieve a desired effect and still have it look physically "correct". Likewise, Chenney and Forsyth [33] discuss techniques for not simulating things which are not seen by the user. Events not seen by the user are ones which can be heavily modified without breaking the illusion of physical realism and controlling such events could help, though not solve, the problem of directing characters.

Two approaches to overcoming the controllability problem are discussed here. Spacetime optimization and controller optimization are the two primary ways dynamics has been used for generating motion. Spacetime optimization determines DOF trajectories given initial conditions, a set of constraints which frame out the overall animation, and a quality function to be minimized. It is an open-loop process and can therefore be categorized as stateless. Controller optimization couples the forward dynamics process with a feedback controller to achieve a desired animation by modifying a set of control parameters, like torques, from one time-step to the next. optimization

Spacetime optimization

Forward dynamics can be used to calculate the position of a character over time given an initial state and a set of torque functions for the DOFs. It could be expressed as

$$P = f(\mathbf{I}, T)$$

where P denotes a character's position over time, $\mathbf{I}$ the initial position and velocity of the character's DOFs, T the torque functions, and f the forward dynamics process. Typically, everything is known save P.

Inverse dynamics calculates the torque functions required to achieve a given set of position functions. In the terms above,

$$T = f^{-1}(\mathbf{I}, P).$$

Here the position and initial conditions are known. The inverse dynamics process, f^{-1}, calculates the torque functions need to achieve the given P and $\mathbf{I}$.

Isaacs and Cohen [79] described the DYNAMO (DYNamic MOtion) system at SIGGRAPH'87. It incorporated ideas from keyframing and dynamics. The system used keyframing to generate a position function P for a subset of the DOFs in a system. Given these trajectories, torques could be calculated on the other DOFs using inverse-dynamics. Positions could then be calculated for the reamining DOFs using forward dynamics. Figure 2.9 shows a problem solved by this system. The whip handle trajectories are known, specified by an animator with keyframing. Given that, torques could be calculated on the whip DOFs. From these, position trajectories for the whip could be determined.

Witkin and Kass [147] first coined the term "spacetime constraints" in their SIGGRAPH'88 paper. Unlike the work of [79], these constraints could be on position, velocity, or acceleration of the joints. Given the P, a set of force

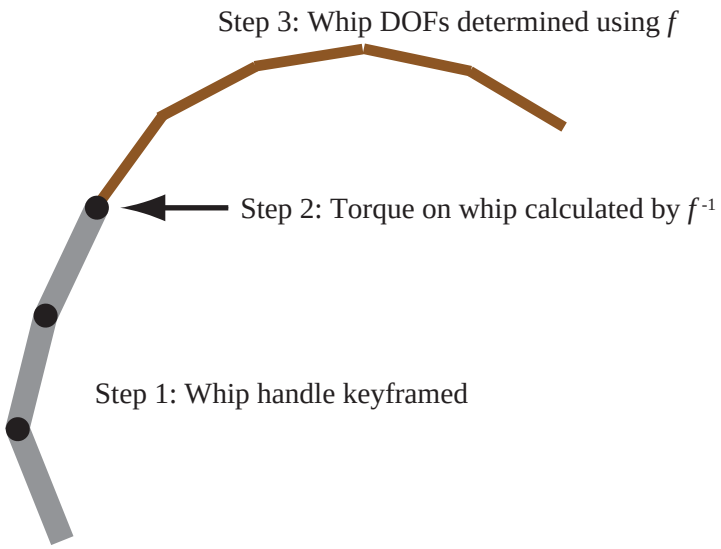

Figure 2.9: Whip motion can be generated using hybrid kinematics and inverse-dynamics

functions T are generated satisfying P. In general, there can be an infinitely large (or empty) set of such functions. The quality of the solution is rated using a fitness functions, typically used energy. A non-linear optimizer is used to drive the used energy towards zero.

Spacetime optimization thus attempts to solve for the position functions, subject to a set of constraints while minimizing a given quality rating.

$$P = \mathcal{S}(\mathbf{C}, O)$$

where P is unknown, $\mathcal{S}$ indicates the spacetime process, $\mathbf{C}$ a set of kinematic constraints, and O a fitness function. Typically, the fitness is proportional to the energy expended, or

$$O(P) \propto T^2$$

the square of the torque functions integrated through time. Spacetime constraint optimization, therefore, often uses inverse-dynamics in its inner loop to rate the quality of a proposed position function, P. The constraints could

be set either by an animator or procedurally by a system. If the optimization process were sufficiently fast, this could be done in real time. For complex human figures, however, this goal has not been achieved, at least for torque-minimization spacetime optimization, due to the complexity of constrained non-linear optimization and the time required to compute the quality function O using inverse-dynamics.

Cohen [35] improves upon the initial work of Witkin and Kass in two ways: representation and interaction. He changed the representation of the DOF curves from piecewise-constant to B-spline, which is continuous and also requires fewer coefficients (in general). By working with "spacetime windows", subregions of time for optimization, he enabled greater animator interaction with the optimization. Liu, Gortler, and Cohen [91] note that by changing to a multi-res representation, namely B-spline wavelets, faster convergence times can be achieved. Speed improvements were also be achieved through compiler techniques such as graph reduction. Liu and Cohen [89] later improve upon this by switching to a better dynamics formulation. Refinements on the interface enable easier animator interaction with the system in [90].

In Rose et al [124] we used the Balafoutis linear recursive dynamics formulation for spacetime optimization of motion transitions for a human figure with 44 DOFs. Quality transitions enabled the joining of two segments of motion capture or hand animated source. The intuition for torque minimal transitioning is from Burdett, Skrinar, and Simon [27]. Joint torques, they found, are a reasonable predictor of metabolic energy. Experience has shown that motion which minimizes metabolic energy looks natural. This leads to the minimization problem:

$$minimize\ e = \int_{\tau_1}^{\tau_2} \sum_j E_j(\tau)^2 d\tau$$

where E_j is the energy function for the jth DOF. A more complete description

of this technique, the full motion equations, and results is found in Appendix B.

Controller optimization

A controller is an object that observes the state of a dynamically simulated figure, directing it to perform some actions in order to meet a goal. Typically, a controller is an object with three kinds of variables. First are the state variables used to sense the system being controlled, such as a linked figure existing in a dynamically simulated world. Control parameters encode the desired state of the system. This state is the goal that the controller, in concert with the simulated figure, is trying to achieve. Control parameters may include things like speed or a desired 3D location, such as in [132] and [141]. The inner workings of the controller (the controller function) transform control parameters and state variables into actions, typically forces applied at the joints. In other words,

$$T = f(\mathbf{S}, \mathbf{C})$$

where T are the torques, f the controller function, and $\mathbf{S}$ and $\mathbf{C}$ the state and control variables. Often, the controller function is itself parameterized by another set of variables known as weights or gains, so

$$T = f(\mathbf{S}, \mathbf{C}, \mathbf{G}).$$

This is an advantageous form since it allows a designer to construct the general form for the controller and then tune it by manipulating the gains until a desired result is achieved. Another option is to use non-linear optimization to automatically find gain settings that cause a controller/figure to perform a certain task.

This task, if stated simply enough, leads naturally to a fitness function that rates the effectiveness of a proposed set of gains. An example fitness

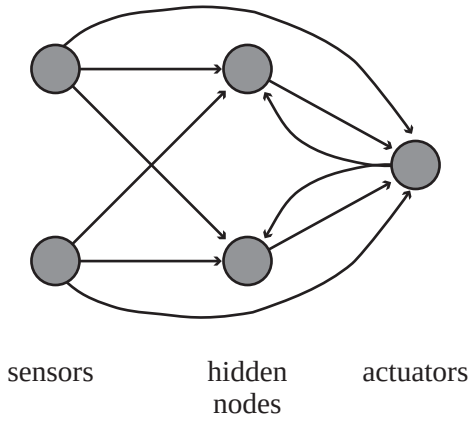

Figure 2.10: Sensor actuator network

function would be the magnitude of the distance traveled by the figure over a certain period of time. The faster the motion, the "better" the controller. Given a fitness function, optimization can be performed using a variety of techniques to find an effective controller.

Sensor actuator networks is a model proposed by van de Panne and Fiume [141]. A sensor actuator network, or SAN, is similar in form to a neural network, though with time delays on propagation of values through the network. A layer of sensors is connected to a layer of hidden nodes, which are in turn connected to a layer of actuators that drive the system, as shown in Figure 2.10. Each of the arcs has a weight that is determined by the controller optimization. The weights for these arcs are the gain variables that tune the controller. A node outputs 1 if its weighted input sum is positive, otherwise it outputs 0. If the actuators receive a positive input, they apply torque to their DOF. Three fitness functions were used in this work. One judged distance traveled, another jump height, and a third location tracking. The optimization method used was to build an initial set of random controllers (i.e. random weights on the arcs) and then perform a hill climbing improvement on the most promising members of the initial set.

A number of locomotion schemes like walking, running, or bounding were discovered using the SAN formulation for a number of different figure topologies. Earlier work [142] used controllers of in the joint-torque state space to generate motions like a back-flip or a parking car. Huang and van de Panne [78] use search to discover the parameters to tune a controller to get their articulated figures to backflip and hop.

Ngo and Marks in *Spacetime Constraints Revisited* [110] describe a genetic algorithm for finding coefficients for a stimulus-response model of motion– a kinematic simulator with feedback control. The controller is structured as a set of stimulus-response pairs which match joints, trigger conditions, and desired response. A stimulus is a function of the form:

$$W\left(1 - \max_j \left(\lambda_j (v_j - v_j^0)\right)^2\right)$$

where the λ_j and v_j^0's are the values (gains) determined by the controller search, v_j the sense variables which gauge the state of the world, and W a normalization factor. The highest valued stimulus at a given step determines which response is activated for the next time-step. The response is a pose towards which the character approaches. The stimulus-response model did not have control parameters. The controllers could perform only one task. The optimization method used a genetic algorithm, which is particularly adept at finding good minima in complex (and potentially discontinuous) non-linear spaces. Ngo and Marks's use of the genetic algorithm for optimization is noteworthy in the animation field. An extensive overview of genetic optimization can be found in Holland [75] and Goldberg [57]. Using this formulation, different locomotion styles for different kinds of characters were generated.

Continuations to Ngo & Marks' initial work are described in Auslander, et. al. [4]. While these authors use the term "spacetime constraints", this is fundamentally different work from what is typically intoned by the phrase.

Their stimulus-response formulation is a closed-loop controller approach to animation, which differs from the open loop energy minimization work of Witkin, Kass, Cohen, *et. al.*, or the kinematic constraint work of Gleicher.

Tu, Terzopoulos, and Grzeszczuk [139] [136] use dynamics, motion controllers, and state machines to simulate artificial fish, including feeding and predation. Grzeszczuk, Terzopoulos, and Rabbie [59] [135] describe learning methods which can be used to generate the needed motion controllers. With an eye to efficiency, Grzeszczuk, Terzopoulos and Hinton's latest work [60] learns to approximate accurate physically correct behavior.

Karl Sims took the controller search idea to a new level by specifying only the simulator and the language in which creatures and their controller could be described in [132] [131]. Using genetic programming, he developed a system which could simultaneously find a creature's structure and its controller in order to select for such traits as speed, jump-height, or the ability to track a user-specified source point in 3D. One interesting discovery he made is that genetically-programmed creatures find bugs in a dynamic simulator. One such creature oscillated at just the right frequency to cause an overflow bug so that it could fly skyward with a healthy dose of free kinetic energy. The creatures Sims found have been described as "spooky" in their behavior, bringing forth visions of digital trilobites. This work is some of the most visionary in animation research, combining dynamic simulation and genetic programming. It has yet to prove effective, however, for human figure animation. The space of creatures and controllers is very large; human figures moving in humanlike ways occupy a tiny subspace. Koza [81] [82] provides the most extensive overview of the field of genetic programming.

Biomechanics and its applications

The work on controller optimization, while quite interesting, has not lead to controllable human figure animation, arguably the most important form of motion. It is unlikely, for example, that a search through creature and controller space is going to discover human motion without guidance. Biomechanical knowledge, therefore, has been integral for designing effective controllers for human-like figures.

Biomechanics is a huge discipline spanning the robotics, biology, neurology, and psychology fields. This section will detail some of the biomechanics work that has proven the most useful to the animation community, together with the successful applications to which this knowledge has been applied. Biomechanics has been used by the animation community in many ways: basic dynamics, musculature, balance, arm positioning, and locomotion. I'll overview a few of the contributions from each area.

Body and balance

Dempster and Gaughran's *Properties of Body Segments Based on Size and Weight* [40] describes the dynamical properties of the human body such as inertia tensor and mass distribution. In macabre detail, this paper describes the cadaver sectioning used to isolate the body segments studied and the methods for calculating the body's dynamical properties. It was used by Hodgins' group in designing their figures for their work in [70].

Chen and Zeltzer [32] developed techniques for simulating a human musculoskeletal system. This work included a model of muscle shape, force, and motion. Scheepers, Parent, Carlson, and May [127] describe a muscle modeling technique which makes use of the animator's eye when defining the musculature of the human form. Wilhelms and Van Gelder [145] show a similar

system which models muscle form and includes a skinning model. These last two works did not seek to simulate the functioning muscle, but rather describe the appearance of the musculature in certain configurations. In that respect, these works are similar to the *Verbs & Adverbs* system in that they model the phenomenon rather than the underlying process which give rise to it.

Badler's *Jack*$^{(TM)}$ system described in *Simulating Humans* [7] and in numerous other publications, makes great use of biomechanical know-how. Military anthropometry data, for example was used to define the range of size for the *Jack* model. Lee, Wei, Zhao, and Badler [88] use knowledge of human strength limits and planning techniques to synthesize motions. Zhao, Tolani, Ting, and Badler [150] describe the use of optimal control to simulate human movements. An example includes getting set up to take a basketball shot. Metaxas [101] uses control techniques together with dynamical simulation of articulated figures to achieve motions like bending, shooting, reaching, and ladder climbing.

Balance is a key issue for a standing figure. Eng, Winter, MacKinnon and Patla [45] show how certain angles are coupled to maintain posture during upper body motions, like far reaches and dips. Forssberg and Hirschfeld [50] perform experiments which show some of the muscle control triggers at work to maintain posture during a differently executed motions. Boulic and Thalmann use information about the mass distribution of the articulated figure together with traditional IK techniques to solve balancing problems in [23].

Arm movement

In his often-cited paper *An Organizing Principle for a Class of Voluntary Motions* [71], Hogan proposed that some human motions can be modeled to optimize for minimal jerk, i.e. minimized rate of change of acceleration. Hogan

and Flash [49] performed experiments to confirm this hypothesis. Hogan described how the human joints can be modeled as *spring-like* objects [73] [74], or systems which relate displacement to force in a possibly non-linear, possibly discontinuous, possibly non-energy-conservative manner. He shows how muscle coordination can be modeled in this framework. Hogan uses feedback control of a dynamical system to control motion in [72].

Mussa-Ivaldi, *et. al.* [107] structure different IK problems and the creature performing them as a network. This network is used together with control techniques to execute the IK problems in a biomechanically sound way.

Bizzi, *et. al.* [19] show that a system more complex than equilibrium points (points where the joint's two muscles exert equal force) must be present for motor control. Mussa-Ivaldi, Hogan, and Bizzi [106] perform experiments to measure the restoring forces for arms which have been displaced from equilibrium point postures. Dean and Brüwer [38] describe the interaction between arm positioning movement and the presence of obstacles and their relation to the usage strategies for the joints.

SIGGRAPH was first introduced to the arm work mentioned in the last several paragraphs by Koga, *et. al.* [80]. This biomechanical knowledge, together with a motion planner was used to simulate a chess game between a human and robot player. Biomechanics aided the realism of the arm motions during chess piece manipulation. An earlier paper by Rijpkema and Girard proposed using a knowledge base of preferred human grasping methods [123]. They created a system for performing high-level grasping motions able to deal with different cup sizes, handle styles, and the like. While not modeling underlying processes, the system used knowledge of the human phenomena to create convincing grasping motions.

Locomotion

J. Rose and Gamble's *Human Walking* [125] is the definitive collection of articles describing the kinematics, dynamics, and pathologies of human walking. The ideas presented there, most notably gait-cycle information, were used to advantage by Bruderlin and Calvert [25] to generate one of the most compelling controllable human walking systems. Similar work [24] handles human running.

McGeer [98] [99] studied walking as a passive rather than active dynamical system. He found that very little force is needed to keep a walk going once started, making walking a very efficient form of locomotion, a important but not surprising result. To show this, he built physical passive robots which could walk with only the kinetic energy potential of an inclined surface to keep them going.

Legged Robots that Balance [118] describes the work of Raibert, Hodgins, and others in building physical robots together with control systems able to locomote through hopping, jumping, and running. Hodgins and Raibert [69] [119] continued this work using improved controllers to create parameterized motion and robot gymnastics.

Jessica Hodgins' group at Georgia Tech has been the most active one in recent years trying to create dynamically simulated human motion, as was demonstrated in their film *Atlanta in Motion*. Hodgins, *et. al.* [70] use control schemes to simulate human sports motions like running, pole-vaulting, cycling, and jumping, many of which were seen in the film. Efficiency is a major concern for dynamic simulation. Carlson and Hodgins describe a system for using levels of detail on the simulator to make maximal use of a system's computing power in [29].

One problem with dynamically simulated motion is the time required

to tune the controller for a particular body. If the body changes size, gender, weight, or age, this can impact the effectiveness of the controller. Hodgins and Pollard [68] interpolate over the space of control laws to facilitate a change in skeleton. The control functions are parameterized by gains which are interpolated for multiple skeletons. By interpolating the control, for an action such as running, they can alter the physical characteristics continuously from child to adult or from male to female.

Faure and Debunne use dynamic analysis of walking data to generate a biomechanical model of walking. Using this, they can then synthesize controllers to generate walks in [46].

Laszlo, van de Panne, and Fiume [86] describe the use of closed-loop feedback control to simulate human walking. They also describe the use of controller interpolation to yield some control over the resulting animations.

At the present time, biomechanically simulated motions are inferior in quality to hand-designed animations. This is due to an inadequate understanding of the human body in motion, not to a defect in the goal of using a first-principles approach to animation. As the biomechanics community produces a richer understanding of human motion and groups like Hodgins' capitalize on that to produce ever-better animation, the appeal of interpolated and procedural animation may decline.

This is not to say interpolated methods are destined to become obsolete. It is unclear, for example, how a first-principles approach would model classic (and unrealistic) animation metaphors like squash and stretch. It may very well be that cartoon dynamics is much less amenable to biomechanical know-how than real dynamics. Interpolated methods like *Verbs & Adverbs*, however, will continue to be useful even in the presence of dynamically incorrect motion.

Synergy-based IK and learning

Encoding knowledge of how much a particular joint contributes to a particular kinematic task, a pairing known as a *synergy*, enables high quality inverse-kinematics. This technique is not subject to the problems of biomechanical implausibility associated with the other IK techniques detailed in Section 2.3. Much of the groundwork for this technique was developed in the Princeton Human Information Processing (HIP) lab and robotics groups. The technique has been considerably strengthened since and is currently being used by *Katrix Inc.*, for projects such as *Ride the Comix*(TM), a location-based game currently at *DisneyQuest*(TM) in Orlando, Florida.

Motor synergies, collections of joint-gain pairs and neural-network learning were used to control Princeton's planar, human-like robot, SLIM, by Lane, Handelman, and Gelfand in [84]. Gelfand, *et. al.* [52] compare three different learning schemes for a human-like robot performing reaching tasks. Gullapalli, *et. al.* [62] use synergies together with a control system, to learn how to perform human-like motions such as position/force control during a sanding motion.

Learning systems may take too long to converge for complex tasks, so Handelman and Lane [65] propose using supervised learning, where a human operator nudges the learning algorithm into the correct region of the search space from which the automated learning system can then converge. Once this is done, more specialized control systems can be employed.

Synnergy-based IK provides a good middle-ground between a purely kinematic system and a dynamic one. Dynamically correct looking behavior is achieved without the expensive hit of a full dynamics simulator.

2.7 Procedural motion

Procedural animation uses code fragments to compute the DOF values at a particular time. The procedures can be as sophisticated as needed in order to provide different motion styles or to react to different conditions of the simulated environment. Unlike dynamic simulation, these procedures can be very efficient, thus allowing for real-time performance on contemporary hardware.

The *Jack*(TM) system has many parts which are procedurally driven, as is detailed in [7]. Phiilips and Badler [114] use inverse kinematics to effect near real time manipulation of human motion. Properties like balance are derived using procedural use of IK.

Blumberg and Galyean [20] use competing procedural behaviors to generate an agent capable of responding to real-time human stimulus.

Latombe's *Robot Motion Planning* [87] describes different schemes for generating path plans. These plan procedures can be used to effectively navigate a creature through an environment.

Perlin's *Improv* system is probably the best known work depending primarily upon procedurally generated animation. Perlin and Goldberg use procedural animation together with noise functions to yield controllable animations in [112] [113]. Perlin also uses blending to handle transitions between different animations. Some of Perlin's recent work in 2- and 3-D facial animation is detailed in [44].

Procedural animation can be used to model actions of groups as well as invidual creatures. Reynolds [120] [122] [121] uses non-global interactions to yield emergent behaviors like bird flocking. Musse and Thalmann describe a model of collision detection schemes in group behavior, so the characters can behave reasonably in a crowd situation in [108].

Cassel, et. al. [30] describe a system for generating conversations, in-

cluding gestures and facial expression changes keyed to the dialog using the *Jack*(TM) system.

Unfortunately, writing animation routines is a task which will alienate most animators, who are the people with the most talent for designing quality motion. Additionally, motion capture is not well suited for use with the procedural approach, though Perlin's noise functions can be used to augment the realism of a motion captured sequence. With these two sources of high-quality animation potentially denied it, procedural animation will not be a complete solution.

Levels of control for procedural motion is a common theme. Tasks can be broken into smaller, more tractable subtasks. Cohen describes a three-level interaction scheme and a learning methodology in [34]. Thalmann and Thalmann [95] describe a decomposition of an animation system into MCMs, or motion control methods. Badler [5] proposes a model, called a parameterized action representation, for decomposing action and intends to build it upon his PATNET (PArallel Transition NETwork) abstraction. Cremer, Kearney, and Papelis describe HCSM, a framework for control in [36]. It uses hierarchical finite state machines to control a driving simulator. Its framework, developed by the operations community, may prove useful to the animation community. Indeed, the commercial product *Motivate*(TM) made by *The Motion Factory* uses a hierarchical finite state machine approach to control and simulation.

2.8 Interpolated motion

The final classification of parameterized animation is interpolated animation, the class into which *Verbs & Adverbs* fits. Interpolated motions are constructed by blending between and extrapolating from pieces of motion source, either hand crafted or motion captured. The idea of altering existing animation to

produce different characteristics is not new. Unuma *et al.* [140] use Fourier techniques to interpolate and extrapolate motion data. Amaya *et al.* [2] alter existing animation by extracting an "emotional transform" from example motions which is then applied to other motions. For example, "anger" from an angry walk is applied to a run to generate an "angry" run. *Verbs & Adverbs* does not follow this approach in that it does not apply characteristics of one motion to another, but instead assumes that the initial library of motions contains these emotions. Unlike these two techniques, *Verbs & Adverbs* method is not based in the frequency domain and thus can handle non-periodic motions which earlier methods fail to capture.

Bruderlin and Williams [26] use multitarget interpolation with dynamic timewarping to blend between motions, and displacement mappings to alter motions such as grasps. Witkin and Popović [148] present a similar system for editing motion capture clips. The former work is in the same spirit as ours, and addresses many of the same difficulties, specifically the necessity of selecting appropriate key times for interpolation and the consequent need for time warping. One difference between the two approaches lies in the choice of interpolation techniques: Bruderlin and Williams use multiresolution filtering of joint angles in the frequency domain, whereas our technique decouples solution representation from interpolation mechanism.

Both Wiley and Hahn [144] and Guo and Robergé [63] produce new motions using linear interpolation on a set of example motions. Both techniques require $O(2^d)$ examples, where d is the dimensionality of the control space. The *Verbs & Adverbs* technique using radial B-splines requires $O(n)$ examples to establish the baseline approximation and $O(n^3)$ to compute the resulting answer. To compare, a Delaunay triangulation of the data would require $O(n^{\mathrm{ceil}(d/2)})$ to compute, when $d \geq 3$.

The *Verbs & Adverbs* system also differs from the work of Wiley and Hahn by using non-uniform time-scaling based on key events. While the uniform time-scaling of Wiley and Hahn obviates the need for an animator to select structurally similar poses during motions, it assumed that the repertoire of motions being interpolated between must be very similar in time. When this assumption is violated, oddities in the motion can result. Wiley and Hahn also reparameterize and sample their motions on a multidimensional grid and then perform simple interpolations at runtime. This requires computation and storage exponential in the number of parameters.

Additionally, neither Wiley and Hahn nor Guo and Robergé discuss the blending of subsets of examples, which would arise when new examples are placed "between" old examples as a designer refines the interpolated motion space. Our technique, on the other hand, is refined by more examples as required. As *Verbs & Adverbs* uses an approximation method based upon radial B-splines with compact support to perform the interpolation, examples have limited effect over the space of animations, thus ensuring that subsets of the examples are used at runtime as appropriate.

An important distinction in *Verbs & Adverbs* is that the interpolation is performed simultaneously in real-time over multiple dimensions, such as emotional content and physical characteristics. Although we apply the techniques to interpolating trajectories characterized by coefficients of spline curves, the methods presented here are also applicable to coefficients in the Fourier and other domains. It may also be possible to apply similar ideas to control the parameters of a physically based model.

In the context of autonomous agents, *Verbs & Adverbs* presents a back-end for applications such as games, the *Improv* [113] system, and the work proposed by Blumberg and Galyean [20]. The high level control structures

	Pro	Con
Spacetime dynamics	Stateless Physical interaction with environment	Mocap and hand animation difficult to reuse Lack of biomechanics limits human realism Slow (relatively)
Dynamics and controller	Deep understanding of motion Physical interaction with environment	Mocap and hand animation difficult to reuse Must maintain state Controllers difficult to design Slow
Procedural animation	High level parameterization Stateless or stated efficient	Mocap and hand animation difficult to reuse Realism dependent upon procedures
Interpolated animation	High level parameterization Stateless efficient	Potentially loose parameterization Realism dependent upon source material

Table 2.2: Strengths and weaknesses of controllable animation techniques

found in such applications are capable of selecting verbs and adverbs while the work we present here provides the low level animation itself. Thus, we create the motion in real-time for "directable" creatures as discussed by Blumberg and Galyean.

2.9 Conclusions

Each of the four major parameterized animation techniques has strengths and weaknesses. Table 2.2 illustrates this for each of the four styles of animation studied here. Given this context of human figure animation, the *Verbs & Adverbs* system can be well described. It is an interpolated technique and as such needs one thing before all others: *example motions*. That is the subject of the next chapter.

[21] [64] [77] [103] [117] [3] [126]

Chapter 3

Acquisition of examples

The most compelling animations in the 2- or 3-D realm are done using time tested hand animation techniques augmented with computer assistance. The computer, for the main part, is used simply for process enhancement, not process reengineering. Motion capture for animation grew out of the biomechanist's desire to track human motion in order to better understand its form. Hardly a panacea, motion capture for animation is expensive and time consuming. Its ability to create "realistic" motions, however, can make it worth the effort.

Procedural and dynamically simulated animations, like the work of Perlin [112] [113], Badler [7], and Hodgins [69] [119] [70] [68], have an advantage over key-framed or motion-captured animations in that they are controllable at run-time and can therefore be *interactive*. Unfortunately, these techniques require an animator to express their vision in a way quite alien to their training. As was mentioned earlier, this dissertation details a method, called *Verbs & Adverbs*, for creating controllable motion from motion capture segments or hand-crafted 3D animation clips. This allows animators to continue working in a way comfortable to them and leverages the effectiveness of expensive motion capture data. Before the *Verbs & Adverbs* technique can be applied, however, the designer must first obtain a set of good *examples*.

In this chapter, the **example** is examined. An example is an animation clip in a particular form. A set of not overly strict requirements must be met for a motion segment to be considered an example. Motion capture processing is examined in the context of these requirements. A motion formalism is introduced and a number of example-related topics are explored. A motion editing system can be built using this formalism to assist the designer in the construction of examples.

3.1 What is a motion-snippet?

In traditional animation, sequences are created with the goal of making scenes and later, films. Interactive animation systems typically use a smaller unit of animation, a self-contained sequence which could be called a *motion snippet*. Examples of snippets would include *punch*, *fall*, *reach*, or a *walk-cycle*. Interactive systems involve ways to mix and match these snippets to form seamless, reactive animation sequences.

Rich, parameterized snippets, *verbs*, form the basic unit from which a controllable interactive system is built. Verb construction is the primary thrust of this chapter and the next. Verb construction requires structured, rather than random, motion snippets, i.e. good examples.

The skeleton, together with values for its degrees of freedom, defines a static pose of the character. A motion is a changing over time of pose. In procedural terms, a motion $\mathbf{M}$ is a function in the time domain which returns a set of degree of freedom values.

Take, for example, a basic walk. Figure 3.1 shows a skeleton, this one affectionately named George. George has 44 DOFs: 6 root DOFs and 38 internal revolute DOFs. Figure 3.2 shows a pair of DOF curves for a basic walking motion designed for the George skeleton, namely those of the knees. Note the

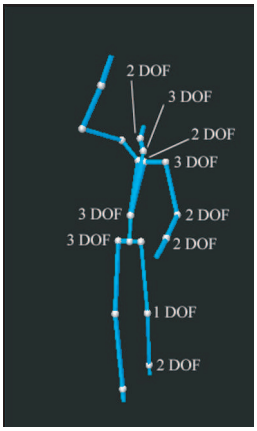

Figure 3.1: George

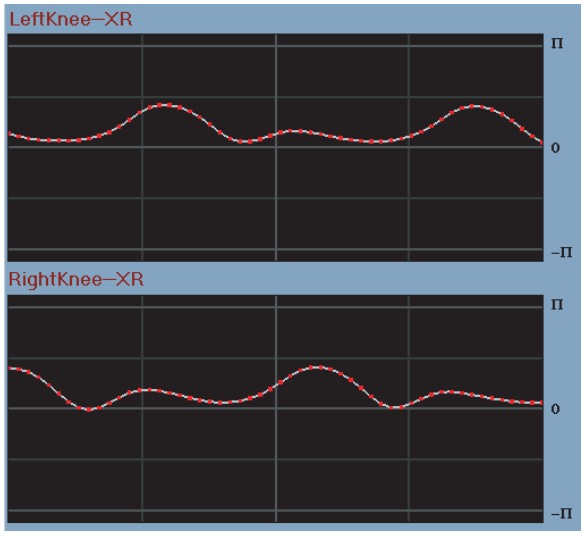

Figure 3.2: Knee DOFs for a walk

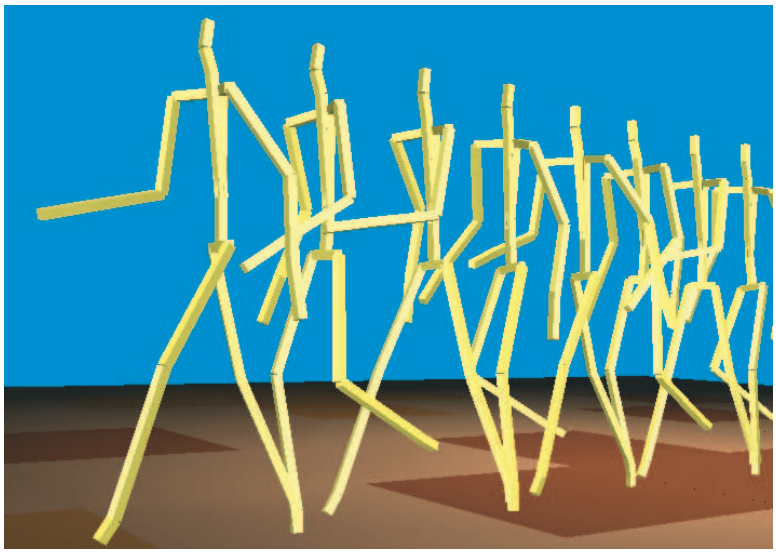

Figure 3.3: A simple walking motion

out of phase similarity of the curves, consistent with a biomechanical analysis of the human walk cycle [125]. Figure 3.3 shows a time-lapse picture of this walking motion.

3.2 The skeleton's DOF ordering

The hierarchy of George's DOFs is shown in Figure 3.4. Of particular interest is the ordering of the first six DOFs that define the position and orientation of the root relative to the global frame of reference. As the multiplication of rotation matrices is not commutative, achieving the same global effect with different orderings requires different sets of values. The choice of this order will affect many of the algorithms presented in this chapter and of those in Chapter 5. The order used in this work is the **X**, **Y**, and **Z** translations, followed by the **Y**, **Z**, and **X** rotations. The algorithms presented in this dissertation will be described in terms of this ordering.

The **Y** axis, in computer graphics, is typically the one pointing "up" from

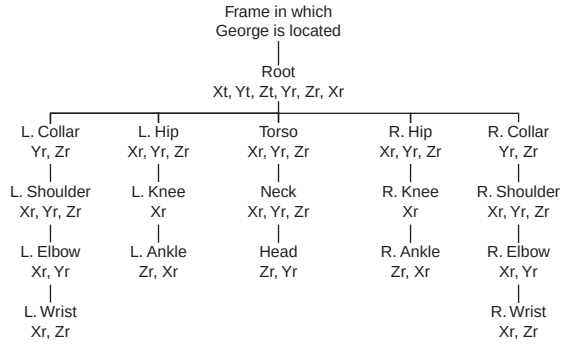

Figure 3.4: The skeleton hierarchy

the ground plane. Thus, motions for a biped where the character is standing, walking, or running, having the **Y** axis rotation first allows easy orienting of the direction of the character. The initial DOF order is shown in Figure 3.5. This DOF will hereafter be referred to as the character's *heading*.

3.3 Motion's relation to the skeleton

The skeleton is like a marionette with many strings to pull, those being the degrees of freedom. Keyframing and motion capture, two techniques for designing simple animation segments, were described in Sections 2.2 and 2.4. Animations of this form have some advantages over procedural animations; they easily leverage the talents of an animator or physical actor. One disadvantage, however, is their close connection to the skeleton for which they were designed. A motion, when applied to a skeleton for which it was not designed, is likely to produce unacceptable results. Most often this becomes evident with visually obvious violations of kinematic constraints, such as foot slide. Foot slide is when the foot moves globally when it should logically be supporting the weight of the character– planted solidly upon the virtual floor.

Figure 3.6 shows the problems caused by switching the skeleton's proportions and then applying an unmodified motion to it. The picture on the

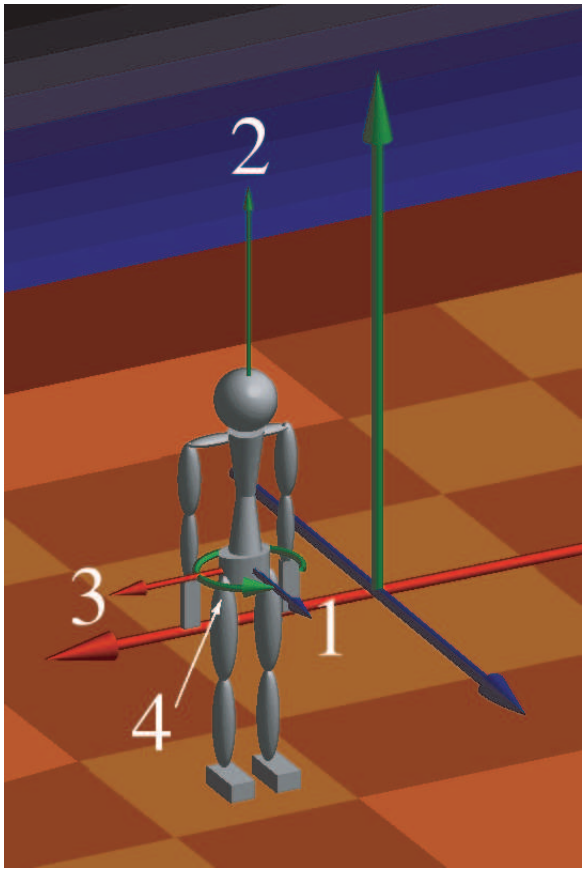

Figure 3.5: The initial DOF ordering

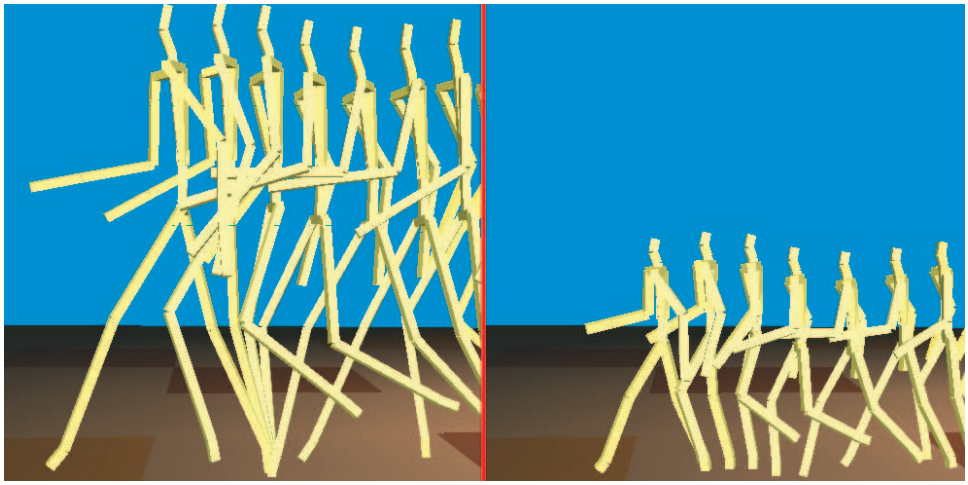

Figure 3.6: Different skeleton results in foot slide

left shows the walk as it was meant to appear. Notice the stability of the foot placement during the support phases of the walk cycle. With a smaller skeleton, the motion's once solid support phases are replaced by skidding. Gleicher [55] [56] has studied this problem extensively and has developed fast methods for fixing such problems based on spacetime constraints.

3.4 Time

Time is a simple notion in the day-to-day sense. Animation systems, however, often require a more sophisticated treatment of time, separating "time" into different types. *Verbs & Adverbs* uses four different kinds of time.

The most general form of time, *clock-time*, is the common-sense notion of time. For this dissertation, clock time can be considered to have begun a long time ago, at $-\infty$, and will continue indefinitely towards $+\infty$. 0 can be placed at any arbitrary moment. How about now? Clock-time will not often be used in this dissertation, but when it is, the symbol $\mathcal{T}$ will be used.

Animation-time is the first major time used here and is the most closely related to clock-time. In general, animation time ranges over a finite region of the clock-time timeline and will be denoted by τ. An animation can be placed in time to play at any region of the animation-time timeline. Two special animation-times τ^s and τ^e mark the start and end times of the animation. It is different from clock time in that clock time is assumed to have a global calibration, i.e. everybody knows what time it is.

Verb-time, T, always starts at 0 and ends at T^d, the duration of the animation. Verb-time is used to normalize animations on the timeline; no shortening or lengthening of the animation's duration takes place. T^d, therefore, is equal to $\tau^e - \tau^s$.

So far, all the time types have been roughly equivalent up to an offset

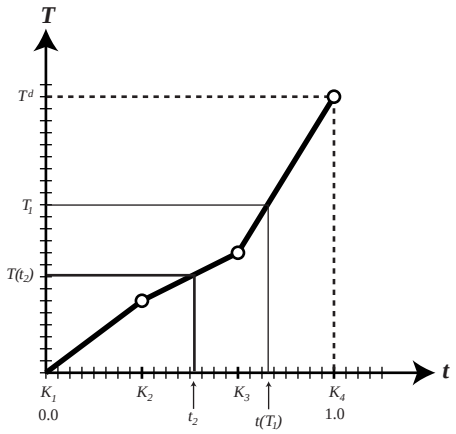

Figure 3.7: Connection between canonical-time, t, and verb-time, T

in clock-time. *Canonical-time*, t, is different and has domain and range from $[0 \ldots 1]$ always. Canonical time is extremely important to this dissertation as it is used to time the structural elements of a motion. A walk motion, for example, could be described by events like heel-strike, toe-off, swing-phase, etc. An extensive overview of the structure of human walking is found in [125].

These structural milestones are defined by a set of user-specified *key-times*. All motions have two implied key-times at 0 and T^d, thus tying the verb-time and canonical-time timelines together. If the motion has no other key-times, canonical time acts as percentage time, thus

$$T = t \cdot T^d.$$

In the presence of three or more key-times, however, the mapping is more complex. Figure 3.7 depicts the mapping for a motion with four key-times (two implied and 2 user-specified). Given a verb-time T, we can project that into the canonical timeline using the equation

$$t(T) = \left((m-1) + \frac{T - K_m}{K_{m+1} - K_m} \right) \frac{1}{NumKeyTimes - 1} \tag{3.1}$$

for the largest m such that $T > K_m$ and keeping in mind that $t(0) = 0$. In other words, at each key-time m, $t(K_m) = \frac{m-1}{NumKeyTimes-1}$. For instance, at the

third of four key-times, $t(K_3) = \frac{2}{3}$ as the key-times will fall at 0, $\frac{1}{3}$, $\frac{2}{3}$, and 1. Between key-times, t is linearly interpolated.

Given a moment in canonical time, t, we can calculate the associated T by linearly interpolating from the key-times between which a particular t falls:

$$T(t) = K_m + \left(\frac{t - t(K_m)}{t(K_{m+1}) - t(K_m)} \cdot (K_{m+1} - K_m) \right) \tag{3.2}$$

where t is in the range $[t(K_m) \ldots t(K_{m+1}))$ and keeping in mind that

$$T(1) = \mathrm{T}_i^d = K_{NumKeyTimes}.$$

The symbols "τ", "T", and "t" are used interchangeably to indicate value and projection from one kind of time to another. "T", for example, stands for a particular verb-time. $T(t)$ is the projection of the canonical time t to the verb-time timeline. While this may seem to clash, it helps to minimize the number of symbols used in this dissertation, a goal which will become clearly important as this chapter continues.

3.5 What makes a good example?

As will be shown in Chapter 4, interpolation is at the core of the *Verbs & Adverbs* system. Interpolation imposes a number of constraints on the example motions to be interpolated. These include:

1. similar motion structure,
2. same skeleton,
3. continuous DOF-curves,
4. anatomically-plausible use of joint angles,
5. similar use of joint angles for similar motions, and

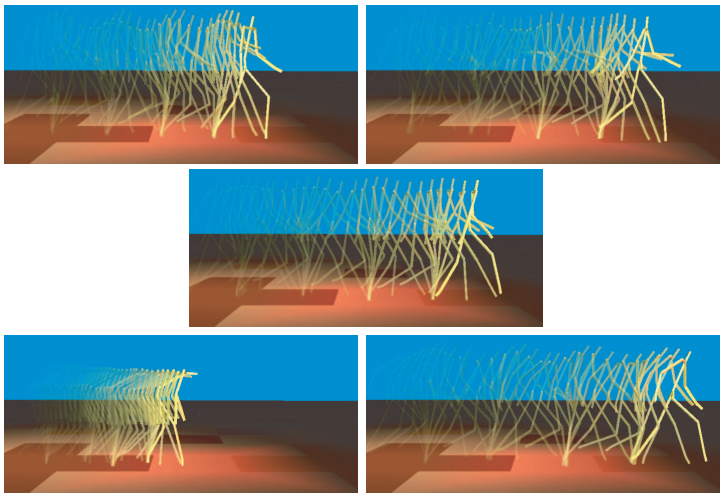

Figure 3.8: Different walking styles

6. same initial placement and heading at the beginning of the example.

7. in canonical timeline

8. identical DOF function encoding schemes

Many of the functions for motion editing to be described in Section 3.9 can be used to prepare an example from raw motion data.

Structural similarity

Of primary importance is the structural similarity requirement. Creating a controlled walk, for example, requires a repertoire of walks exhibiting various desired walking styles. These walks all need to start and end at the same points in the walking-cycle. Likewise, the actor needs to swing his or her arms in a consistent off-phase behavior, and not perform any spurious motions, like a head-scratch, fidget, or angry fist raised against the world. Within this restriction, a great variety of motions is found, as shown in Figure 3.8. Figure 3.9 shows to structurally dissimilar walks.

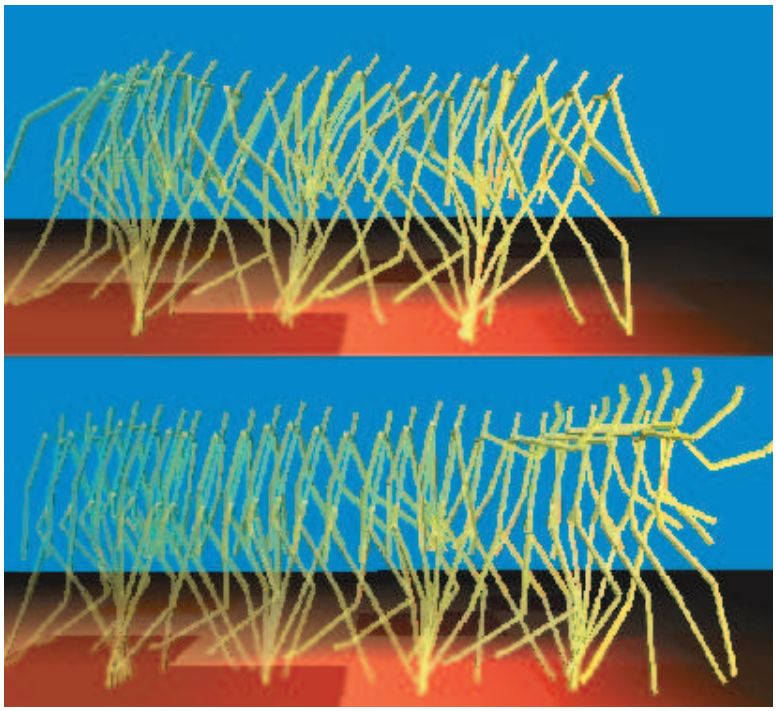

Figure 3.9: Two walks not displaying structural similarity

As was discussed in Section 3.4, motions have a set of key-times, instants when important milestones take place. All motions to be combined to form a verb must have the same set of key-times in the same order. Given this key-time information, all the examples can be put in the same canonical timeframe, ensuring interpolation of corresponding instants in each of the motions. Annotation of key-times and other structural constraints will be further described in Sections 4.2 and 4.4.

Same skeleton

As was discussed in Section 3.3, applying a motion to a skeleton for which it was not designed leads to unpredictable results. As all of the examples are going to be applied to one skeleton in the interpolation step, the examples need to be designed for the same skeleton. Using the techniques of the previously

mentioned section, however, can help a designer transform a set of motions for differing skeletons into a set for one.

Continuous DOF-curves

DOF-curves are required to be continuous. Often basic motion capture is delivered as a collection of poses, each representing a discrete moment in time when the motion was captured and analyzed. Since human figures are highly redundant systems and given the incomplete knowledge of the human musculoskeletal system and its flexibility range [21], there is no guarantee that the joint angles will be continuous from frame to frame even for slow deliberate motions with little change from one frame to the next.

This is an issue since the *Verbs & Adverbs* system samples the DOF-curves at times other than the frame times. This is not itself a fundamental problem– we could use a piecewise-constant representation. Non-continuous curves, however, are often associated with non-similar curves for similar motion. This restriction will be described shortly. For this reason, the desire to sample the curves off the frame times, and the desire to use curve representations other than piecewise-constant, DOF-curve continuity is required.

Section 3.7 describes how motion capture data analysis can be performed in order to ensure continuous DOF curves.

Anatomical plausibility

A degree of freedom for our skeleton is an approximation of some motion which a human can perform. Given that, the DOF must be set in a manner consistent with a human's abilities.

For example, take the wrist curling motion, shown in Figure 3.10. The extent of the allowable curl is bounded by θ_{min} & θ_{max}. Anatomically plausible

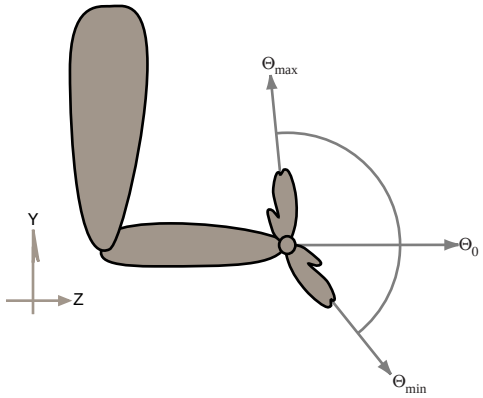

Figure 3.10: Wrist curl extent

settings of this DOF fall between those two extents.

This model of plausibility is a simplification. The "real" values of θ_{min} & θ_{max} are not static, but are functions of other DOFs at the same body site, such as the wrist bend about the **Y** axis, or of motions at other parts of the body. This too is a simplification in that it does not handle self-intersection with the body or the fact that the human body is only approximated by a hierarchical collection of rigid limbs connected by rotary joints.

Why, therefore, is this simplified model being used? The biomechanics community has yet to furnish a complete kinematic model of the human form. When and if they do, it is not clear that it will be low dimensional enough to be of practical use to animators or in motion capture analysis. The shoulder complex, for example, was studied by Maurel, et. al., [97], and had 18 DOFs per arm and shoulder, an impractically large number for many animation purposes. Simplified models have been the most useful ones for the animation community to date.

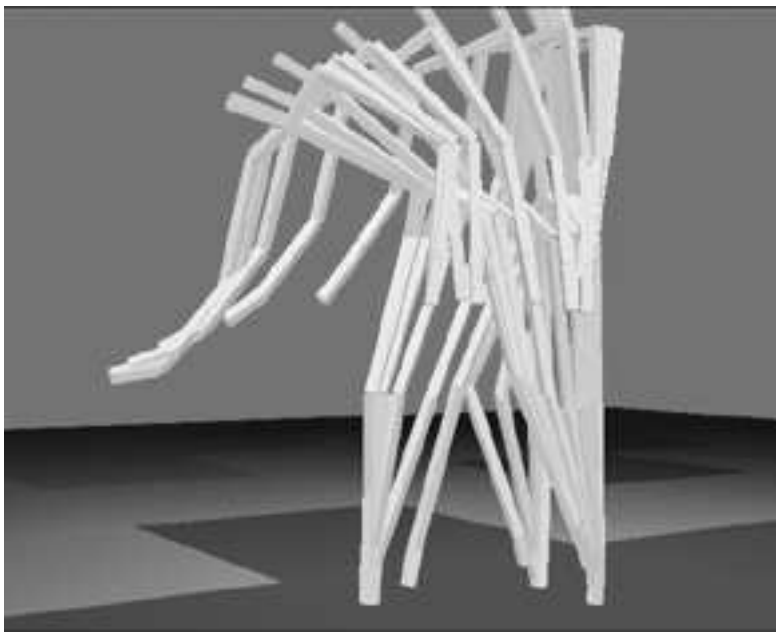

Figure 3.11: A medium reach

Similar use of joint angles

Redundant manipulators (redundant kinematic figures) can be posed in a globally unique configuration using a potentially infinite number of DOF settings. *Verbs & Adverbs* requires that DOF values be similar for similar global configurations. Motion capture analysis can be done in a way which supports this. The technique will be detailed later in Section 3.7.

Figure 3.13 shows two sets of shoulder curves for two different reaching motions shown in Figures 3.11 and 3.12. The curves were generated using a basic motion capture analysis technique. Note that both sets pass the (admittedly liberal) plausibility test. As they are different motions, they should have differing curves, but it is reasonable to assume that if the motions are similar overall, then their DOF-curves should be somewhat comparable. Blending these two motions yields the strange result shown in Figure 3.15.

Using the motion capture method to be detailed in Section 3.7 yields

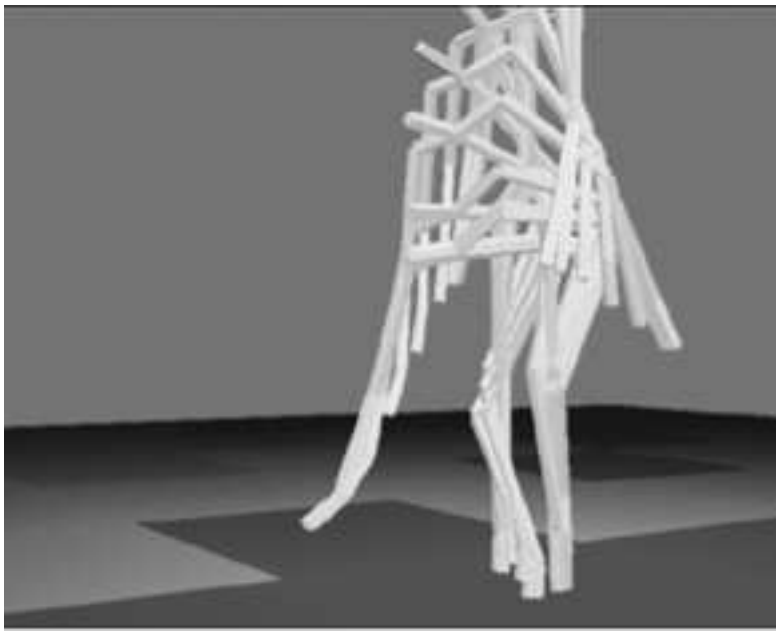

Figure 3.12: A low reach

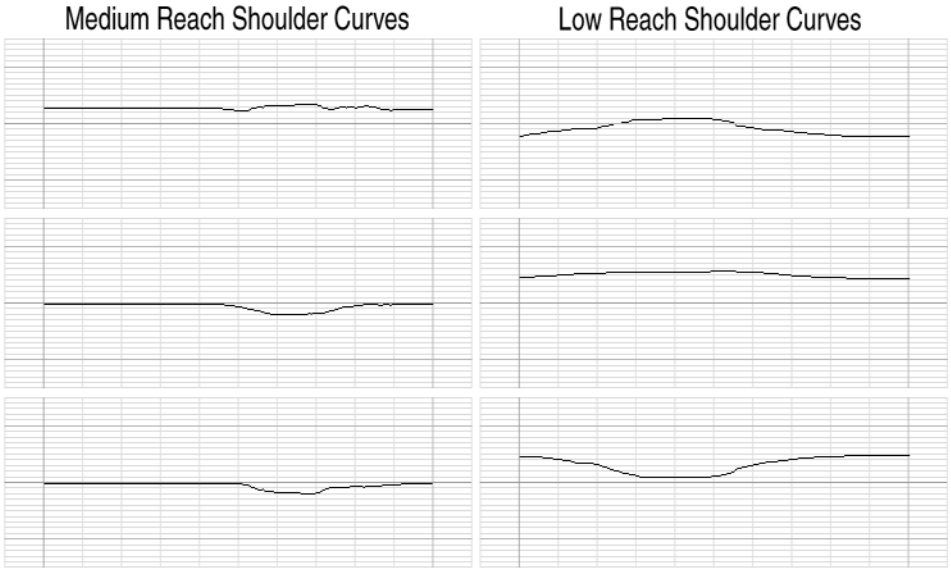

Figure 3.13: Dissimilar use of joint angles

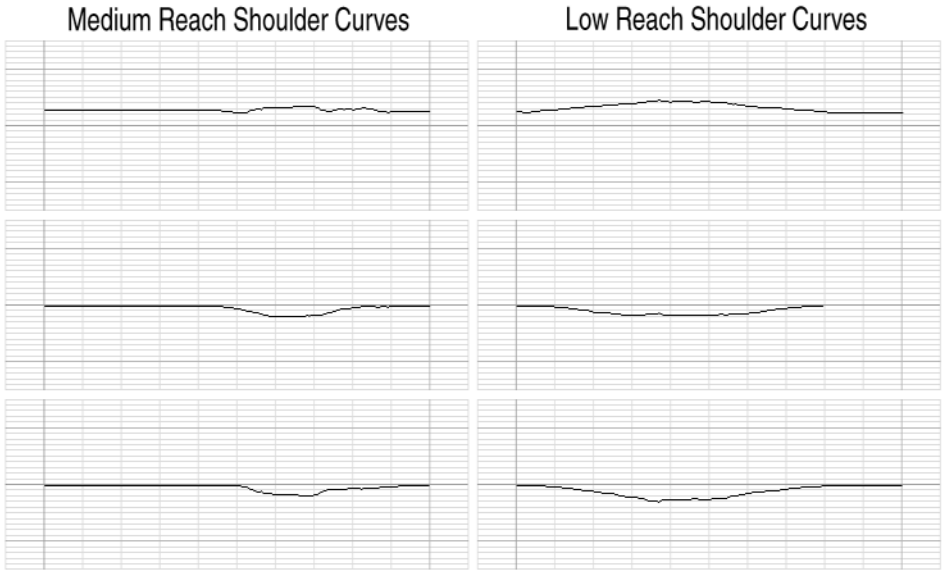

Figure 3.14: Similar use of joint angles

Figure 3.15: Poor motion blend due to dissimilar DOF curves

Figure 3.16: Good motion blend with similar DOF curves

consistent use of joint angles, as shown in Figure 3.14. These result in the exact same reaching motions shown in Figures 3.11 and 3.12. When blended, the result is much more reasonable, as shown in Figure 3.16.

Placement and heading at $t = 0$

The initial placement and heading requirement holds that all motions begin at the same $[\mathbf{X}, \mathbf{Z}]$ location in space oriented along the same heading. That is, at $t = 0$, the first, third, and fourth DOFs of each example must match.

A useful side product of the initial DOF ordering introduced in Section 3.2 is the ability to reorient and reposition the character using a simple algorithm. Thus, any motion can be put into the proper form to satisfy the placement and heading requirement.

Assume that all the examples for a verb are to start at $[x, z] = [0, 0]$ and heading along the positive $\mathbf{Z}$ axis, i.e. the global $\mathbf{Y}$ rotation is 0. Usually, the clips start at an arbitrary location and head off in an arbitrary direction. This is shown as motion $\mathbf{M}$ in Figure 3.17. To be useful as an example, $\mathbf{M}$ must be

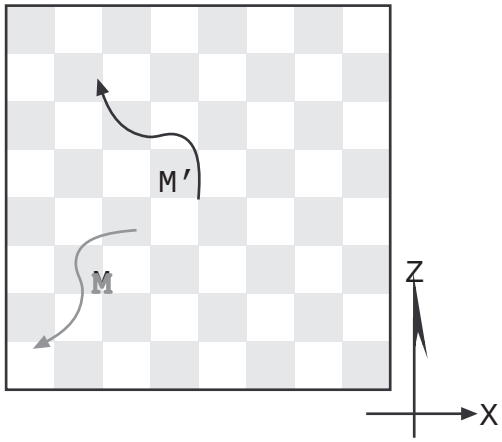

Figure 3.17: Reorienting the character

transformed into **M**'. The method is shown by the following algorithm:

Algorithm 3.1 Steps to reorient

```
for each frame i
{
```

$$\hat{\mathbf{Y}}_i^r = \mathbf{Y}_i^r - \mathbf{Y}_0^r$$

$$\hat{\mathbf{X}}_i^t = \cos(-\mathbf{Y}_0^r)(\mathbf{X}_i^t - \mathbf{X}_0^t) + \sin(-\mathbf{Y}_0^r)(\mathbf{Z}_i^t - \mathbf{Z}_0^t)$$

$$\hat{\mathbf{Z}}_i^t = -\sin(-\mathbf{Y}_0^r)(\mathbf{X}_i^t - \mathbf{X}_0^t) + \cos(-\mathbf{Y}_0^r)(\mathbf{Z}_i^t - \mathbf{Z}_0^t)$$

```
}
```

A "t" superscript indicates a translational DOF and an "r" a revolute one. $\mathbf{X}^t$, $\mathbf{Z}^t$, and $\mathbf{Y}^r$ are arrays of the first, second, and fourth DOF values for each frame of **M**. $\hat{\mathbf{X}}^t$, $\hat{\mathbf{Z}}^t$, and $\hat{\mathbf{Y}}^r$ are the adjusted DOF values corresponding to motion **M**'. The algorithm adjusts each frame by translating so that the initial frame is at $[0, 0]$ and rotating by the negative of the initial heading, thus ensuring that the initial heading is 0. A simple extension of this algorithm can reposition a motion so that at time $T = 0$ it is in a certain position heading

in a certain direction. This is shown in the following algorithm:

Algorithm 3.2 General reorient and reposition

Given $\hat{\mathbf{X}}_0^t, \hat{\mathbf{Y}}_0^t, \hat{\mathbf{Z}}_0^t, \hat{\mathbf{Y}}_0^r$,
$\mathbf{X}_0^t, \mathbf{Y}_0^t, \mathbf{Z}_0^t, \mathbf{Y}_0^r$,
$\mathbf{X}_T^t, \mathbf{Y}_T^t, \mathbf{Z}_T^t, \mathbf{Y}_T^r$, construct $\hat{\mathbf{X}}_T^t, \hat{\mathbf{Y}}_T^t, \hat{\mathbf{Z}}_T^t, \hat{\mathbf{Y}}_T^r$

$$\Delta\mathbf{X}^t = \mathbf{X}_T^t - \mathbf{X}_0^t$$
$$\Delta\mathbf{Z}^t = \mathbf{Z}_T^t - \mathbf{Z}_0^t$$

$$\Delta\mathbf{Y}^t = \hat{\mathbf{Y}}_0^t - \mathbf{Y}_0^t$$
$$\Delta\mathbf{Y}^r = \hat{\mathbf{Y}}_0^r - \mathbf{Y}_0^r$$

$$\hat{\mathbf{X}}_T^t = \cos(-\Delta\mathbf{Y}^r)\Delta\mathbf{X}^t + \sin(-\Delta\mathbf{Y}^r)\Delta\mathbf{Z}^t + \hat{\mathbf{X}}_0^t$$
$$\hat{\mathbf{Z}}_T^t = -\sin(-\Delta\mathbf{Y}^r)\Delta\mathbf{X}^t + \cos(-\Delta\mathbf{Y}^r)\Delta\mathbf{Z}^t + \hat{\mathbf{Z}}_0^t$$

$$\hat{\mathbf{Y}}_T^t = \mathbf{Y}_T^t + \Delta\mathbf{Y}^t$$
$$\hat{\mathbf{Y}}_T^r = \mathbf{Y}_T^r + \Delta\mathbf{Y}^r$$

The $\hat{\mathbf{X}}_0^t, \hat{\mathbf{Y}}_0^t, \hat{\mathbf{Z}}_0^t$, and $\hat{\mathbf{Y}}_0^r$ terms indicate the desired start time state of the character. The $\mathbf{X}_0^t, \mathbf{Y}_0^t, \mathbf{Z}_0^t$, and $\mathbf{Y}_0^r$ terms contain the actual start time state of the character. The natural state for time T is contained in the $\mathbf{X}_T^t, \mathbf{Y}_T^t, \mathbf{Z}_T^t$, and $\mathbf{Y}_T^r$ terms. The $\mathbf{Y}$ translation is unaffected by the rotation about the $\mathbf{Y}$-axis. The $\mathbf{Y}^r$ DOF will be rotated to maintain the illusion that the motion started heading off in the direction $\hat{\mathbf{Y}}_0^r$, when in reality it started off with the heading $\mathbf{Y}_0^r$. The $\mathbf{X}$ and $\mathbf{Z}$ translations are shifted and rotated to keep the motion consistent with the desired initial configuration. This algorithm will

often be used in the remainder of this dissertation to line up motions one after another.

A question to ask is whether an alternate handling of the root would render the reprojection details moot. A velocity based approach would be to store some or all of the initial DOFs as velocity curves, rather than positional curves, and then integrate the values to yield the final result. One such initial ordering is $\dot{\mathbf{X}}^t$, $\dot{\mathbf{Y}}^t$, $\dot{\mathbf{Z}}^t$, $\dot{\mathbf{Y}}^r$, $\dot{\mathbf{Z}}^r$, $\dot{\mathbf{X}}^r$. A simple integration from one time to the next would yield the appropriate changes in positioning the skeleton, handling all of the work done by the reprojection. Unfortunately, this method has two serious drawbacks which make it unappealing as a solution. First, integration is expensive. Integration by Gaussian quadrature, for example, must sample the curves at a number of distinct times in order to build up a reliable answer. The longer the time, the more samples needed for reasonable confidence in the solution.

The primary fault with this method is not efficiency, however, but accuracy. Unless the integration is performed from the beginning of time for each frame, some information gathered in one frame will be used to calculate the next frame and this leaves open the possibility of compounding error. For example, buildup of error for the $\mathbf{Z}^r$ rotation could cause the character to spin off its feet and onto its head. Only favorable luck would keep this from happening.

Keeping a subset of the initial DOFs as velocity curves and some as position curves can make for a useful hybrid approach. Such an approach might have an initial ordering of $\dot{\mathbf{X}}^t$, $\mathbf{Y}^t$, $\dot{\mathbf{Z}}^t$, $\dot{\mathbf{Y}}^r$, $\mathbf{Z}^r$, $\mathbf{X}^r$. Buildup of error can appear in the $\mathbf{X}$ and $\mathbf{Z}$ translation, as well as the $\mathbf{Y}$ rotation component of the root's motion. The character is moving in the $\mathbf{X}$-$\mathbf{Z}$ plane and rotating about the $\mathbf{Y}$ axis as it turns, i.e. these values are typically unbounded. Such errors,

which are likely to be small from any given frame to the next, will likely go unnoticed. This hybrid scheme would probably be a good initial configuration and is worthy of use. It was not used as the scheme for the *Verbs & Adverbs* system as described in this dissertation, however, and will not be discussed further.

3.6 Hand-designed examples

The *Verbs & Adverbs* system currently accepts both hand-animated and motion-captured examples. In Chapter 2, keyframing, the process that animators use to make 3D animations, was overviewed. Making *examples* using keyframing systems like *SoftImage*$^{(TM)}$ and *3D-Studio/Max*$^{(TM)}$ requires placing a few restrictions upon the animator in order to satisfy the example criteria introduced earlier in this chapter. Currently, *Verbs & Adverbs* has been set up to work with *SoftImage*$^{(TM)}$.

At the current time, *Verbs & Adverbs* requires one skeleton per verb, i.e. one skeleton per set of examples. Some traditional animation metaphors, like squash and stretch, therefore, cannot be used as they alter the size of the skeleton. One goal of our current research is to remove this restriction to enable the design of cartoonish verbs. The animator must also make sure to maintain the structural similarity of the examples. A confused walking motion, for instance, cannot convey its confusion with a head scratch, but must rather convey emotion using timing and posture only. The rest of the example restrictions can be satisfied automatically and are things with which the animator need not be concerned.

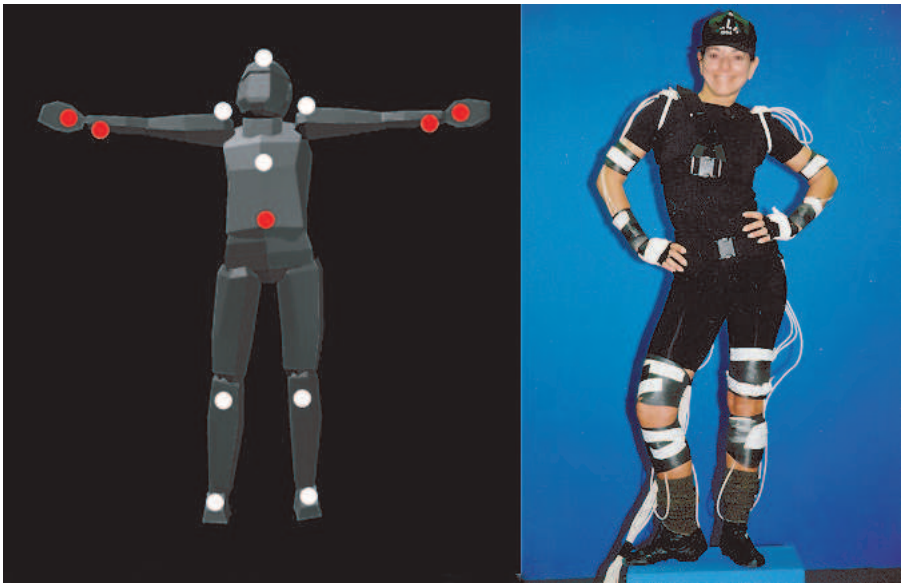

Figure 3.18: Placement of sensors for motion capture analysis

3.7 Motion captured examples

A more complete description of our motion capture process was detailed at Eurographics Workshop on Computer Animation and Simulation 1997 in *The Process of Motion Capture: Dealing with the Data* [21].

Most of the data used in this dissertation was motion captured. Our motion capture data was generated from an *Ascension MotionStar*$^{(TM)}$ system input directly into a 3D modeling and animation program, *SoftImage*$^{(TM)}$, at capture time. Data is was sampled at 144Hz. This high sampling rate is advisable when fast motions, such as sports motions, are captured; using slower sampling rates for such motions can often produce problems in the inverse kinematics phase, since there is less frame-to-frame coherence. Actors are suited using from 13 to 18 six-DOF sensors, the typical locations of which are shown in Figure 3.18. Statistical analysis is used to ignore gross errors and outlying data in the skeleton generation and inverse-kinematic phases of the analysis process.

Given a motion capture dataset, the goal is to construct an articulated, hierarchical rigid body model. The George skeleton (Figure 3.1) is the topology used, though it must sized to fit different motion capture actors. The first task is to extract the best limb lengths from the motion capture data. Once the scale of the segments is determined, an inverse-kinematics solution is calculated to determine the joint angles for the figure. Our inverse-kinematics routine uses penalty functions to constrain the joint angles to approximate a human's range of motion.

Motion capture data is noisy and often contains gross errors. The source of the noise is primarily the magnetic sensors themselves, although we note that in our experience optical data is as noisy. The size of the skeleton is determined by finding the distances of the translated joint locations over a motion or repertoire of motions. Using the simple arithmetic mean to compute these distances results in answers distorted by a few gross errors. Unfortunately, editing the data by hand to remove outliers is impractical. As an example, gross errors in fast motions such as throwing may, for a frame or two, give a distance between the elbow and wrist of over three meters. A robust statistical procedure, such as described in [64], can be used to remove the outliers resulting in accurate measurements of the skeleton. Outliers can also be tagged and then ignored in the inverse-kinematic stage of the analysis.

Fitting the cleaned data to the skeleton

Once the hierarchical model has been determined using robust statistical analysis, each frame of data must be analyzed to produce a set of joint angles. To obey the example criteria, the resulting DOF curves must make consistent use of DOFs across multiple motions and must be continuous. A piecewise-linear or B-spline representation of the DOF is used. Care is taken to insure that these

curves contain no extreme accelerations between frames. In contrast, many commercial data sets often contain discontinuities in the rotational data from frame to frame which makes off-frame sampling impossible without representing the joints as quaternions and using quaternion interpolation, as described in Shoemake [129].

The data sets yield information about many areas of the body, giving us a highly constrained kinematic problem. As mentioned in Section 2.3, such problems can be solved using a non-linear optimization technique which seeks to minimize the deviation between the recorded data and the hierarchical model. A modification to the technique presented in Zhao and Badler [150] is used.

Recall that Zhao's fitness function to minimize is defined as

$$\begin{aligned} F(\Theta) = \sum_{j \in J} & w_{p_j}(P_j(\Theta) - \hat{P}_j)^2 + \\ & w_{O_{0,j}}(O_{0,j}(\Theta) - \hat{O}_{0,j})^2 + \\ & w_{O_{1,j}}(\Theta) - \hat{O}_{1,j})^2 + \\ & w_c C_j^2 \end{aligned} \tag{3.3}$$

where Θ is the set of joint angles for the set of joints J, $P_j(\Theta)$ the global location of the jth joint given Θ and $\hat{P}_j$ the recorded joint position from the capture phase. $O_{0,j}(\Theta)$ and $O_{1,j}(\Theta)$ are two vectors defining the global orientation of the joint with $\hat{O}_{0,j}$ and $\hat{O}_{1,j}$ being the recorded orientations. Two vectors are used, as, together with their cross product, they will form a coordinate system. The quantities w_{p_j}, $w_{O_{0,j}}$, and $w_{O_{1,j}}$ are scalar weights which can be tuned on a DOF by DOF basis to achieve better results. Additionally, we employ a joint angle constraint term, C_j, in the form of a penalty function. Joint angle constraints for humans have been measured and can be found in the biomechanical literature, such as in Houy [77]. C_j is calculated as the

distance of Θ_j from the range $[\Theta_{j_{\min}} \ldots \Theta_{j_{\max}}]$.

The quasi-Newton BFGS optimization technique [53] is used to solve the system and uses the gradient of the fitness function, given by

$$\begin{aligned}\frac{\partial F_j}{\partial \Theta_i} =& 2w_{p_j}(P_j(\Theta) - \hat{P}_j)(u_j \times d_{ji}) + \\ & 2w_{O_0}(O_{0,j} - \hat{O}_{0,j})(u_j \times O_{0,j}) + \\ & 2w_{O_1}(O_{1,j} - \hat{O}_{1,j})(u_j \times O_{1,j}) + \\ & 2w_c C_j\end{aligned} \tag{3.4}$$

where u_j is the effective axis of rotation about the jth DOF in global terms and d_{ji} the vector from the jth DOF to the ith DOF. If the data is relatively non-noisy and the skeleton is well formed, this technique will work well. It can produce poor results if these conditions are not present. Robust statistics helps to insure these conditions by making the best skeleton and by marking data points which are considered outliers. Since a hierarchical description of a skeleton is a biological simplification and since non-linear optimization does not guarantee finding a global minimum, this analysis can still fall into an insufficient local minimum if the starting guess for the optimization is far from the desired solution.

If we assume a good starting guess for the first frame, then a good IK solution for that frame will very likely be found by Zhao and Badler's minimization. As the sampling rate on the motion capture system is high, the pose from one frame to the next will not change much. The solution for one frame, therefore, provides an excellent starting guess for the next frame. For many motions, this technique works admirably. It suffers if the data and the skeleton are mismatched near where the skeleton goes through a singularity or where the data points are too far apart in time for a given motion's velocity. Additionally, it will suffer if it never converges to a good solution for an initial frame. Over the shoulder reaching, fast motions, and motions where the arm

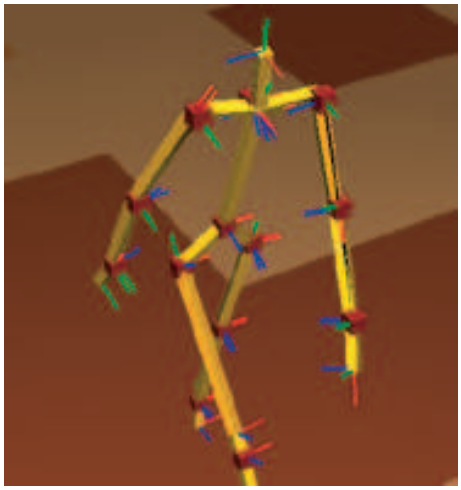

Figure 3.19: Fitting motion capture data to the skeleton

is extended to its limit are examples. If this happens, the solution can jump over to another local minimum and stay there. This behavior is not desirable, as it will likely break the DOF consistency requirement of examples.

Analyzing a walk motion of 6.7 seconds duration at 30 frames per second required 306 seconds on a Pentium 133 machine with 4389 BFGS iterations for satisfactory convergence of the solution. A selected frame showing the fit of the skeleton (yellow) to the data (black) is shown in Figure 3.19. Notice that the fit is extremely good and shows only a slight discrepancy in the left arm. The resulting walk motion is shown in Figure 3.20.

Bootstrapping

A further refinement of the motion capture analysis presented here is to use motions to bootstrap one another by providing good starting guesses to the BFGS optimization. The assumption for this technique is that many motions of similar structure are to be analyzed, i.e. a set of examples, and that motions of similar structure will have relatively similar DOF curves. Such data sets might include reaches, runs, walks, etc.

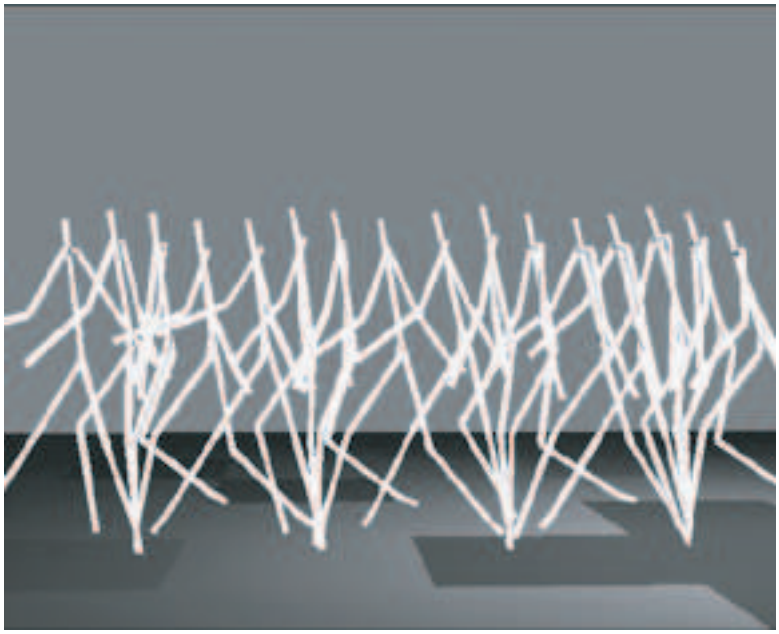

Figure 3.20: Walk motion from motion capture data

Assume that there is a motion $\mathbf{M}$, a set of DOF curves, for a motion in a set of examples. If we have a motion capture dataset for a structurally similar motion (Section 3.5), the joint angles will be similar to those for motion $\mathbf{M}$. The main difference between the solution for the new desired motion $\tilde{\mathbf{M}}$ and $\mathbf{M}$ will be a time warping to account for differences in the phrasing (relative timing) between $\mathbf{M}$ and the captured data. Thus a scaling in time on the data sets is needed. We mark a set of correspondence times, key-times (Section 3.4), in $\mathbf{M}$, and in the data set. We time-warp $\mathbf{M}$ and then use that as the starting guess for the inverse kinematics optimization described earlier. Section 3.9 describes piecewise-linear time-warping and Chapter 4 describes the use of key-times in depth.

Thus, this technique will not propagate errors, whereas in the previous technique a bad starting guess may result in a bad solution, which can propagate from frame to frame. Near singular conditions can also cause the previous technique to jump from one local minimum to another, which is then

propagated to all the remaining frames. With this technique, similar motions will make similar use of their joint angles when analyzed, which is key for using motion capture for a set of examples or for a technique like Witkin and Popović [148]. Note that this technique requires operator intervention to mark the key times, and thus it is only employed for groups of data-sets when the previous technique did not work.

The details of the *Verbs & Adverbs* interpolation will not be described until the next chapter, but in short, DOF values are generated through interpolating (blending) between multiple example motions. Section 3.5 stated that examples must make similar use of DOFs for similar motions. Why is this?

DOFs for the two reaching motions from the naive motion capture analysis were shown in Figure 3.13. Note that the curves are quite different even though they achieve a similar global effect. Figure 3.15 showed an unsuccessful blend between the two motions. The motions did not analyze reasonably due to noise in the sensors and inadequacies of the skeletal model and the joint angle constraints model. Using the medium reach motion as a reference motion and a time-warp to align it as the starting guess, we can obtain a more consistent use of shoulder angles, as shown in Figure 3.14. Notice that, up to a time warp, these sets of shoulder angles are very similar. Figure 3.16 showed a simple blend which yielded reasonable results.

3.8 The motion formalism

Early in this chapter, the *motion* was introduced. Motions, denoted by the symbol "**M**", were defined to be short, finite-duration animations. Later sections introduced the three varieties of time (Section 3.4) and detailed a special class of motions called examples– motions which obey a set of restrictions

(Section 3.5). $\mathbf{M}_i$ refered to the ith example in a set. In more abstract terms, however, $\mathbf{M}_i$ is just a particular motion. $\mathbf{M}_{i'}$, therefore, would refer to a different motion.

What do these two motions have in common? They may be encoded using different DOF curve representations, for example. Each would have a set of DOF curves. As will be seen later in this section, they may be even less closely related than that. What all motions share is the ability to answer a common set of questions or, in other words, *all motions share a common interface.* This interface is called the *motion formalism.* The remainder of this chapter will detail this formalism and describe a set of concrete objects that implement it. This formalism will tie together the notions of DOF-functions, time, and a number of other ancillary values like key-times.

Motions were defined to contain a number of data items, such as key-times and inverse-kinematic constraints. More formally, a *motion* is an entity which can respond to a number of questions and which can produce a number of values. Table 3.1 details the simple values a motion needs to be able to produce. To take an example, T_i^d is the duration of the ith motion in terms of verb-time. Subscripts i will indicate motion and j DOF. Superscripts will differentiate between different values of the same type, such as τ_i^s for animation-time start and τ_i^e for animation-time end.

As was introduced in Section 3.4, the three types of time which must be considered are animation-time, verb-time, and canonical time. Motions, therefore, must be able to project time from one type to another. As stated previously, the symbols "τ", "T", and "t" are used both in functional and non-functional forms. $\tau(T)$, for instance, will project the verb-time T into the animation-timeline while τ alone simply indicates a particular animation-time. $\tau_i(T)$ indicates the projection of the verb-time T into the animation-timeline

data-item	*values*	*description*
K_{im}	$[0 \ldots +\infty)$	mth key-time for motion $\mathbf{M}_i$
τ_i^s	$(-\infty \ldots +\infty)$	start-time of motion $\mathbf{M}_i$
τ_i^e	$(-\infty \ldots +\infty)$	end-time of motion $\mathbf{M}_i$
T_i^d	$[0 \ldots +\infty)$	duration of motion $\mathbf{M}_i$
$\mathbf{p}_i$	$\Re^{NumAdverbs}$	adverb values for motion $\mathbf{M}_i$

Table 3.1: Basic motion values: key-times, time-bounds, duration, and adverbs for motion $\mathbf{M}_i$.

θ is the DOF position function for a motion, which takes verb-time as a parameter. $\theta_{ij}(T)$, therefore, would return the value for the jth DOF for the ith motion at verb-time T. Likewise, the DOF velocity and acceleration functions are indicated by $\dot{\theta}_{ij}$ and $\ddot{\theta}_{ij}$. Script-I, $\mathcal{I}$, is used to indicate the active constraints placed upon a motion at particular time. Script-D, $\mathcal{D}$, is used to indicate the DOF-usage for the DOFs at a particular time, as will be described in Section 3.9 under the sub-heading "composition". Table 3.2 indicates the different functions supported by the motion formalism, their parameters, results, and meanings.

Unless otherwise indicated, the duration of a motion $\mathbf{M}_i$ is calculated simply as

$$\mathrm{T}_i^d = \tau_i^e - \tau_i^s.$$

Likewise, the functions $T_i(t)$ and $t_i(T)$ are functions of key-times and can be calculated by using Equations 3.1 and 3.2 unless otherwise indicated.

The motion-formalism is an abstraction and such abstractions beckon to computer scientists to define many concrete implementations, this being the core idea of object oriented programming. This idea was used effectively in the *Verbs & Adverbs* system. In Section 3.9, some other kinds of objects

function	*result*	*range*	*description*
$\tau_i(T)$	τ	$(-\infty \ldots +\infty)$	verb-time to animation-time
$T_i(t)$	T	$[0 \ldots +\infty)$	canonical-time to verb-time
$t_i(T)$	t	$[0 \ldots 1]$	verb-time to canonical-time
$\theta_{ij}(T)$	position	$\Re$	return DOF position
$\dot{\theta}_{ij}(T)$	velocity	$\Re$	return DOF velocitie
$\ddot{\theta}_{ij}(T)$	acceleration	$\Re$	return DOF acceleration
$\mathcal{D}_{ij}(T)$	DOF-usage	{**required**, **defined** **undefined**}	return DOF-usage on a DOF-by-DOF basis
$\mathcal{I}_i(T)$	constraints		IK constraints active at T

Table 3.2: Motion functions: time projections and kinematic operators

supporting the formalism will be introduced. In Chapter 4, verb construction will be detailed. Verbs, not surprisingly, will been shown to be simply another object supporting the motion formalism.

Since some key concepts are introduced at different times, the motion formalism grows as this dissertation progresses. The full motion-formalism and all of its different concrete implementations is detailed in Appendix A for easy reference.

Basic motions

So far, the only kind of motions which have been introduced are the simplest ones, those defined by a set of DOF-curves. These motions will be called *basic-motions*. A basic motion $\mathbf{M}_i$ is defined as

Definition 1 *basic motion* $\mathbf{M}_i = \left\{\mathbf{K}_i, \mathcal{I}_i, \mathbf{p}_i, \mathrm{T}_i^d, C_i\right\}$

where the key-times $\mathbf{K}_i$, constraints $\mathcal{I}_i$, adverbs by $\mathbf{p}_i$, duration T_i^d, and DOF-curves C_i are DOF-curves represented using methods such as described in Section 2.2. We use C_i to indicate the distinction between the $\theta_{ij}(T)$ operator, which must be able to respond to queries about any DOF j, many of which may not be defined for a particular basic-motion.

Basic motions implement the formalism as follows:

$$
\begin{aligned}
\tau_i^s &= 0 \\
\tau_i^e &= \mathrm{T}_i^d \\
\tau_i(T) &= T \\
\theta_{ij}(T) &= \begin{cases} C_{ij}(T) & \mathcal{D}_{ij}(T) = \textbf{required} \\ 0 & \text{otherwise} \end{cases} \\
\dot{\theta}_{ij}(T) &= \begin{cases} \dot{C}_{ij}(T) & \mathcal{D}_{ij}(T) = \textbf{required} \\ 0 & \text{otherwise} \end{cases} \\
\ddot{\theta}_{ij}(T) &= \begin{cases} \ddot{C}_{ij}(T) & \mathcal{D}_{ij}(T) = \textbf{required} \\ 0 & \text{otherwise} \end{cases} \\
\mathcal{D}_{ij}(T) &= \begin{cases} \textbf{required} & \forall \text{ DOFs } j \text{ in } C_i \\ \textbf{undefined} & \text{otherwise} \end{cases}
\end{aligned}
$$

To ensure that basic motions may be considered as examples, they are defined to begin at $T = 0$. Using the reprojection algorithm (Algorithm 3.2), proper placement and heading at $T = 0$ can be ensured.

3.9 Functional composition of motions

Now that the motion formalism has been defined and the basic motion type detailed, other kinds of motions can be described which combine to form an editing tool. These "other" motions embody key editing concepts like clipping, mirroring, time-warping, and the like. If the basic motion is thought of as a

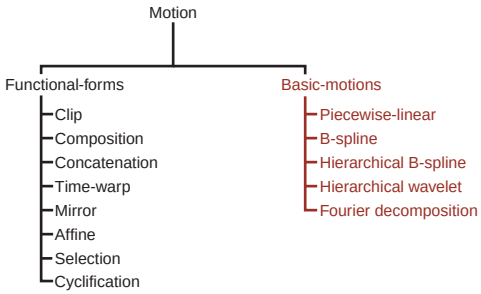

Figure 3.21: A hierarchy of motion types

particular kind of motion in the object-oriented sense, the different motion types form an object hierarchy, as shown in Figure 3.21. So far, the reader has been introduced to the sub-tree shown in red, rooted at *basic-motions*.

This motion hierarchy and the functional relationships it encapsulates form a language for expression of motion. Rather than being separate entities, motions are organized as an acyclic graph of relationships, the parameters of which can be altered at any time, thus allowing freedom to edit, try new ideas, and undo mistakes. As each type of motion object must be able to support the motion formalism introduced in Section 3.8, posing a question to one node in the graph is answered by posing a cascade of questions further down in the graph. These answers are combined or arbitrated between to produce a final result. The operative variable, which is manipulated from one level of the graph to the next, is verb-time, T.

Motion clipping

The simplest kind of edit a designer may wish to perform is to clip out a piece of a motion. This could be done by extracting the relevant coefficients from the original motion, but this approach has two drawbacks. First, the decision is final. Short of redoing the operation against the original motion, there is no way to change the clip points, unless it is to make the clip even shorter. Sec-

ondarily, multiple clips of one motion which overlap will unnecessarily contain duplicate information.

By maintaining a relationship between the clip and the motion being clipped, both pitfalls are avoided. The term "clip" refers here to this relationship, not the operation of clipping out coefficients. The clip-relationship is embodied in a clip-type motion, the second kind of motion from the hierarchy. A clip, $\mathbf{M}_i$, of another motion $\mathbf{M}_{i'}$, therefore, is defined as

Definition 2 *clip motion* $\mathbf{M}_i = \{t^s_{i'}, t^e_{i'}, \mathbf{M}_{i'}\}$

where $t^s_{i'}$ and $t^e_{i'}$ mark the start and stop region of the clip in motion $\mathbf{M}_{i'}$. These values are expressed in the canonical timeline of motion $\mathbf{M}_{i'}$.

The duration of the clip is derived by projecting the clip times from the canonical timeline of the motion being clipped into its verb timeline, so

$$\mathrm{T}^d_i = T_{i'}(t^e_{i'}) - T_{i'}(t^s_{i'}).$$

We define the clip to begin at $\tau = 0$, so

$$\begin{aligned} \tau^s_i &= 0 \\ \tau^e_i &= \mathrm{T}^d_i \end{aligned}$$

where T^d_i is calculated as shown.

The DOF position, velocity, and acceleration functions are calculated by offsetting T by the projected clip start:

$$\begin{aligned} \theta_i(T) &= \theta_{i'}(T + T_{i'}(t^s_{i'})) \\ \dot{\theta}_i(T) &= \dot{\theta}_{i'}(T + T_{i'}(t^s_{i'})) \\ \ddot{\theta}_i(T) &= \ddot{\theta}_{i'}(T + T_{i'}(t^s_{i'})). \end{aligned}$$

Likewise, the DOF usage function, $\mathcal{D}_{ij}$, is implemented as

$$\mathcal{D}_{ij}(T) = \mathcal{D}_{i'j}(T + T_{i'}(t^s_{i'})).$$

The DOF position function is actually more complicated than $\theta_i(T) = \theta_{i'}(T + T_{i'}(t^s_{i'}))$. In order for a clipped motion to be an example (Section 3.5), it must start at the origin pointed along the **Z**-axis. This can easily be ensured using Algorithm 3.2 for general repositioning and orienting of the root. For simplicity's sake, however, this and other motion types will be specified as if this step were avoided. Note, however, that at $T = 0$, all motions to be used as examples must be in the standard position and orientation. Potentially, we would like all kinds of motions to be able to serve as examples.

A clip of a motion may not encompass all of the key-times of the original motion, but even if the clip does, all the key-times must change to account for the change in duration. Let us assume that all the internal key-times of motion $\mathbf{M}_{i'}$ are also in the clip $\mathbf{M}_i$. A simple example of this is shown in Figure 3.22. The key-times in the figure would be determined as follows:

$$\begin{array}{lcll} \mathbf{K}_{i1} & = & 0 & \text{all motions start at } T = 0 \\ \mathbf{K}_{im} & = & \mathbf{K}_{i'm} - T_{i'}(t^s_{i'}) & \text{for internal keytimes (here } 2 \ldots 4) \\ \mathbf{K}_{i5} & = & \mathrm{T}^d_i & \text{the duration of the clip} \end{array}$$

The equation also works for the case where the clip does not encompass all the key-times in the original motion, save that the m subscript must be adjusted to account for the missing key-times, so for the internal key-times,

$$\mathbf{K}_{im} = \mathbf{K}_{i'm'} - T_{i'}(t^s_{i'})$$

where m' is the adjusted subscript. Once the key-times are determined, the time mapping functions, $T_i(t)$ and $t_i(T)$, work as developed in Section 3.4.

The kinematic constraint function, $\mathcal{I}_i(T)$, is designed to exclude all those constraints which fall outside the clip region.

An example clip motion is shown in Figure 3.23. It shows a walk that has been clipped from $t_s = 0.44s$ to $t_e = 1.33s$. The original motion ran from

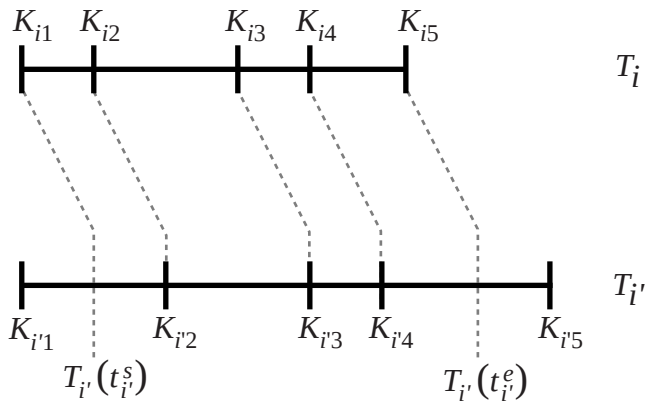

Figure 3.22: Shifting the key-times for a clip motion

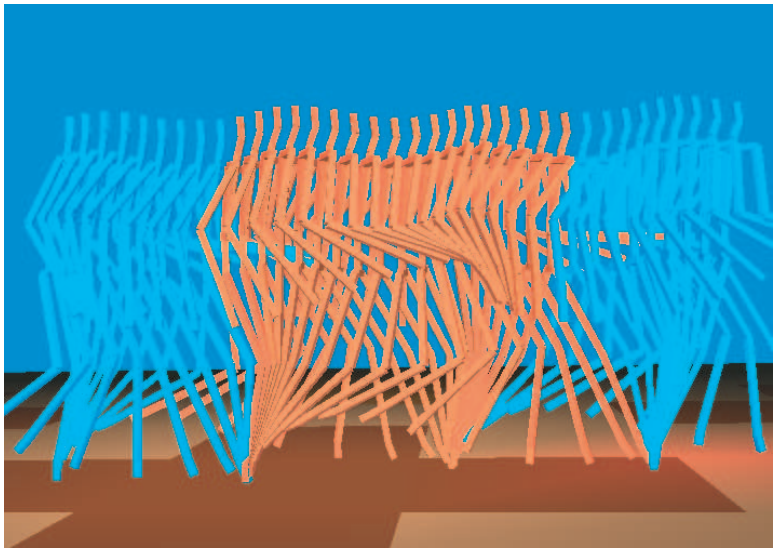

Figure 3.23: A clipped walk motion

$\tau = 0s$ to $1.76s$. The clipped region is shown in red and the unclipped excess shown in blue. Note that the clipped region begins at the standard position and orientation as required by Section 3.5

Affine

Shifting and scaling time is an important operation and is done using an *affine* relationship. This operation enables the designer to put multiple motions in the correct temporal relationship to one another for later compositing, as will be described later in this section. Using an affine, a designer can also perform an overall speed up or slow down to achieve a desired effect. The affine motion is defined as

Definition 3 *affine motion* $\mathbf{M}_i = \{\tau_i^s, s, \mathbf{M}_{i'}\},\ s > 0$

where τ_i^s is the moment in animation-time when the motion is set to begin and s a scaling factor which changes the overall duration of the motion.

The motion formalism is implemented using the following formulae:

$$\begin{aligned}
\tau_i^s & \quad \text{given} \\
\mathrm{T}_i^d &= s \cdot \mathrm{T}_{i'}^d \\
\tau_i^e &= \tau_i^s + \mathrm{T}_i^d \\
\tau_i(T) &= \tau_i^s + s \cdot T \\
T_i(t) &= s \cdot T_{i'}(t) \\
\mathbf{p}_i &= \mathbf{p}_{i'}
\end{aligned}$$

$$
\begin{aligned}
\mathbf{K}_{im} &= s \cdot \mathbf{K}_{i'm} \\
\mathcal{D}_{ij}(T) &= \mathcal{D}_{i'j}\left(\frac{T}{s}\right) \\
\mathcal{I}_i(T) &= \mathcal{I}_{i'}\left(\frac{T}{s}\right) \\
\theta_i(T) &= \theta_{i'}\left(\frac{T}{s}\right) \\
\dot{\theta}_i(T) &= \dot{\theta}_{i'}\left(\frac{T}{s}\right) \\
\ddot{\theta}_i(T) &= \ddot{\theta}_{i'}\left(\frac{T}{s}\right).
\end{aligned}
$$

As the affine changes only time, the position and heading at $T = 0$ are unchanged.

Figure 3.24 shows 3 motions, a walk at normal speed in the middle, a slow down at top, and a speed up at bottom. Each was multiply-exposed at 0.05 second intervals for a duration of 1 second. None was shifted in time.

Time-warping

In addition to shifting and scaling a motion in time, an animator may want to alter the relative durations of some pieces of a motion. This is the primary operation which projects a motion in and out of the canonical timeline, for example. This type of motion is known as the *time-warp*.

Time-warping relies upon a function, $\mathcal{U}$, to warp time by slowing or speeding it as a motion progresses. The general form of the time-warp motion is defined as

Definition 4 *time-warp motion* $\mathbf{M}_i = \{\mathcal{U}, \mathbf{M}_{i'}\}$

where $\mathcal{U}$ is a monotonically increasing function with domain and range [0..1]. The $\mathcal{U}$ function warps the relative duration of the segments of the motion.

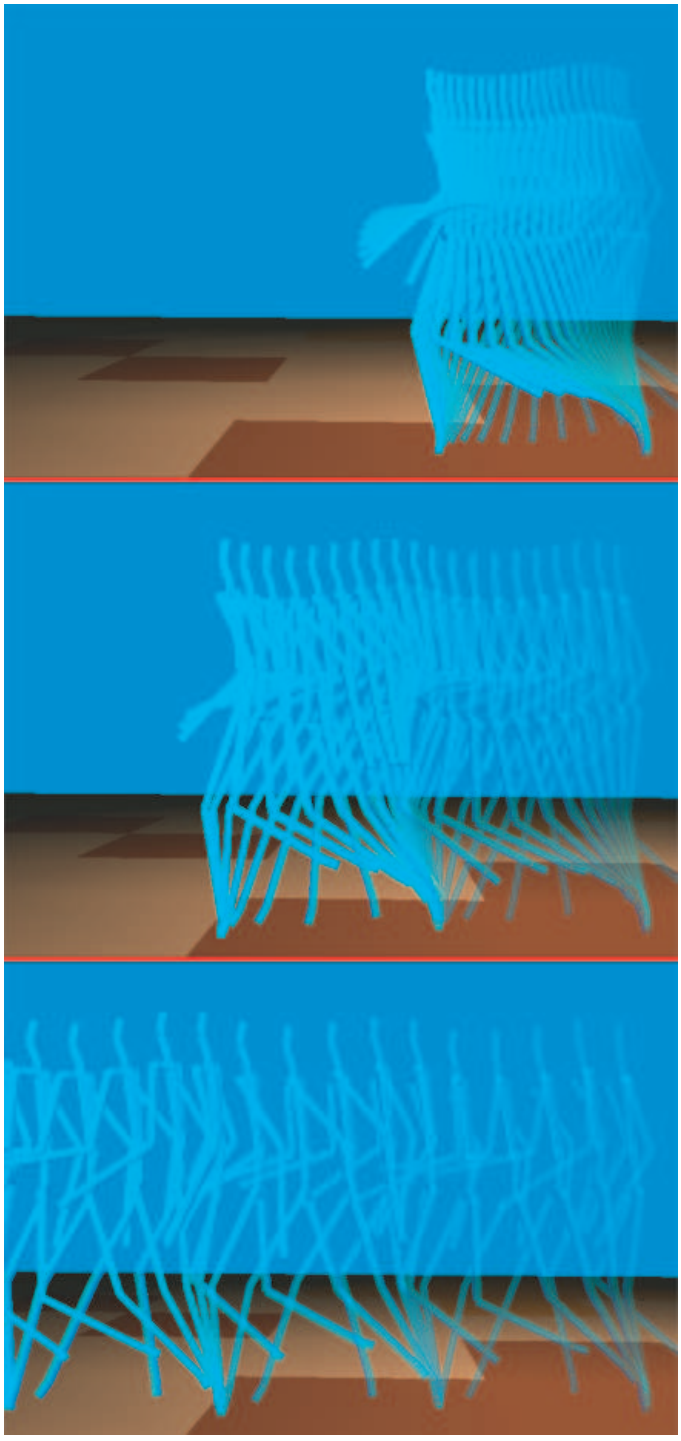

Figure 3.24: A walk and 2 affines

The motion formalism is implemented as follows:

$$\begin{aligned}
\tau_i^s &= \tau_{i'}^s \\
\tau_i^e &= \tau_{i'}^e \\
\mathrm{T}_i^d &= \mathrm{T}_{i'}^d \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\tau_i(T) &= T + \tau_i^s \\
T_i(t) &= \text{ refer to Equation 3.2} \\
t_i(T) &= t_{i'}\left(T + \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\mathbf{K}_{im} &= \mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{\mathbf{K}_{i'm}}{\mathrm{T}_i^d}\right) \\
\theta_i(T) &= \theta_{i'}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\mathcal{D}_{ij}(T) &= \mathcal{D}_{i'j}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\mathcal{I}_i(T) &= \mathcal{I}_{i'}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right)
\end{aligned}$$

$\dot{\theta}_i$ and $\ddot{\theta}_i$ are similar to θ_i, $\mathcal{I}_i$, and $\mathcal{D}_i$. Like the affine, the time-warp does not alter the position or heading at $T = 0$, so motion $\mathbf{M}_i$ will be in the standard position and orientation if $\mathbf{M}_{i'}$ is.

Figure 3.25 shows 3 motions. The top is a non-time-warped walk. The middle is one which has been *eased-in* using the function $\mathcal{U}(t) = t^2$. The bottom has had its middle sped up using the function

$$\mathcal{U}(t) = \frac{1}{2} + \frac{1}{2}\sin\left(\pi\left(t - \frac{1}{2}\right)\right).$$

Mirroring

Another often needed operation is the *mirror*, which switches the left/right direction of a motion. Animation snippets are expensive to create, so being able

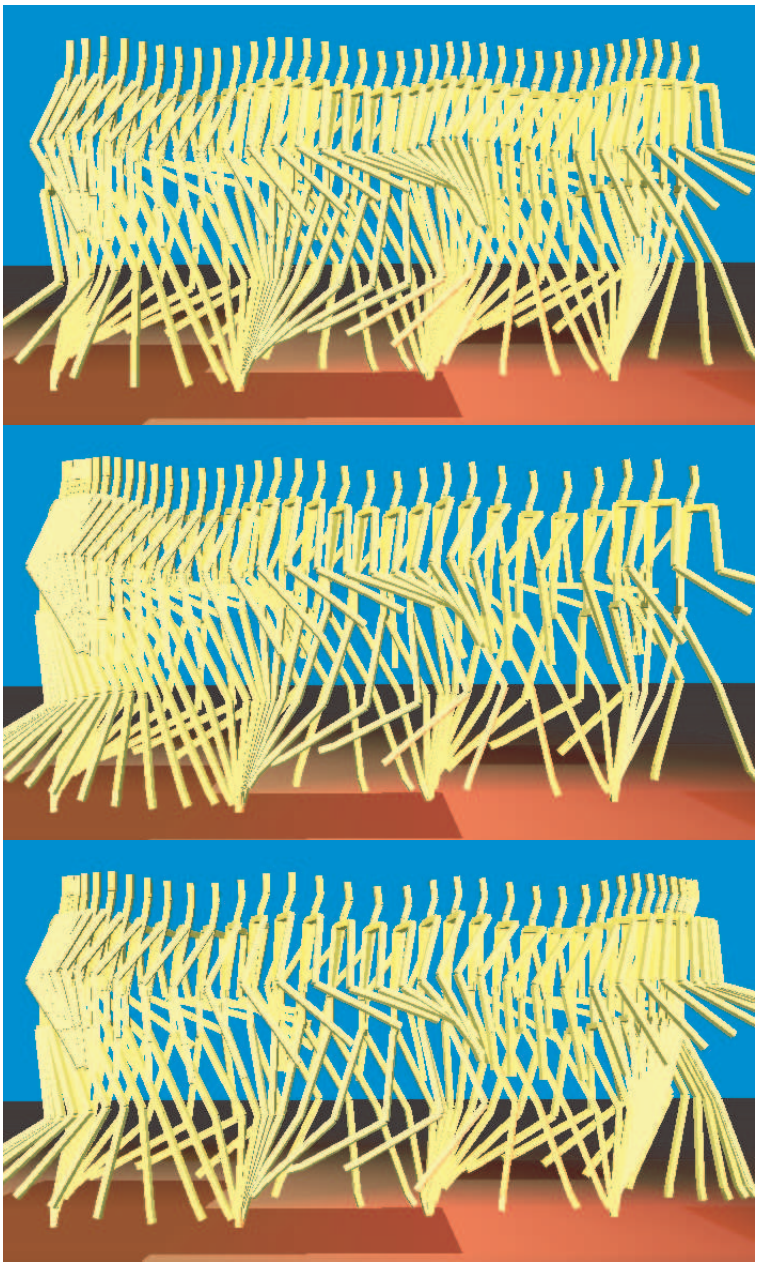

Figure 3.25: Time-warping

to take a walk to the left and turn it into a walk to the right is advantageous. It is also a relatively simple task. The mirror of a motion is defined as

Definition 5 *mirror motion* $\mathbf{M}_i = \{A, S, \mathbf{M}_{i'}\}$

where motion $\mathbf{M}_{i'}$ is the motion to be mirrored and A and S are sets of DOF pairs $\{j, j'\}$ which are anti-symmetrical and symmetrical as defined below. As the mirroring does not alter the duration or timing of the example, most of the motion formalism can be implemented simply:

$$\begin{aligned}
\tau_i^s &= \tau_{i'}^s \\
\tau_i^e &= \tau_{i'}^e \\
\mathrm{T}_i^d &= \mathrm{T}_{i'}^d \\
\tau_i(T) &= \tau_{i'}(T) \\
T_i(t) &= T_{i'}(t) \\
t_i(T) &= t_{i'}(T) \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\mathbf{K}_{im} &= \mathbf{K}_{i'm}.
\end{aligned}$$

The $\mathbf{K}_{im}$ function may change the meaning of the key-times. A left-foot-down key-time in $\mathbf{M}_{i'}$ will need to be interpreted as a right-foot-down key-time in $\mathbf{M}_i$.

The other functions require more work. Figure 3.26 shows the skeleton projected onto the **X**-**Y** plane raising its arm by rotating about the **Z**-axis at the shoulder by an angle of Θ_j (assume the shoulder is the jth DOF). If that value were simply applied to the corresponding point on the other side of the body, the arm would rotate up into the torso. The negative of this angle, $-\Theta$ is used instead. Similarly, a **Y** rotation would have the same anti-symmetry. These corresponding pairs in the DOF values need to be exchanged and have

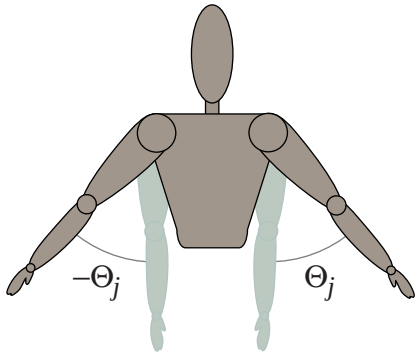

Figure 3.26: Mirroring anti-symmetries

their signs flipped. Other DOFs need to be copied directly from one side of the body to the other, i.e. the symmetry pairs.

For the George skeleton (Figure 3.1), the anti-symmetry pairs are used to exchange the **Y** and **Z** rotations for the left and right sides of the body in addition to changing their sign. The change of sign for the root and spine **Y** and **Z** rotations, in addition to the root **X** translation are also encoded using the anti-symmetry pairs. The symmetry pairs are used to apply the **X** rotations for the left side of the body to the right side.

The position, θ_i, function is defined as

$$\theta_{ij}(T) = \begin{cases} \theta_{i'j'}(T) & \forall \text{ symmetry pairs } \{j, j'\} \in \mathbf{M}_i \\ -\theta_{i'j'}(T) & \forall \text{ anti-symmetry pairs } \{j, j'\} \in \mathbf{M}_i \\ \theta_{i'j}(T) & \text{otherwise.} \end{cases}$$

The DOF velocity and acceleration functions, $\dot{\theta}_{ij}$ and $\ddot{\theta}_{ij}$, are similar. The DOF-usage function $\mathcal{D}_i$, must return DOF-usages for the opposite side of the body, so would be defined:

$$\mathcal{D}_{ij} = \begin{cases} \mathcal{D}_{i'j'}(T) & \forall \text{ symmetry and anti-symmetry pairs } \{j, j'\} \in \mathbf{M}_i \\ \mathcal{D}_{i'j}(T) & \text{otherwise.} \end{cases}$$

Since the spine is its own symmetry and anti-symmetry target, this will return the correct answer even though some of the DOF-usages will not actually shift

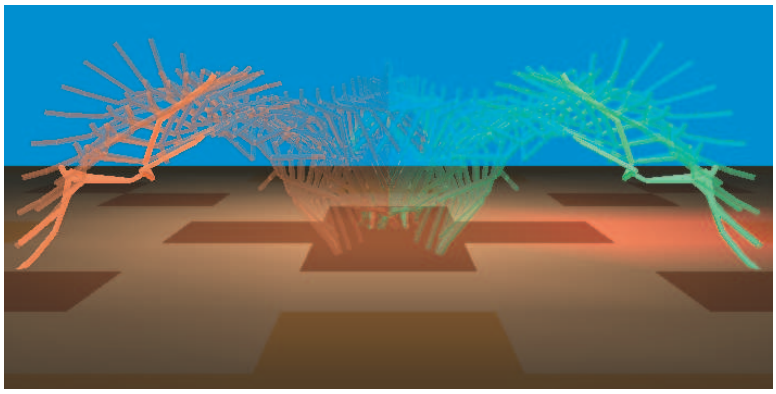

Figure 3.27: A mirrored jump-dive

from one side of the body to another.

The constraint function, $\mathcal{I}_i\ (T)$, requires that the constraints involving the DOFs for one side of the body be translated to use their appropriate counterparts on the other side of the body, if appropriate. Constraints involving the right foot, for example, in motion $\mathbf{M}_{i'}$, should report as involving the left foot for motion $\mathbf{M}_{i'}$.

Figure 3.27 shows a jump-dive motion in red and its mirror in green.

Composition

Motion compositing is the process of taking multiple motions and executing them simultaneously, producing a layered motion. For example, an animator may want to take a wave and compose it with a walk to construct a walking wave, as shown in Figure 3.28. A composition is defined as:

Definition 6 *composition motion* $\mathbf{M}_i = \{\mathbf{p}_i, \mathbf{M}_{i_0}, \mathbf{M}_{i_1}, \ldots \mathbf{M}_{i_n}\}$

where the $\mathbf{M}_{i_c}$'s are the motions to be composed with one another. The adverb value $\mathbf{p}_i$ is designer specified since there is no reasonable procedural way to assign an overall adverb value to a motion composed of many different motions with many different, and potentially conflicting, adverbs settings. The motions

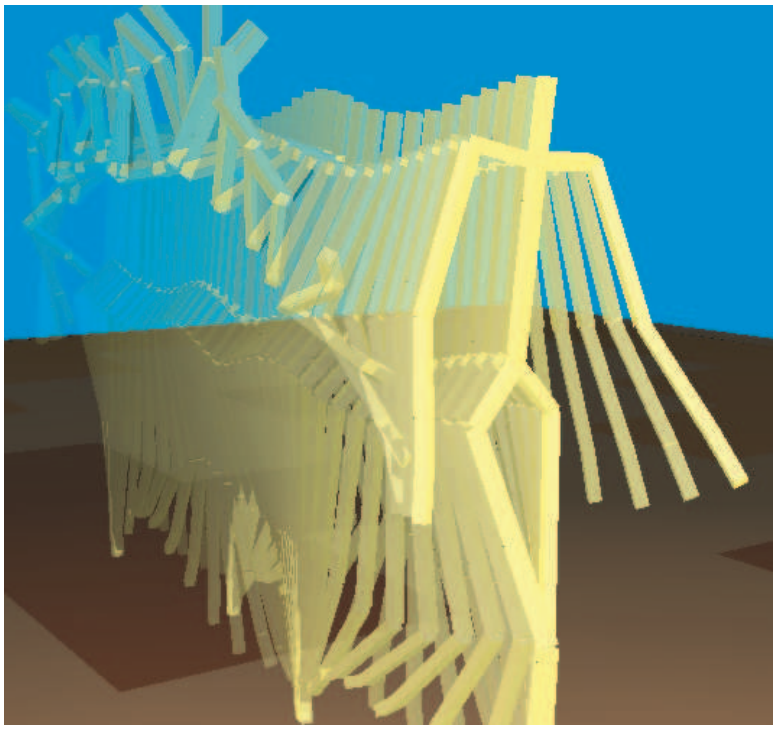

Figure 3.28: A walk/wave composition

may be placed anywhere on the animation timeline (possibly through use of an *affine*), so the animation time bounds can be calculated as:

$$\begin{aligned} \tau_i^s &= \min_c \tau_{i_c}^s \\ \tau_i^e &= \max_c \tau_{i_c}^e. \end{aligned}$$

The composition has the union of all the keys for its components. For each key-time $\mathbf{K}_{i_c m'}$ in a constituent motion,

$$\mathbf{K}_{im} = \tau_{i_c}(\mathbf{K}_{i_c m'}) - \tau_i^s.$$

Likewise, the constraint function $\mathcal{I}_i$ would return the union of all the constraints present in the constituent motions:

$$\mathcal{I}_i = \bigcup_c \mathcal{I}_{i_c}(T).$$

The other functions of the motion formalism are not as simple. When two motions overlap in time, how are their effects or kinematic constraints combined to produce a reasonable effect? For this, a mechanism for arbitrating the composibility of two (or more) motions must be designed.

Degrees of freedom & motion composibility

Many such arbitration schemes are possible and no one method is currently the preferred one. The *Verbs & Adverbs* system makes use of a 3-valued DOF-*usage* value on a DOF-by-DOF basis to determine composibility. The three types of DOFs used are those which are *required*, *defined*, and *undefined*.

A **required** DOF is one which is integral to the effectiveness of a motion. An example would be the knee rotation of a walk cycle. A **defined** DOF is one for which information has been provided but which is not integral to the successful completion of the motion's primary goal. An example would be the wrist rotation of the walk. Finally, an **undefined** DOF is one for which no information is provided. A waving motion, for example, would not need to define the knee rotation.

Figure 3.29 shows the different DOF usages for a walking motion and waving motion. Yellow DOFs are **required**, red DOFs are **defined**, and gray ones are **undefined**.

For sufficiently complicated motions, these distinctions may not be fine enough. Take, for example, the walk/wave composition motion shown in Figure 3.28. If it were to be used basic motion in a further motion edit, what would its DOF-usages be? One answer would be that the arm and legs are **required** and the other DOFs **defined**, as shown in Figure 3.30.

Rather than define a static value for the DOF-usages of a motion, motions should provide a mechanism for returning the DOF-usage of any DOF at a given

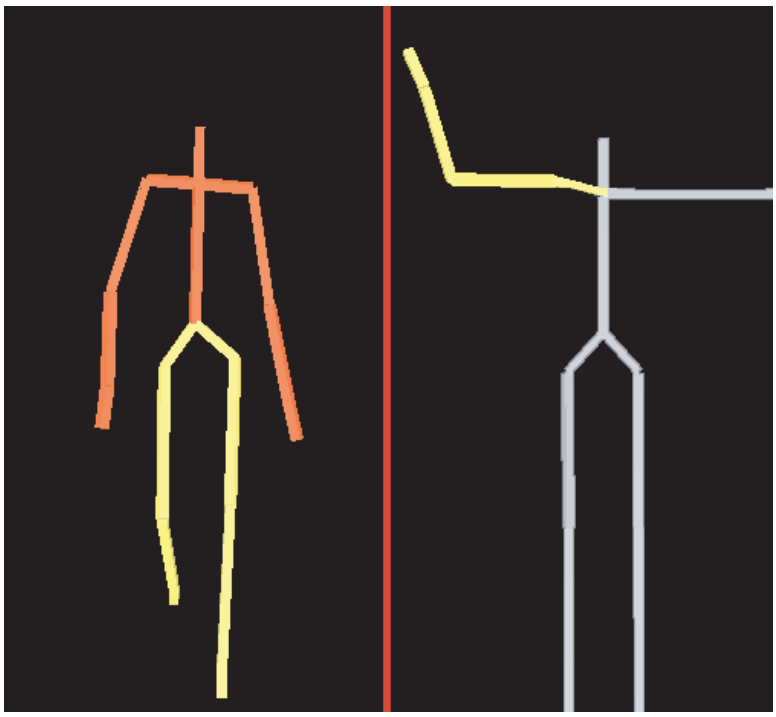

Figure 3.29: DOF-classes for walk/wave motions

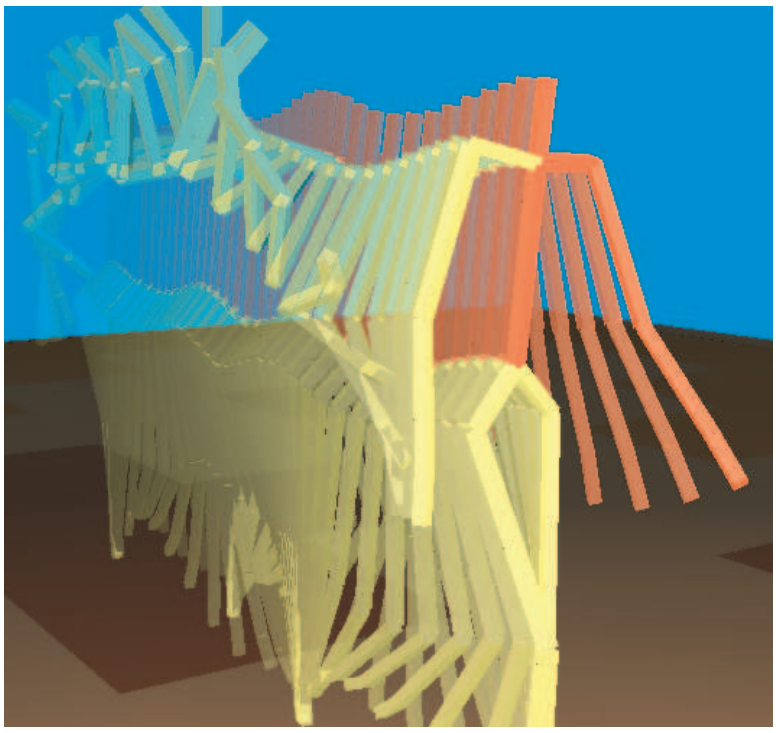

Figure 3.30: DOF-usage values too broad

time. Hence the DOF-usage function $\mathcal{D}_{ij}\ (T)$.

Motion compositing determines DOF-usages procedurally, by querying the motions which are being composited. The motion with the highest class at a given time is granted control of the DOF and the DOF-usage is set at that level. Motions with conflicting DOFs can be either tagged as ill-constructed, further arbitrated, or both. In this work, conflicting DOFs are arbitrated in a first-come-first-served basis depending upon the order in which the constituent motions of a composition were added into the composition. Additionally, flags indicating a potential problem are set which can be queried. So,

$$\mathcal{D}_{ij}(T) = \max_{c} \mathcal{D}_{i_c j}(T)$$

where **undefined** < **defined** < **required**.

Finally, the position function, θ_{ij}, is

$$\theta_{ij}(T) = \text{Arbitrate}\left(\theta_{i_0 j}(T_{i_0}), \theta_{i_1 j}(T_{i_1}), \ldots, \theta_{i_n j}(T_{i_n})\right)$$

where $T_{i_c} = T_{i_c}\left(\tau_i(T)\right)$.

A time-lapse picture showing the DOF-usages on the figure through time is depicted in Figure 3.9. Note that the arm is **defined** at the beginning, becomes **required** as the motion moves into the wave period, and returns to **defined** at the end. No DOFs where **undefined** in this composition.

Concatenation

Concatenating a number of motions means to place them end-to-end on a new timeline. This differs from a composition and affine-time-shift in that special care must be taken when dealing with the root DOFs, so that it appears that the motions follow one another reasonably in space.

A concatenation motion is defined as

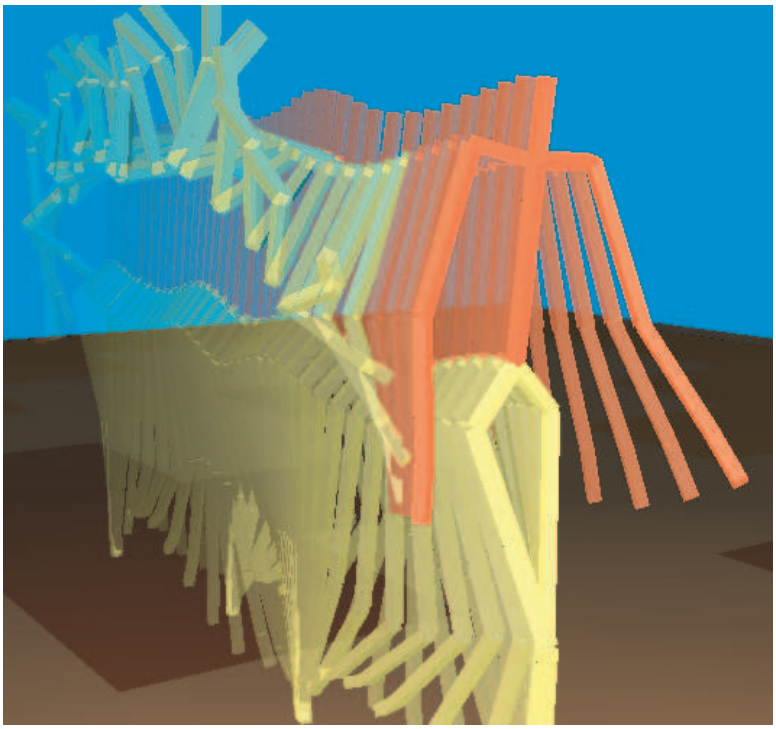

Figure 3.31: Walk/wave DOF-usage time lapse

Definition 7 *concatenation motion* $\mathbf{M}_i = \{\mathbf{p}_i, \mathbf{M}_{i_0}, \mathbf{M}_{i_1}, \ldots \mathbf{M}_{i_n}\}$

where the $\mathbf{M}_{i_c}$'s are the motions being concatenated in order from $0..n$. Like the composition motion, the overall adverb value is specified by the designer as $\mathbf{p}_i$. The concatenation as a whole is defined to start at $\tau = 0$, so $\tau_i^s = 0$. The total duration of the concatenation is the sum of the durations of the $\mathbf{M}_i$'s, so the basic time mapping functions are defined as follows:

$$\begin{aligned} \tau_i^s &= 0 \\ \mathrm{T}_i^d &= \sum_{c=0}^{n} T_{i_c}^d \\ \tau_i^e &= \mathrm{T}_i^d \\ \tau_i(T) &= T \end{aligned}$$

Only one motion is active at a given time τ, so the elapsed time spend in

any motion at τ is defined as

$$T_{\text{elapsed}} = T - \sum_{c=0}^{n-1} T_{i_c}^d$$

where n is the maximum n which satisfies

$$\sum_{c=0}^{n-1} T_{i_c}^d \leq T.$$

If no n satisfies, then the first motion has not yet been exhausted, so $T_{\text{elapsed}} = T$.

Key-times can now be defined. For any key-time $\mathbf{K}_{i_n m'}$ in a constituent motion,

$$\mathbf{K}_{im} = \begin{cases} \mathbf{K}_{i_n m'} + \sum_{c=0}^{n-1} T_{i_c}^d & n > 0 \\ \mathbf{K}_{i_0 m'} & \text{otherwise} \end{cases}$$

where m and m' indicate the key-time offset due to the concatenation having key-times from all the constituent motions. Once the key-times are determined, the time functions $T_i(t)$ and $t_i(T)$ can be calculated using the standard formulae from Section 3.4.

The kinematic functions are calculated using

$$\begin{aligned} \theta_i(T) &= \theta_{i_c}(T - T_{\text{elapsed}}) \\ \dot{\theta}_i(T) &= \dot{\theta}_{i_c}(T - T_{\text{elapsed}}) \\ \ddot{\theta}_i(T) &= \ddot{\theta}_{i_c}(T - T_{\text{elapsed}}) \end{aligned}$$

The position function is not as simple as shown here. In order for examples to be concatenated in space, the start of a motion $\mathbf{M}_{i_{c+1}}$ must be lined up with the end of a motion $\mathbf{M}_{i_c}$. A general reorientation (Algorithm 3.2) can be used to line them up, making the actual θ_i not much more complicated than as above.

This concatenation mechanism does not smooth between motions other than to line them up at the root. Discontinuous change in the body rotations

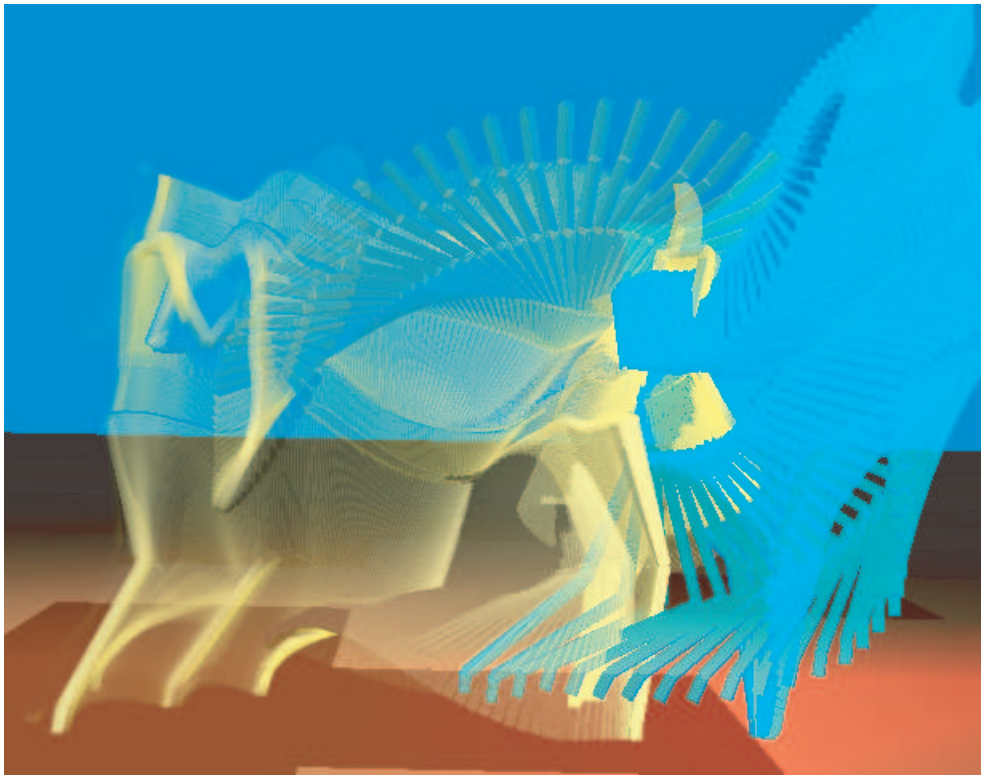

Figure 3.32: A concatenation

will make for strange concatenations. Transitioning mechanisms, described in Section 5.5 and Appendix B, can smooth out these discontinuities. An example concatenation is shown in Figure 3.32, discontinuities and all. Color indicates the different motions in the concatenation. Note the discontinuities between the motions.

Selection

Another motion type, this one a helper in the construction of more complicated motions, is the *selection*. This kind of motion is particularly simple, as it turns on and off DOFs, i.e. it changes some DOF-usages for some of the DOFs. While simple, it is useful for complex editing sessions. Imagine a waving verb, such as seen earlier in this chapter. When the wave came out of the motion capture

analysis system, its DOFs are of equal weight. The designer can go in and annotate the DOFs and assign some of them to **required** and some to **defined**. It is possible, however, that this motion may be used in two places, one where the arm is **required**, and one where the arm is simply **defined**. The designer may want the motion to have the DOF values **defined** for the non-arm parts of the body in one circumstance and **undefined** elsewhere. This is where the selection motion comes in handy.

The selection of a motion $\mathbf{M}_i$, is defined as

Definition 8 *selection motion* $\mathbf{M}_i = \{\mathbf{M}_{i'}, \{j, usage\}^*\}$

where $\mathbf{M}_{i'}$ is the motion having its DOF-usages adjusted and *usage* either **undefined**, **defined**, or **required**. The motion formalism is easily implemented for this motion as

$$
\begin{aligned}
\tau_i^s &= \tau_{i'}^s \\
\tau_i^e &= \tau_{i'}^e \\
\mathrm{T}_i^d &= \mathrm{T}_{i'}^d \\
\tau_i(T) &= \tau_{i'}(T) \\
T_i(t) &= T_{i'}(t) \\
t_i(T) &= t_{i'}(T) \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\mathbf{K}_{im} &= \mathbf{K}_{i'm} \\
\theta_{ij}(T) &= \begin{cases} \theta_{i'j}(T) & \forall \text{ DOFs not } \textbf{undefined} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

$$\mathcal{D}_{ij}(T) = \begin{cases} \textbf{undefined} & \forall \text{ DOFs } j \textbf{ undefined} \text{ by the selection} \\ \textbf{defined} & \forall \text{ DOFs } j \textbf{ defined} \text{ by the selection} \\ \textbf{required} & \forall \text{ DOFs } j \textbf{ required} \text{ by the selection} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{I}_i(T) = \mathcal{I}_{i'}(T) - \text{ those involving } \textbf{undefined} \text{ DOFs}$$

The DOF velocity and acceleration functions, $\dot{\theta}_i$ and $\ddot{\theta}_i$, are similar to θ_i.

3.10 Cyclification

As reported in *Efficient Generation of Motion Transitions using Spacetime Constraints* [124], hand animation or motion capture often yields motions which are almost cyclic, but which should logically be perfectly cyclic. Variations in human motion and motion capture data noise yield non-cyclic results. Animators who do not take pains to produce exactly cyclic motion are also likely to produce almost cyclic motions.

Some simple processing can help to ensure the cyclicity of a motion. For example, take the walking motion introduced earlier in this chapter. The designer marks out two times where they believe a cycle is located. A region around this proposed cycle is searched to try to find the most closely matched times τ_0 and τ_1. Once they are found, spreading the remaining error throughout the cycle yields a smooth cyclic motion. Typically, this construction is then fit through a cyclic B-spline type motion, which yields $C-2$ continuity at the end-points. Not all of the DOFs are cyclified, however. As is typical, the root requires some special handling. The **X**-translation, **Z**-translation, and **Y**-rotation DOFs are not cyclified. The concatenation mechanism keeps these motions flowing from one to another in these DOFs. Cyclification can also often be done using the transitioning mechanism detailed in Section 5.5.

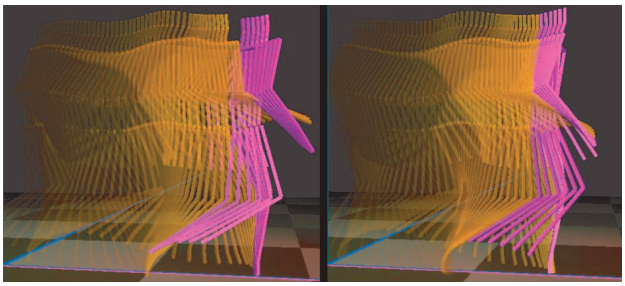

Figure 3.33: Cylification smooths out the cycle for seamless concatenation

Figure 3.33 shows the effectiveness of this cyclification technique. The picture on the left shows the cycle-clip (as defined above) simply concatenated with itself. Note the discontinuity at the color change. The picture on the right, however, was cyclified and exhibits a smooth transition from one repetition of the cycle to the next.

Rather than alter the DOF-curve coefficients to achieve a perfect cycle, however, a functional cycle form obeying the motion formalism can be constructed. Assume that we have constructed a clip motion $\mathbf{M}_{i'}$ which marks the beginning and ending of a cycle in amother motion $\mathbf{M}_{i''}$ as determined above. A cycle motion $\mathbf{M}_i$ of $\mathbf{M}_{i'}$, therefore, is defined as

Definition 9 *cycle motion* $\mathbf{M}_i = \{\mathbf{c}_i, \mathbf{M}_{i'}\}$

where $\mathbf{c}_i$ is a vector of booleans indicating which DOFs are cyclified. Remember that the root **Z** translation is typically not-cyclified. Using the boolean vector generalizes that notion. Implementing the motion formalism is as follows:

$$\begin{aligned} \tau_i^s &= \tau_{i'}^s \\ \tau_i^e &= \tau_{i'}^e \\ \mathrm{T}_i^d &= \mathrm{T}_{i'}^d \end{aligned}$$

$$
\begin{aligned}
\tau_i(T) &= \tau_{i'}(T) \\
T_i(t) &= T_{i'}(t) \\
t_i(T) &= t_{i'}(T) \\
\mathbf{K}_{im} &= \mathbf{K}_{i'm} \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\mathcal{D}_{ij}(T) &= \mathcal{D}_{i'j}(T) \\
\mathcal{I}_i(T) &= \mathcal{I}_{i'}(T) \\
\theta_{ij}(T) &= \begin{cases} \theta_{i'j}(T) & \mathbf{c}_{ij} \text{ false} \\ \theta_{i'j}(T) + \frac{T \cdot d_{ij}}{\mathrm{T}_i^d} - \frac{d_{ij}}{2} & \mathbf{c}_{ij} \text{ true} \\ \text{where } \; d_{ij} = \theta_{i'j}(\mathrm{T}_{i'}^d) - \theta_{i'j}(0) & \end{cases}
\end{aligned}
$$

The DOF velocity and acceleration functions, $\dot{\theta}_i$ and $\ddot{\theta}_i$, are similar to θ_i.

3.11 Conclusions

In this chapter, the idea of an *example* was presented. Examples are motions which meet a number of criteria, none of them overly strict. A motion formalism was introduced which structures the way in which information is extracted from a motion to facilitate playback. Tools for manipulating example motions were described which implement the motion formalism. Rather than treating a clip differently than a composition, the motions can be handled using the same methods. The motion objects can be used to construct an animation editing system which allows for undo at any stage of the editing process without seriously adding to the memory size requirements of the system, as the motion data is never copied.

Chapter 4

Verbs & adverbs

> *Our hero walks into a moonlit room turning sharply to the left and then right, scanning purposefully. Dejected, he realizes that he is alone. He walks further into the room in that slow, lackluster way of the disappointed. Off in the distance, he hears a familiar welcoming voice and walks to its source full of life, brimming with pleasure in each carefree step. His intended had not left, after all.*

Earlier in this dissertation, both hand animation and motion capture were detailed. Using these tools, a competent animator could design a sequence for the above scene. What if, however, the unseen voice was of a hated foe, or an unknown person. The walk, on the one hand carefree, would need to be modified to fit the new scene. For a movie, modifying the motion using the same tools used for the original motion would be a fine solution. If this were an interactive game or a shared virtual environment, however, the circumstances might not be known until runtime. In order to fit the scene, either a large collection of appropriate motions must be gathered or the motions themselves must be parameterized, able to handle many different situations.

This chapter describes a method for using multidimensional function interpolation to derive controllable motions, *verbs*, from sets of examples. Ac-

quiring these examples and putting them into the proper form was described in Chapter 3.

4.1 Overview

There are many ways to parameterize motions. Chapter 2, the overview and literature review chapter provides a synopsis of the primary techniques, breaking them into three primary groupings: procedural, simulated, and interpolated. The *Verbs & Adverbs* technique is an interpolated method used to construct motions, or *verbs*, which respond to a set of control knobs, known as *adverbs*. The goal of this technique is to create these verbs using as input primarily that which an animator or motion capture system will typically produce. Thus, this system is designed to leverage talents or properties of each.

Please recall that examples were defined to have a number of desirable properties, as was detailed in Section 3.5. These were:

1. similar motion structure,
2. same skeleton,
3. continuous DOF-curves,
4. anatomically-plausible use of joint angles,
5. similar use of joint angles for similar motions, and
6. same initial placement and heading at the beginning of the example.
7. in canonical timeline
8. identical DOF function encoding schemes.

The last restriction requires that all the examples be encoded using the same mechanism, for example, a B-Spline, and that they each have the same number of coefficients. This does not mean that each of the DOFs for a particular example i have the same encoding, but rather that the jth DOF for each example be encoded in the same way. The next-to-last restriction ensures that a given coefficient in a particular example encodes the same structural portion of a motion as that coefficient for a different example.

If the coefficients are so constrained, then it is possible to interpolate the motion by constructing independent problems over the coefficients. Each of these separate problems can be solved using a non-linear multi-dimensional interpolation technique. As will be described later, due to the sparsity of data for the interpolated animation problem, we solve for these interpolations using radial basis functions. In the example acquisition phase, the designer assigned key-times, which were used to reproject the motions into a canonical timeline. The canonical timeline ensures that the coefficients can be interpolated meaningfully, but removes timing information from the example set. An interpolation scheme for timing must also be used in order to recapture that information. The timing information is merely the key-times, which can be thought of as further coefficients to be interpolated.

There are many ways to perform multi-dimensional interpolation. Wavelets, for example, have proven a useful encoding for many computer graphics problems such as radiosity, surface approximation, and video-lookup. One problem, however, is the large number of samples required by this encoding, typically $\mathbf{O}\left(2^D\right)$ where D is the dimension of the function being encoded. In radiosity calculations, for example, this is not a concern since the problem is typically over-constrained. For animation, however, each sample represents a hard-won animation, a precious commodity. An interpolation scheme needing

fewer samples must be used if the number of control knobs desired is high. For this reason, a radial basis function, or RBF, encoding was used. In particular, the *Verbs & Adverbs* system uses a radial B-spline as its basis function. This encoding has the advantage that it leads to an interpolating space, thus preserving exactly each example animation. Wavelets, on the other hand, are typically approximating encodings. Further advantages include speed of evaluation and fitting. RBF approximation has potential pitfalls which will be discussed in this chapter. Extensions to the basic technique can be used to overcome these pitfalls.

There are a number of symbols and subscripts used in the description of the interpolation technique. Table 4.1 synopsizes the main ones used in this chapter.

4.2 The canonical timeline

In Chapter 3, the concepts of the canonical timeline and key-time were introduced. This section explains the connection between the canonical timeline and motion blending. As will be introduced later in Section 4.3, the *Verbs & Adverbs* system blends motions together by setting up interpolations of DOF-curve coefficients. For a two-example verb, for instance, the blend is quite simple. Assume the verb had one adverb. The blend would be related to how close the desired motion was to the examples' adverb values. If the desired motion was half way from $\mathbf{M}_1$ to $\mathbf{M}_2$ in adverb, then a 50% blend would be used.

To illustrate the importance of blending examples in the canonical timeline, rather than in the timeline in which the examples were designed, consider the motions shown in Figure 4.1. At first glance, the blend would be obvious: right up the center. A closer look at the distribution of the temporal

Object	Variable	Subscript	subscript meaning	subscript range
Motion example	M	i	Motion example number	$1..NumExamples$
DOF	θ	i	Motion example number	$1..NumExamples$
		j	DOF index	$1..NumDOF$
B-spline	B	k	B-spline index	$1..NumCP$
B-S Control Point	b	i, j, k		
Point in Adverb Space	$\mathbf{p}$	i	Motion example number	$1..NumExamples$
Key-time	K	m	Keytime index	$1..NumKeyTimes$
Radial basis	R	i	basis associated with M_i	
Radial Coefficient	r	i, j, k		
Linear Basis	A	l	Adverb index	$1..NumAdverbs$
Linear Coefficient	a	j, k, l		
Distance	d	i	Distance to p_i	
Time	Variable	range		
Clock time	τ			
Key-time time	T	$0..K_{NumKeyTimes}$		
Generic time	t	$0..1$		

Table 4.1: Terminology

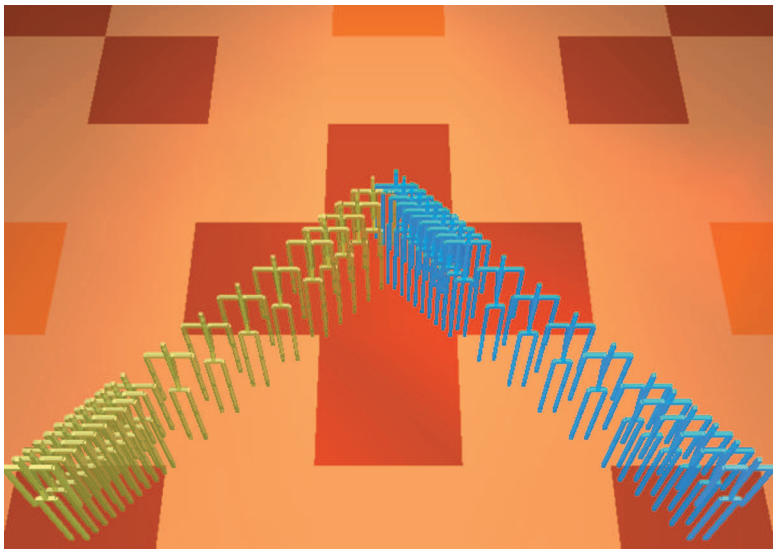

Figure 4.1: Motions $\mathbf{M}_1$ and $\mathbf{M}_2$

samples, however, will reveal that there is a different phrasing to the two motions. Figure 4.2 shows a plot of one of the DOF-curves for each of the two motions, showing how they differ. Each motion has four key-times: *start*, *first-speed-change*, *second-speed-change*, and *stop*.

If blended without respect to these key-times, the resulting motion is strange, as shown in Figure 4.3. By lining up the motions in the canonical timeline, however, the blend is better. Figure 4.4 shows this. The timing information lost in canonical timeline can be synthesized from the examples. Recapturing expressive timing is often vital to the success of an interpolated motion.

Key-times

Key-times and the canonical timeline were both introduced in Section 3.4. A graph of key-times plotted against canonical time for a collection of walks is shown in Figure 4.5. The key-times were placed at the foot-down events, as shown in Figure 4.6. While the timelines progress in a relative lockstep, small

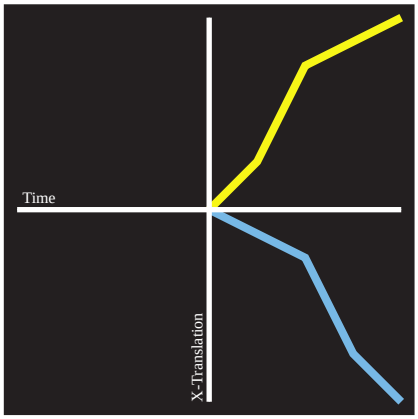

Figure 4.2: Plot of **X**-translation for $\mathbf{M}_1$ and $\mathbf{M}_2$

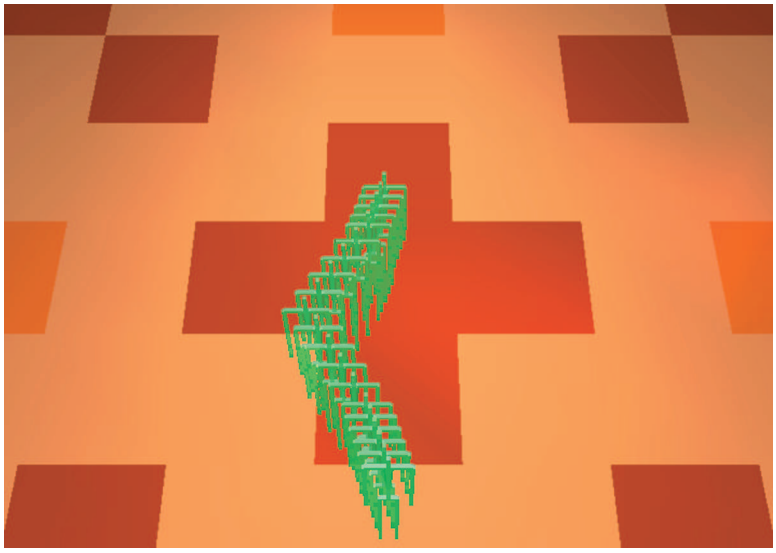

Figure 4.3: Strange blend due to incompatible timelines

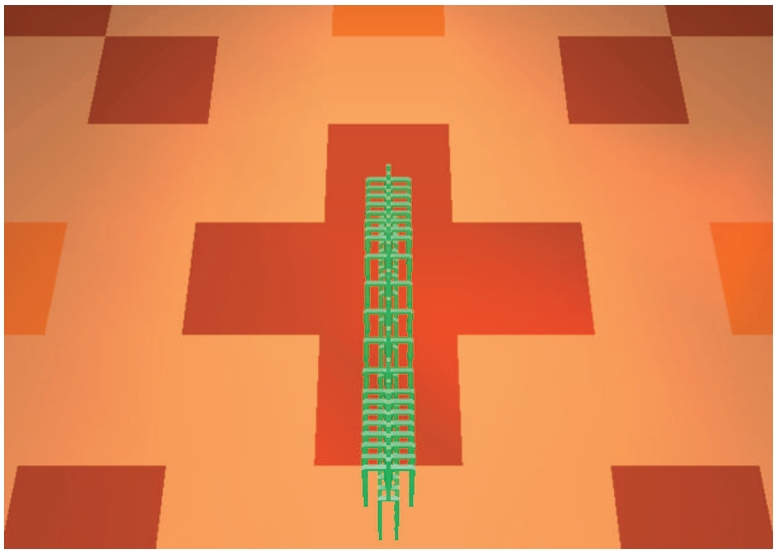

Figure 4.4: Good blend with canonical timeline

variation is seen. These variations correspond to differences in the relative durations of portions of the walking cycle in the different examples.

Examples need not have such similar key-time graphs, however. A graph for a selection of idling motions is shown in Figure 4.7. Notice the greater variation in timing for these motions. For example, take two extreme idling motions, one a despondent hands on hips motion and another a brusque, angry hands on hips. For these two motions, the key times would be at the following moments: hands leave side, hands on hips, hands leave hips, hands at side. These key-times are represented in canonical time, and result in the graph shown in Figure 4.8.

The sad timeline shown in brown and the angry in red. The angry idler keeps his hands at his side and snaps them quickly to his hips, as evidenced by the small time it takes to get from the $\mathbf{K}_1$ to $\mathbf{K}_2$, shown by the short region between the two red dashed lines on the vertical (verb-time) axis. If we were to blend the two motions at a particular verb-time T, for example, the result would be that shown in Figure 4.9. Notice the odd backtracking in the path

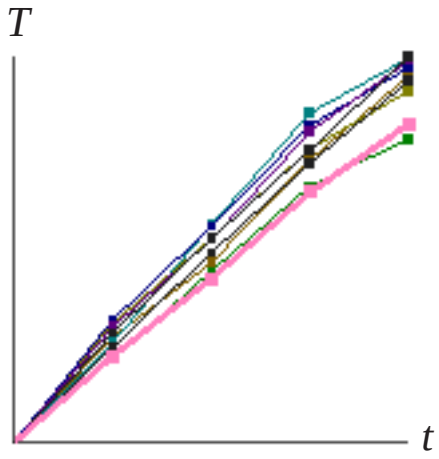

Figure 4.5: Key-times for a walking repertoire

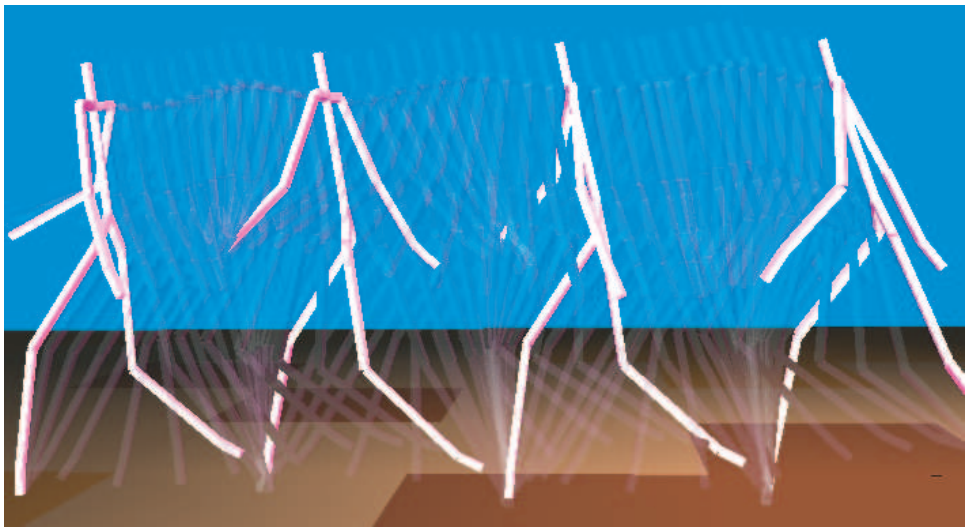

Figure 4.6: Walking key-times placed at foot-down events

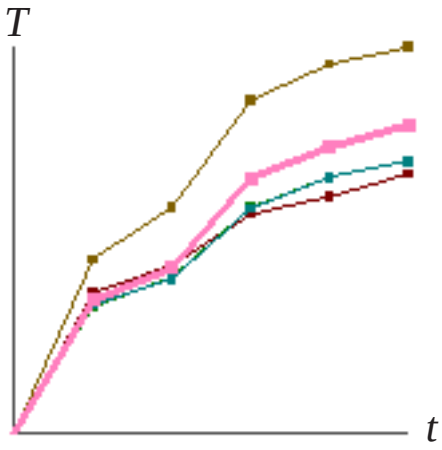

Figure 4.7: Key-times for a idling repertoire

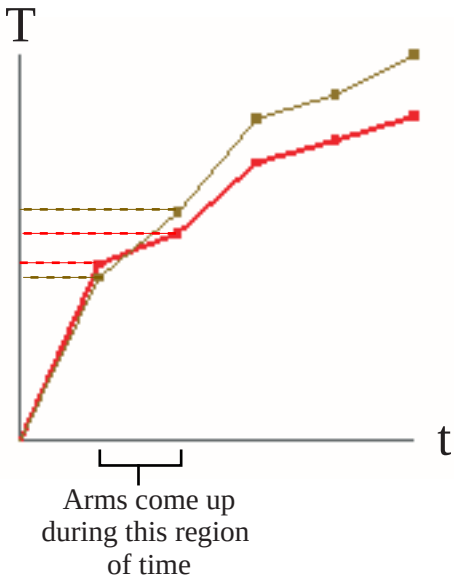

Figure 4.8: Key times for two extreme hands-on-hips idles

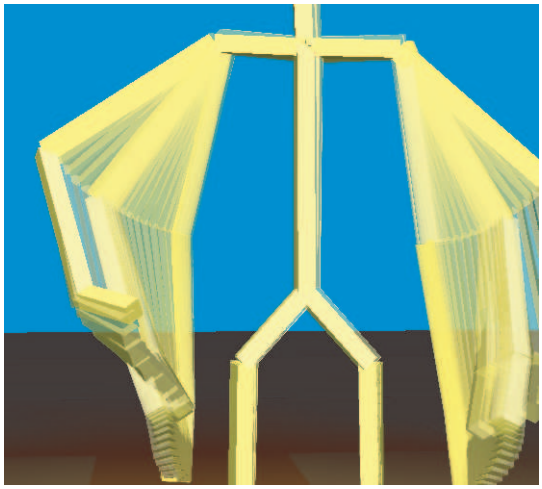

Figure 4.9: No key times blend

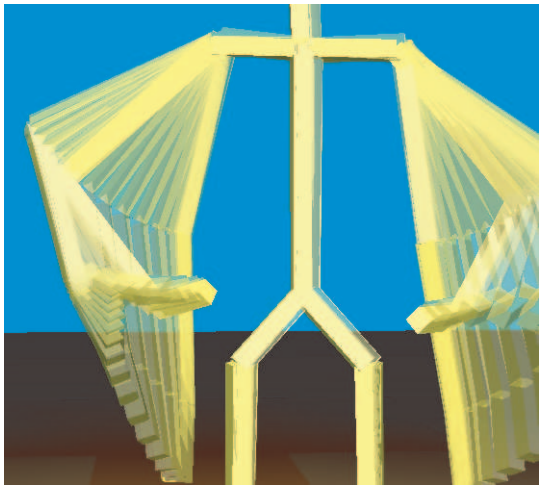

Figure 4.10: Use key times

of the arms. If the blend is done in the canonical timeline referenced by the key-times, the blend is over corresponding moments in the two motions, and results in a better blend shown in Figure 4.10 as evidenced by the smoother arm trajectories.

The examples are sampled before they are used in the construction of the interpolating spaces. As the DOF-curves must be continuous, this is a simple process. One potential problem with using the canonical timeline, however, is that it destroys timing information. Recall the previous example. The "anger" conveyed in the hands on hips is conveyed partly in the differing arch of the back, but the bulk of the emotional content was expressed in the timing of the motion. Clearly, some mechanism must be used in order to preserve timing and duration information while using the canonical timeline for interpolation.

Each motion has a set of key times (the last of which is the duration), however, and these can be considered yet another set of coefficients to be interpolated using the same multidimensional interpolation scheme as used for the DOF curve coefficients.

So, the number of separate interpolation spaces needed for a motion $\mathbf{M}$ is $(NumCP \times NumDOF + NumKeyTimes)$ for a motion where each DOF is encoded using the same representation and $(TotalCoefficients + NumKeyTimes)$ for a motion with a heterogeneous set of DOF representations. The latter can be useful for cutting down on the number of coefficients required to encode relatively low-frequency DOF-curves, such as often occur in the spine and knee.

4.3 Verb construction

Chapter 3 showed how to construct a set of examples obeying the example criteria. Recall that the examples, by the verb-construction stage, are in canonical form, which includes being in the canonical timeframe. The next step in

the production pipeline is to construct from them a verb. This verb represents a continuous "space" of motions parameterized by a set of adverbs. A point in this space represents a complete animation for a particular combination of adverb settings. These settings may change from one moment to the next if, for example, the emotional state of the actor changes or the environment changes.

The goal is to produce at any point $\mathbf{p}$ in the adverb space a new motion $\mathbf{M}(\mathbf{p}, t)$ derived through interpolation of the basis motions. Since animation source data is previous, interpolating the examples is an added requirement. Therefore, when $\mathbf{p}$ is equal to the adverb setting for a particular example motion i, $\mathbf{M}(\mathbf{p}, t) = \mathbf{M}_i(t)$.

Each example motion has one free variable for each control point defining the DOF curves and one free variable for each key-time. The time warping described in Section 4.2 ensures that corresponding control points in each example motion specify similar moments in each motion, even if the overall lengths and internal phrasing of the example motions differ. This allows us to treat the example motion interpolation as a separate problem for each control point and each key-time (i.e. $\mathit{TotalCoefficients} + \mathit{NumKeyTimes}$ individual interpolation problems).

The standard problem of multivariate interpolation is as follows: given N distinct points $\mathbf{p}_i$ in $\mathbf{R}^n$ and N values v_i in $\mathbf{R}$, find a function $f : \mathbf{R}^n \rightarrow \mathbf{R}$, such that for all i, $f(\mathbf{p}_i) = v_i$, and such that f does "the right thing" between the points $\mathbf{p}$. In general, "the right thing" means filling in the space between data points as smoothly as possible. The potentially high dimensionality of the space defined by the adverbs, coupled with the desire to require few example motions (perhaps only two or three times the number of adverbs), presents difficulties for many interpolation methods. Given these requirements, a combi-

nation of radial basis functions and low order (linear) polynomials was selected for this problem. The polynomial function provides an overall approximation to the space defined by the example motions. It also allows for extrapolation outside the convex hull of the locations of the example motions. The radial bases then locally adjust the polynomial to interpolate the example motions themselves.

Radial basis functions have the form:

$$R_i\left(d_i(\mathbf{p})\right) \tag{4.1}$$

where R_i is the radial basis associated with $\mathbf{M}_i$ and $d_i(\mathbf{p})$ a measure of distance between $\mathbf{p}$ and $\mathbf{p}_i$, most often the Euclidean norm $\|\mathbf{p}-\mathbf{p}_i\|_2$. Because the sums of radial bases cannot represent an affine or polynomial function, radial bases are often augmented by adding a polynomial of fixed degree.

Details of the mathematics for this type of interpolation can be found in the seminal work of Micchelli [103] and in the survey article by Powell [117]. Radial basis functions have been used in computer graphics for image warping by Ruprecht and Müller [126], and Arad et al. [3]. Pighin, *et. al.*, [115] used radial basis functions to perform blending of facial texture data.

Each of the DOF curves θ_j are defined by B-spline coefficients, b_{jk} where k varies from $[1 \ldots NumCP]$. The value of each interpolated DOF curve coefficient in this space, $b_{jk}(\mathbf{p})$, is defined as

$$b_{jk}(\mathbf{p}) = \sum_{i=1}^{NumExamples} r_{ijk} R_i(\mathbf{p}) + \sum_{l=0}^{NumAdverbs} a_{jkl} A_l(\mathbf{p}) \tag{4.2}$$

where the r_{ijk} and R_i are the radial basis function weights and radial basis functions themselves and the a_{jkl} and A_l the linear coefficients and linear bases as explained in Section 4.3. Interpolated key-times are similarly defined as

$$K_m(\mathbf{p}) = \sum_{i=1}^{NumExamples} r_{im} R_i(\mathbf{p}) + \sum_{l=0}^{NumAdverbs} a_{lm} A_l(\mathbf{p}). \tag{4.3}$$

For each verb there are ($NumCP \times NumDOF$) (or *TotalCoefficients* for a mixed representation verb) control point interpolations (Equation 4.2) and *NumKeyTimes* key-time interpolations (Equation 4.3).

The remaining problems are choosing the specific shape of the radial bases and determining the linear and radial coefficients. The radial basis shapes are determined by the spacing of the examples in the adverb space. The coefficients are determined in two steps, by first solving for the linear coefficients and then for the radial basis coefficients.

Linear approximation

In the first step, the linear coefficients are found by fitting a hyperplane through the adverb space that best fits the variation across the example motions of the selected control point or keytime. The linear basis functions are simply $A_l(\mathbf{p}) = \mathbf{p}_l$, the l^{th} component of $\mathbf{p}$, and $A_0(\mathbf{p}) = 1$. An ordinary least squares solution determines the $NumAdverbs + 1$ linear coefficients, a_{jkl}, that minimize the sum of squared errors between

$$\tilde{b}_{ijk}\left(\mathbf{p}_i\right) = \sum_{l=0}^{NumAdverbs} a_{jkl} A_l\left(\mathbf{p}_i\right), \tag{4.4}$$

and b_{ijk}, the actual B-spline control point (or key-time) being interpolated, where $\mathbf{p}_i$ is the adverb setting for the i^{th} example motion. Letting $\mathbf{b}_{jk}$ and $\tilde{\mathbf{b}}_{jk}$ denote vectors of each $b_{ijk}(\mathbf{p}_i)$ and $\tilde{b}_{ijk}(\mathbf{p}_i)$ for a fixed j and k, the linear approximation leaves the residuals

$$\bar{\mathbf{b}}_{jk} = \mathbf{b}_{jk} - \tilde{\mathbf{b}}_{jk}. \tag{4.5}$$

It is the job of the radial basis to interpolate these residuals.

Radial basis function approximation

At this stage, one radial basis function is defined for each example motion. Later, in order improve the technique, a hierarchy of radial basis functions will be used. This will be explored in Section 4.6. The radial basis functions are solely a function of the distance, $d_i(\mathbf{p}) = \|\mathbf{p} - \mathbf{p}_i\|_2$ between a point in the adverb space, $\mathbf{p}$, and the point in the adverb space corresponding to example motion i, $\mathbf{p}_i$. The radial basis itself, $R_i(\mathbf{p})$, has its maximum at $\mathbf{p}_i$ (i.e., where $d = 0$). In order to limit each example motion's influence to a local region of the adverb space, a radial basis function with compact support is used. This allows, as will be explained in coming sections, local refinement of the verb. There are a number of choices for the specific shape of the radial basis. For its combination of simplicity and C^2 continuity, a radial B-spline was used with a cross section of a dilated B-spline, $B(\frac{d}{\alpha})$. The dilation factor, $\frac{1}{\alpha}$, is chosen for each example motion to create a support radius for the B-spline equal to twice the Euclidean distance to the nearest other example motion. For $\alpha = 1$, the cubic B-spline has a radius of 2.0, thus α is simply the minimum separation to the nearest other example in the adverb space. Given this definition, it is clear that the example motions must be well separated.

The coefficients, r_{ijk}, can now be found for each DOF curve coefficient and key-time by solving the linear system,

$$\mathbf{D}\mathbf{r}_{jk} = \bar{\mathbf{b}}_{jk}$$

where $\mathbf{r}_{jk}$ is a vector of the r_{ijk} terms for a fixed j and k, and $\mathbf{D}$ is a square matrix with terms equal to the value of the radial basis function centered on motion i_1 at the location of motion i_2. Thus

$$\mathbf{D}_{i_i,i_2} = B_{i_1}\left(\frac{R_{i_1}\left(\mathbf{p}_{i_2}\right)}{\alpha_{i_1}}\right).$$

Results using this technique are shown in Chapter 6.

4.4 Kinematic constraints

Earlier, key-time annotation was introduced as a method for making the *Verbs & Adverbs* system handle motions with different duration or internal timing. When motions are blended, previously solid foot constraints can become somewhat wobbly, especially in regions of the interpolation space far from any example. The animations produced in these regions of the adverb space can be corrected by fixing kinematic constraint violations using standard techniques.

An important aspect of a motion is the constraints imposed upon it. For example, a walk is defined by a period of support during which one or both of the feet are stationary on the floor. Along with the DOF values, a motion needs to be able to answer questions concerning the active set of constraints at a time T.

Why is this important when the example motions encode the constraints implicitly? When a motion $\mathbf{M}$ is defined on a skeleton $\mathbf{S}$, the constraints will be correct, but often the motion or skeleton is modified or multiple motions are blended as in during a transition. In these situations, the constraints can be violated slightly. Since constraints like foot support are visually crucial to the perceived overall quality of a motion, even minimal violations are distracting. Effective IK is essential for cleaning up these problems and is essential for run-time-modifiable 3D figure animation.

Constraint specification language

A skeleton can be simply defined as a hierarchical collection of joints, some articulated and some not, each separated by an offset, possibly of zero length. Kinematic constraints are defined upon these joints. A common constraint, one used to describe support phases of a walk, for example, could be called a *stay-put* constraint. This can be defined by the tuple $\{t^s, t^e, \mathbf{j}, \mathbf{P}\}$, i.e. from

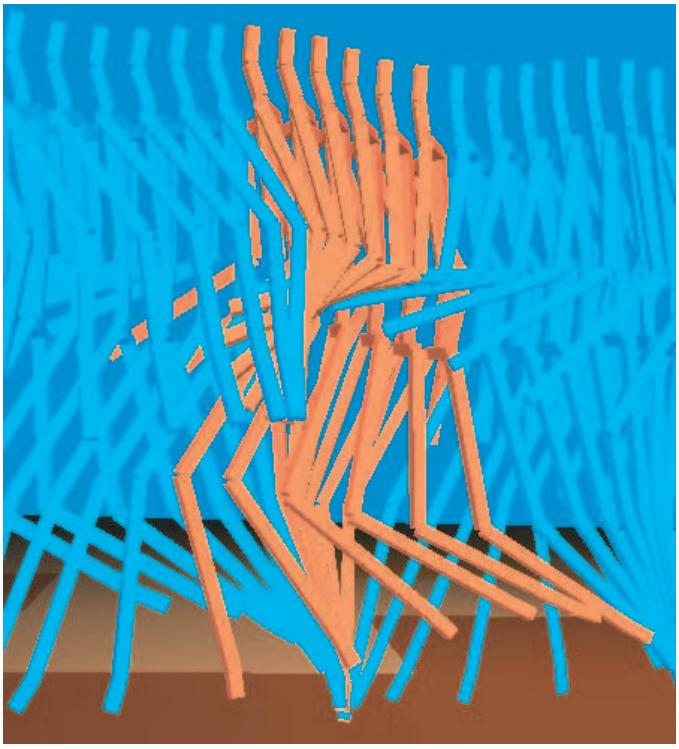

Figure 4.11: Support constraint for a walk

t^s to t^e the joint $\mathbf{j}$ should remain at point $\mathbf{P}$ relative to the coordinate frame in which the skeleton is rooted. The times are expressed in canonical in order to deal with the changing duration and timing caused by changing adverbs. More generally, a stay-put constraint, $\mathbf{C}$, can be defined by

$$\mathbf{C} = \{t^s, t^e, \mathbf{j}, \mathbf{P}, \mathbf{F}\} \tag{4.6}$$

where $\mathbf{F}$ is the frame of reference in which $\mathbf{P}$ is defined. Such a constraint can be used to handle many tasks. A common example would be the walk, as shown in Figure 4.11.

The red region shows the support constraint from $t = 0.38$ to $t = 0.68$. These verb times were recorded for the verb at a particular adverb setting $\mathbf{p}$. $\mathbf{F}_G$, the global frame is the constraint reference frame, $\mathbf{P}$ the point where the foot touched down in $\mathbf{F}_G$, and $\mathbf{j}$ the foot.

Another, less simple, example is catching a ball. Suppose one knew that the ball was to be caught at a certain time t^s and held until t^e. The constraint would be

$$\mathbf{C}_{\text{catch}} = \left\{t^s, t^e, \mathbf{j}_{\text{hand}}, (0,0,0), \mathbf{F}_{\text{ball}}\right\}.$$

This constraint, when enforced, would mold the hand to the trajectory of the ball as defined by the ball motion, when this is probably not the desired effect. A better solution would be to impose 2 constraints, one to catch the ball, and another to hold the ball:

$$\begin{aligned}\mathbf{C}_1 &= \left\{t^s, t^h, \mathbf{j}_{\text{hand}}, (0,0,0), \mathbf{F}_{\text{ball}}\right\} \\ \mathbf{C}_2 &= \left\{t^h, t^e, \mathbf{j}_{\text{ball}}, (0,0,0), \mathbf{F}_{\text{hand}}\right\}.\end{aligned}$$

$\mathbf{C}_1$ constrains the skeleton's hand to be at the ball from t^s to t^h, when the catch occurred. $\mathbf{C}_2$ constrains the ball, yet another actor, to be at the skeleton's hand from t^h to t^e.

These few examples form a basis of how a constraint specification language would be formed. Other constraint forms, like *look-at* which lines up a frame of reference along a vector to another frame of reference, and *track-path* generalizes the *stay-put* by replacing the position with a function returning position, round out a constraint specification. Other forms could be added as expressiveness dictates.

Automatic Kinematic Constraints

Some of the inverse-kinematic constraints in a motion can be detected automatically. Michael Gleicher takes this to an extreme in his animation work, automatically calculating hundreds of spacetime constraints on a motion [55]. Short of this, however, some simple techniques can be used to find some of the constraints, offloading some of the burden from the animator or motion capture technician.

Support constraints are the easiest to detect as they place a part of the skeleton, typically a foot, in a single spot in a reference coordinate frame over a short interval of time. A support constraint, $\mathbf{C}$, would be one of the form:

$$\mathbf{C} = \{t^s, t^e, \mathbf{j}_s, \mathbf{P}_s, \mathbf{F}_g\} \tag{4.7}$$

where $T(t^e) - T(t^s)$ is larger or equal to some prespecified minimal duration. The support joint, $\mathbf{j}_s$, needs to remain at $\mathbf{P}_s$ in the global reference frame, $\mathbf{F}_G$. Noting that motion capture data is noisy and that hand animation is rarely designed in a state of absolute perfection, the point $\mathbf{P}_s$ is unlikely to be pristine, so the search is for periods of time where $\mathbf{P}_s$ moves a small amount. If this is below some defined threshold, the centroid of that region is tagged as a support point. This technique was used effectively by Rose, *et. al.*, in the detection of support points duration the creation of torque-minimal transitions [124], discussed in Appendix B.

4.5 The verb design loop

The mechanism described thus far can be used to produce controllable animations which keep the aesthetic of motion capture of hand designed source, but another level of quality and interactivity of design can be achieved through refining the initially constructed verb. This process is depicted by Figure 4.12.

Once the designer creates a verb from the initial examples, the verb is inspected by executing the verb at different adverb settings, $\mathbf{p}$, both inside and outside the convex hull of the examples. In general, the verb's output is tolerable (but not necessarily optimal) inside this hull and is useful for extrapolation in a region near the hull. As $\mathbf{p}$ drifts further from the examples, the verb's output generally becomes unacceptable since linear approximation is an insufficient interpolation mechanism for animation. By fixing problem

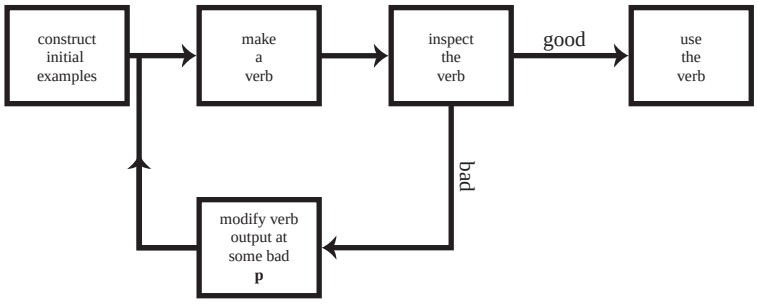

Figure 4.12: Verb refinement process

areas inside the hull and outside, the animator can both improve the overall quality of the interpolations and provide for a greater range of extrapolation.

If during the inspection process the designer finds a region which is unacceptable, the output of the verb at that $\mathbf{p}$ is taken. Note that this is a simple (basic) motion defined by DOF-curves which are themselves defined by coefficients. This motion is modified and reinserted as a new example. If this $\mathbf{M}(\mathbf{p}, \tau)$ is inside the convex hull of examples, typically it is relatively close to being acceptable. If it is outside, it can be close or far from useful depending upon from where in the adverb space it was taken. By iterating this process, more and more of motions defined for the adverb space become quality motions.

$\mathbf{M}(\mathbf{p}, \tau)$ is improved either through recapturing a trained motion capture actor or, more likely, with keyframing tools such as *SoftImage*$^{(\text{TM})}$ or *3D-Studio/Max*$^{(\text{TM})}$. Adding a new example into the verb requires setting the key-times and resolving the linear systems which define the radial B-spline and linear approximations. This can be done quickly since the current solutions are likely to be nearly correct for the new example. This refinement process becomes a new way for an animator to design animations and a new way for the animator to think about interactive animation.

A simple walk with one adverb, happiness, was designed by an anima-

tor trained with the *Verbs & Adverbs* system. The initial verb, defined with three examples, is shown in Figure 4.13. The examples for this verb span $\mathbf{p} = \{-10 \ldots 10\}$. The figure shows some extreme extrapolations, some useful ones and good interpolations. The animator chose $\mathbf{p} = 18$ (Figure 4.14) as a candidate for refinement. He fixed the foot slide problem, dealt with some of the more extreme eccentricities, and added this as an example to the verb. The results are shown in Figure 4.15 and exhibit a greater range of useful motions. This figure shows the happy side of the scale, showing the new example at +20. A comparison of the changes our animator made to the initial too-happy walk are shown in Figure 4.16. The initial too-happy is yellow and the new example motion made by changing it is shown in green.

Unfortunately, the animator loop can interact poorly with the verb mechanism as described so far. This interaction can turn a refinement step into one that decreases the overall quality of the interpolations for a large portion of the adverb space. This occurs when two (or more) examples are placed too near to one another. Fixing this shortcoming is the subject of the next section.

4.6 Multiresolution radial basis function approximation

Let us reconsider the walk example from the last section. It is a simple one-adverb verb so will illustrate the problem of example closeness well. The original verb consisted of three examples at happiness -10, 0, and 10. A small subset of the RBF approximation curves for some of the DOFs are shown in Figure 4.17. These curves are nicely smooth and interpolate the examples.

If we duplicate the middle example and place it at happiness 1.0, the smoothness of the approximations deteriorates as shown in Figure 4.18. This

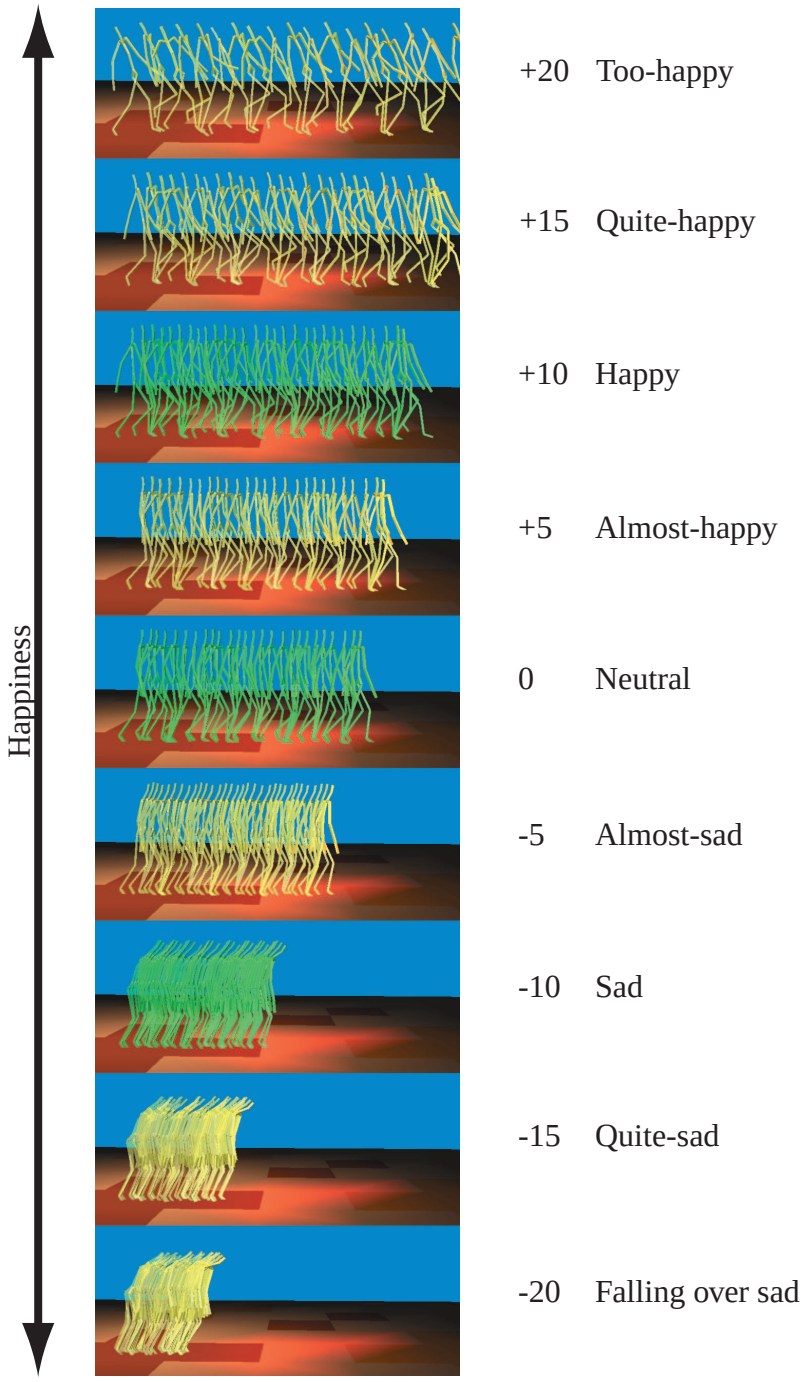

Figure 4.13: Initial verb to be refined

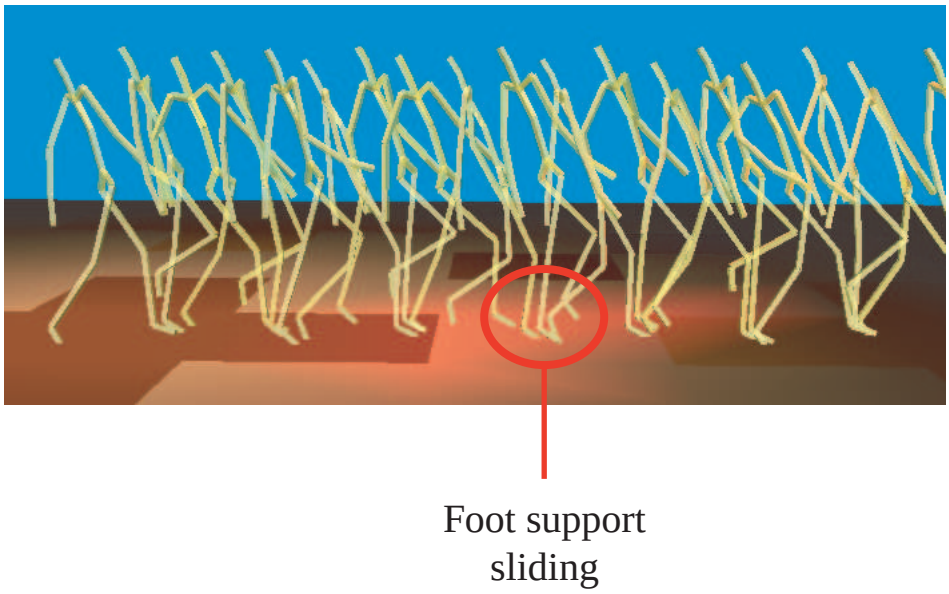

Figure 4.14: What's wrong with the overly happy walk?

is due to the way in which the radii for the radial B-splines are chosen: twice the distance to the nearest example. In essence, this causes the two middle examples to have a very local effect. The linear approximation represented by the line in the figures is also affected since the two middle examples are given double weight. The lack of smoothness in the coefficient interpolations affects the resultant motions as shown in Figure 4.19. Notice that the motions in the column A (no extra example) form a more smooth interpolation from neutral (happiness = 0) to near-sad (happiness = -3) than those in the other columns. Also notice that as the examples get closer together (columns B, C, and D), the approximation becomes one between the two extreme examples (happiness = 10 and happiness = -10) with a spike in the middle. In contrast, the column A interpolation is much more a function of all the examples all over the space.

The RBF approximation scheme can be improved through by taking two key steps: recursive clustering of close points and recursive multi-resolution approximation given the cluster assignments. The technique is detailed in the

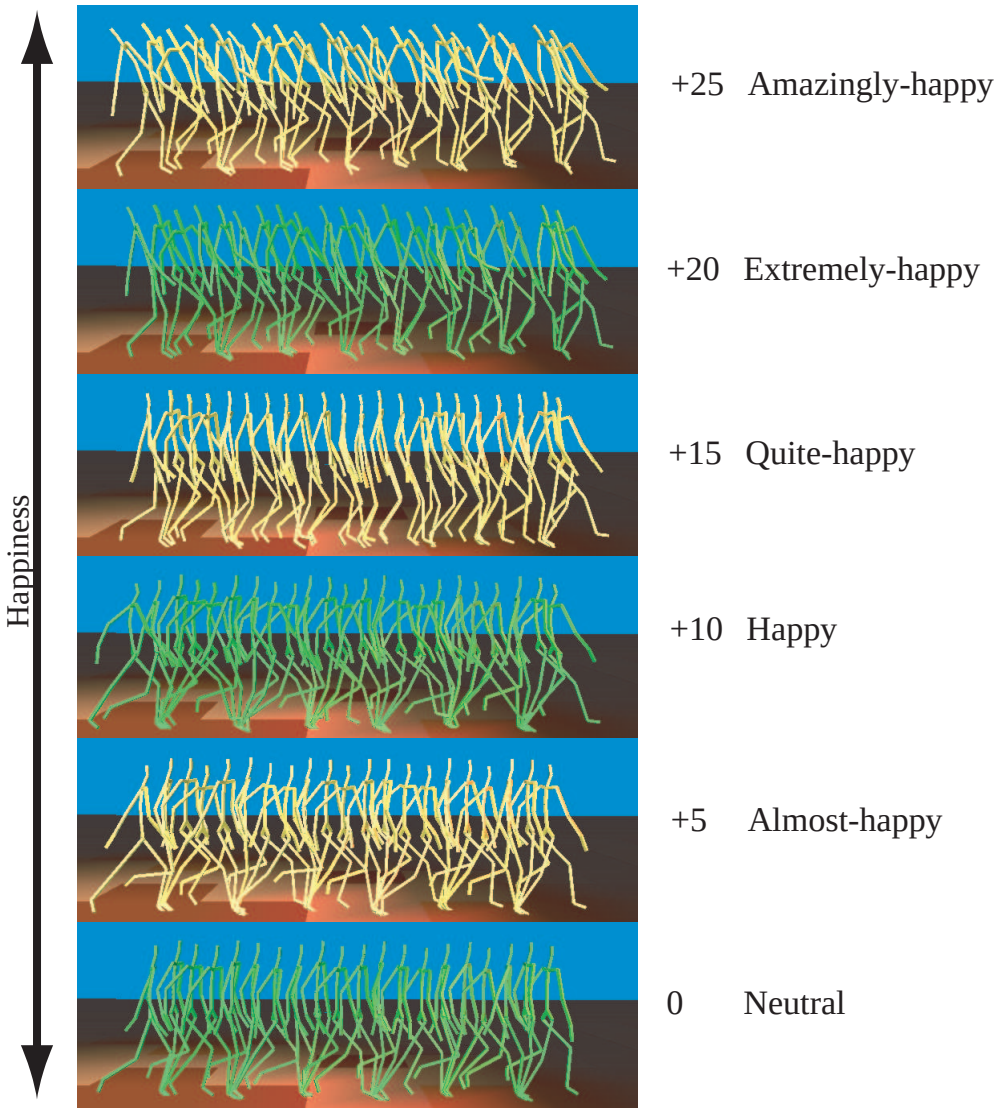

Figure 4.15: Improved walk

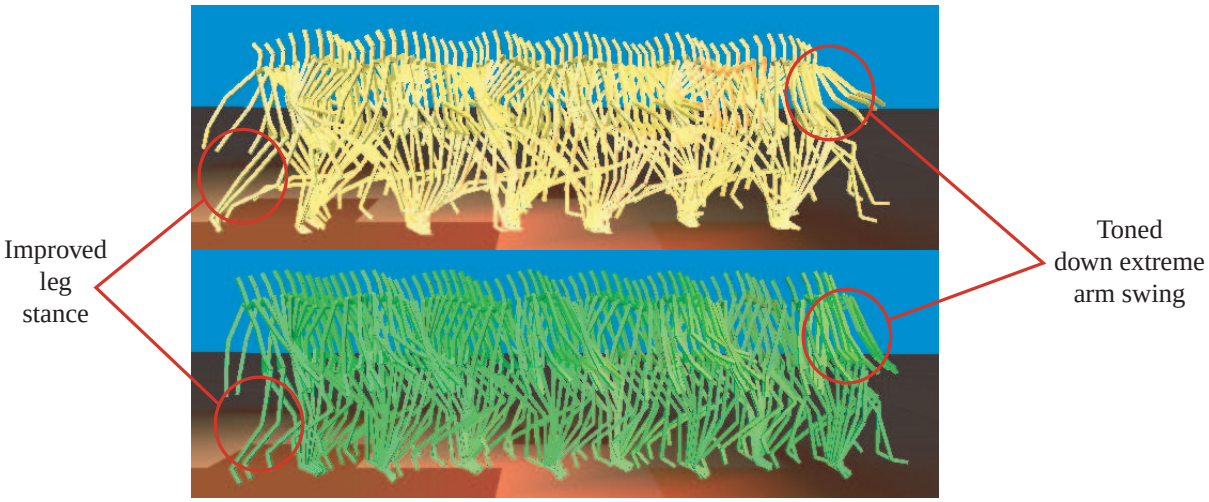

Figure 4.16: Differences at happiness = +20

remainder of this section.

Clustering

The problems with the approximation described thus far are two-fold. First, the radii for the two center basis functions are too small. This in effect cancels the contribution of these example throughout most of the space. Second, the center of the approximation is given too much weight in the construction of the linear approximation. Both of these problems can be alleviated through *clustering*. Clustering is the process of taking multiple points and treating them as one. Figure 4.20, for instance, shows the middle two points treated as a cluster. The faint line shows one level of the approximation with the two middle points treated as one. The thicker line shows the full approximation that interpolates all the example points.

The reader will likely see the value of this: clustering is used to reduce concentrations of dots into single (synthesized) examples and then RBF approximation is used on this new (reduced) data set. This is followed by a recursive refinement of the space thus yielding a multi-resolution RBF approx-

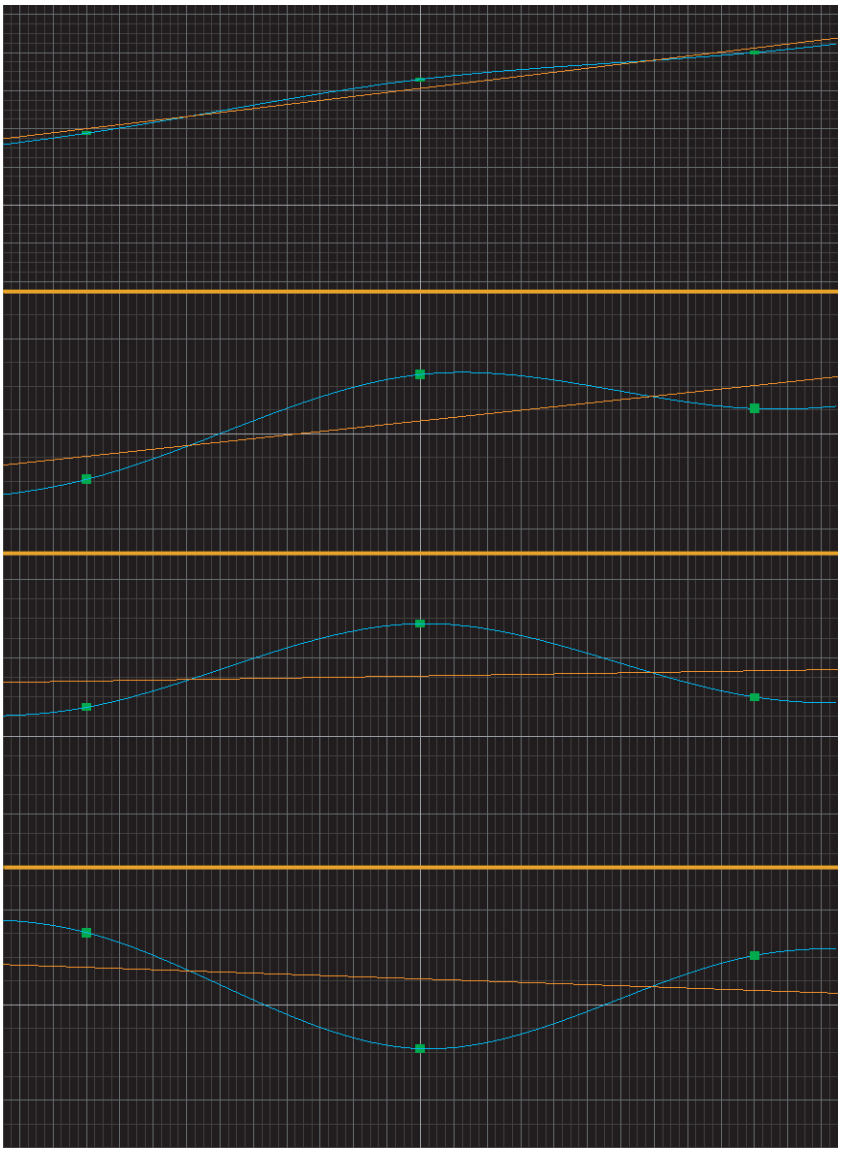

Figure 4.17: The simple technique works well for these examples. The green dots are the examples, the orange line the linear approximation and the blue line the complete approximation.

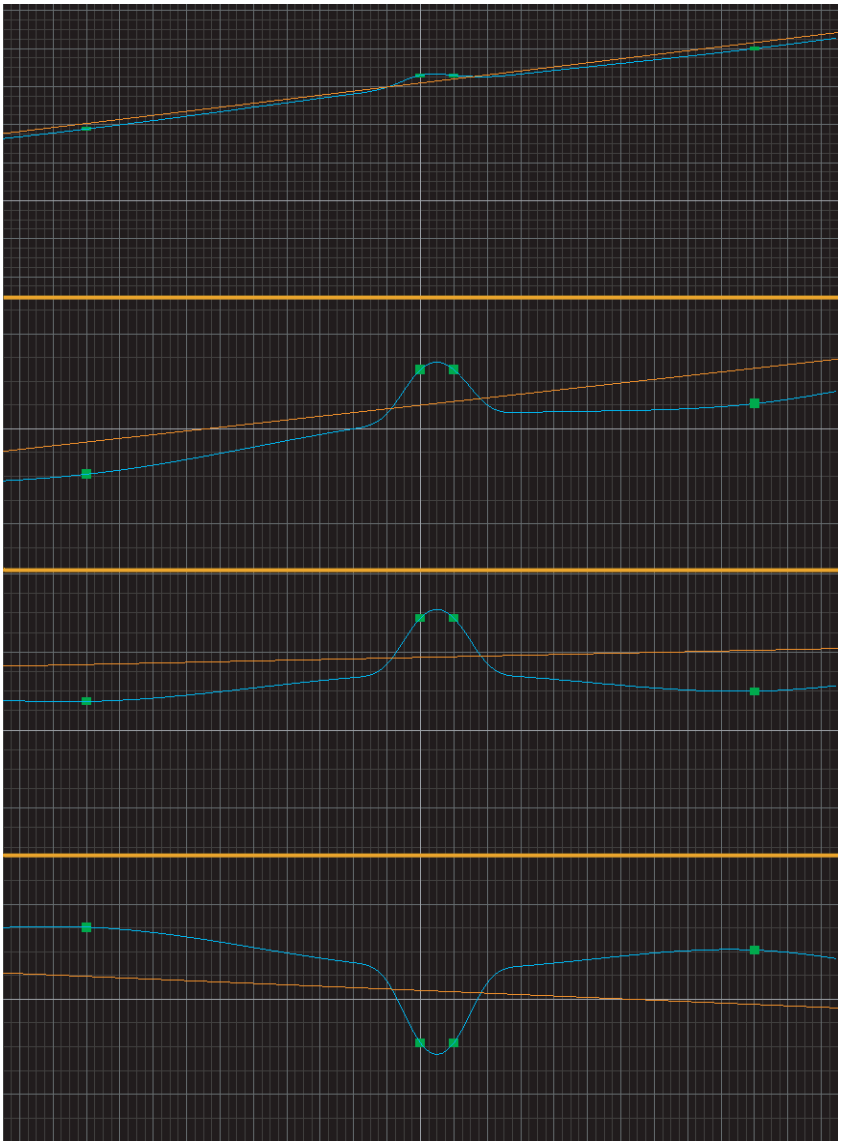

Figure 4.18: The simple technique fails for these examples. The green dots are the examples, the orange line the linear approximation and the blue line the complete approximation.

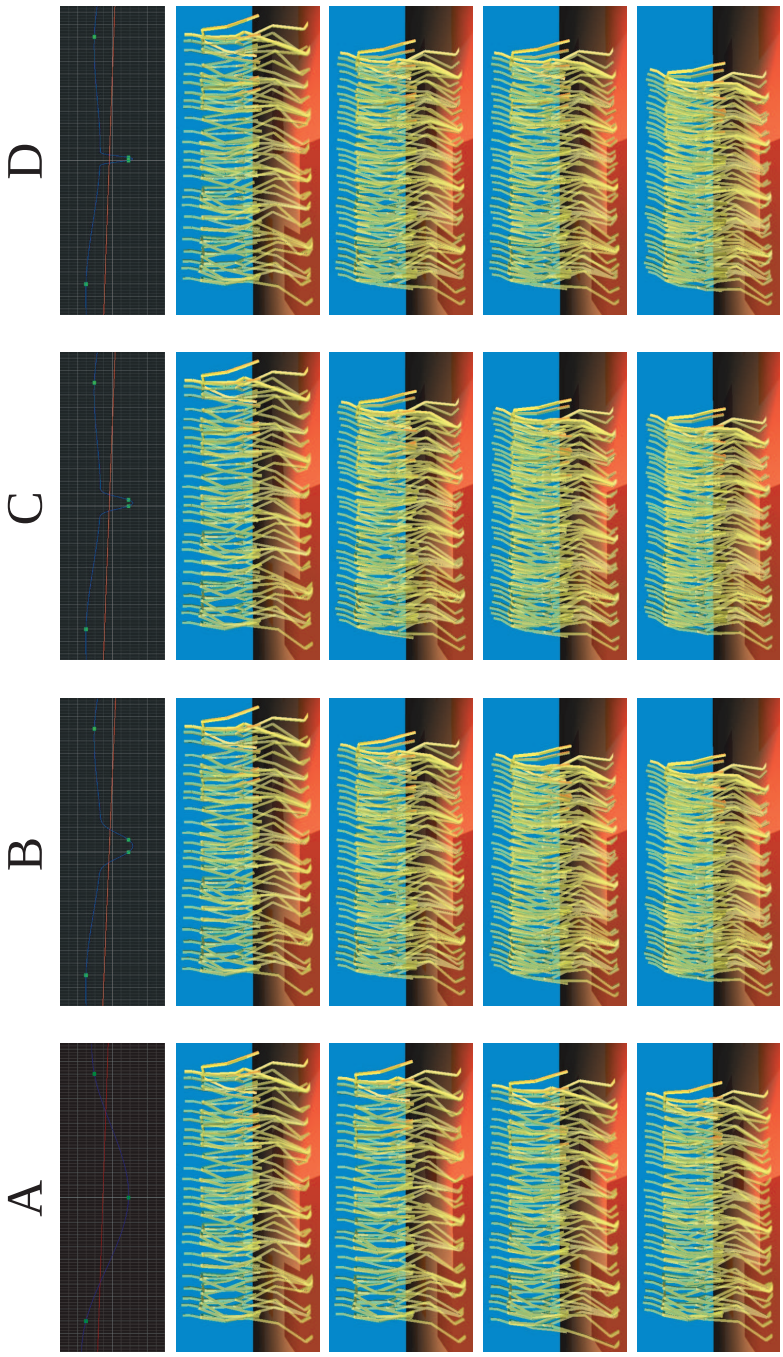

Figure 4.19: Ill-behaved interpolations lead to unsatisfactory results

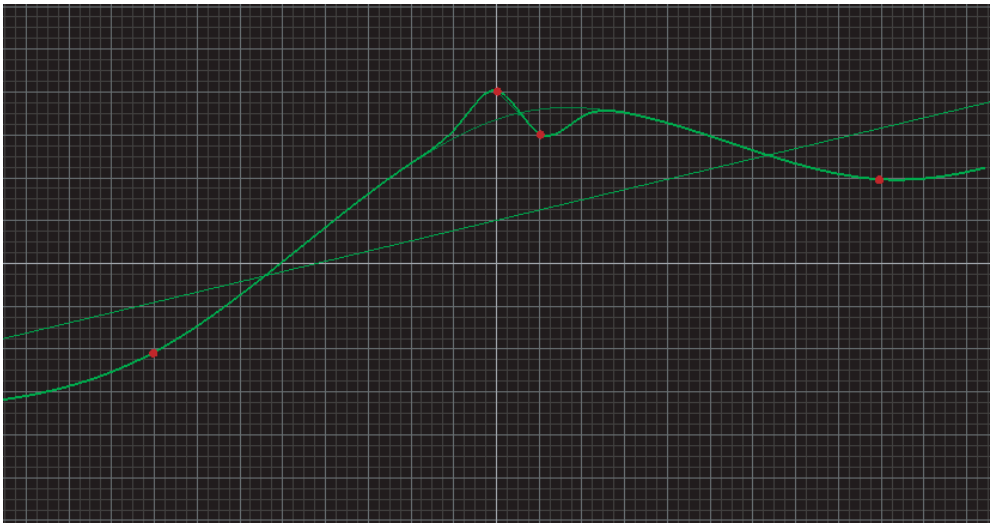

Figure 4.20: Two close examples made into one cluster

imation scheme. Before I describe the approximation scheme, the process of cluster construction is outlined.

For a verb with d adverbs, $d + 1$ examples are the minimum needed to establish the linear approximation. A least $d + 1$ clusters, therefore, must be used. The number of clusters, however, should be determined from the structure of the data. Regions in space finely sampled with examples should be treated as distinct clusters.

Clusters are established here using a randomized algorithm run many times in order to establish confidence in the solution. The basic algorithm is detailed in Algorithm 4.1.

Algorithm 4.1 Basic clustering scheme

```
BasicClustering (P ,C ,n)
01  // n , the number of clusters, C , to form over the k example points P
02
03  Establish n initial clusters C located randomly within the convex-hull of P
04
05  for each P_i , assign it to the nearest cluster C_j
06
07  for any empty clusters C_j , steal a point randomly from the clusters with
08       2 or more to establish a singleton cluster
09
10  // now the initial case is now established
11
12  repeat until no cluster reassignments occur
13      center each C_j as the average of the P_i in C_j
14
15      for each point P_i
16          if there is a cluster C_j closer than the one to which it is currently assigned
17              reassign P_i to C_j unless doing so would empty C_j
```

This algorithm performed many times and the best result is chosen to establish the final clustering. The best result is the one with the smallest cluster radii where each cluster radius is the distance from the center of that clusters to the farthest point in that cluster from the center. The algorithm converges very quickly, so it can be run often to establish a reasonably high confidence in the solution. A globally-minimal solution is not required. At line 10, each cluster is guaranteed at least one member. Line 17 ensures that no cluster is ever emptied. Singleton clusters, of course, are allowed. The quality of the clustering is assigned to be the sum of the radii of the individual clusters. If there are n clusters for n points, the clustering will have zero cost. Randomization is used to establish confidence in the solution rather than an analytical non-linear optimization.

Figure 4.21 shows the iterations for one run of this algorithm. The colors

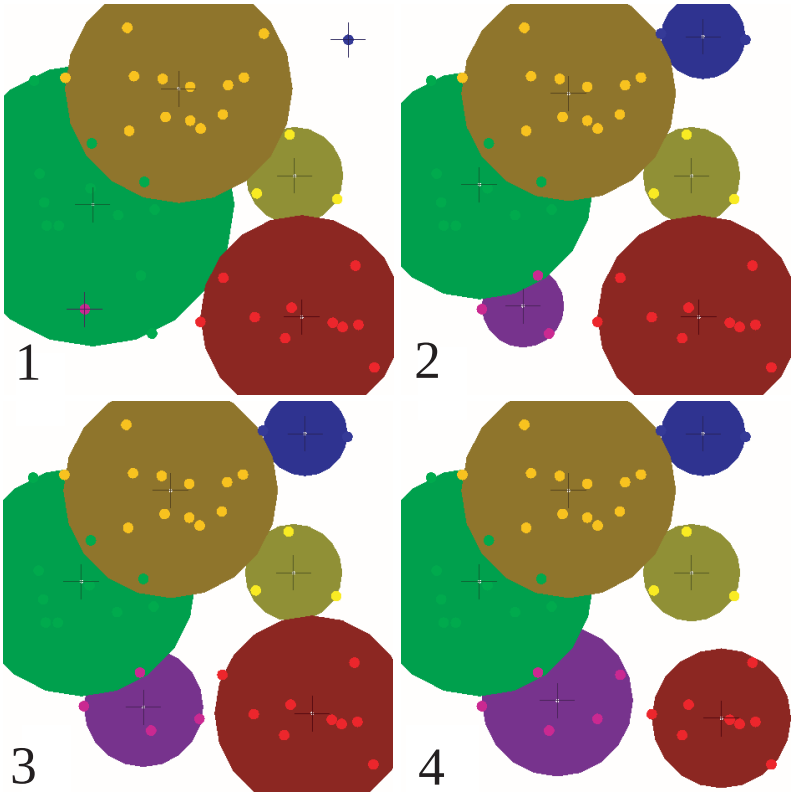

Figure 4.21: Clustering algorithm in action

indicate the clustering. Small circles are the examples. Color indicates cluster assignment. Cross-hairs mark each cluster center. Large colored circles indicate the cluster radii. Iterations 1 to 2 show the largest difference with the purple and blue clusters taking points away from the green and tan clusters respectively. The initial purple cluster is one of the "stolen" point taken from green, i.e. purple initially had no elements before the end of iteration 1. Iterations 3 and 4 see a balancing of the purple and red clusters. The algorithm took 4 iterations to arrive to a local minimum. Note that the cost is assured to monotonically decrease due to the nature of the algorithm. Running this algorithm many times with random starting guesses will build up confidence in the solution. As stated before, finding the global minimum is not a requirement.

One problem with this algorithm is that it takes as input the number

of desired clusters where this is a quantity ideally determined in a programmatic manner. Improving the algorithm, however, is not difficult, as seen in Algorithm 4.2.

Algorithm 4.2 Improved clustering algorithm

```
ImprovedClustering (P , C , maximum-cost, maximum-clusters)
    clustercount = d

    repeat until no splits occur
        BasicClustering (P ,C ,clustercount)

        cost = ∑(radii of C)

        if ((cost > maximum-cost) && (clustercount < maximum-clusters))
            clustercount++
```

C is the completed cluster assignments, and d the number of adverbs, dimensionality, of the points, P.

This algorithm will modify the clustering until the best result is reach subject the supplied restrictions on cost and cluster-count. For circumstances where the animator has clearly defined clusters due to the pattern of their modifications, however, it is not unreasonable to ask them to supply the cluster count, in which case the basic scheme of Algorithm 4.1 will work fine.

Multiresolution technique

Each cluster acts as an example point, thus collapsing groups of proximal examples. The base cluster level has few points widely spaced from one another. The radius assignment of twice the distance to the nearest example (cluster), therefore, does not ruin the approximation.

The linear term, Equation 4.4, was formulated as

$$\tilde{b}_{ijk}\left(\mathbf{p}_i\right) = \sum_{l=0}^{NumAdverbs} a_{jkl} A_l\left(\mathbf{p}_i\right).$$

In order to take into account the cluster level, another subscript is needed for the examples (clusters) $\mathbf{p}$:

$$\tilde{b}_{ijk}\left(\mathbf{p}_{0,i}\right) = \sum_{l=0}^{NumAdverbs} a_{jkl} A_l\left(\mathbf{p}_{0,i}\right).$$

The zero-th level of clustering corresponds to the coarse approximation. The highest (most fine grain) level is simply all the example points treated as singleton clusters and with cluster radii 0. Note that they do not have radial basis radii of 0. Those are calculated using the normal technique of twice the distance to the nearest other cluster (or on the finest level, nearest other example).

The residuals, $\bar{b}_{jk} = b_{jk} - \tilde{b}_{jk}$ (Equation 4.5), need to be solved for by the radial basis functions. Rather than doing this in one step, however, it is done iteratively from coarse to fine resolution. The coefficients, now $r_{c,ijk}$ to account for this residual can be found for each DOF coefficient and keytime by solving the linear system

$$\mathbf{D}_c \mathbf{r}_{c,jk} = \bar{\mathbf{b}}_{c,jk} \tag{4.8}$$

where c is the cluster level from 0 (coarse) to *NumClusterLevels* (all-examples) and $\mathbf{r}_{c,jk}$ is a vector for the $r_{c,ijk}$ for a fixed c, j, and k, and $\mathbf{D}$ a square matrix with terms equal to the value of the radial B-spline centered on point p_{c,i_1} at the location of p_{c,i_2}. Thus

$$\mathbf{D}_{c_{i_1,i_2}} = B_{c,i_1}\left(\frac{R_{c,i_1}(p_{c,i_2})}{\alpha_{c,i_1}}\right). \tag{4.9}$$

The value of $\bar{\mathbf{b}}_{c,jk}$ is the sum of the linear approximation and the contribution of all the layers up to $c-1$. The interpolated coefficients, therefore, can be calculated as

$$b_{jk}(\mathbf{p}) = \sum_{l=0}^{NumAdverbs} a_{jkl} A_l(\mathbf{p}) + \sum_{c=0}^{NumClusterLevels} \sum_{i=1}^{NumExamples} r_{c,ijk} R_{c,i}(\mathbf{p}). \tag{4.10}$$

4.7 Efficiency concerns

There are three main efficiency concerns, the speed of the system at author time and run time, and the storage efficiency of the representation.

Runtime evaluation

This technique is efficient. It can be used to drive animations for multiple figures at real-time frame rates. Indeed, the system is bound primarily by the frame-rate of the 3D rendering of the figure. Assuming a constantly changing adverb setting $\mathbf{p}$, a certain number of coefficients must be evaluated at each timestep. The number of the coefficients is $(4 \cdot NumDOF + NumKeyTimes)$ for a B-spline encoded solution since for a given time τ, 4 coefficients are required for each DOF curve and all the key-times are required to transform τ into canonical time t.

Evaluating each coefficient requires an evaluation of a hyperplane and the radial basis functions for each of the examples and the clusters. Each clustering level must have at least one fewer cluster than the next higher level, which could lead to $\mathbf{O}(n^2)$ evaluation time where n is the number of examples in finest level of clustering, i.e. the original number of examples. If we assume a halving of examples (clusters) at each level of clustering, a conservative assumption, we can simplify to $1 + 2 \cdot n$ or just $\mathbf{O}(n)$.

Author time evaluation

While runtime evaluation time is of paramount importance, the cost to set up the interpolation spaces is also an issue, especially considering the iterative refinement aspect of the animator loop.

Setting up the initial hyperplane requires $\mathbf{O}(n^3)$ in order to solve the non-

square linear system using singular value decomposition, which was chosen for its robustness. As n, the number of examples (or coarse level clusters) is typically small, this use of SVD does not overly impact efficiency. We use Gauss-Seidel at each level of the RBF approximation, since the $\mathbf{D}$ matrices are square and diagonally dominant. This iterative algorithm will converge quickly as the solutions at each level of the hierarchy are already reasonably well approximated by the approximation at the lower levels of the hierarchy.

Furthermore, clever bookkeeping can facilitate the updating of the solution without starting over. Thus, in conjunction with Gauss-Seidel we achieve fast re-solving of the systems as the animator moves example points around. Large changes in the clustering will require new solutions.

Storage requirements

The MRBF verb encoding is space efficient and serves as a reasonable compression scheme for repertoires of similar motion when compared with typical techniques commonly employed in the game industry. The example motions that constitute a verb are stored in some form either as simple motions such as B-spline or piecewise-linear, or as functional compositions with simple motions as the leaves in that expression graph (Chapter 3). Let us assume they are stored simply with each example requiring $(NumDOF \times NumCP)$ coefficients of storage. This is not a restriction since any functional composition can be well approximated with a simple representation as desired.

The total coefficients required for the examples, therefore, is $(NumDOF \times NumCP \times NumExamples)$. For the the walk verb from Christian's motion capture session, Figure 6.2, that is $(44 \times 32 \times 20) = 28160)$ coefficients.

The MRBF approximation requires $(NumKeyTimes + NumDOF \times NumCP)$ separate interpolation spaces. Each of those requires $(NumAdverbs + 1)$ coeffi-

cients for the linear approximation and $\sum_c NumExamples_c$ coefficients for the radial basis functions. Using the same conservative estimation of cluster halving, $2 \cdot NumExamples_{NumClusterLevels}$ coefficients are required, twice the number of examples before clustering. For the walk, therefore, each MRBF space needs $(4 + 1) + (2 \times 20) = 45$ coefficients for a grand total of $45(4 + 1408) = 63540$ coefficients, not appreciably more than the storage requirement for the examples themselves. Once the verb is complete, the examples need no longer be stored.

A fine question to ask is whether fewer basis motions are needed in a *Verbs & Adverbs* based application than another. To get the same gradation of motion, games will typically store many variations, often many more than the number of examples needed to construct a verb with the same range of motion. A verb, therefore, is a more efficient representation of a particular motion than storing a number of variations. This is not surprising, as the MRBF interpolation seeks to capture what a particular motion does over a range of variation.

4.8 MRBF interpolation and human biomechanics

As will be seen in Chapter 6, walking, running, reaching, and idling have all been shown to be amenable to the *Verbs & Adverbs* technique. Smooth blends of many motions in the multi-dimensional adverb space yields convincing controllable animations. Whether this technique can, in general, work for a myriad of human motions with many adverbs, is still an open question. The *Verbs & Adverbs* technique interpolates the motions produced by a biological system, rather than the control system used to generate the motions. Is

smooth interpolation of motion a reasonable things to do?

A human's motion is dictated by three primary factors: the environment in which he is placed, his internal control system (brain, neurons, reflexes), and the dynamical properties of the underlying biomechanical system (joints, tendons, muscles, bones). Hogan [71] [72] shows that the two latter systems, when coupled, form, in essence, a single biomechanical system. Furthermore, he showed that in the absence of new forces (typically muscle forces), the system only decreases its overall kinetic energy. This is due to factors such as friction in the joints. Biomechanical systems are thus critically damped and since muscles can only inject a finite amount of energy, the system as a whole will always have bounded (and in general decreasing) energy. The end result, Hogan showed, is that motions produced by biological systems are smooth.

This is hardly surprising, but extremely important to establish the usefulness of interpolation for simulating biomechanical systems. The damped, smooth nature of human physiology has been used effectively in the graphics community. Hodgin's group relies upon the critically damped nature of the system in their work on controllers [70] [68]. Grzeszczuk [59] showed the usefulness of neural-network approximation for learning dynamics control strategies and, lately, for learning dynamics systems in total [60]. Neural-networks, of course, are smooth n-d function interpolators like RBFs. RBFs are often formulated using neural-network-like structure and nomenclature. Gelfand, Lane, Handelman, Gullapalli [84] [62], and others showed how neural-network-like entities called CMACs (Cerebellar Model Articulation Controllers) could be used to effectively generate human motion control strategies in concert with a dynamics simulator.

Dynamically simulated motions have a major drawback in that the animator, a talented ally in the production environment, is removed from the

process. Interpolating control systems, therefore, is insufficient for making best use of an animator or a library of motion capture data. Actual motions, the things animator's develop, are what must be interpolated.

A biological system's motion is proportional to the second derivative of the forces acting upon it ($\mathbf{F} = m\mathbf{a}$). These accelerations are driven by the control system, which was shown to be smooth. Continuous acceleration imply smooth velocities and, by extension, positions. Smooth changes in the controller yield smooth changes in the motions produced by the system. Interpolating motions using a C-2 continuous interpolation scheme like MRBFs, therefore, is a reasonable way to interpolate human motion.

4.9 Some further problems

There are some questions which could be posed concerning the *Verbs & Adverbs* mechanism:

- What is an "accurate" parameterization of the examples?
- How do the adverbs interact with one another, i.e. are the chosen adverbial axes orthogonal?
- What axes are needed in order to capture the space of human emotion in movement?

Each of these constitutes an open problem related to *Verbs & Adverbs*.

Example parameterization

Ensuring the accurate assignment of adverbs is difficult. For some adverbs, namely the structural ones, this is a relatively easy process since the adverb

value is objective. In these cases, adverb assignment can be done automatically. The reach verb, as will be seen in Chapter 6 (Figure 6.4) has three axes: the **X**, **Y**, and **Z** offset of the hand goal from the position of the body root at the start of the motion. Once the goal-reached key-time is set, these adverbs can be set automatically.

If, on the other hand, the adverbs were subjective, such as "happiness", setting the adverbs is a trickier task. The meaning of a 10.0 happiness value is, at best, ill-defined. While psychological studies of human motion might prove useful, the work presented here trusts the designer to recognize the differences between the examples and set the parameters in an appropriate way.

Additionally, the designer must ensure that the suite of verbs which will later be used to construct a verb-graph (Chapter 5) are parameterized in a consistent way. A 10.0 happy should have the same emotional impact for a walk verb as it does for a idle verb. Likewise, if verbs from many designers are used to form a single application, care will need to be taken to ensure even treatment of the adverbs so as to avoid any strange transitions. This could occur, for example, if one verb had a distribution of examples along a happiness axis from $[-100 \ldots 100]$ and another from $[-10 \ldots 10]$ when in fact the ranged over the same gamut of human happiness.

Adverb orthogonality

The walking verb (Figure 6.2) used a number of examples which had non-zero components in multiple adverbs. The "angry" example, for instance, was deemed to be unhappy and knowledgeable. Thus, the unhappiness stemmed from an understood source yielding an angry reaction. The "despondent" walker was possessed with unhappiness not understood. This pop-psychology mixing of human emotion is open to many pitfalls. So far, these problems have

not proved too troublesome.

When the walk verb was designed, the axes were chosen and a number of examples placed in the space defined by those adverbs. The unhappy plus knowledge equals anger was a rationale used to place the angry example. A different set of axes, however, could easily have been chosen. "Angriness" could be one such axis.

Furthermore, the orthogonality of happiness and knowledge is in question. Whether a person can be happy with no correlation of knowledge is a psychological question outside the scope of computer graphics. Primarily, the *Verbs & Adverbs* group chose to ignore this potential problem since it has not seemed to affect the common-sense usage of our system from the human figure animation standpoint. If *Verbs & Adverbs* were to find usage in the psychological research community, these issues would need to be addressed more satisfactorily. An overview of one such effort follows.

Complete parameterizations

The emotional axes used in our examples form a very ad-hoc set of this the author and animators in the project thought useful. Clearly, happiness and sadness are insufficient adverbs if one's goal is to parameterize the complete gamut of human emotion. An interesting question to ask, therefore, is what set of adverbs can be used to describe the emotional content of all human emotion. Movement analysts have been interested in this very subject. Bartenieff [17] provides a good introduction to the field of Laban notation, which is one of the major techniques for analyzing and parameterizing human motion.

Rudolf Laban created a form of analysis which has come to be called Laban analysis or Laban notation, used often by movement analysts and dancers. He identified a number of parameters which can be used to describe the qual-

Effort	Indulging	Figthing
space	indirect	direct
weight	light	strong
time	sustained	sudden
flow	free	bound

Table 4.2: Range of effort in Laban notation

itative and structural characteristics of a motion.

Effort is on the of the key ideas in the notation, and is divided into four sub-categories: space, weight, time, and flow. Each sub-category ranges from indulgent, or non-resisting, to fighting, resistive motions. Table 4.2 shows these ranges.

These four parameters; space, weight, time, and flow; can be considered adverbs in the construction of a verb with the addition of some parameters for simple structural elements such as direction of motion, turning radius, etc. An angry motion, for example, would be direct, strong, and sudden. A sad motion would be sustained and bound. Whether Laban notation truly has enough parameters to place the emotion characteristics of human motion is an open question.

While Laban analysis provides a systematic and consistent framework, it is not a calibrated one. A space of 10.0 is not a well-defined quantity, just as a happiness of 10.0 is not as mentioned before. A talented movement analyst trained in Laban notation, as a talented animator using an ad-hoc parameterization, can develop a reasonable calibration and from them, reasonable verbs. We have not, at present, worked with one so trained, though we may in the future.

4.10 Conclusions

This chapter described the *Verbs & Adverbs* mechanism, detailing the mathematics behind both single- and multi-resolution radial B-spline approximation. Examples of the system's output were followed by a short discussion of open problems. Additionally, the connection between modeling human phenomena and human processes was made.

Verbs are short segments that can perform a motion in an infinite variety of ways. In order to create long animations, however, a mechanism for gluing verbs together must be constructed. The *verb-graph* is one such mechanism and is the subject of the next chapter.

Chapter 5

The verb graph:

a verb management scheme

Using multiresolution radial B-spline approximation, controllable animation segments, *verbs*, can be constructed. These verbs can exhibit subtle aesthetic variations as well as structural difference. Verbs, however, are short movements. In order to make use of these verbs, a scheme for concatenating and transitioning must be designed. In this section, the *verb graph* is detailed, which is one possible solution to this problem. The work of Perlin and Goldberg [113], Blumberg and Galyean [20], and Badler [5], as well as *The Motion Factory*'s *Motivate*$^{(TM)}$ product, are alternate methods. A transitioning mechanism will be introduced in Section 5.5 which is used to bridge the gaps between motions to yield a seamless animation under the control of the user, an automated system, or a combination of the two.

5.1 Overview

The verb graph is an object and a set of algorithms designed to structure the overall flow of an animation. Verbs are the basic unit of expression and are chosen to correspond to a logical unit of action, such as a walk-cycle. While larger

Figure 5.1: A linear verb graph

sequences could be designed using the *Verbs & Adverbs* technique, breaking long animations up into their most primal sequences enables more opportunity for greater interactivity. This has been shown to be a valuable technique by other researchers. Additionally, many interactive character games are designed using this technique.

A *transition* occurs when one verb flows into another and is necessitated by splitting animations into small units. The mechanism used in this work is low-overhead and can only transition relatively similar motions. A transition, for example, should not be used to bring a character from a run all the way into a sleeping motion. These actions are quite dissimilar and a transitioning mechanism capable of bridging these motions would need to be quite sophisticated. For similar motions, however, a relatively simple transitioning mechanism can be used in a convincing way. More sophisticated transitions limited to non-parameterized motions were explored during the early phases of the *Verbs & Adverbs* work. These transitions seek to minimize the amount of torque needed and are detailed in Appendix B. Note that even this technique would still not have the knowledge to transition from a run to a sleep motion effectively; there are simply too many logical steps the actor must perform to bridge these two motions.

The verb graph is used to structure what types of motions can follow which other motions. The nodes of the graph are the different actions a character can be called upon to perform and the arcs are the transitions that the designer builds using a transitioning mechanism. The graph becomes the object manipulated by a system in order to make the animation progress.

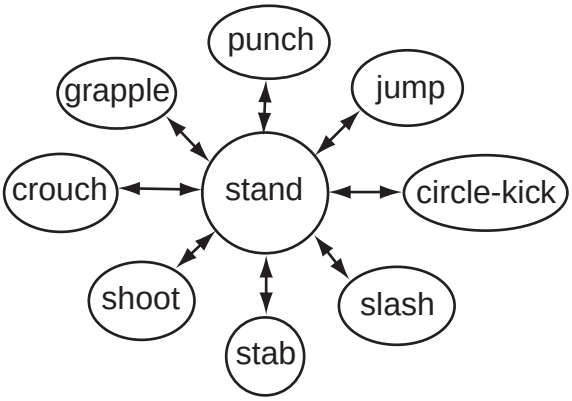

Figure 5.2: A "home position" style verb graph

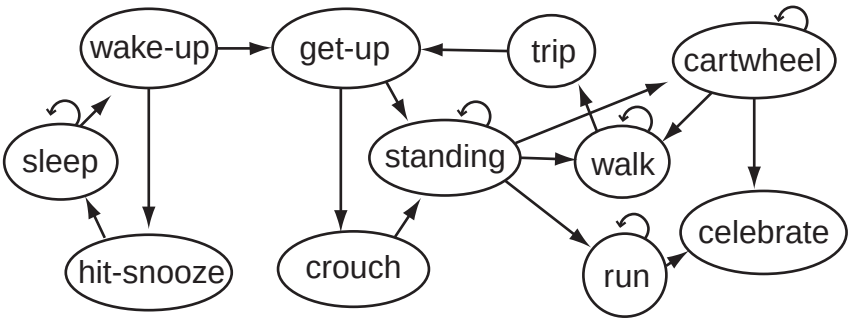

Figure 5.3: An arbitrary verb graph

Different verb graph topologies can be developed. Each has its use, though this work will deal mainly with the unrestricted graph topology. An animated movie, for example, could be thought of as a linear verb graph of the form depicted in Figure 5.1. A slightly more complicated verb graph is often used in video games, the *home-position* style graph, as seen in Figure 5.2. It has a special stance in which all animations begin and end, thus making the transition-design task easier and the overall bookkeeping for the animation engine simpler. This kind of graph is seen in popular games like *Mortal Combat*$^{(\text{TM})}$, *Primal Rage*$^{(\text{TM})}$, and *Tekken*$^{(\text{TM})}$, all successful fighting games in arcades around the world. The work here is primarily concerned with unrestricted graphs, an example of which is shown in Figure 5.3. Complicated relationships between verbs can be described with these graphs, forcing the animation to flow along a logical path from motion to motion. Transitions take place at logical junctions, not simply at convenient times. Home position style graphs are much more restricted in the scope of the motions they can reasonably contain. The bookkeeping and control of unrestricted graphs, however, is more difficult than that of a simple graph.

5.2 The verb graph formalism

The verb graph is a tuple of the form

$$\{V, T, S, G\} \tag{5.1}$$

where V is a set of verbs, parameterized or otherwise, T a set of transitions generated using the method to be described in Section 5.5, S the set of starting verbs where $S \subseteq V$, and G a set of gesture verbs, those which may be played concurrently with the verbs in V using the gesturing mechanism to be described in Section 5.6. The state of the verb graph will be introduced later,

but for the sake of the next section, there is a queue of pending verbs. The first verb on the queue, Q_0 is the one currently playing.

Time and the verb graph

As was seen earlier in Section 3.4, time is not a trivial notion for an animation system. Animation-time, τ, is the natural timeframe of an animation. An animation designed to play for a certain time τ^d is defined for a portion of the real timeline from $[\tau_i^s \ldots \tau_i^e]$. In chapter 4, the construction of parameterized verbs was detailed. Parameterized motions are problematic because their realtime durations can change due to their current parameterization. Additionally, the parameter setting can change the relative duration of the pieces of an animation. Another kind of time, canonical time, t, was introduced to deal with these changes in phrasing or duration. One can be sure for any conceivable adverb setting $\mathbf{p}$, that a given t indicates the same structural portion of a motion. Canonical time is therefore useful for annotating the verb with information like kinematic constraints and sound events like a footfall. It is also useful, as will be seen in Section 5.5, for specifying transitions. Animation-time is the type of time used for the verb graph. It starts at $\tau = 0$ and progresses towards $+\infty$. Verb-time, T, implies a finite duration, T^d, so animation-time is used instead for the verb-graph. Verb-graphs, as will be seen, can be played for an indefinitely long period of time.

A verb graph is initialized to start at $\tau = 0$ at the beginning of one of the verbs in its start set, i.e. at verb-time $T = 0$ for that particular verb. When the front of the verb queue, Q_0, is completed or transitioned from, that time needs to be recorded. τ^{in} will mark the animation time when the current Q_0 came to the fore of the queue. When the item at the front of the queue is a verb (as opposed to a transition), it is rare that it was entered at the exact

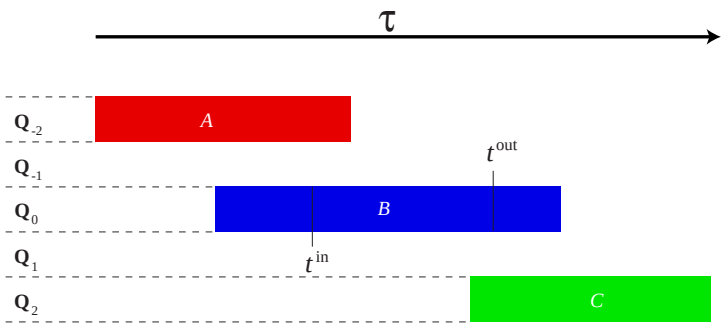

Figure 5.4: Verb queue times

beginning of that verb or that it will exit at the very end due to the nature of the transitions. Hence, two times t^{in} and t^{out}, expressed in canonical time, will mark the portion of the Q_0 verb which will actually be played as τ progresses. t^{in} is defined by the transition which came before Q_0 and t^{out} by the transition in spot Q_1. These times are indicated by Figure 5.4.

As the queue past spot Q_0 is not fixed, t^{out} can change at any time as a new transition is queued to exit from the currently executing verb. At a given time, however, t^{out} is always known. If we know τ^{in}, t^{in}, and t^{out}, we can estimate when the current Q_0 will be exhausted, i.e. τ^{out}, with this equation

$$\tau^{\text{out}} = \tau^{\text{in}} + T_0(t^{\text{out}}) - T_0(t^{\text{in}}) \tag{5.2}$$

where $T_0(t^{\text{out}})$ projects t^{out} into the verb-time of the verb at the front of the queue, Q_0, and similarly for $T_0(t^{\text{in}})$.

The last few paragraphs have assumed that Q_0 is a verb, not a transition, but since verbs and transitions must each adhere to the motion formalism (Chapter 3), very little changes. The t^{in} and t^{out} for a transition in spot Q_0 are simply 0 and 1 since an entire transition gets played. It takes the entire transition to transition properly from one verb into another.

When τ reaches the currently evaluated τ^{out}, the τ^{in} and t^{in} numbers must be recalculated and the queue advanced. Since they are not fixed, τ^{out} and

t^{out} are always calculated when needed given the current adverb setting **p**. Algorithm 5.1 resets the static values when the item at the front of the queue is exhausted.

Algorithm 5.1 Reseting the times

```
UpdateQueue (Q , τ^in , t^in , t^out)

    if Q_0 is a transition and Q_1 is the verb we are entering
        t^in = t_e^B from Q_0 (exit from the end of the current transition)
        t^out = t_s^A beginning of transition out of the verb we're entering now

    else Q_0 is the verb we're leaving and Q_1 is a transition which will be played in full
        t^in = 0
        t^out = 1
    end if

    τ^in = τ^out

    pop Q
```

The verb graph's state

The verb graph is merely a description of the possible courses an animation could take. Another entity is required to detail the state of the actor being controlled with the graph and the internal state of the graph. Separating these two allows a system to control multiple actors with one verb graph object. The verb graph state object is a tuple of the form

$$\left\{\tau^{\text{now}}, t^{\text{in}}, t^{\text{out}}, \tau^{\text{in}}, Q, G, \Theta, C\right\} \tag{5.3}$$

where τ^{now} is the current animation time of the animation measured from 0, when it began, Q, the primary verb queue, G the set of active gestures (to be described in Section 5.6), Θ the vector describing the actor's configuration, i.e., its DOF values, and C the set of active constraints at τ^{now}. τ^{in}, t^{in}, and

t^{out}, were described previously in this chapter. The first six entries in the Θ vector describe the position and orientation of the root of the creature. They are used specially by some of the algorithms to be presented in this chapter. Of particular interest are the **X** and **Z** translations and the **Y** rotation, otherwise known as an actor's "heading".

The verb queue indicates what verb is currently playing and which verbs are pending. Verb queues are of the form

$$\{v, t\}^* v$$

or

$$\{t, v\}^* tv$$

where the v's are verbs and the t's transitions between them. Note that the primary verb queue cannot be allowed to empty as the animation would have nowhere to "go" and would need to end or restart in a non-smooth manner.

The root of the actor comprises the first 6 DOFs, though only the **X** and **Z** translations as well as the **Y** rotation (the heading) are of interest. As τ^{now} marches forward, the root DOFs need to be handled so as to make the animation look like one seamless piece. This is handled in a way similar to the handling of the concatenation verb as detailed in section 3.9.

Updating the state of the verb graph from the currrent time τ^{now} to a new time τ^{next}, $\tau^{\text{now}} < \tau^{\text{next}}$, is detailed in Algorithm 5.2.

Algorithm 5.2 Verb-graph increment function

```
PositionDOFs (τ^next ,S)
    T^now = S.τ^now − S.τ^in
    T^next = τ^next − S.τ^in

    for root-DOFs j
        Δ Θ_j = θ_0j(T^next) − θ_0j(T^now)
        Θ_j = R(Δ Θ_j)

    for non-root-DOFs j
        Θ_j = θ_0j(T^next)

    S.τ^now = τ^next
```

τ^{next} is the time to which the system wishes to advance, S the state of the verb graph. θ_{0j} is the DOF position function for the verb at the front of the queue (Q_0). The R indicates a concatenation of the delta change in the root according to the general concatenation algorithm introduced in the previous chapter.

Assuming no transitions occur from τ^{now} to τ^{next}, to update the state of the graph from τ^{now} to τ^{next}, the information for the root DOFs is calculated for these two times. At first glance, this might seem strange. After all, the actor currently contains the information for τ^{now}. There are two shortcomings with using this information, one trivial and one more serious. The information for the root is transformed to follow smoothly from the last time it was evaluated rather than the raw root information gathered from the verb itself, which is likely near the point of origin. A verb graph can take the actor far from the origin. Of course, the raw information could be saved for reference. The important reason, however, is that the verbs are parameterized, therefore changing, so if we are to find out what change occurred for the root motion at a given parameterization from τ^{now} to τ^{next}, both times must be evaluated at

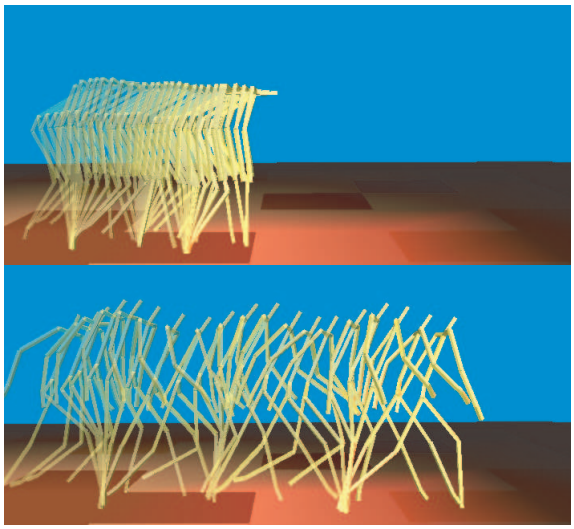

Figure 5.5: A simple verb parameterized by adverb *happiness*

the current parameterization.

For example, verb $\mathbf{M}_i$ is parameterized by 1 adverb, *happiness*, which has the side effect of moving the actor further along the $\mathbf{X}$ when the motion is happy then when it is unhappy. Figure 5.5 shows the actor walking at different parameter values.

Remember that verbs alone do not "remember" from a time τ^{now} to τ^{next}. Evaluating a verb at a given time is completely separate from evaluating it at another time. Verbs are simply templates detailing how a particular action is performed at different parameter settings. Therefore, if we vary the parameter setting, we can achieve strange looking animations. Figure 5.6 shows a quadratic ramp up in *happiness* for the aforementioned verb which achieves very high velocities in the resulting motion. Foot support phases break up. Ramping down the motion could cause the character to go backwards if the forward motion resulting from the advancing T were more than offset by the decreasing happiness parameter. Verbs alone, therefore, cannot achieve coherent animations in the presence of changing parameters.

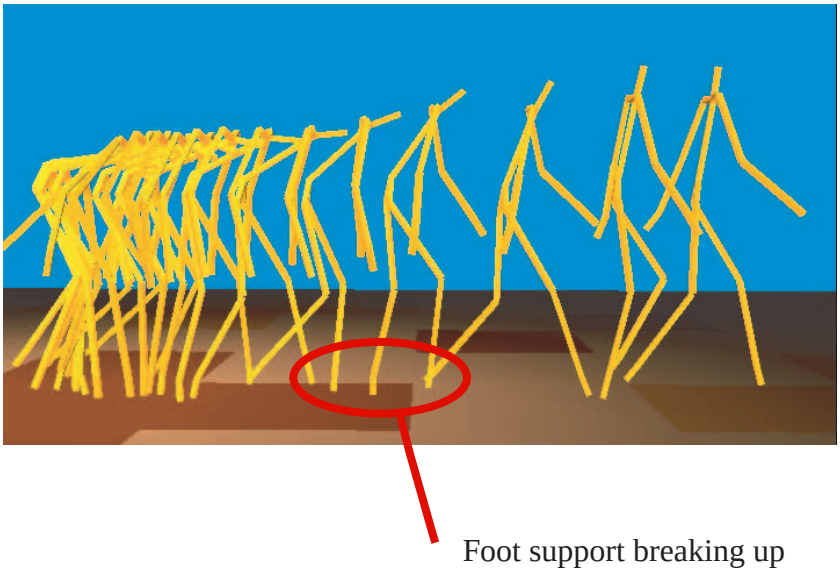

Figure 5.6: Velocity boost

Calculating the root info for τ^{now} and τ^{next} yields deltas, thus ensuring no velocity boosts or backtracking unless, of course, properly dictated by the verb at its current parameterization.

If the τ^{now} is outside the current verb, calculating this delta is difficult since the values for the verbs are only intra-verb consistent. In this case, a neutral $\hat{\tau}$ is chosen at the end of the current verb to serve as a bridge between the two verbs. Root calculations are performed from τ^{now} to $\hat{\tau}$ and then from $\hat{\tau}$ to τ^{next}. If τ^{now} and τ^{next} are separated by more than 1 item on the queue, then the process is iterated. The algorithm which deals with this is found in Algorithm 5.3.

Algorithm 5.3 Updated verb-graph algorithm

```
UpdateVerbGraph (τ^next ,S)
    τ̂ = S.τ^in + T_0(S.t^out) − T_0(S.t^in)

    while (τ^next ≥ τ̂)
        PositionDOFs (τ̂ ,S)        Algorithm 5.2
        UpdateQueue (S)            Algorithm 5.1
        τ̂ = S.τ^in + T_0(S.t^out) − T_0(S.t^in)

    PositionDOFs (τ^next , S)
```

While the root is being updated, the constraint set must also be kept consistent. If a constraint is on a body part, such as the left-foot, for example, the system needs to know when the foot goes from one constraint into another. If the frame rate is too low, constraints can be held for far too long as periods of unconstraint may be ignored. In a walk cycle, the foot support could be kept one support phase too long. Instead, the system needs to keep track of which constraints go in and out of existence.

5.3 Restrictions on the verb graph state

The goal of a program using the verb graph, aside from constructing animations, is to keep the graph in a *continuable* state. Being continuable means that that the animation has something left to do. The Q_0 verb should always leave through a transition in the Q_1 spot. In order for a verb-graph to be considered continuable, there must always be an *achievable* transition leading out of Q_0. At a given moment, $\tau^{\text{now}} - \tau^{\text{in}}$ indicates how much time has been spent in Q_0. If t^{now} were the current clock time τ^{now} projected into the canonical timeframe of Q_0, the achievable transitions leading from Q_0 are all those with

a transition start time, $t_0^s \geq t^{\text{now}}$ where t_0^s the time when the transition begins to leave the verb in spot Q_0 expressed in Q_0's canonical timeframe.

Typically, when a verb Q_0 comes to the front of the queue, an achievable transition and verb are added to the queue if Q_0 was the last item on the queue. The transition chosen is a weighted random choice based upon relative weights assigned to the transitions by the designer. It is assumed that the designer never designs a transition which enters into a verb beyond the last exit out of the verb. This kind of pathological condition can be easily detected at verb graph design time.

The item at the front of the queue will play until a transition leaving it is reached. At any time, an achievable transition may be placed in the Q_1 spot followed by its exit verb in Q_2. The Q_0 verb cannot be halted and control given to another verb without going through this transition mechanism. To do so would break the continuity of the overall animation. The designer is tasked with designing the transitions and the graph in such a way that interactivity is not lost due to long uncontrollable sequences of canned animation.

A verb with adverb settings $\mathbf{p}$ will be very similar to one with adverb settings $\mathbf{p}'$ where $\mathbf{p}$ and $\mathbf{p}'$ differ by a small epsilon ϵ. This is guaranteed by the continuity of the radial basis function approximation used to construct the verbs. As ϵ grows, however, continuity is not assured. As τ^{now} moves forward, therefore, $\mathbf{p}$ must not be allowed to change drastically lest the animation loose its continuity. For this reason, $\mathbf{p}$ is constrained to move at most $\epsilon \cdot dt$ for an ϵ chosen to reflect the magnitude of the adverb parameterization.

5.4 Non-standard graph layouts

Most of the verb graphs studied here have no articulation points, i.e. no transition which, when removed, would result in a disconnected graph. The

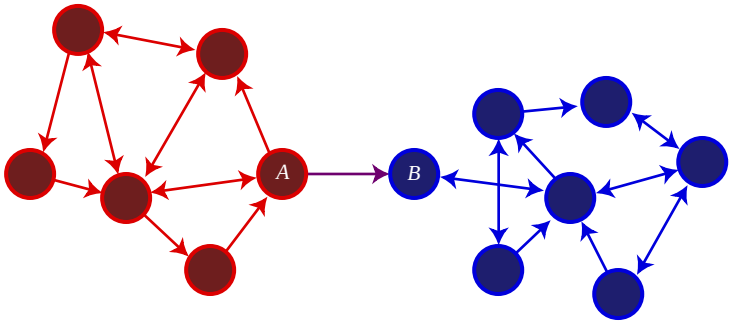

Figure 5.7: A one-way graph

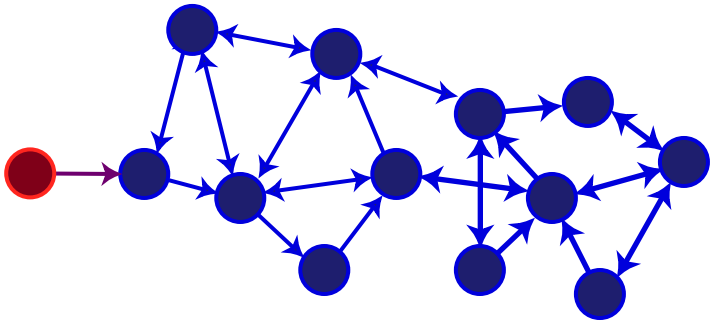

Figure 5.8: A graph with a special start sequence

implication of this is that all verbs are reachable from all other verbs by some path of transitions. This property is not a requirement, however, just a useful form of graph.

A graph which contains one articulation point connects two separate sets of verbs. From one set, all verbs are reachable. From the other set, only verbs within that set are reachable. Verb graphs of this form will be called *one-way* due to the one-way nature of the articulating transition. A graph of this form is shown in Figure 5.7.

Clearly, once the purple transition is taken from the **A** verb to the **B** verb, the queue can never again contain a verb from the red set. Any attempt to search the graph for a path from a verb in the blue set to one in the red set will fail. A graph like this could prove useful, however, as a way to separate motion

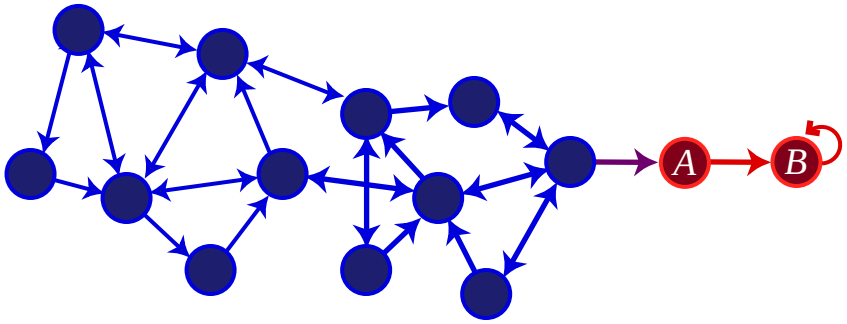

Figure 5.9: A sub-graph containing a special death

into distinct modes or for playing a special start-up motion which can never be revisited, as shown in Figure 5.8. This is a common video game metaphor. Similarly, the character's demise could be handled by a one-way graph like the one shown in Figure 5.9. The red verbs are the death sequence, with *A* being the actual death and *B* the lying around dead loop, as in *Id*'s genre defining games *Doom*(TM) and *Quake*(TM). Capturing these kinds of common game elements is essential for a system from which a game could be built.

5.5 Transitioning

Previously, the concatenation motion object was described (Section 3.9). One of the primary problems with this object was its inability to smooth out the transitions from one motion to another. That is the role of transitioning. This topic has been dealt with by Rose, Perlin, and others [124] [113]. The first type of transition which will be described is the simple blending kind of transition used in Perlin's work. An extension for the root DOFs allows this transition to be of greater use. Quaternions [129], have proven useful for transitioning and are used here for non-root DOFs.

A special case transitioning mechanism, torque-minimal transitioning, will be described with its related topics in Appendix B. Not real time, this tech-

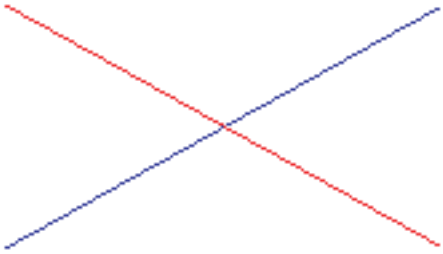

Figure 5.10: Linear blending function, $\alpha(t) = t$

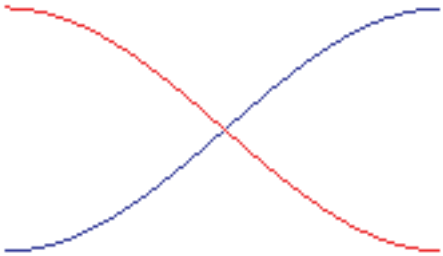

Figure 5.11: A sigmoid blending function, $\alpha(t) = \frac{\cos(-\pi+t\pi)+1}{2}$

nique has the added restriction of working only with static motions, making it inappropriate for parameterized verbs.

Simple blending

Perlin [112] [113] describes a technique for blending between motions using blending functions to achieve a weighted average of the two motions. A blending function, α, is a non-decreasing function, possibly discontinuous, with domain and range of [0..1]. Two plausible candidate functions, both used in the work here, are shown in Figures 5.10 and 5.11.

Transitioning can be thought of as another type of functional relationship, this one between a motion being transitioned from and one being transitioned to. These, of course, can be the same motion, which allows the transitioning object to be of use for constructing motion cycles. The transition motion, $\mathbf{M}_i$,

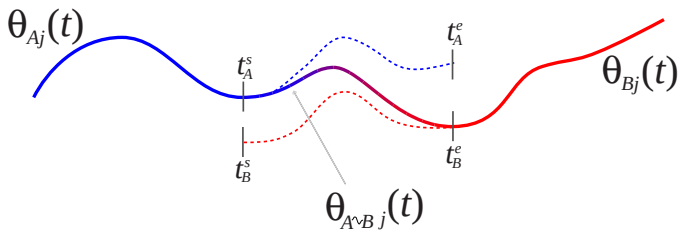

Figure 5.12: The transition

between two motions $\mathbf{M}_A$ and $\mathbf{M}_B$ is defined as:

Definition 10 *transition motion* $\mathbf{M}_i = \{\alpha, \mathbf{M}_\mathrm{A}, t^s_A, t^e_A, \mathbf{M}_\mathrm{B}, t^s_B, t^e_B\}$

where α is the blending function as defined previously, $\mathbf{M}_\mathrm{A}$ and $\mathbf{M}_\mathrm{B}$, the motions being blended from and to respectively, and $[t^s_A \ldots t^e_A]$ and $[t^s_B \ldots t^e_B]$ the blending regions in the two motions expressed in canonical time. Figure 5.12 shows the transition diagrammatically. As τ progresses through motion **A** through the transition $\mathbf{A} \rightsquigarrow \mathbf{B}$ and into **B**, a smooth blend of the two motions will occur. If the designer placed the transition plausibly, the viewer will not be able to tell that the motion was not originally designed as a single unit.

The duration of the "from" region is calculated as $T^e_A - T^s_A$ and the "to" region $T^e_B - T^s_B$ where $T^s_A = T_A\,(t^s_A)$ (the canonical-time to verb-time projection for motion $\mathbf{M}_A$) and similarly for T^e_A, T^s_B, and T^e_B. In general, these durations should be chosen to be roughly equal, but that is not a requirement. The duration, T^d_i, of the transition $\mathbf{M}_i$, therefore, is

$$\mathrm{T}^d_i = \frac{(T^e_A - T^s_A) + (T^e_B - T^s_B)}{2}$$

Transitions are defined to begin at $T = 0$, so the time mapping functions for the transition, are:

$$\begin{aligned} T_i(t) &= t \cdot \mathrm{T}^d_i \\ t_i(T) &= \frac{T}{\mathrm{T}^d_i} \end{aligned}$$

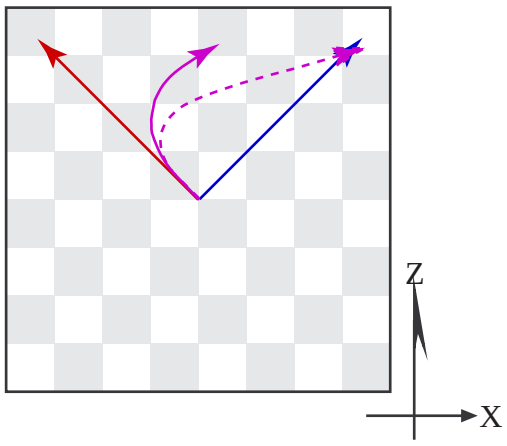

Figure 5.13: A bad root transition

with T_i^d the duration of the transition as defined above. The constraint set function, $\mathcal{I}_i$, will be handled using an IK constraint arbitration scheme to be described in Section 5.5. The position function, θ_{ij}, is is a weighted sum of the position functions of the from and to motions,

$$\theta_{ij}(T) = (1 - \alpha(t))\theta_{Aj}\,(T + T_A^s) + \alpha(t)\theta_{Bj}\,(T + T_B^s) \tag{5.4}$$

where $t = t_i(T)$ and T_A^s and T_B^s as defined previously.

Blending with root integration

Equation 5.4 works well for non-root DOFs. If the same method used for transitioning the non-root DOFs were applied to the root, however, some potentially undesirable results can occur, especially on turning motions when the character is in motion. Imagine the two motions, **A** and **B** shown in Figure 5.13 in red and blue respectively. These motions would blend to lead the actor to the end point in motion **B**. This path is shown by the dashed purple line. The overall accelerations would be increased, however, as the transitioned motions would swing left and then right. A better transition would be one which blended the velocities of the two constituent motions, as is shown by the solid

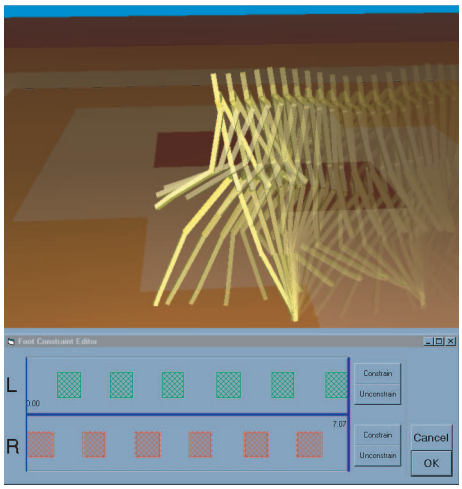

Figure 5.14: Walk with foot support constraints indicated

purple line. This blend, when integrated, yields better position values than a simple blend. For dynamic motions like this example, integration yields better results. Position, θ_{ij}, therefore, is achieved by

$$\theta_{ij}(T) = \begin{cases} (1-\alpha(t))\theta_{Aj}\left(T+T_A^s\right) + \alpha(t)\theta_{Bj}\left(T+T_B^s\right) \\ \qquad \text{for non-root DOFs } i \\ \int_{t'=0}^{t}(1-\alpha(t'))\dot{\theta}_{Aj}\left(T'+T_A^s\right) + \alpha(t')\dot{\theta}_{Bj}\left(T'+T_B^s\right) \\ \qquad \text{otherwise} \end{cases}$$

where $t = t_i(T)$, $T' = T_i(t')$, and T_A^s and T_B^s as defined previously.

Inverse kinematic arbitration and enforcement

An animation segment has information about its kinematic constraints, as were detailed by the designer or derived automatically as described in Section 4.4. An example of an annotated track, for a walking motion, is shown Figure 5.14.

As an animation plays through this walk segment, the constraints will be enforced at the appropriate times. As an animation transitions from one motion to another using the transitioning mechanism, however, the constraints need to be arbitrated to yield a reasonable constraint regimen.

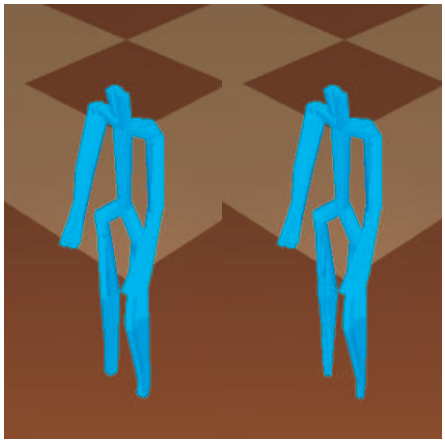

Figure 5.15: A milling about transition

The kinematic constraint arbitration mechanism used in this dissertation is a simple one which works reasonably well in practice: constraints at both ends of the transition are held throughout. Otherwise, constraints in the *from* motion $\mathbf{M}_A$ are enforced in the first half of the transition and constraints in the *to* motion $\mathbf{M}_B$ enforced in the last half. Constraints are eased out when a constraint period ends. Likewise, constraints can be eased in. Constraint ease-out keeps the motion "pop" free. Constraints are enforced using the linear technique described in Section 2.3 weighted by an ease-out period.

Figure 5.15 shows a motion generated for a character as he transitions from one milling-about motion to another. Note the slight kinematic constraint violations of the feet as the transition occurs shown in the left multiple-exposure picture in the figure. The right picture shows the IK enforced result. While the foot slide is not large, it can be jarring. Constraints can be used to avoid small but noticeable indicators of error.

One caveat to this IK arbitration scheme is that body parts need to be unconstrained every so often. Imagine continuing the milling about animation. Each motion clip is going to move the root a little bit, unless it was specifically designed not to. Assuming it does and assuming the feet are never released,

the character can end up being far from where its feet are constrained after a lone period of time. Allowing one foot to shift while the other supports is an effective and plausible way to deal with this and is accomplished by relaxing one foot constraint occasionally.

5.6 Gesturing

A *gesture* is defined as a motion that uses a subset of the DOFs and which does not require the motion of the root. In terms of the composibility rules introduced in Section 3.9, the root motion can be either **defined** or **undefined** but not **required**. The remaining DOFs can be **required**, **defined**, or **undefined** as appropriate for the gesture.

Being able to compose a gesture asynchronously upon the current primary action is a powerful notion. It can be used to extend the usefulness of verbs created for the system and can cut down on the number or complexity of verbs needed. Instead of needing verbs for walk-with-wave, walk-with-head scratch, and walk-plainly, one good walk verb plus gestures for wave and head-scratch are needed. As all the walking variants are more complicated to create than the gestures, this is already a win. As the waving gesture would be useful with other base verbs, such as stand-around or sit-down, the total number of verbs potentially required can be considerably smaller.

Assuming no DOF-requirement incompatibility problems, layering a gesture atop another verb requires a smooth transition into the gesture verb and a smooth transition from it back to the primary action when the gesture is complete. The same transition mechanism used in Section 5.5 is used here. Figure 5.16 shows a timeline indicating the weight of the primary verb vs. gesture verb during the gesture. The "primary" verb in this figure indicates all the actions which take place in the primary verb queue during the duration

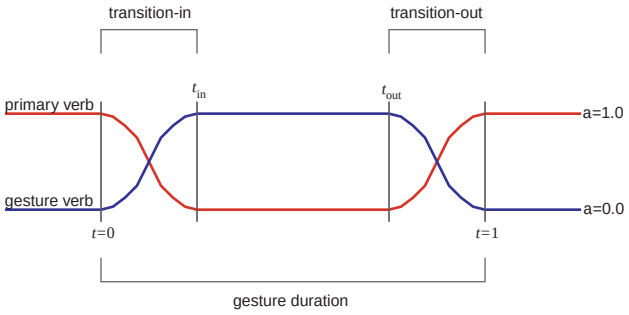

Figure 5.16: Primary vs. gesture weight during a gesture

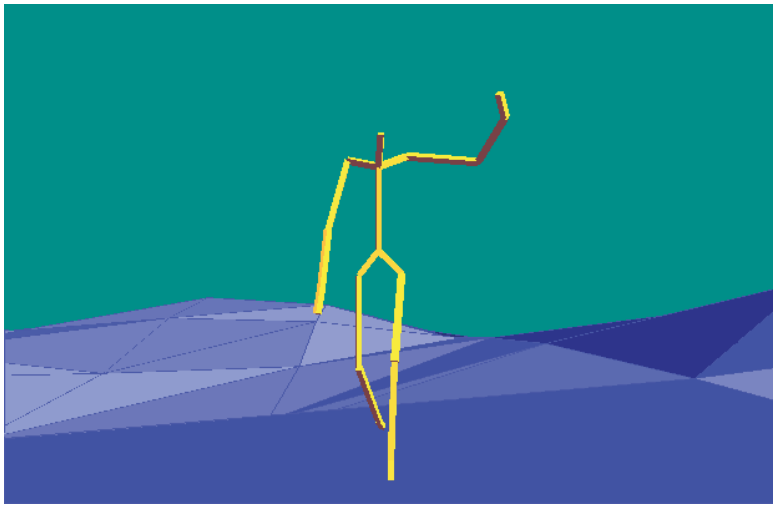

Figure 5.17: A walk verb with the wave gesture overlaid atop it

of the gesture. A walk/wave verb/gesture result is shown in Figure 5.17.

Note that the gestures need not be simple non-parameterized verbs, but can be complete verbs constructed using the *Verbs & Adverbs* mechanism or another such as shown in [26].

Algorithm 5.4 Simple gesture positioning

```
PositionGesture (G_active, τ^now, τ^s_G)
{
    T^active = τ^now − τ^s_G
    if T^active > T^d_G then
        mark the gesture verb G_active no longer active
    else
    {
        t^active = t_G (T^active)

        if (0 ≤ t^active ≤ t^in) then
            α = U( t^in − t^active / t^in )
        else if (t^out ≤ t^active ≤ 1) then
            α = U( 1 − (1−t^active)/(1−t^out) )
        else
            α = 0

        for all defined or required joints j for gesture G_active
          Θ_j = α · Θ_j + (1 − α) θ_Gj(T^active)
    }
}
```

Algorithm 5.4 shows the steps needed to layer a gesture atop an already evaluated primary verb. G_{active} is the gesture, τ^s_G the animation time the gesture began and τ^{now} the current animation time for the verb-graph. T^{active} and t^{active} is the time spent in the gesture in verb and canonical time. U is the blending function, typically sigmoid, and α the resulting blending factor. t^{in} and t^{out} are shown in Figure 5.16.

Multigesture support

One gesture, not surprisingly, is often insufficient. As was detailed in Section 6.1, verbs can be used to control things like arm position, mood, or as

seen previously, waving. Instead of a single active gesture, therefore, a set of active gestures is maintained. The only requirement is that they do not interfere with one another vis-á-vis the composibility rules from Section 3.9.

Algorithm 5.5 Multigesture positioning algorithm

```
PositionGestures (τ^now)
{
    for all active gestures G_i which started at τ_i^s
        PositionGesture (G_i, τ^now, τ_i^s)
}
```

In order for Algorithm 5.5 to function properly, the gestures must be checked for compatibility. This can be done by ensuring that no two verbs have a **required** DOF in common. A first-come-first-served approach can be taken when dealing with conflicts. If any part of the new gesture conflicts with an old gesture, however, the entire new gesture should be discarded since verbs and gestures will often do unpredictable things if DOFs are simply removed.

Incompatibility with the primary action

A walking verb typically has the following DOF requirements: the root and legs are **required**, the main body **defined**, and the face, if present in the skeleton, **undefined**. Laying a wave gesture atop this, for instance, is a reasonable thing to do. The functioning of the basic motion will not be compromised with this added motion. If the character were to walk up to cliff-face and begin climbing, however, it would need to stop waving, even if the wave were only "half-finished".

Such incompatibilities are determined using the composition rules for the front verb in the primary verb queue. If the DOF requirements for that verb

change (or if the verb transitions to a new verb with new DOF requirements) which are incompatible with an active gesture, then that gesture is marked and a transition from it is begun, even if the normal gesture t_{out} has not been reached.

5.7 Motion snippets are verbs too

In earlier chapters, the term "verb" has been reserved for rich parameterized entities, capable of directing the motion of a synthetic actor in a variety of ways. Examples included the *walk*, *run*, and *reach*. Many verbs which would be desirable to have in the verb graph, however, do not fit into this form. A celebrating motion like the one shown in Figure 5.18 is the kind of motion which would be difficult to design in enough different, but similar ways to yield a sampled-enough space for verb construction. Verbs like this are quite useful when designing a system like a game and should be played as is. Including the entire motion hierarchy in a greater hierarchy of verbs yields a consistent treatment of examples and verbs and is shown in Figure 5.19. This hierarchy deals consistently with all of the motion types introduced for this work. As basic motions, functional motions (like a clip motion), transitions, and verbs must all support the motion formalism, this leads naturally into an object-oriented class hierarchy, which is how all the different kinds of motions were implemented for the *Verbs & Adverbs* system.

An obvious extension to this idea is to make a place at the table for other kinds of verbs, like those introduced by Perlin and Goldberg in the Improv system [112] [113]. As processing power increases, physically simulated systems [68] [132] can be dealt with similarly, leading to a unified real-time animation system that makes use of the best aspects of each method.

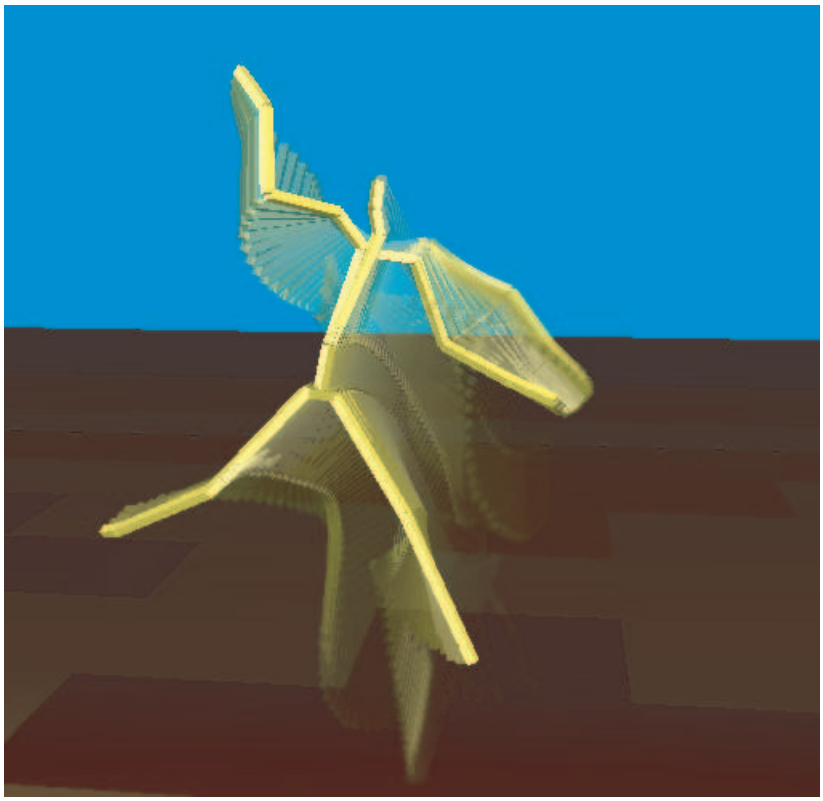

Figure 5.18: Celebrate good times come home

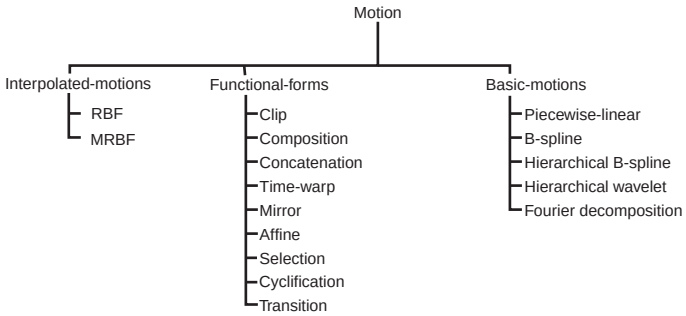

Figure 5.19: The completed hierarchy

5.8 Conclusions

This chapter introduced the verb-graph, an object used to structure the overall flow of an animation. A designer, not the system, determines the logical places for transitions to occur and designs those transitions using a transition mechanism introduced in this chapter. Gestures can extend the usefulness of the verbs and are used for on-the-fly compositing of small motions like waving. The verb-graph, not the verbs themselves, is the object controlled by the high-level system such as Perlin or Blumberg's. Chapter 6 describes a demo application and verb graph used to test the system.

Chapter 6

Results & user study

Chapter 3 introduced examples, time, and the motion formalism. An editing system, built from the formalism, helps a designer transform motions into examples. Chapter 4 described both the single- and multi-resolution versions of the radial B-spline interpolation used in the *Verbs & Adverbs* system. It also described a number of issues related to verbs. Finally, Chapter 5 introduced the verb-graph and transitioning mechanisms. Using these concepts, long, seamless animations can be constructed from verbs as well as from simple, non-parameterized motions. This chapter presents results generated using the methods developed in the previous three chapters. It will also present analyses of the effectiveness of the system through measurement and user-studies.

6.1 Verbs

A library of motion capture was used to construct a number of different parameterized verbs: walking, jogging, reaching, and idling. Some verbs, such as "walk", have a large number examples representing different emotions such as happy, sad, angry, clueless, tired, delirious, determined, frenzied, ashamed, bored, goofy, and grief-stricken, as well as walks at different inclinations and radius of turn.

The data for the example motions discussed here was motion captured with an *Ascension MotionStar*(TM) system sampled at 120 Hz. with 15 six degree-of-freedom sensors positioned on the body. The raw data was preprocessed to fit the rigid body model with hierarchical joint rotations and fewer DOFs that correspond to the limitations of human joints. The methods described by Bodenheimer *et al.* [21] ensure that the motion capture data makes consistent use of joint angles for redundant DOFs, one of the example restrictions introduced in Section 3.5. The final model has 40 joint DOFs in addition to six DOFs at the root, located between the hips, for global positioning and orientation.

Walk

A parameterized walk was the first goal of the *Verbs & Adverbs* project. The data used for this walk was motion captured from a physical actor, named Christian. A pair of stills from this motion capture session is shown in Figure 6.1 showing the depth of expression he was able to produce with his posture. The walk verb has four adverbs: happiness, knowledge, inclination (slope), and turning. A sampling of this walk at a particular key-time along the two emotional axes is shown in Figure 6.2. A sampling of this walk at different turning radii and levels of happiness is shown in Figure 6.3. In each of these figures, and in the figures to follow, the green figures were examples and the yellows some interpolated and extrapolated motions. While only few synthesized motions are shown, the reader should note that there is an infinite variety of motions between the examples.

Figure 6.1: Christian walking

Reach

A reaching verb was parameterized by the adverbs representing the x, y, and z offsets of the reach goal from the starting position of the reach. It used 18 examples and has good coverage inside the convex hull of the examples. The reaching adverbs can be extrapolated 10% beyond the convex hull before the resulting motion deteriorates unacceptably. A sampling at the reach apex key-time along x and y axes is shown in Figure 6.4.

Even given an accurate parameterization of the verb's examples, it is still not a guarantee that a linear change along the adverbs will produce a linear change in the resulting motion. Take the reach verb again for an illustration. This undesirable situation is due to the fact that both articulated figures and radial basis approximation are non-linear processes. In general, the desired location of a reach and the actual location will only match exactly at the examples. Figure 6.5 shows a sampling of the errors in the bounding box of the example reach points (blue spheres). The green and red spheres indicate the desired and actual location of a reach for a particular adverb setting **p**. Green lines are small errors which blend to red lines for the worst errors. Note the lower right corner of the figure. The errors near the blue sphere (an

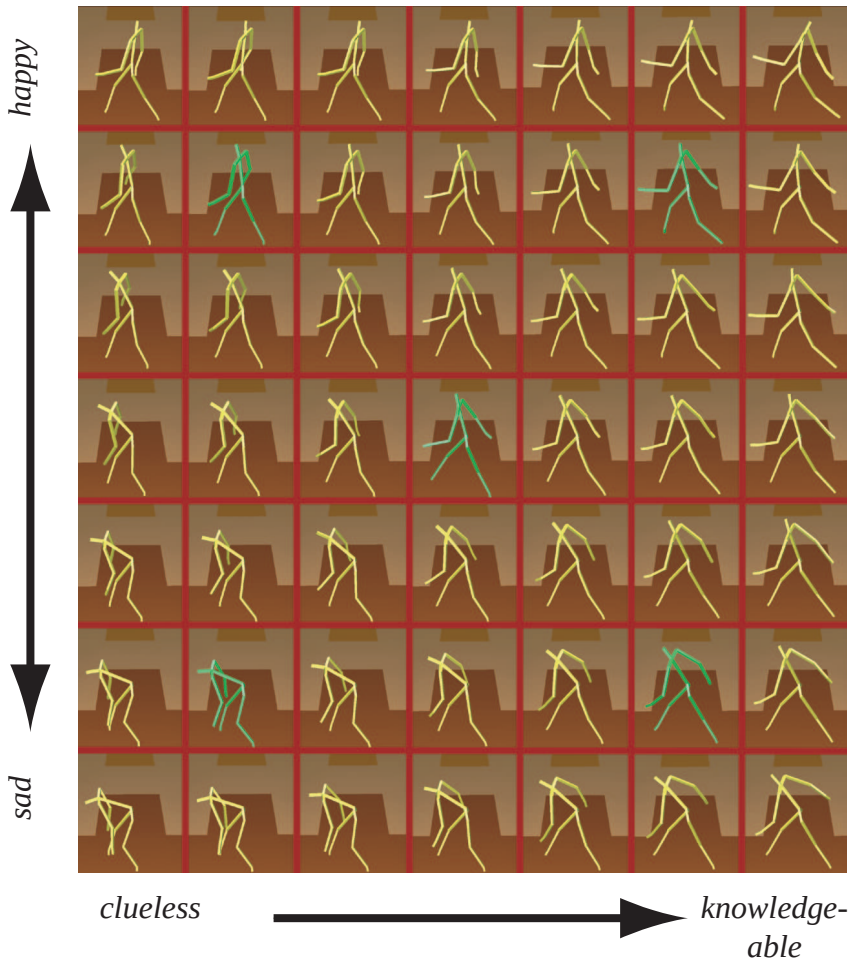

Figure 6.2: A walk sampled across two emotional axes. The green figures are the example motions. The rest are created through the verb/adverb mechanism.

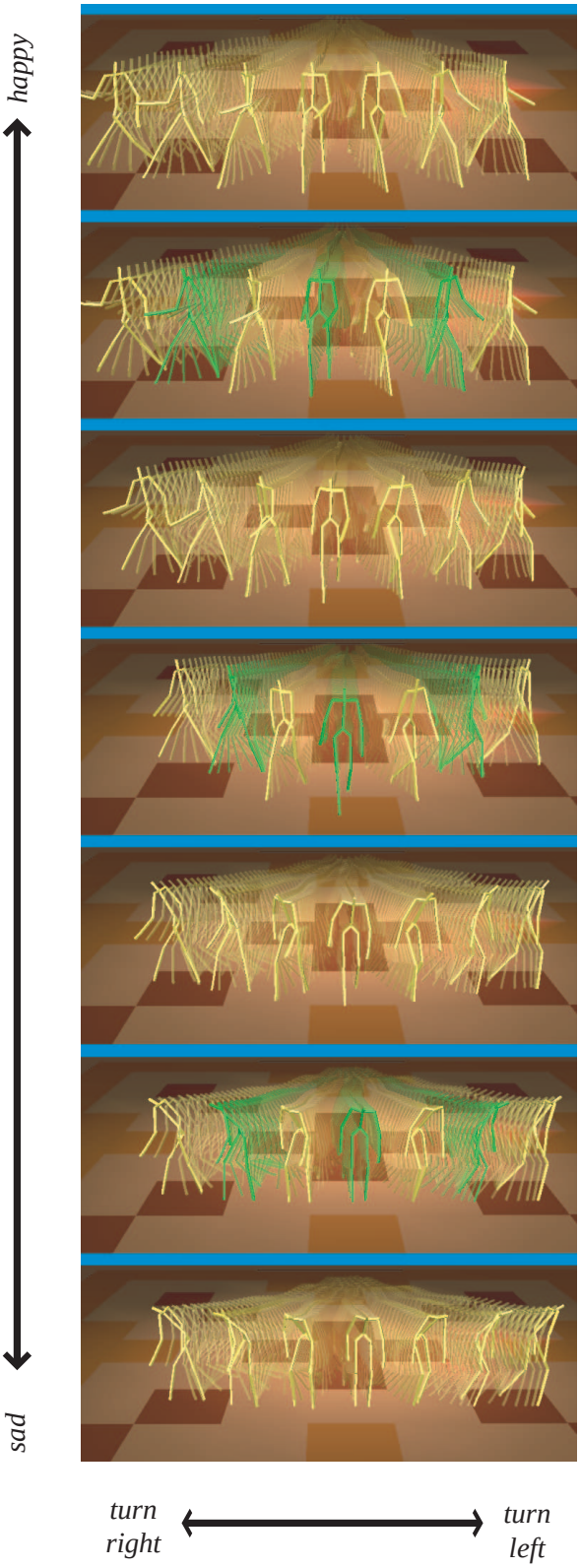

Figure 6.3: Emotive turns

Figure 6.4: A reach sampled along the x and y axes

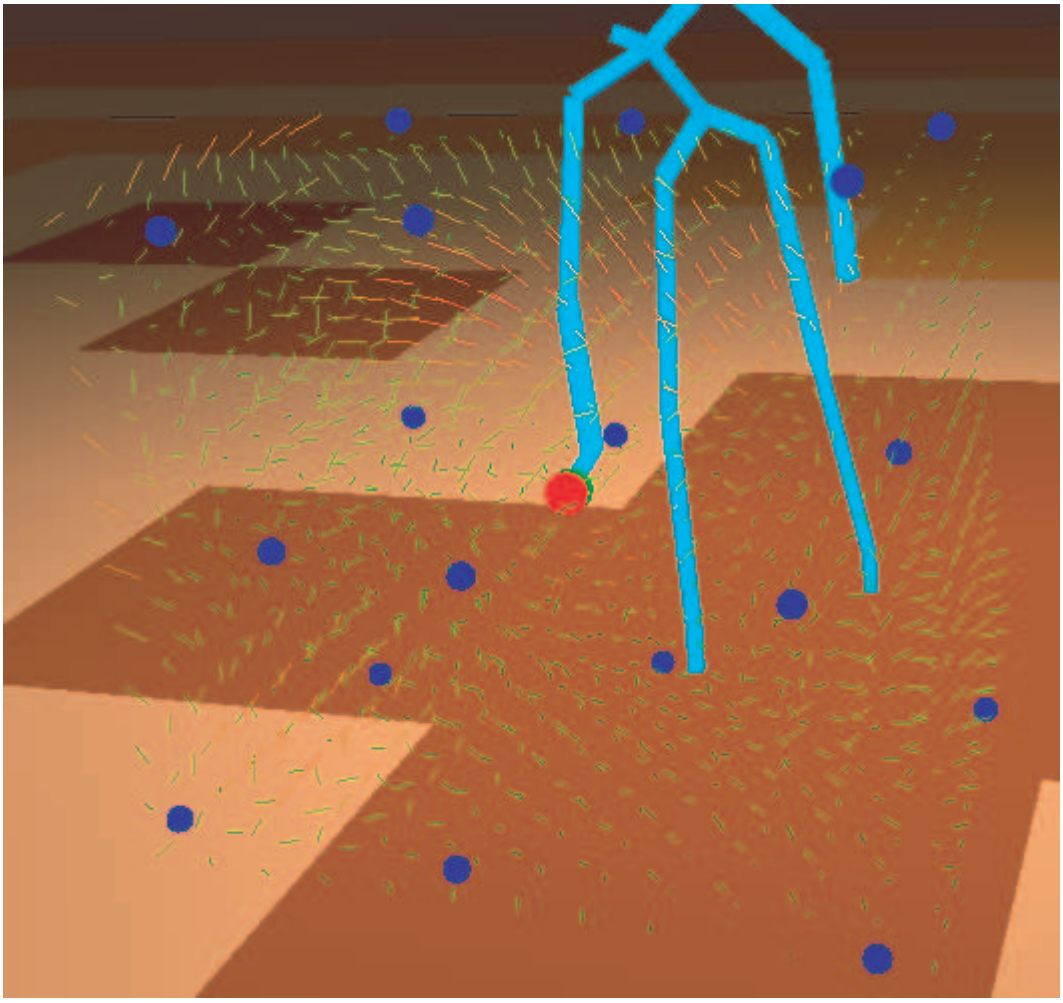

Figure 6.5: A sampling of reach errors

example) are small. The minimum sampled error (none of the examples were sampled as they appeared off the sample grid), was 0.00311, or a 0.212% error. Maximum error was 0.13576, or 9.68%. Average error was 0.0453, or 3.12%.

There are a number of ways to solve this problem, assuming an average 3.12% error is unworkable. The simplest solution is to use IK as we did earlier for foot support constraints. As these errors are small, a simple IK solution will converge very quickly to a solution, possibly in 1 iteration. Another way is to use more example reaches. Errors near the examples are lower, so blanketing the space with more examples will yield an overall improvement in parame-

Figure 6.6: Comparison of raw versus reparameterized reaching. The spike closer to zero for the reparameterized verb indicates lower overall error.

terization. As evaluation time grows linearly with number of examples, this may not be a workable solution if the number of examples required to achieve a desired error tolerance was high. Also, examples are precious and may not be available in abundance. Another solution is to reparameterize the adverb space in order to achieve a more linear solution. While this is still an active area in our research program, I'll detail a simple non-general solution for this 3-D verb. It should be stressed that it is not a general n-D solution and is exponentially costly in terms of storage.

The error sampling used in Figure 6.5 is a regular lattice of error vectors. Any given set of adverbs, $\mathbf{p}$, within the bounding box of the samples will fall within one of the boxes in this lattice. Blending the errors of this box with multi-target interpolation yields an interpolated error vector, $\mathbf{v}$, so $\mathbf{p} - \mathbf{v}$ will project to a place in the parameterization will likely lower the error of the desired parameterization. We performed this test which yielded the following minimum, average, and maximum error percentages: 0.015%, 0.61%, and 2.8% respectively, a marked improvement over the non-reparameterized reach. A histogram showing the distribution of errors for the raw parameterization and the reparameterized approach is shown in Figure 6.6. Note that the reparameterized motion has many more error samples near zero with a much lower frequency of high errors.

Linearizing emotional adverbs is much more difficult due to the subjective nature of the adverbs. A skilled designer, however, will do this as part of the refinement loop introduced in Section 4.5 using the improved mechanism from Section 4.6.

Jog

A jog was constructed from two examples, a run straight ahead and a run to the right. It is an instructive instance that shows the power of the editing system built on top of the motion formalism from Chapter 3. Jogs are difficult to motion capture, especially with a tethered system like the *MotionStar*$^{(TM)}$.

The goal for this verb was a jog that could run to the left, right, straight ahead, or anywhere in-between. The two examples captured from the motion capture system were a neutral run forward and a run to the right. A simple mirror (Definition 5) operation yielded a run to the left. The key-times, however, then become wrong as the mirror operation changes the usage of the feet. A run beginning on the left foot becomes one from the right foot once mirrored. As the overall length of the jog sample was short, a simple clip (Definition 2) was insufficient. A cyclification step (Section 3.10) on the mirrored walk, followed by a concatenation (Definition 7) and a clip yielded the third example. The relationships among the examples is shown in Figure 6.7. Results of the jog verb are shown in Figures 6.8 and 6.9.

Milling about

A set of three controlled idles was constructed. A sampling of one is shown in Figure 6.10. While some emotional content is covered in pose (note the difference in the back), these verbs depend most heavily on phrasing.

6.2 Verb-graphs

A simple application was generated to test the verb and verb graph mechanisms. Figure 6.11 shows the system. The upper left shows a simple control system for indicating mood (face), direction (slider), and command over pri-

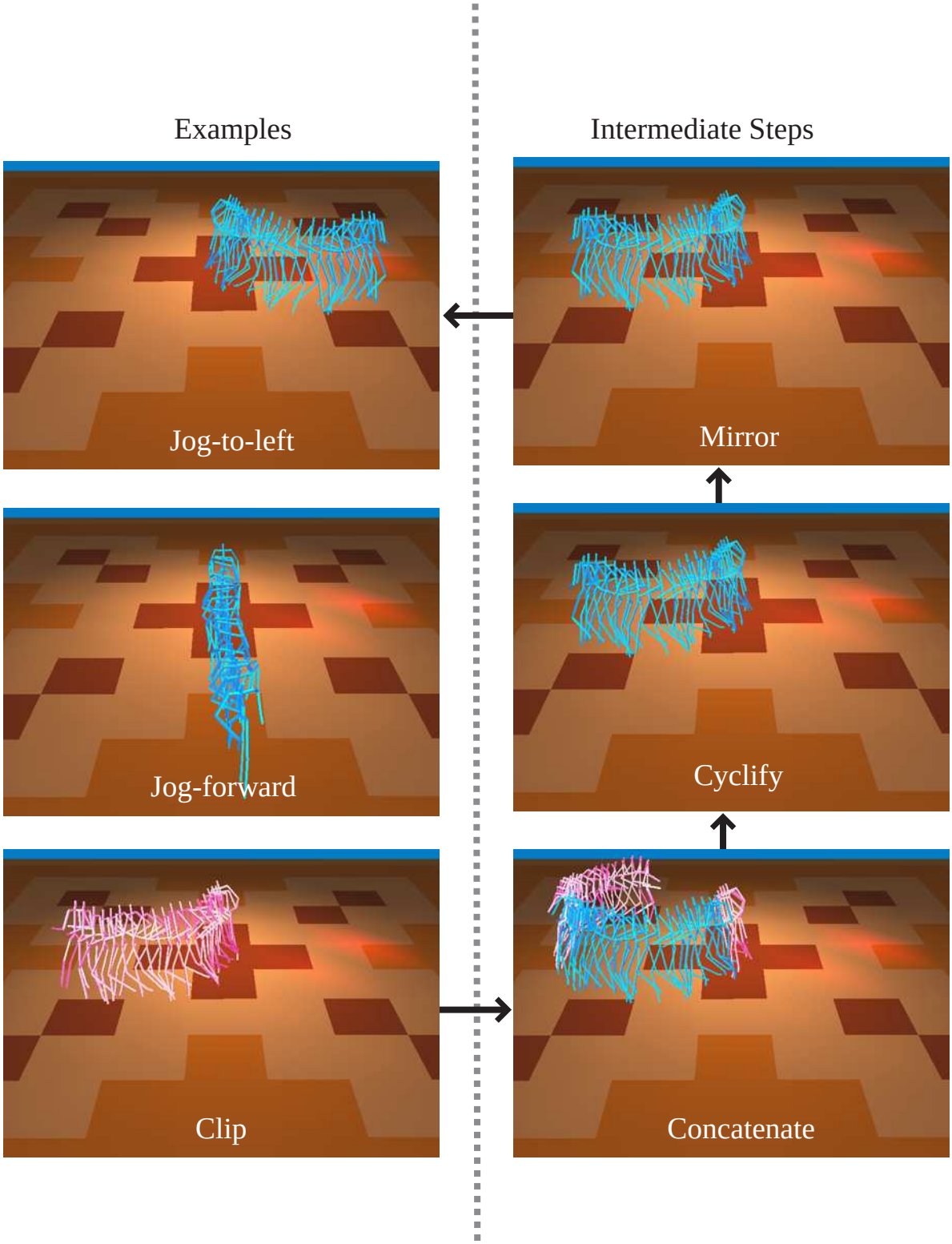

Figure 6.7: Three examples from two basis motions

Figure 6.8: A jogging verb

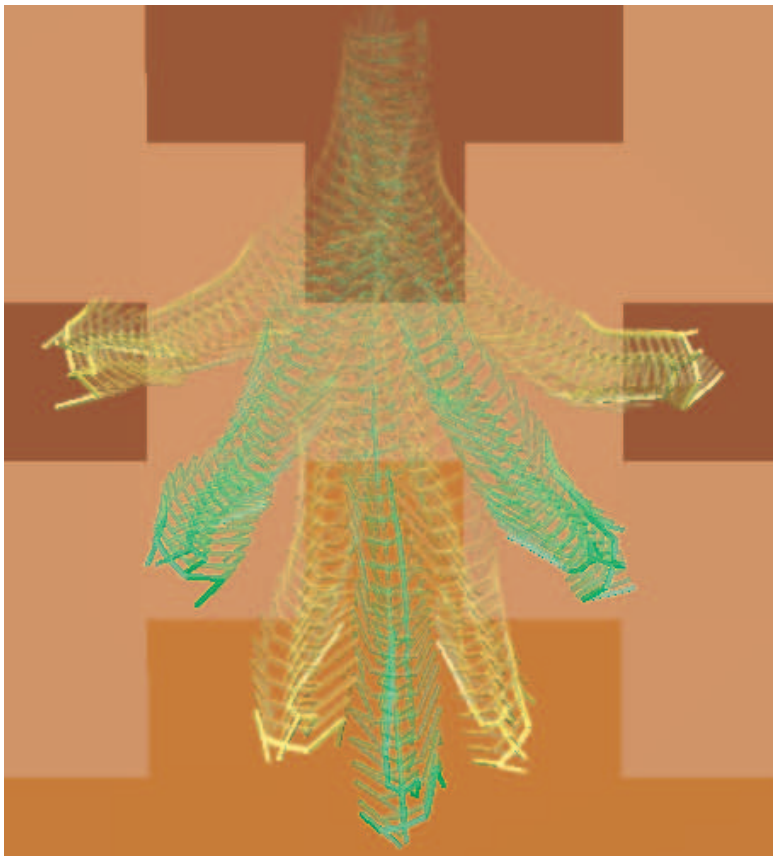

Figure 6.9: The jogging verb from overhead

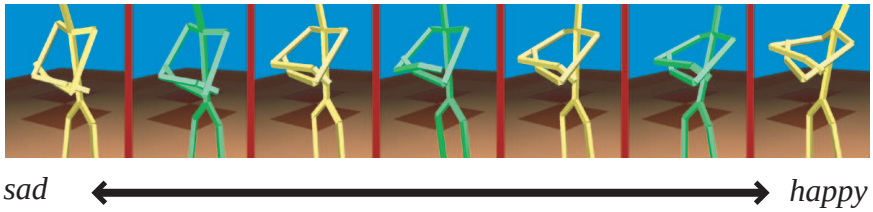

Figure 6.10: A sampling of an idle motion

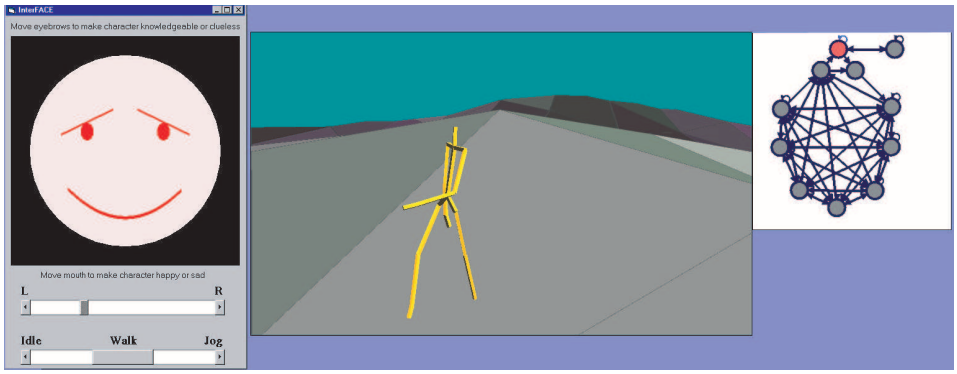

Figure 6.11: Demo application

mary and gesture verbs (buttons). The middle is the view upon the virtual world in which the character is placed. Far right shows the verb graph and its current state. The currently executing verb is in red.

A discrete event simulator serves as the runtime system's main loop. This system tracks the clock and sequentially processes events placed on its event queue. Each event has a time stamp and an associated callback function. The system inserts events in the event queue (in time stamp order) and processes them by invoking the callback function with the time stamp as a parameter. Events may be one of three types: normal, sync, or optional. The system processes normal events as they are reached in the queue, independent of the relative values of the time stamp and the clock. Sync events wait for the clock to catch up to the time stamp if the clock time is less than the time stamp; they execute immediately otherwise. The system skips optional events if the

clock has passed the time stamp; otherwise, optional events act like normal events.

The most common events are "render" and "display" events. A render event evaluates the DOFs at the time indicated by the time stamp to set the creature's pose, then renders (but does not display) an image. The render event has the "normal" flag and thus creates an image as soon as the event reaches the fore of the queue. A display event with the same time stamp but with a sync flag waits for the clock to reach the time stamp and then displays the rendered image. The render event also measures the amount of clock time between frames and estimates the best time stamp for the next render or display events and inserts them into the event queue. This way, the frame rate dynamically adjusts to the computational load.

The verb graph used for the demo system is shown in Figure 6.12. Gray nodes are simple verbs. Blue nodes are parameterized ones and are annotated by their axes in italic and the number of constituent example motions. Transition likelihoods separate the graph into different regions indicated by the dashed lines. The verb graph will never automatically shift from a jog to a walk or idle unless told to do so by a higher level entity such as the runtime system or user due to these transition likelihoods, i.e. transitions crossing the boundaries are given a likelihood of zero, so they are never automatically chosen. The system randomly perturbs the parameters and verb queue if left untouched for too long.

This application allows the character to stand around, navigate around the world through walking or jogging, and interact with the world through reaching. The user or system can invoke gestures, such as a wave.

The verb-graph and verbs/adverbs mechanisms can be run at high frame rates. On a Pentium-Pro 200MHz machine (a 1996 generation machine), the

raw frame rate is approximately 240 frames per second with inverse-kinematics turned on or 800 fps without. *Verbs & Adverbs* is bound primarily by the rendering performance of the graphics card. As detailed in Chapter 4, the *Verbs & Adverbs* mechanism is space efficient. The number of coefficients needed is within a factor of 2 the total number of coefficients needed to store the example motions. The added storage for verb-graph objects, such as transitions, is negligible.

6.3 User study

Quantifying the effectiveness of a technique which grants aesthetic control is difficult. Often in computer graphics, the "looks good to me, does it look good to you" test is used. In this section, a user study will be introduced which seeks to measure the qualitative aesthetic aspects of the system along with the results of these tests.

As was described in Chapter 4, the system produces a set of verbs which are controllable by a set of aesthetic and structural axes, i.e., the adverbs.

Our test judges the effectiveness of the technique at generating plausible motions. Rather than test directly whether a motion looks plausible, which would be more likely to judge the effectiveness of the animator or motion capture system, the ability of an observer to discern *example* from *interpolated* motion is judged.

This test, a line-up, will be structured as follows. The subject is shown a series of line-ups, such as shown in Figure 6.13, of characters performing similar motions, one of which is an example. The subject is asked to rate the motions on a scale of 1 to 5, using each rating once. They are told that one is motion capture and to rate the motions from 1 (least natural) to 5 (most natural). The motion captured examples are natural looking and will

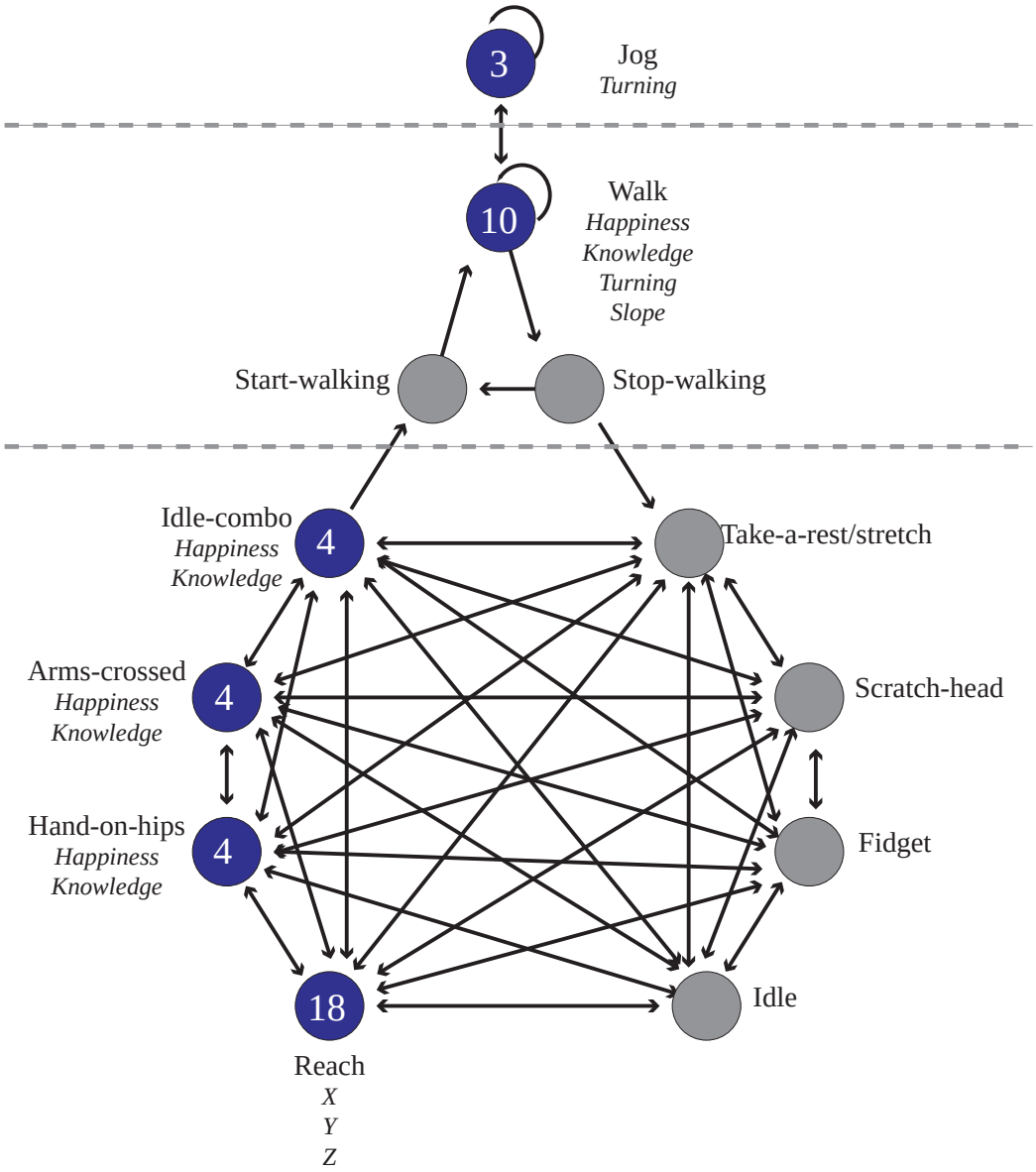

Figure 6.12: The final verb graph for the demo application

Figure 6.13: A line-up

be easily identifiable unless the *Verbs & Adverbs* mechanism does a good job of synthesizing natural looking motions.

Twenty test subjects were asked to rate four line-ups, two walking and two reaching. Four of the five motions in each line up were generated from randomly selected adverb values within an expanded bounding box in adverb space. This box was 20% bigger than the actual bounding box containing all of the examples. This allowed for extrapolations as well as interpolations. Complete success would correspond to a random distribution of ratings for the real motion, i.e. approximately 20% of the examples rated as 1, 2, etc.

User ratings for the examples of the walking verb are shown in Table 6.1. The table shows the values assigned to the examples by the test subjects. If the motion captured ones were completely obvious, then 100% of them would be assigned 5. The table shows that *Verbs & Adverbs* does far better than that. Less than 20% were rated as 5, but 47% were rated either 4 or 5 (the two most natural ratings), showing that the examples are on average slightly better than the synthesized motions. The synthesized walks, however, were convincing. The results for the reaching motion are less compelling and work will need to be done in order to improve it. User ratings are shown in Table 6.2. 42% of the examples were rated as 5 and another 22% rated as 4.

6.4 Conclusions

This chapter presents a synopsis of the results of the *Verbs & Adverbs* project to date. A collection of verbs was shown, with notes on the examples needed to make them. An analysis of reach effectiveness was shown. The verb-graph and demo application used to showcase the system were also described. Finally,

1	25%
2	16.7%
3	11.1%
4	27.8%
5	19.4%

Table 6.1: Distribution of user-study ratings for the real walking verb example. "5" represents what the user perceived as "most natural". If our system were unable to generate convincing motion, all motion-captured examples would receive a "5".

1	8.3%
2	13.9%
3	13.9%
4	22.2%
5	41.7%

Table 6.2: Distribution of user-study ratings for the real reaching verb example. "5" represents what the user perceived as "most natural". If our system were unable to generate convincing motion, all motion-captured examples would receive a "5".

a user-study showing the effectiveness of the technique was performed and analyzed.

Chapter 7

Conclusions & future directions

This dissertation presented a technique for constructing controllable animation segments from sets of similar animation data, called "examples". Each controllable segment performs a specific task, such as running, walking, or reaching with a particular hand. We call these succinct units "verbs". Their control parameters define an infinite space of variation for these verbs; we therefore call them "adverbs". Chapter 3 describes the processing of raw animation data into examples.

Example motions are encoded using curves that are themselves commonly represented using summed basis functions weighted by coefficients. Controllable motions, therefore, can be generated using parameterized coefficients. Chapter 4 describes in detail the multi-resolution radial B-spline interpolation scheme used by *Verbs & Adverbs*. Time-warping example data ensures the interpolation is performed between moments with similar meaning.

Chapter 5 described one way of generating seamless animations of indeterminate length using the short verb segments. A transition scheme that can bridge two verbs leads to a verb-graph, an object which controls the overall flow of a reactive real-time animation.

Finally, results using the system were detailed in Chapter 6. An analysis of a small user study showing the effectiveness of *Verbs & Adverbs* was included.

Additionally, an analysis of the reach verb showed that the system can be used to effectively and efficiently simulate inverse-kinematics on a figure of known topology without the problems commonly associated with IK.

The *Verbs & Adverbs* system is a promising technique for realizing the goal of compelling interactive 3D human figure animation. It is, however, just one piece of a complete system.

7.1 Integration with other techniques

Verbs & Adverbs presents an interpolated technique, and as such is subject to the limitations of such methods. Inverse kinematics constraints were found to be useful in augmenting the effectiveness of the interpolations, especially for things like foot supports. Other animation traditions could prove useful to the system. Interpolation, for example, can cause the figure to move in ways not plausible or comfortable for a real human. The balance and biomechanics work mentioned in Chapter 2 could be integrated to further improve the effectiveness the system.

Integration with procedural work like Perlin's, or dynamical simulation, like Hodgins', could be useful. Perlin's use of noise, for example, increases the believability of a motion could be easily integrated with *Verbs & Adverbs*. Some tasks are naturally well suited to procedural or dynamical methods such as procedural head turns or gazes or dynamical balancing. Such techniques could augment the *Verbs & Adverbs* system and help create a more convincing real-time animation system. Standards efforts like HANIM [5] could be a facilitating event making such integration more possible.

Higher level control has not been addressed in this dissertation. Even the verb graph, which abstracts some of the lower level problems, is not high level control scheme. A high level controller is the object that studies the state of

the world and then controls the verbs and adverbs to achieve a goal. High level control, for instance, may involve planning a course, such as through a maze, and then direct the actor along that path, dealing with unforeseen circumstances as they arise. Animation generated with *Verbs & Adverbs* could be used by a system like Perlin's *Improv* or those developed in Blumberg's Synthetic Character Group at MIT's Media Lab. *Motivate*(TM), a product of *The Motion Factory*, could also prove a useful system for driving *Verbs & Adverbs*-based animation segments.

7.2 Skinning & musculature animation

Hodgins, O'Brien, and Tumblin [67] describe some tests indicating viewers of animation systems have difficulty discriminating between subtle differences in animation when the characters are represented as simple models, such as lines. "Models" here refers to the graphical representation of a figure. Hodgins, *et. al.* showed that accurate depictions of human figures are superior to simple models, leaving open the question of whether viewers can discriminate differences on a model somewhere between these two extremes. Despite this, their work is compelling enough that the *Verbs & Adverbs* system will soon include more realistic human figures.

This dissertation showed how multi-resolution radial B-spline interpolation could lead to motion synthesis. The MRBF formulation, however, will likely prove useful in many areas of computer graphics, as did wavelets in recent computer graphics history. Skinning, for example, is an obvious candidate and is particularly complementary to the motion work. Artist directed skinning, parameterized by joint angle, effort, emotion, and physical statistics could prove a very effective way to improve one of the worst aspects of human figure animation: the disturbing nature of our models, which appear more like

mannequins than living beings.

7.3 Facial animation

Finally, there has been much work recently on facial animation. Guenter, Grimm, et. al. showed how facial expressions could be synthesized from examples in [61], thus allowing for very low-bandwidth transmission of high-quality faces. Pighin et. al. also used blended examples to form high-quality facial animation in [115].

DeCarlo et. al. showed how human anthropometry data could be used to generate parameterized facial models in [39]. Large quantities of accurate anthropometry data combined with an interpolation method such as MRBF may be the key highly adaptable human figure models. Current anthropometry efforts by the U.S. military might prove useful in a few years as the data becomes available.

7.4 Open issues

The *Verbs & Adverbs* project is now seeking to work more extensively with animators. Through further collaboration, we seek to understand what things an artist finds empowering or limiting, intuitive or frustrating about *Verbs & Adverbs*. How easily do animators grasp, for example, the notion of an n-D adverb space of largely independent adverb axes?

The *Verbs & Adverbs* system is currently a loose collection of tools, a circumstance that slows the verb refinement loop. This loop is the process the animator uses to work with the system, so it is important that overhead not hamper the artist. One investment we plan to make is to improve the authoring system, perhaps through integration with a commercial 3-D package such as

3D-Studio/Max(TM) or *SoftImage*(TM).

In addition to improving *Verbs & Adverbs* for use in animation, we also intend to discover where else MRBF interpolation may prove applicable. That process may uncover deficiencies in the interpolation mechanism. Improvements in the basic interpolation as well as in clustering may be needed in order to use the technique in other problem domains. Non-spherical radial bases, for example, are one possible avenue to explore.

Reparameterization of the adverb space is likely to prove important. A walking verb, for example, may need to be individualized. A saddened person may have a hunched back, but so also might a perfectly happy older person. Without changing the walk, individualization might be accomplished through reparameterization. It will also likely prove useful for linearizing the naturally non-linear nature of adverbs for problems such as reaching (Chapter 6).

Appendix A

The motion formalism

This appendix synopsizes the motion formalism, which was developed in Chapter 3, Section 3.8.

A motion $\mathbf{M}_i$ is an object which can respond to a number of questions posed to it. The primary question is of DOF position, $\theta_i(T)$, meaning "what are the values for the DOFs for motion i and time T", where T is expressed in verb-time as defined in Section 3.4. This is a syntactic simplification for the sake of making this formalism less cluttered, but could be expressed in a more general form: $\theta(\mathbf{M}_i, T)$, meaning "apply the position operator to motion $\mathbf{M}_i$ at T".

The data items a motion must be able to supply were detailed in Table 3.1, but are repeated here in Table A.1. Likewise, the full set of questions a motion must be able to answer were first introduced in Table 3.2, but are repeated here in Table A.2. Figure 3.21 first introduced the full complement of motion types used in the *Verbs & Adverbs* system. It has been expanded to include the motions introduced in Chapters 4 and 5 and appears in Figure A.1.

This appendix will proceed by introducing each of the motion types developed in Chapters 3, 4, and 5, very briefly and will then show how the motion formalism is implemented. This appendix is not meant to re-describe the rationale behind these implementations, but rather provide a concise reference.

data-item	*values*	*description*
K_{im}	$[0 \ldots +\infty)$	mth key-time for motion $\mathbf{M}_i$
τ_i^s	$(-\infty \ldots +\infty)$	start-time of motion $\mathbf{M}_i$
τ_i^e	$(-\infty \ldots +\infty)$	end-time of motion $\mathbf{M}_i$
T_i^d	$[0 \ldots +\infty)$	duration of motion $\mathbf{M}_i$
$\mathbf{p}_i$	$\Re^{NumAdverbs}$	adverb values for motion $\mathbf{M}_i$

Table A.1: Basic motion values: key-times, time-bounds, duration, and adverbs

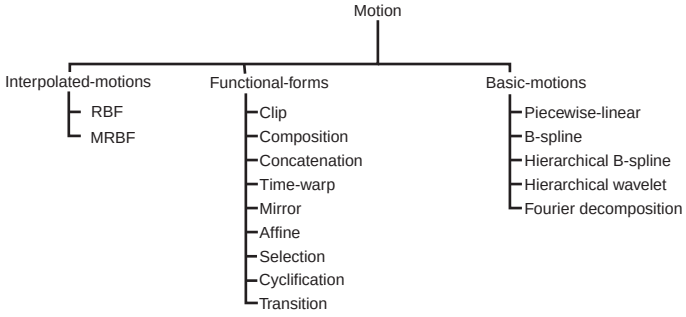

Figure A.1: A hierarchy of motion types

The reader should refer to the chapter and section indicated to find more out about a particular motion type.

A.1 Basic motion

The basic motions form the right-most sub-tree in the motion hierarchy shown in Figure A.1. These are simple motions described by a set of curves supplied by the motion designer most likely through motion capture or hand animation. Basic motions were first introduced in Section 3.8 by Definition 1 as

$$\text{basic motion} \quad \mathbf{M}_i = \left\{\mathbf{K}_i, \mathcal{I}_i, \mathbf{p}_i, \mathrm{T}_i^d, C_i\right\}$$

function	*result*	*range*	*description*
$\tau_i(T)$	τ	$(-\infty \ldots +\infty)$	verb-time to animation-time
$T_i(t)$	T	$[0 \ldots +\infty)$	canonical-time to verb-time
$t_i(T)$	t	$[0 \ldots 1]$	verb-time to canonical-time
$\theta_{ij}(T)$	position	$\Re$	return DOF positions
$\dot{\theta}_{ij}(T)$	velocity	$\Re$	return DOF velocities
$\ddot{\theta}_{ij}(T)$	acceleration	$\Re$	return DOF accelerations
$\mathcal{D}_{ij}(T)$	DOF-usage	{**required**, **defined** **undefined**}	return DOF usage on a DOF-by-DOF basis
$\mathcal{I}_i(T)$	constraints		IK constraints active at T

Table A.2: Motion functions: time projections and kinematic operators

where the key-times $\mathbf{K}_i$, constraints $\mathcal{I}_i$, adverbs by $\mathbf{p}_i$, duration T_i^d, and DOF-curves C_i are all user-supplied values. C_i is a collection of DOF-curves for some or all of the possible DOFs encoded using any curve representation which can respond to position, velocity, and acceleration queries.

Basic motions implement the formalism as follows:

$$\begin{aligned} \tau_i^s &= 0 \\ \tau_i^e &= \mathrm{T}_i^d \\ \mathrm{T}_i^d &= \text{designer supplied} \\ \mathbf{K}_{im} &= \text{designer supplied} \end{aligned}$$

$$
\begin{aligned}
\mathbf{p}_i &= \text{designer supplied} \\
\tau_i(T) &= T \\
T_i(t) &= \text{Refer to Equation 3.2} \\
t_i(T) &= \text{Refer to Equation 3.1} \\
\theta_{ij}(T) &= \begin{cases} C_{ij}(T) & \mathcal{D}_{ij}(T) = \textbf{required} \\ 0 & \text{otherwise} \end{cases} \\
\dot{\theta}_{ij}(T) &= \begin{cases} \dot{C}_{ij}(T) & \mathcal{D}_{ij}(T) = \textbf{required} \\ 0 & \text{otherwise} \end{cases} \\
\ddot{\theta}_{ij}(T) &= \begin{cases} \ddot{C}_{ij}(T) & \mathcal{D}_{ij}(T) = \textbf{required} \\ 0 & \text{otherwise} \end{cases} \\
\mathcal{D}_{ij}(T) &= \begin{cases} \textbf{required} & \forall \text{ DOFs } j \text{ in } C_i \\ \textbf{undefined} & \text{otherwise} \end{cases} \\
\mathcal{I}_i(T) &= \text{designer supplied}
\end{aligned}
$$

A.2 Clip motion

A clip motion extracts a short piece from a longer piece of animation. It was first introduced in Section 3.9 and by Definition 2 as

$$\text{clip motion} \quad \mathbf{M}_i = \{t^s_{i'}, t^e_{i'}, \mathbf{M}_{i'}\}$$

where $t^s_{i'}$ and $t^e_{i'}$ mark the start and stop region of the clip in the canonical timeline of motion $\mathbf{M}_{i'}$.

The clip motion implements the motion formalism as follows:

$$
\begin{aligned}
\tau_i^s &= 0 \\
\tau_i^e &= \mathrm{T}_i^d \\
\mathrm{T}_i^d &= T_{i'}(t_{i'}^e) - T_{i'}(t_{i'}^s) \\
\mathbf{K}_{im} &= \mathbf{K}_{i'm'} - T_{i'}(t_{i'}^s) \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\tau_i(T) &= T \\
T_i(t) &= \text{Refer to Equation 3.2} \\
t_i(T) &= \text{Refer to Equation 3.1} \\
\theta_i(T) &= \theta_{i'}(T + T_{i'}(t_{i'}^s)) \\
\dot{\theta}_i(T) &= \dot{\theta}_{i'}(T + T_{i'}(t_{i'}^s)) \\
\ddot{\theta}_i(T) &= \ddot{\theta}_{i'}(T + T_{i'}(t_{i'}^s)) \\
\mathcal{D}_i(T) &= \mathcal{D}_{i'}(T + T_{i'}(t_{i'}^s)) \\
\mathcal{I}_i(T) &= \mathcal{I}_i(T + T_{i'}(t_{i'}^s)) - \text{constraints excluded by the clip}
\end{aligned}
$$

A.3 Affine motion

An affine motion $\mathbf{M}_i$ shifts and scales another motion $\mathbf{M}_{i'}$ in time. It was first introduced in Section 3.9 and by Definition 3 as

$$\text{affine motion} \quad \mathbf{M}_i = \{\tau_i^s, s, \mathbf{M}_{i'}\}, s > 0$$

where τ_i^s is the moment in animation-time when the motion is set to begin and s a scaling factor which changes the overall duration of the motion.

The motion formalism is implemented using the following formulae:

$$
\begin{aligned}
\tau_i^s &= \text{designer supplied} \\
\tau_i^e &= \tau_i^s + \mathrm{T}_i^d \\
\mathrm{T}_i^d &= s \cdot \mathrm{T}_{i'}^d \\
\mathbf{K}_{im} &= s \cdot \mathbf{K}_{i'm} \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\tau_i(T) &= \tau_i^s + s \cdot T \\
T_i(t) &= s \cdot T_{i'}(t) \\
t_i(T) &= \text{Refer to Equation 3.1} \\
\theta_i(T) &= \theta_{i'}\left(\frac{T}{s}\right) \\
\dot{\theta}_i(T) &= \dot{\theta}_{i'}\left(\frac{T}{s}\right) \\
\ddot{\theta}_i(T) &= \ddot{\theta}_{i'}\left(\frac{T}{s}\right) \\
\mathcal{D}_{ij}(T) &= \mathcal{D}_{i'j}\left(\frac{T}{s}\right) \\
\mathcal{I}_i(T) &= \mathcal{I}_{i'}\left(\frac{T}{s}\right)
\end{aligned}
$$

A.4 Time-warp motion

A time-warp motion $\mathbf{M}_i$ distorts the natural timeline of another motion $\mathbf{M}_{i'}$ using a time-warping function specified by the designer. It was first introduced in Section 3.9 and by Definition 4 as

$$\text{time-warp motion} \quad \mathbf{M}_i = \{\mathcal{U}, \mathbf{M}_{i'}\}$$

where $\mathcal{U}$ is a monotonically increasing function with domain and range [0..1]. The $\mathcal{U}$ function warps the relative duration of the segments of the motion.

The motion formalism is implemented as follows:

$$
\begin{aligned}
\tau_i^s &= \tau_{i'}^s \\
\tau_i^e &= \tau_{i'}^e \\
\mathrm{T}_i^d &= \mathrm{T}_{i'}^d \\
\mathbf{K}_{im} &= \mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{\mathbf{K}_{i'm}}{\mathrm{T}_i^d}\right) \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\tau_i(T) &= T + \tau_i^s \\
T_i(t) &= \text{ Refer to Equation 3.2} \\
t_i(T) &= t_{i'}\left(T + \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\theta_i(T) &= \theta_{i'}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\dot{\theta}_i(T) &= \dot{\theta}_{i'}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\ddot{\theta}_i(T) &= \ddot{\theta}_{i'}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\mathcal{D}_{ij}(T) &= \mathcal{D}_{i'j}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right) \\
\mathcal{I}_i(T) &= \mathcal{I}_{i'}\left(\mathrm{T}_i^d \cdot \mathcal{U}\left(\frac{T}{\mathrm{T}_i^d}\right)\right)
\end{aligned}
$$

A.5 Mirror motion

A mirror motion $\mathbf{M}_i$ of a motion $\mathbf{M}_i$ switches the left-right direction and body motion. It is useful for changing a walk to the right, for example, into a walk to the left. It was first introduced in Section 3.9 and by Definition 5 as

$$\text{mirror motion} \quad \mathbf{M}_i = \{A, S, \mathbf{M}_{i'}\}$$

where motion $\mathbf{M}_{i'}$ is the motion to be mirrored and A and S are sets of DOF pairs $\{j, j'\}$ which are anti-symmetrical and symmetrical.

The motion formalism for the mirror motion is implemented as follows:

$$
\begin{aligned}
\tau_i^s &= \tau_{i'}^s \\
\tau_i^e &= \tau_{i'}^e \\
\mathrm{T}_i^d &= \mathrm{T}_{i'}^d \\
\mathbf{K}_{im} &= \mathbf{K}_{i'm} \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\tau_i(T) &= \tau_{i'}(T) \\
T_i(t) &= T_{i'}(t) \\
t_i(T) &= t_{i'}(T) \\
\theta_{ij}(T) &= \begin{cases} \theta_{i'j'}(T) & \forall \text{ symmetry pairs } \{j,j'\} \in \mathbf{M}_i \\ -\theta_{i'j'}(T) & \forall \text{ anti-symmetry pairs } \{j,j'\} \in \mathbf{M}_i \\ \theta_{i'j}(T) & \text{otherwise.} \end{cases} \\
\dot{\theta}_{ij}(T) &= \begin{cases} \dot{\theta}_{i'j'}(T) & \forall \text{ symmetry pairs } \{j,j'\} \in \mathbf{M}_i \\ -\dot{\theta}_{i'j'}(T) & \forall \text{ anti-symmetry pairs } \{j,j'\} \in \mathbf{M}_i \\ \dot{\theta}_{i'j}(T) & \text{otherwise.} \end{cases} \\
\ddot{\theta}_{ij}(T) &= \begin{cases} \ddot{\theta}_{i'j'}(T) & \forall \text{ symmetry pairs } \{j,j'\} \in \mathbf{M}_i \\ -\ddot{\theta}_{i'j'}(T) & \forall \text{ anti-symmetry pairs } \{j,j'\} \in \mathbf{M}_i \\ \ddot{\theta}_{i'j}(T) & \text{otherwise.} \end{cases} \\
\mathcal{D}_{ij} &= \begin{cases} \mathcal{D}_{i'j'}(T) & \forall \text{ symmetry and anti-symmetry pairs } \{j,j'\} \in \mathbf{M}_i \\ \mathcal{D}_{i'j}(T) & \text{otherwise.} \end{cases} \\
\mathcal{I}_i &= \mathcal{I}_{i'} \text{ modified by the symmetry and anti-symmetry pairs}
\end{aligned}
$$

A.6 Composition

Motion compositing overlays multiple motions to create a gestalt motion from the pieces. It was first introduced in Section 3.9 by Definition 6 as

$$\text{composition motion} \quad \mathbf{M}_i = \{\mathbf{p}_i, \mathbf{M}_{i_0}, \mathbf{M}_{i_1}, \ldots \mathbf{M}_{i_n}\}$$

where the $\mathbf{M}_{i_c}$'s are the motions to be composed with one another. The adverb value $\mathbf{p}_i$ is designer specified since there is no reasonable procedural way to assign an overall adverb value to a motion composed of many different motions with many different, and potentially conflicting, adverbs settings.

The motion formalism is implemented as follows:

$$\begin{aligned}
\tau_i^s &= \min_c \tau_{i_c}^s \\
\tau_i^e &= \max_c \tau_{i_c}^e \\
\mathrm{T}_i^d &= \tau_i^e - \tau_i^s \\
\mathbf{K}_{im} &= \tau_{i_c}(\mathbf{K}_{i_c m'}) - \tau_i^s \\
\tau_i(T) &= T + \tau_i^s \\
T_i(t) &= \text{Refer to Equation 3.2} \\
t_i(T) &= \text{Refer to Equation 3.1} \\
\mathbf{p}_i &= \text{designer specified} \\
\theta_{ij}(T) &= \text{Arbitrate}\left(\theta_{i_0 j}(T_{i_0}), \theta_{i_1 j}(T_{i_1}), \ldots, \theta_{i_n j}(T_{i_n})\right) \\
\dot{\theta}_{ij}(T) &= \text{Arbitrate}\left(\dot{\theta}_{i_0 j}(T_{i_0}), \dot{\theta}_{i_1 j}(T_{i_1}), \ldots, \dot{\theta}_{i_n j}(T_{i_n})\right) \\
\ddot{\theta}_{ij}(T) &= \text{Arbitrate}\left(\ddot{\theta}_{i_0 j}(T_{i_0}), \ddot{\theta}_{i_1 j}(T_{i_1}), \ldots, \ddot{\theta}_{i_n j}(T_{i_n})\right) \\
\mathcal{D}_{ij}(T) &= \max_c \mathcal{D}_{i_c j}(T) \\
\mathcal{I}_i &= \bigcup_c \mathcal{I}_{i_c}(T)
\end{aligned}$$

where $T_{i_c} = T_{i_c}\left(\tau_i(T)\right)$.

A.7 Concatenation

Concatenation places a number of motions one after another on the timeline to create a longer sequence. It was first introduced in Section 3.9 by Definition 7 as

$$\text{concatenation motion} \quad \mathbf{M}_i = \{\mathbf{p}_i, \mathbf{M}_{i_0}, \mathbf{M}_{i_1}, \ldots \mathbf{M}_{i_n}\}$$

where the $\mathbf{M}_{i_c}$'s are the motions being concatenated in order from $0..n$. Like the composition motion, the overall adverb value is specified by the designer as $\mathbf{p}_i$.

The motion formalism is implemented as follows:

$$\begin{aligned}
\tau_i^s &= 0 \\
\mathrm{T}_i^d &= \sum_{c=0}^{n} T_{i_c}^d \\
\tau_i^e &= \mathrm{T}_i^d \\
\mathbf{K}_{im} &= \begin{cases} \mathbf{K}_{i_n m'} + \sum_{c=0}^{n-1} T_{i_c}^d & n > 0 \\ \mathbf{K}_{i_0 m'} & \text{otherwise} \end{cases} \\
\mathbf{p}_i &= \text{Designer supplied} \\
\tau_i(T) &= T \\
T_i(t) &= \text{Refer to Equation 3.2} \\
t_i(T) &= \text{Refer to Equation 3.1} \\
\theta_i(T) &= \theta_{i_c}(T - T_{\text{elapsed}}) \\
\dot{\theta}_i(T) &= \dot{\theta}_{i_c}(T - T_{\text{elapsed}}) \\
\ddot{\theta}_i(T) &= \ddot{\theta}_{i_c}(T - T_{\text{elapsed}}) \\
\mathcal{D}_{ij}(T) &= \mathcal{D}_{i_c j}(T - T_{\text{elapsed}}) \\
\mathcal{I}_i(T) &= \mathcal{I}_{i_c}(T - T_{\text{elapsed}})
\end{aligned}$$

where

$$T_{\text{elapsed}} = T - \sum_{c=0}^{n-1} T_{i_c}^{d}$$

and where n is the maximum n which satisfies

$$\sum_{c=0}^{n-1} T_{i_c}^{d} \leq T.$$

If no n satisfies, then the first motion has not yet been exhausted, so $T_{\text{elapsed}} = T$. Motion $\mathbf{M}_{i_c}$ is the motion active at the verb-time T in question.

A.8 Selection

A selection type motion $\mathbf{M}_i$ modifies the DOF-usage function, $\mathcal{D}$, of another motion $\mathbf{M}_{i'}$. It was first introduced in Section 3.9 by Definition 8 as

$$\text{selection motion} \quad \mathbf{M}_i = \{\mathbf{M}_{i'}, \{j, \mathit{usage}\}^*\}$$

where $\mathbf{M}_{i'}$ is the motion having its DOF-usages adjusted and *usage* either **undefined**, **defined**, or **required**. The motion formalism is easily implemented for this motion as:

$$\begin{aligned}
\tau_i^s &= \tau_{i'}^s \\
\tau_i^e &= \tau_{i'}^e \\
\mathrm{T}_i^d &= \mathrm{T}_{i'}^d \\
\mathbf{K}_{im} &= \mathbf{K}_{i'm} \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\tau_i(T) &= \tau_{i'}(T) \\
T_i(t) &= T_{i'}(t) \\
t_i(T) &= t_{i'}(T)
\end{aligned}$$

$$\theta_{ij}(T) = \begin{cases} \theta_{i'j}(T) & \forall \text{ DOFs not } \textbf{undefined} \\ 0 & \text{otherwise} \end{cases}$$

$$\dot{\theta}_{ij}(T) = \begin{cases} \dot{\theta}_{i'j}(T) & \forall \text{ DOFs not } \textbf{undefined} \\ 0 & \text{otherwise} \end{cases}$$

$$\ddot{\theta}_{ij}(T) = \begin{cases} \ddot{\theta}_{i'j}(T) & \forall \text{ DOFs not } \textbf{undefined} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{D}_{ij}(T) = \begin{cases} \textbf{undefined} & \forall \text{ DOFs } j \textbf{ undefined} \text{ by the selection} \\ \textbf{defined} & \forall \text{ DOFs } j \textbf{ defined} \text{ by the selection} \\ \textbf{required} & \forall \text{ DOFs } j \textbf{ required} \text{ by the selection} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{I}_i(T) = \mathcal{I}_{i'}(T) - \text{ those involving } \textbf{undefined} \text{ DOFs}$$

A.9 Cyclification

A cycle motion $\mathbf{M}_i$ transforms a motion $\mathbf{M}_{i'}$ into one which is perfectly cyclic, so that no discontinuities occur if the motion is repeated. It was first introduced in Section 3.10 by Definition 9 as

$$\text{cycle motion} \quad \mathbf{M}_i = \{\mathbf{c}_i, \mathbf{M}_{i'}\}$$

where $\mathbf{c}_i$ is a vector of booleans indicating which DOFs are cyclified. Remember that the root **Z** translation is typically not-cyclified. Using the boolean vector generalizes that notion. Implementing the motion formalism is as follows:

$$\begin{aligned} \tau_i^s &= \tau_{i'}^s \\ \tau_i^e &= \tau_{i'}^e \\ \mathrm{T}_i^d &= \mathrm{T}_{i'}^d \end{aligned}$$

$$
\begin{aligned}
\mathbf{K}_{im} &= \mathbf{K}_{i'm} \\
\mathbf{p}_i &= \mathbf{p}_{i'} \\
\tau_i(T) &= \tau_{i'}(T) \\
T_i(t) &= T_{i'}(t) \\
t_i(T) &= t_{i'}(T) \\
\theta_{ij}(T) &= \begin{cases} \theta_{i'j}(T) & \mathbf{c}_{ij} \text{ false} \\ \theta_{i'j}(T) + \frac{T \cdot d_{ij}}{\mathrm{T}_i^d} - \frac{d_{ij}}{2} & \mathbf{c}_{ij} \text{ true} \\ \text{where } \; d_{ij} = \theta_{i'j}(\mathrm{T}_{i'}^d) - \theta_{i'j}(0) & \end{cases} \\
\dot{\theta}_{ij}(T) &= \begin{cases} \dot{\theta}_{i'j}(T) & \mathbf{c}_{ij} \text{ false} \\ \dot{\theta}_{i'j}(T) + \frac{T \cdot d_{ij}}{\mathrm{T}_i^d} - \frac{d_{ij}}{2} & \mathbf{c}_{ij} \text{ true} \\ \text{where } \; d_{ij} = \dot{\theta}_{i'j}(\mathrm{T}_{i'}^d) - \dot{\theta}_{i'j}(0) & \end{cases} \\
\ddot{\theta}_{ij}(T) &= \begin{cases} \ddot{\theta}_{i'j}(T) & \mathbf{c}_{ij} \text{ false} \\ \ddot{\theta}_{i'j}(T) + \frac{T \cdot d_{ij}}{\mathrm{T}_i^d} - \frac{d_{ij}}{2} & \mathbf{c}_{ij} \text{ true} \\ \text{where } \; d_{ij} = \ddot{\theta}_{i'j}(\mathrm{T}_{i'}^d) - \ddot{\theta}_{i'j}(0) & \end{cases} \\
\mathcal{D}_{ij}(T) &= \mathcal{D}_{i'j}(T) \\
\mathcal{I}_i(T) &= \mathcal{I}_{i'}(T)
\end{aligned}
$$

A.10 Transition

A transition motion $\mathbf{M}_i$ bridges the gap between to motions $\mathbf{M}_A$ and $\mathbf{M}_B$. It was first introduced in Section 5.5 by Definition 10 as

$$\text{transition motion} \;\; \mathbf{M}_i = \{\alpha, \mathbf{M}_\mathrm{A}, t_A^s, t_A^e, \mathbf{M}_\mathrm{B}, t_B^s, t_B^e\}$$

where α is the blending function as defined previously, $\mathbf{M}_\mathrm{A}$ and $\mathbf{M}_\mathrm{B}$, the motions being blended from and to respectively, and $[t_A^s \ldots t_A^e]$ and $[t_B^s \ldots t_B^e]$ the blending regions in the two motions expressed in canonical time.

The motion formalism is implemented as follows:

$$
\begin{aligned}
\tau_i^s &= 0 \\
\tau_i^e &= \mathrm{T}_i^d \\
\mathrm{T}_i^d &= \frac{(T_A^e - T_A^s) + (T_B^e - T_B^s)}{2} \\
\mathbf{K}_{im} &= \text{any key-times } \mathbf{K}_{Am}, \mathbf{K}_{Bm} \text{ in the blend regions for } \mathbf{M}_A \text{ and } \mathbf{M}_B \\
\mathbf{p}_i &= \text{assigned by the designer or blended from } \mathbf{M}_A \text{ and } \mathbf{M}_B \\
\tau_i(T) &= T \\
T_i(t) &= \text{Refer to Equation 3.2} \\
t_i(T) &= \text{Refer to Equation 3.1} \\
\theta_{ij}(T) &= \begin{cases} (1-\alpha(t))\theta_{Aj}\,(T + T_A^s) + \alpha(t)\theta_{Bj}\,(T + T_B^s) \\ \qquad \text{for non-root DOFs } i \\ \int_{t'=0}^{t}(1-\alpha(t'))\dot{\theta}_{Aj}\,(T' + T_A^s) + \alpha(t')\dot{\theta}_{Bj}\,(T' + T_B^s) \\ \qquad \text{otherwise} \end{cases} \\
\dot{\theta}_{ij}(T) &= (1-\alpha(t))\dot{\theta}_{Aj}\,(T + T_A^s) + \alpha(t)\dot{\theta}_{Bj}\,(T + T_B^s) \\
\ddot{\theta}_{ij}(T) &= (1-\alpha(t))\ddot{\theta}_{Aj}\,(T + T_A^s) + \alpha(t)\ddot{\theta}_{Bj}\,(T + T_B^s) \\
\mathcal{D}_{ij}(T) &= \begin{cases} \mathcal{D}_{Aj}(T + T_A^s) & t_i(T) \le 0.5 \\ \mathcal{D}_{Bj}(T + T_B^s) & \text{otherwise} \end{cases} \\
\mathcal{I}_i(T) &= \begin{cases} \mathcal{I}_A(T + T_A^s) & t_i(T) \le 0.5 \\ \mathcal{I}_B(T + T_B^s) & \text{otherwise} \end{cases}
\end{aligned}
$$

A.11 Verbs

The verb is the central object in this thesis. A full description of its workings is can be found in Sections 4.3 and 4.6. Its definition is

$$\text{verb motion } \mathbf{M}_i \; = \left\{\mathcal{D}_i, \mathcal{I}_i, {\mathbf{M}_{i'}}^{+}\right\}$$

where the $\mathbf{M}_{i'}{}^{+}$ are the example motions as detailed in Chapters 3 and 4. There must be at least $NumAdverbs + 1$ examples in order to establish a baseline interpolation for a motion with *NumAdverbs* adverbs. Some of mechanisms for verbs are too complicated to be shown here in a compact form, so the motion formalism will refer the reader back to the appropriate sections when necessary. Otherwise, the motion formalism is implemented as:

$$
\begin{aligned}
\tau_i^s &= 0 \\
\tau_i^e &= \mathbf{K}_{i,NumKeyTimes} \text{ see below and Section 4.3} \\
\mathrm{T}_i^d &= \tau_i^e - \tau_i^s \\
\mathbf{K}_{i0} &= 0 \\
\mathbf{K}_{im} &= \sum_{l=1}^{NumAdverbs} a_{lm} A_l(\mathbf{p}_i) + \sum_{c=0}^{NumClusterLevels} \sum_{i'=0}^{NumExamples} r_{c,i'm} R_{c,i'}(\mathbf{p}_i) \\
\mathbf{p}_i &= \text{Variable; used to control verbs and is set at run time} \\
\tau_i(T) &= T \\
T_i(t) &= \text{Refer to Equation 3.2} \\
t_i(T) &= \text{Refer to Equation 3.1} \\
\theta_{ij}(T) &= f(b_{jk}(\mathbf{p}_i)) \text{ Refer to Section 4.3} \\
\dot{\theta}_{ij}(T) &= \dot{f}(b_{jk}(\mathbf{p}_i)) \text{ Refer to Section 4.3} \\
\ddot{\theta}_{ij}(T) &= \ddot{f}(b_{jk}(\mathbf{p}_i)) \text{ Refer to Section 4.3} \\
\mathcal{D}_{ij}(T) &= \text{Designer specified at verb construction time} \\
\mathcal{I}_i(T) &= \text{Designer specified at verb construction time}
\end{aligned}
$$

Appendix B

Dynamics equations & torque-minimal transitioning

As stated in Chapter 2, our SIGGRAPH paper *Efficient Generation of Motion Transitions Using Spacetime Constraints* [124] used Balafoutis and Patel's [8] linear recursive dynamics formulation for spacetime optimization for motion transitions for a human figure with 44 DOFs. Quality transitions enable the joining of two segments of motion capture or hand animated source. The intuition for torque minimal transitioning is from Burdett, Skrinar, and Simon [27]. Joint torques, they found, are a reasonable predictor of metabolic energy. Experience has shown that motion which minimizes metabolic energy looks natural. This leads to the minimization problem:

$$minimize\ e = \int_{\tau_1}^{\tau_2} \sum_j E_j(\tau)^2 d\tau \tag{B.1}$$

where E_j is the energy function for the jth DOF. This appendix will describe that minimization process, will detail the equations in the Balafoutis formulation and their derivatives, and will show examples of spacetime transitioning.

Please note that this appendix will use a different scheme than the rest of the thesis for terms and subscripts. This is to keep it in line with [124] in addition to Balafoutis and Patel [8].

Representations used in the past for the DOF function,

$$\mathbf{q}(\tau) = (q_1(\tau), \ldots, q_n(\tau))$$

where $q_j(\tau)$ is the value of DOF j at time τ, include piecewise constant [147], B-splines [35], and B-spline wavelets [91]. B-spline wavelets show good convergence properties for spacetime optimization when the number of basis functions in a single degree of freedom is large, e.g. more than 20 or 30. Since transitioning generally involves a short amount of time, on the order or 1 second or less, good paths can be represented with 5 to 10 B-spline coefficients. Our experience has been that very few iterations are required to achieve convergence with a B-spline basis, so the extra complexity and computation f the B-spline wavelet basis was not justified. For these reasons, we use cubic B-splines as the basis functions for $\mathbf{q}(\tau)$.

We use the BFGS optimization algorithm [53] to find a minimum of integral of Equation B.1. BFGS belongs to the class of quasi-Newton algorithms which progress toward a solution by using the gradient of the objective function

$$\mathbf{g} = \nabla e.$$

The gradient is used to incrementally update a matrix decomposition of a psuedo-Hessian matrix, $\mathbf{H}$, and to compute a new step direction

$$\mathbf{d} = -\mathbf{H}^{-1}\mathbf{g}$$

The relative amount of computation for each subtask required at every iteration of the algorithm is common to several quasi-Newton algorithms: gradient computation, pseudo-Hessian, and computation of the step direction.

Since each of the E_j is potentially a function of all the q, $\dot{q}$, and $\ddot{q}$, the gradient requires the evaluation of $\mathbf{O}(n^2)$ partial derivatives where n is the number of DOFs in the body. This is in fact a lower bound for the asymptotic

time complexity of spacetime algorithms which use gradient-based optimization techniques.

If m is the number of B-spline coefficients used to define the time function of each degree of freedom then the pseudo-Hessian is a square matrix with nm rows and columns. The update of the pseudo-Hessian and computation of the step direction are both $\mathbf{O}\left((nm)^2\right)$. For m small, less than 20, and n large, more than 30, the time required to compute $\mathbf{g}$ dominates all other computation thus an efficient formulation for $\mathbf{g}$ will pay the greatest dividends in reducing computation.

Computing $\mathbf{g}$ requires finding the joint torques and a variety of subsidiary quantities, such as angular velocity and acceleration. This is the inverse dynamics problem which has been extensively studied in the robotics literature. Balafoutis provides a good overview of many of these algorithms in his book. Many inverse dynamics formulations have been proposed ranging from $\mathbf{O}\left(n^4\right)$ non-recursive to $\mathbf{O}\left(n\right)$ recursive algorithms. The inverse dynamics formulation we use was developed by Balafoutis and Patel. It is an $\mathbf{O}\left(n\right)$ recursive one which requires $96n - 77$ multiplications and $84n - 77$ additions to solve the inverse dynamics problem for a articulated figure with n DOFs. This is faster than the $\mathbf{O}\left(n\right)$ Legragian recursive formulation developed by Hollerbach [76] which was used by Liu and Cohen [89] which requires $412n - 277$ multiplications and $320n - 201$ additions.

The efficiency of Balafoutis and Patel's formulation derives from the computation efficiency of Cartesian tensors and from the recursive nature of the computations. These efficiencies carry over to the computation of the gradient terms. The algorithm proceeds in two steps. In the first step velocities, accelerations, net torques, and forces at each DOF are computed starting from the root DOF and working out to the tips of all the chains in the tree. In

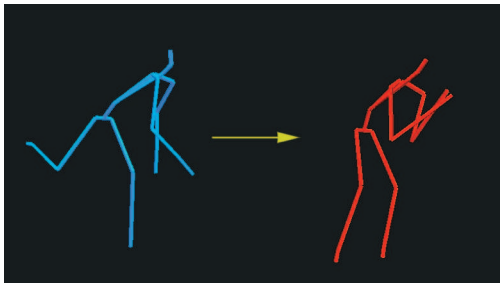

Figure B.1: End position of motion 1 and beginning position of motion 2 for a transition

the second step, the joint torques are computed starting from the tips of the chains back to the root DOF. Integrating over the transition yields the overall energy, which is minimized using the BFGS optimization algorithm.

B.1 Results

We successfully applied the motion transition algorithm on many motions. Figure B.1 shows the endpoints of the motions being bridged. For this example, the transition time was set to 0.6 seconds and the number of B-spline coefficients to 5. The resulting transition is shown in Figure B.2. Our experience has been that successful transitions are quite short, usually in the range of 0.3 to 0.6 seconds. Without a biomechanical model to guide the generation of a large motion, our minimal energy model will often prove insufficient.

The beginning of the transition is colored blue and the end is colored red with intermediate times a linear blend of the two colors. The motion is one transition from a longer animation which has 5 transitions between 6 motions, shown in Figure B.3. Inverse-kinematics is used to augment the effectiveness of the transitioning technique as shown in Figure B.4.

Figure B.5 shows an example of a motion transition which affects only the arm degrees of freedom of the motion. This sequence actually consists of

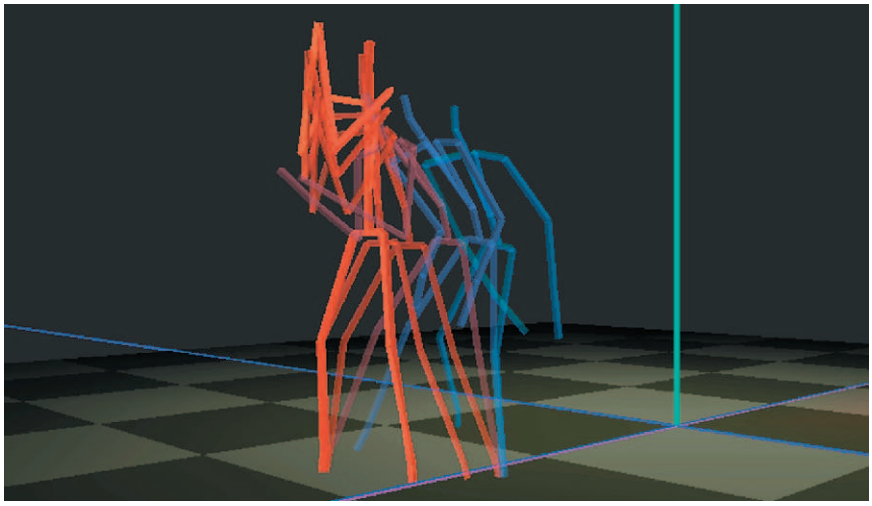

Figure B.2: Multiple time exposure of transition generated from the motions in Figure B.1

Figure B.3: The complete animation with 5 transitions between 6 different motions

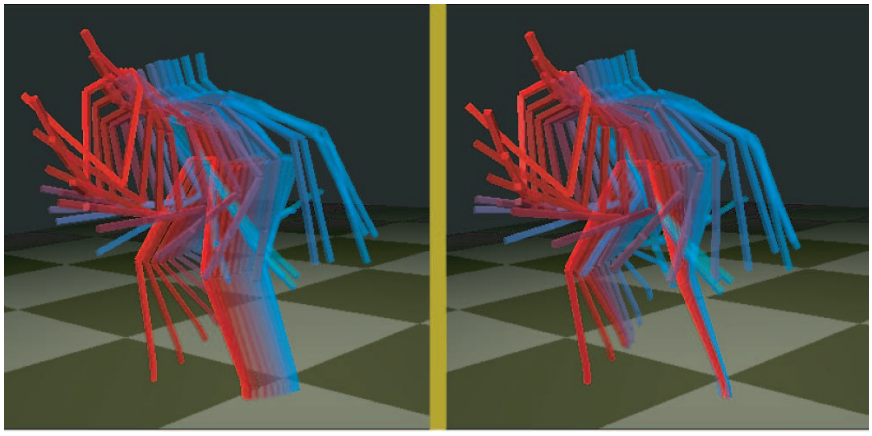

Figure B.4: Inverse-kinematics is used to improve the placement of the feet during transitions

two torque-minimal transitions: one from a walking arm motion to the salute motion and another back to the walking arm motion. Each transition was 0.3 seconds long.

Computation times for transitions are strongly dependent on the number of DOFs involved since the spacetime formulation we use is $\mathbf{O}(n^2)$ in the number of degrees of freedom. For the transition of Figure B.2 generating the spacetime transition motion took 72 seconds. This transition involved 44 DOFs. For the transition of Figure B.5 generating the spacetime transition took 20 seconds. All timings were performed on a 100 MHz Pentium processor.

Spacetime transitions are more costly to generate than transitions generated with joint angle interpolation techniques. They often produce better results, however. One type of motion that demonstrates this superiority is motion that has identical joint space beginning and ending conditions on some of the DOFs of the figure. An example of this type of motion is shown in Figure B.6. This motion begins with the forearm nearly vertical, held close to the shoulder with zero initial velocity. The motion ends with the forearm held horizontal also with zero velocity. Because the upper arm and the wrist have

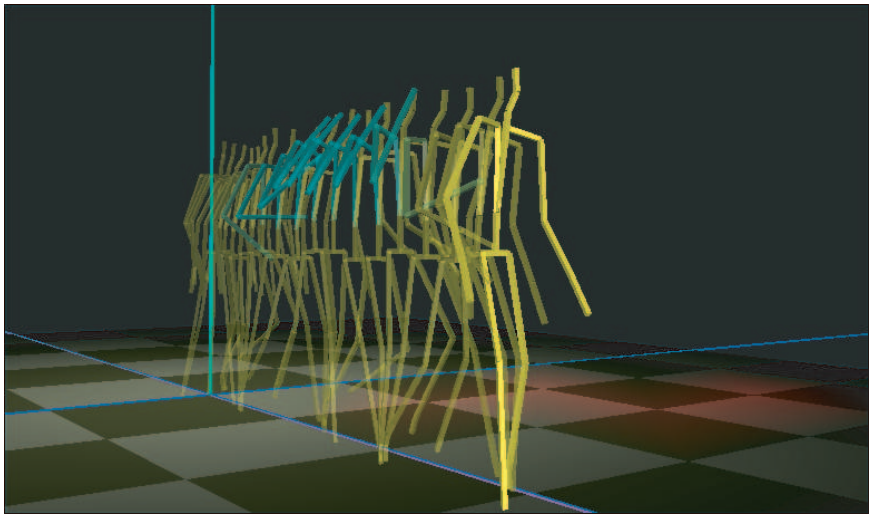

Figure B.5: Arm walk motion transitioning to salute motion and back to walk motion. Arm degrees of freedom affected by the transition are colored green.

identical joint space starting and ending conditions, any simple interpolation technique, which would include linear interpolation, polynomial interpolation, and most other types of interpolation (include those described previously in this thesis) which simply take a weighted sum of the two endpoint conditions will yield a motion such as shown in the left on Figure B.6. This is an unnatural motion since there is no joint space motion at the shoulder or the wrist. The spacetime motion, however, has motion at every joint and looks much more like the kind of motion a person might make.

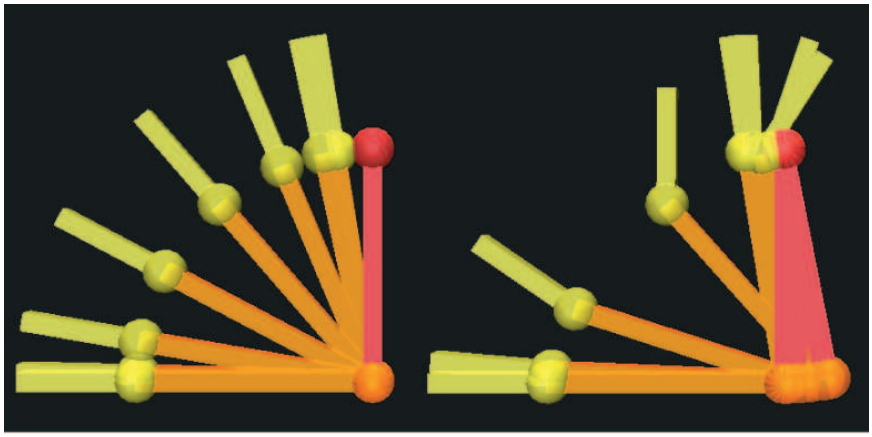

Figure B.6: Joint angle interpolation vs. spacetime optimization

B.2 Equations of dynamics & their derivatives

Constants, symbols, and notation:

$\mathbf{o}_i$	=	origin of the i-th link coordinate frame.
$\mathbf{c}_i$	=	center of mass of the i-th link.
ω_i^i	=	angular velocity of the i-th link .
$\mathbf{z}_i^i$	=	joint axis of the i-thlink expressed in the i-th coordinate frame.
$\mathbf{s}_{i,j}^i$	=	vector from $\mathbf{o}_i$ to $\mathbf{o}_j$ expressed in the i-th coordinate frame.
$\mathbf{r}_{i,j}^i$	=	vector from $\mathbf{o}_i$ to $\mathbf{c}_j$ expressed in the i-th coordinate frame.
$\mathbf{A}_i$	=	3x3 coordinate (or 4x4 homogeneous) transformation relating the i-th coordinate frame to the $(i-1)$-th frame.
$\mathbf{I}_{ci}^k$	=	inertia tensor of the i-th link about $\mathbf{c}_i$ expressed in the k-th coordinate frame.

$\mathbf{J}_{ci}^{k}$ = Euler's inertia tensor of the i-th frame about $\mathbf{c}_i$ expressed in the k-th coordinate frame.

$\mathbf{\Omega}_i^i$ = angular acceleration tensor of the i-th link expressed in the i-th coordinate frame.

$\mathbf{F}_{ci}^{i}$ = force vector acting on $\mathbf{c}_i$ expressed in the i-th coordinate frame.

$\mathbf{M}_{ci}^{i}$ = moment vector about $\mathbf{c}_i$ expressed in the i-th coordinate frame.

$\mathbf{f}_i^i$ = force vector exerted on link i by link $(i-1)$.

η_i^i = moment vector exerted on link i by link $(i-1)$.

τ_i = torque at joint i.

$\mathbf{g}$ = gravity.

m_i = mass of the i-th link.

In the above, the subscript indicates the coordinate frame being represented and superscript the coordinate frame in which it is represented.

We use + and - on index variables to denote relative placement in the joint hierarchy. Thus, i_1 is the predecessor of i which is the predecessor of $i+$. For example, in the equation $\omega_{i+}^{i+} = \mathbf{A}_{i+}^{T}\omega_i^i + \mathbf{z}_{i+}^{i+}\dot{q}_{i+}$, the variable ω_i^i is the angular velocity in the coordinate frame which precedes the coordinate frame of ω_{i+}^{i+}. In other words, coordinate frame i is closer to the root coordinate frame than is frame $i+$. Note that there is no guarantee of a uniquely defined successor.

$$\mathbf{g} = [0.0, -9.80655, 0.0]^T$$

$$\mathbf{J}_{ci}^{i} = \frac{1}{2} trace\left(\mathbf{I}_{ci}^{i}\right) 1 - \mathbf{I}_{ci}^{i}$$

$$
\begin{aligned}
dual(\mathbf{v}) &= \tilde{\mathbf{v}} = \begin{bmatrix} 0 & -\mathbf{v}_3 & \mathbf{v}_2 \\ \mathbf{v}_3 & 0 & -\mathbf{v}_1 \\ -\mathbf{v}_2 & \mathbf{v}_1 & 0 \end{bmatrix} \\
dual(\tilde{\mathbf{v}}) &= \mathbf{v}
\end{aligned}
$$

Forward dynamics equations:

Base conditions at the root of the creature:

$$
\begin{aligned}
\omega_0^0 &= \mathbf{z}_0^0 \dot{q}_0 \\
\dot{\omega}_0^0 &= \mathbf{z}_0^0 \ddot{q}_0 \\
\ddot{\mathbf{s}}_{0,0}^0 &= \mathbf{A}_0^T \mathbf{g}
\end{aligned}
$$

Recursive forward dynamics equations:

$$
\begin{aligned}
\omega_{i+}^{i+} &= \mathbf{A}_{i+}^T \omega_i^i + \mathbf{z}_{i+}^{i+} \dot{q}_{i+} \\
\dot{\omega}_{i+}^{i+} &= \mathbf{A}_{i+}^T \dot{\omega}_i^i + \tilde{\omega}_i^{i+} \mathbf{z}_{i+}^{i+} \dot{q}_{i+} + \mathbf{z}_{i+}^{i+} \ddot{q}_{i+} \\
\mathbf{\Omega}_{i+}^{i+} &= \dot{\tilde{\omega}}_{i+}^{i+} + \tilde{\omega}_{i+}^{i+} \tilde{\omega}_{i+}^{i+} \\
\ddot{\mathbf{s}}_{0,i+}^{i+} &= \mathbf{A}_{i+}^T \left[\ddot{\mathbf{s}}_{0,i}^i + \mathbf{\Omega}_i^i \mathbf{s}_{i,i+}^i \right] \\
\ddot{\mathbf{r}}_{0,i+}^{i+} &= \mathbf{\Omega}_{i+}^{i+} \mathbf{r}_{i+,i+}^{i+} + \ddot{\mathbf{s}}_{0,i+}^{i+} \\
\mathbf{F}_{ci+}^{\ i+} &= m_{i+} \ddot{\mathbf{r}}_{0,i+}^{i+} \\
\tilde{\mathbf{M}}_{ci+}^{\ i+} &= \left(\mathbf{\Omega}_{i+}^{i+} \mathbf{J}_{ci+}^{\ i+} \right) - \left(\mathbf{\Omega}_{i+}^{i+} \mathbf{J}_{ci+}^{\ i+} \right)^T
\end{aligned}
$$

Backward recursive equations (torque equations):

At a joint controlling an end-effector:

$$\begin{aligned} \mathbf{f}_i^i &= \mathbf{F}_{ci}^{\ i} \\ \eta_I^i &= \tilde{\mathbf{r}}_{i,i}^i \mathbf{F}_{ci}^{\ i} + \mathbf{M}_{ci}^{\ i} \\ \tau_i &= \eta_i^i \cdot \mathbf{z}_i^i \end{aligned}$$

At an internal joint:

$$\begin{aligned} \mathbf{f}_i^i &= \mathbf{F}_{ci}^{\ i} + \sum_{i+} \left[\mathbf{A}_{i+} \mathbf{f}_{i+}^{i+} \right] \\ \eta_i^i &= \tilde{\mathbf{r}}_{i,i}^i \mathbf{F}_{ci}^{\ i} + \mathbf{M}_{ci}^{\ i} + \sum_{i+} \left[\mathbf{A}_{i+} \eta_{i+}^{i+} + \tilde{\mathbf{s}}_{i,i+}^i \mathbf{f}_{i+}^i \right] \\ \tau_i &= \eta_i^i \cdot \mathbf{z}_i^i \end{aligned}$$

The energy function & the partials:

$$\begin{aligned} P &= \int_t \sum_i \tau_i^2 dt \\ \frac{\delta P}{\delta \theta_j} &= 2 \int_t \sum_i \frac{\delta \tau_i}{\delta q_j} + \frac{\delta \tau_i}{\delta \dot{q}_j} + \frac{\delta \tau_i}{\delta \ddot{q}_j} \end{aligned}$$

The forward partials & their initial conditions:

$$\begin{aligned}
\frac{\delta\omega_{i+}^{i+}}{\delta q_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \frac{\delta\omega_i^i}{\delta q_j} \\
\frac{\delta\omega_{i+}^{i+}}{\delta q_j} &\underset{i+=j}{=} \frac{\delta\mathbf{A}_j^T}{\delta q_j}\omega_{j-}^{j-}
\end{aligned}$$

$$\begin{aligned}
\frac{\delta\dot{\omega}_{i+}^{i+}}{\delta q_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \frac{\delta\dot{\omega}_i^i}{\delta q_j} + \widetilde{\frac{\delta\omega_i^{i+}}{\delta q_j}}\mathbf{z}_{i+}^{i+}\dot{q}_{i+} \\
\frac{\delta\dot{\omega}_{i+}^{i+}}{\delta q_j} &\underset{i+=j}{=} \frac{\delta\mathbf{A}_j^T}{\delta q_j}\dot{\omega}_{j-}^{j-} + \left(\widetilde{\frac{\delta\mathbf{A}_j^T}{\delta q_j}\omega_{j-}^{j-}}\right)\mathbf{z}_j^j\dot{q}_j
\end{aligned}$$

$$\frac{\delta\boldsymbol{\Omega}_{i+}^{i+}}{\delta q_j} = \widetilde{\frac{\delta\dot{\omega}_{i+}^{i+}}{\delta q_j}} + \widetilde{\frac{\delta\omega_{i+}^{i+}}{\delta q_j}}\tilde{\omega}_{i+}^{i+} + \left(\widetilde{\frac{\delta\omega_{i+}^{i+}}{\delta q_j}}\tilde{\omega}_{i+}^{i+}\right)^T$$

$$\begin{aligned}
\frac{\delta\omega_{i+}^{i+}}{\delta\dot{q}_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \frac{\delta\omega_i^i}{\delta\dot{q}_j} \\
\frac{\delta\omega_{i+}^{i+}}{\delta\dot{q}_j} &\underset{i+=j}{=} \mathbf{z}_j^j
\end{aligned}$$

$$\begin{aligned}
\frac{\delta\dot{\omega}_{i+}^{i+}}{\delta\dot{q}_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \frac{\delta\dot{\omega}_i^i}{\delta\dot{q}_j} + \left(\widetilde{\mathbf{A}_{i+}^T \frac{\delta\omega_i^i}{\delta\dot{q}_j}}\right)\mathbf{z}_{i+}^{i+}\dot{q}_{i+} \\
\frac{\delta\dot{\omega}_{i+}^{i+}}{\delta\dot{q}_j} &\underset{i+=j}{=} \widetilde{\mathbf{A}_j^T\omega_{j-}^{j-}}\mathbf{z}_j^j
\end{aligned}$$

$$\frac{\delta \mathbf{\Omega}_{i+}^{i+}}{\delta \dot{q}_j} = \widetilde{\frac{\delta \dot{\omega}_{i+}^{i+}}{\delta \dot{q}_j}} + \widetilde{\frac{\delta \omega_{i+}^{i+}}{\delta \dot{q}_j}} \tilde{\omega}_{i+}^{i+} + \left(\widetilde{\frac{\delta \omega_{i+}^{i+}}{\delta \dot{q}_j}} \tilde{\omega}_{i+}^{i+} \right)^T$$

$$\frac{\delta \omega_{i+}^{i+}}{\delta \ddot{q}_j} = 0$$

$$\begin{aligned} \frac{\delta \dot{\omega}_{i+}^{i+}}{\delta \ddot{q}_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \frac{\delta \dot{\omega}_i^i}{\delta \ddot{q}_j} \\ \frac{\delta \dot{\omega}_{i+}^{i+}}{\delta \ddot{q}_j} &\underset{i+=j}{=} \mathbf{z}_j^j \end{aligned}$$

$$\frac{\delta \mathbf{\Omega}_{i+}^{i+}}{\delta \ddot{q}_j} = \widetilde{\frac{\delta \dot{\omega}_{i+}^{i+}}{\delta \ddot{q}_j}}$$

$$\begin{aligned} \frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta q_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \left[\frac{\delta \ddot{\mathbf{s}}_{0,i}^i}{\delta q_j} + \frac{\delta \mathbf{\Omega}_i^i}{\delta q_j} \mathbf{s}_{i,i+}^i \right] \\ \frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta q_j} &\underset{i+=j}{=} \frac{\delta \mathbf{A}_j^T}{\delta q_j} \left[\ddot{\mathbf{s}}_{0,j-}^{j-} + \mathbf{\Omega}_{j-}^{j-} \mathbf{s}_{j-,j}^{j-} \right] \end{aligned}$$

$$\begin{aligned} \frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta \dot{q}_j} &\underset{i+>j}{=} \mathbf{A}_{i+}^T \left[\frac{\delta \ddot{\mathbf{s}}_{0,i}^i}{\delta \dot{q}_j} + \frac{\delta \mathbf{\Omega}_i^i}{\delta \dot{q}_j} \mathbf{s}_{i,i+}^i \right] \\ \frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta \dot{q}_j} &\underset{i+=j}{=} 0 \end{aligned}$$

$$\frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta \ddot{q}_j} \underset{i+>j}{=} \mathbf{A}_{i+}^T \left[\frac{\delta \ddot{\mathbf{s}}_{0,i}^{i}}{\delta \ddot{q}_j} + \frac{\delta \boldsymbol{\Omega}_i^i}{\delta \ddot{q}_j} \mathbf{s}_{i,i+}^i \right]$$

$$\frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta \ddot{q}_j} \underset{i+=j}{=} 0$$

$$\frac{\delta \ddot{\mathbf{r}}_{0,i+}^{i+}}{\delta q_j} = \frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta q_j} \mathbf{r}_{i,i+}^{i+} + \frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta q_j}$$

$$\frac{\delta \ddot{\mathbf{r}}_{0,i+}^{i+}}{\delta \dot{q}_j} = \frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta \dot{q}_j} \mathbf{r}_{i,i+}^{i+} + \frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta \dot{q}_j}$$

$$\frac{\delta \ddot{\mathbf{r}}_{0,i+}^{i+}}{\delta \ddot{q}_j} = \frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta \ddot{q}_j} \mathbf{r}_{i,i+}^{i+} + \frac{\delta \ddot{\mathbf{s}}_{0,i+}^{i+}}{\delta \ddot{q}_j}$$

$$\frac{\delta \mathbf{M}_{ci+}^{i+}}{\delta q_j} = \frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta q_j} \mathbf{J}_{ci+}^{\ i+} - \left(\frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta q_j} \mathbf{J}_{ci+}^{\ i+} \right)^T$$

$$\frac{\delta \mathbf{M}_{ci+}^{\ i+}}{\delta \dot{q}_j} = \frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta \dot{q}_j} \mathbf{J}_{ci+}^{\ i+} - \left(\frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta \dot{q}_j} \mathbf{J}_{ci+}^{\ i+} \right)^T$$

$$\frac{\delta \mathbf{M}_{ci+}^{\ i+}}{\delta \ddot{q}_j} = \frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta \ddot{q}_j} \mathbf{J}_{ci+}^{\ i+} - \left(\frac{\delta \boldsymbol{\Omega}_{i+}^{i+}}{\delta \ddot{q}_j} \mathbf{J}_{ci+}^{\ i+} \right)^T$$

$$\frac{\delta \mathbf{F}_{ci+}^{\ i+}}{\delta q_j} = m_{i+} \frac{\delta \ddot{\mathbf{r}}_{0,i+}^{i+}}{\delta q_j}$$

$$\frac{\delta \mathbf{F}_{ci+}^{\ i+}}{\delta \dot{q}_j} = m_{i+} \frac{\delta \ddot{\mathbf{r}}_{0,i+}^{i+}}{\delta \dot{q}_j}$$

$$\frac{\delta \mathbf{F}_{ci+}^{\ i+}}{\delta \ddot{q}_j} = m_{i+} \frac{\delta \ddot{\mathbf{r}}_{0,i+}^{i+}}{\delta \ddot{q}_j}$$

The Reverse Partials:

$$\begin{aligned} \frac{\delta \mathbf{f}_i^i}{\delta q_j} &\underset{\exists i+}{=} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta q_j} + \sum_{i+} \left[\frac{\delta \mathbf{A}_{i+}}{\delta q_j} \mathbf{f}_{i+}^{i+} + \mathbf{A}_{i+} \frac{\delta \mathbf{f}_i^i}{\delta q_j} \right] \\ \frac{\delta \mathbf{f}_i^i}{\delta q_j} &\underset{\nexists i+}{=} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta q_j} \end{aligned}$$

$$\begin{aligned} \frac{\delta \mathbf{f}_i^i}{\delta \dot{q}_j} &\underset{\exists i+}{=} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \dot{q}_j} + \sum_{i+} \left[\mathbf{A}_{i+} \frac{\delta \mathbf{f}_i^i}{\delta \dot{q}_j} \right] \\ \frac{\delta \mathbf{f}_i^i}{\delta \dot{q}_j} &\underset{\nexists i+}{=} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \dot{q}_j} \end{aligned}$$

$$\begin{aligned} \frac{\delta \mathbf{f}_i^i}{\delta \ddot{q}_j} &\underset{\exists i+}{=} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \ddot{q}_j} + \sum_{i+} \left[\mathbf{A}_{i+} \frac{\delta \mathbf{f}_i^i}{\delta \ddot{q}_j} \right] \\ \frac{\delta \mathbf{f}_i^i}{\delta \ddot{q}_j} &\underset{\nexists i+}{=} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \ddot{q}_j} \end{aligned}$$

$$
\begin{aligned}
\frac{\delta \eta_i^i}{\delta q_j} &\underset{\exists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta q_j} + \frac{\delta \widetilde{\mathbf{M}_{ci}^{\ i}}}{\delta q_j} + \\
&\qquad \sum_{i+} \left[\frac{\delta \mathbf{A}_{i+}}{\delta q_j} \eta_{i+}^{i+} + \mathbf{A}_{i+} \frac{\delta \eta_{i+}^{i+}}{\delta q_j} + \widetilde{\mathbf{s}_{i,i+}^i} \frac{\delta \mathbf{A}_{i+}}{\delta q_j} \mathbf{f}_{i+}^{i+} + \widetilde{\mathbf{s}_{i,i+}^i} \mathbf{A}_{i+} \frac{\delta \mathbf{f}_{i+}^{i+}}{\delta q_j} \right] \\
\frac{\delta \eta_i^i}{\delta q_j} &\underset{\nexists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta q_j} + \frac{\delta \widetilde{\mathbf{M}_{ci}^{\ i}}}{\delta q_j}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\delta \eta_i^i}{\delta \dot{q}_j} &\underset{\exists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \dot{q}_j} + \frac{\delta \widetilde{\mathbf{M}_{ci}^{\ i}}}{\delta \dot{q}_j} + \sum_{i+} \left[\mathbf{A}_{i+} \frac{\delta \eta_{i+}^{i+}}{\delta \dot{q}_j} + \widetilde{\mathbf{s}_{i,i+}^i} \mathbf{A}_{i+} \frac{\delta \mathbf{f}_{i+}^{i+}}{\delta \dot{q}_j} \right] \\
\frac{\delta \eta_i^i}{\delta \dot{q}_j} &\underset{\nexists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \dot{q}_j} + \frac{\delta \widetilde{\mathbf{M}_{ci}^{\ i}}}{\delta \dot{q}_j}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\delta \eta_i^i}{\delta \ddot{q}_j} &\underset{\exists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \ddot{q}_j} + \frac{\delta \widetilde{\mathbf{M}_{ci}^{\ i}}}{\delta \ddot{q}_j} + \sum_{i+} \left[\mathbf{A}_{i+} \frac{\delta \eta_{i+}^{i+}}{\delta \ddot{q}_j} + \widetilde{\mathbf{s}_{i,i+}^i} \mathbf{A}_{i+} \frac{\delta \mathbf{f}_{i+}^{i+}}{\delta \ddot{q}_j} \right] \\
\frac{\delta \eta_i^i}{\delta \ddot{q}_j} &\underset{\nexists i+}{=} \widetilde{\mathbf{r}_{i,i+}^i} \frac{\delta \mathbf{F}_{ci}^{\ i}}{\delta \ddot{q}_j} + \frac{\delta \widetilde{\mathbf{M}_{ci}^{\ i}}}{\delta \ddot{q}_j}
\end{aligned}
$$

$$
\frac{\delta \tau_i}{\delta q_j} = \frac{\delta \eta_i^i}{\delta q_j} \cdot \mathbf{z}_i^i
$$

$$
\frac{\delta \tau_i}{\delta \dot{q}_j} = \frac{\delta \eta_i^i}{\delta \dot{q}_j} \cdot \mathbf{z}_i^i
$$

$$
\frac{\delta \tau_i}{\delta \ddot{q}_j} = \frac{\delta \eta_i^i}{\delta \ddot{q}_j} \cdot \mathbf{z}_i^i
$$

Bibliography

[1] Allbeck, J. M., and Badler, N. I. Avatars á la snow crash. In *Computer Animation '98* (June 1998), pp. 19–24.

[2] Amaya, K., Bruderlin, A., and Calvert, T. Emotion from motion. In *Graphics Interface '96* (May 1996), W. A. Davis and R. Bartels, Eds., pp. 222–229.

[3] Arad, N., Dyn, N., Reisfeld, D., and Yeshurun, Y. Image warping by radial basis functions: Application to facial expressions. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing 56*, 2 (Mar. 1994), 161–172.

[4] Auslander, J., Fukunaga, A., Partovi, H., Christensen, J., Hsu, L., Reiss, P., Shuman, A., Marks, J., and Ngo, J. T. Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Trans. Gr. 13*, 4 (Oct. 1995), 311–336.

[5] Badler, N. Virtual humans. In *Virtual Humans: Behaviors and Physics, Acting and Reacting*. SIGGRAPH, July 1998. SIGGRAPH '98 Course Note Series.

[6] Badler, N. I., Hollick, M. J., and Granieri, J. P. Real-time control of a virtual human using minimal sensors. *Presence 2*, 1 (1993), 82–86.

[7] Badler, N. I., Phillips, C. B., and Webber, B. L. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, 1993.

[8] Balafoutis, C. A., and Patel, R. V. *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach*. Kluwer Academic Publishers, 1991.

[9] Baraff, D. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Computer Graphics* (July 1989), pp. 223–232. Proceedings of SIGGRAPH 89.

[10] Baraff, D. Curved surfaces and coherence for non-penetrating rigid body simulation. In *Computer Graphics* (Aug. 1990), pp. 19–28. Proceedings of SIGGRAPH 90.

[11] Baraff, D. Coping with friction for non-penetrating rigid body simulation. In *Computer Graphics* (Aug. 1991), pp. 31–40. Proceedings of SIGGRAPH 91.

[12] Baraff, D. Dynamic simulation of non-penetrating flexible bodies. In *Computer Graphics* (July 1992), pp. 303–308. Proceedings of SIGGRAPH 92.

[13] Baraff, D. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics* (July 1994), pp. 23–34. Proceedings of SIGGRAPH 94.

[14] Baraff, D. Linear-time dynamics using lagrange multipliers. In *Computer Graphics* (Aug. 1995), pp. 137–146. Proceedings of SIGGRAPH 96.

[15] Baraff, D., and Witkin, A. Large steps in cloth simulation. In *Computer Graphics* (July 1998), pp. 43–54. Proceedings of SIGGRAPH 98.

[16] Barr, A. H., Currin, B., Gabriel, S., and Huges, J. F. Smooth interpolation of orientations with angular constraints using quaternions. In *Computer Graphics* (July 1992), pp. 313–320. Proceedings of SIGGRAPH 92.

[17] Bartenieff, I., and Lewis, D. *Body Movement: Coping with the Environment*. Gordon and Breach Publishers, 1980.

[18] Barzel, R., Hughes, J. F., and Wood, D. N. Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation '96* (Sept. 1996), pp. 184–197. Proceedings of the 1996 Eurographics Workshop on Animation.

[19] Bizzi, E., Accornero, N., Chapple, W., and Hogan, N. Posture control and trajectory formation during arm movement. *The Journal of Neuroscience 4* (1984), 2738–2744.

[20] Blumberg, B. M., and Galyean, T. A. Multi-level direction of autonomous creatures for real-time virtual environments. In *Computer Graphics* (Aug. 1995), pp. 47–54. Proceedings of SIGGRAPH 95.

[21] Bodenheimer, B., Rose, C. F., Rosenthal, S., and Pella, J. The process of motion capture: Dealing with the data. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation* (Sept. 1997), pp. 3–18.

[22] Boulic, R., and Thalmann, D. Combined direct and inverse kinematic control for articulated figure motion editing. In *Siggraph 93 Course*

80: Recent Techniques in Human Modeling, Animation, and Rendering (1993).

[23] Boulic, R., and Thalmann, D. Position control of the center of mass for articulated figures in multiple support. In *Proceedings of the 5th EuroGraphics Workshop on Animation and Simulation* (Sept. 1995), pp. 130–143.

[24] Bruderlin, A., and Calvert, T. Knowledge-driven, interactive animation of human running. In *Virtual Humans: Behaviors and Physics, Acting and Reacting*. SIGGRAPH, Aug. 1997. SIGGRAPH '97 Course Note Series.

[25] Bruderlin, A., and Calvert, T. W. Goal-directed, dynamic animation of human walking. In *Computer Graphics* (July 1989), pp. 233–242. Proceedings of SIGGRAPH 89.

[26] Bruderlin, A., and Williams, L. Motion signal processing. In *Computer Graphics* (Aug. 1995), pp. 97–104. Proceedings of SIGGRAPH 95.

[27] Burdett, R. G., Skrinar, G. S., and Simon, S. R. Comparison of mechanical work and metabolic energy consumption during normal gait. *Journal of Orhopaedic Research 1*, 1 (1983), 63–72.

[28] Capin, T., Jovovic, M., Esmerado, J., Aubel, A., and Thalmann, D. Efficient network transmission of virtual human bodies. In *Computer Animation '98* (June 1998), pp. 41–48.

[29] Carlson, D. A., and Hodgins, J. K. Simulation levels of detail for real-time animation. In *Graphics Interface '97* (1997), pp. 1–8.

[30] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Doubille, B., Prevost, S., and Stone, M. Animated conversation: Rule-based generation of facial expression, gesture, & spoken intonation for multiple conversational agents. In *Computer Graphics* (July 1994), pp. 413–420. Proceedings of SIGGRAPH 94.

[31] Chadwick, J. E., Haumann, D. R., and Parent, R. E. Layered construction for deformable animated characters. In *Computer Graphics* (July 1989), pp. 243–252. Proceedings of SIGGRAPH 89.

[32] Chen, D. T., and Zeltzer, D. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *Computer Graphics* (July 1992), pp. 89–98. Proceedings of SIGGRAPH 92.

[33] Chenney, S., and Forsyth, D. View-dependent culling of dynamic systems in virtual environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Apr. 1997), pp. 55–58.

[34] Cohen, M. F. Gracefulness and style in motion control. In *Proceedings of the Workshop on the Mechanics, Control, and Animation of Articulated Figures* (Apr. 1989).

[35] Cohen, M. F. Interactive spacetime control for animation. In *Computer Graphics* (July 1992), pp. 293–302. Proceedings of SIGGRAPH 92.

[36] Cremer, J., Kearney, J., and Papelis, Y. H.c.s.m.: A framework for behavior and scenario control in virtual environments. *ACM Transactions on Modeling and Computer Simulation 5* (1995), 242–267.

[37] da Silva, F. W. S. V., Velho, L., Cavalcanti, P. R., and de Miranda Gomes, J. A new interface paradigm for motion capture based animation systems. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation* (1997), pp. 19–36.

[38] Dean, J., and Brüwer, M. Control of human arm movements in two dimensions: Paths and joint control in avoiding simple linear obstacles. *Experimental Brain Research 97* (1994), 497–514.

[39] DeCarlo, D., Metaxas, D., and Stone, M. An antropometric face model using variational techniques. In *Computer Graphics* (July 1998), pp. 67–74. Proceedings of SIGGRAPH 98.

[40] Dempster, W. T., and Gaughran, G. R. L. Properties of body segments based on size and weight. *American Journal of Anatomy 120* (1965), 33–54.

[41] Denavit, J., and Hartenberg, R. S. A kinematic notation for lower-pair mechanisms based on matricies. *ASME Journal of Applied Mechanics 23* (1955), 215–221.

[42] Duff, T. Quaternion splines for animating orientation. In *Proceedings of the Monterey Computer Graphics Workshop* (Dec. 1985), pp. 54–62.

[43] Earnshaw, R., Thalmann, N. M., Terzopoulos, D., and Thalmann, D. Computer animation for virtual humans. *IEEE Computer Graphics and Applications 18*, 5 (1998), 20–23.

[44] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., and Worley, S. *Texturing and Modeling.* AP Profefessional, 1998.

[45] Eng, J. J., Winter, D. A., MacKinnon, C. D., and Patla, A. E. Interaction of the reactive moments and center of mass displacement for postural control during voluntary arm movements. *Neuroscience Research Communications 2* (1992), 73–80.

[46] Faure, F., and Debunne, G. Dynamic analysis of human walking. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation* (1997), pp. 54–65.

[47] Featherstone, R. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, 1987.

[48] Finkelstein, A., Jacobs, C. E., and Salesin, D. H. Multiresolution video. In *Computer Graphics* (Aug. 1995), pp. 281–290. Proceedings of SIGGRAPH 96.

[49] Flash, T., and Hogan, N. The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience 5* (1985), 1688–1703.

[50] Forssberg, H., and Hirschfeld, H. Postural adjustments in sitting humans following external perturbations: Muscle activity and kinematics. *Experimental Brain Research 97* (1994), 515–527.

[51] Gascuel, M.-P. An implicit formulation for precise contact modeling between flexible solids. In *Computer Graphics* (Aug. 1993), pp. 313–320. Proceedings of SIGGRAPH 93.

[52] Gelfand, J., Flax, M., Endres, R., Lane, S., and Handelman, D. Acquisition of automatic activity through practice: Changes in sensory input. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (July 1992), AIII Press.

[53] Gill, P. E., Murray, W., and Wright, M. H. *Practical Optimization*. Academic Press, 1981.

[54] Girard, M., and Maciejewski, A. A. Computational modeling for the computer animation of legged figures. In *Computer Graphics* (July 1985), pp. 263–270. Proceedings of SIGGRAPH 85.

[55] Gleicher, M. Motion editing with spacetime constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Apr. 1997), pp. 139–148.

[56] Gleicher, M. Retargetting motion to new characters. In *Computer Graphics* (July 1998), pp. 33–42. Proceedings of SIGGRAPH 98.

[57] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[58] Gortler, S. J., Schröder, P., Cohen, M. F., and Hanrahan, P. Wavelet radiosity. In *Computer Graphics* (Aug. 1993), pp. 221–230. Proceedings of SIGGRAPH 93.

[59] Grzeszczuk, R., and Terzopoulos, D. Automated learning of muscle-actuated locomotion through control abstraction. In *Computer Graphics* (Aug. 1995), pp. 63–70. Proceedings of SIGGRAPH 95.

[60] Grzeszczuk, R., Terzopoulos, D., and Hinton, G. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Computer Graphics* (July 1998), pp. 9–20. Proceedings of SIGGRAPH 98.

[61] Guenter, B., Grimm, C., Wood, D., Malvar, H., and Pighin, F. Making faces. In *Computer Graphics* (July 1998), pp. 55–66. Proceedings of SIGGRAPH 98.

[62] Gullapalli, V., Gelfand, J. J., and Lane, S. H. Synergy-based learning of hybrid position/force control for redundant manipulators. In *Proceedings of IEEE Robotics and Automation Conference, Minneapolis MN* (June 1996), IEEE Press, Piscataway, NJ, pp. 3526–3531.

[63] Guo, S., and Robergé, J. A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Computer Animation and Simulation '96* (Sept. 1996), pp. 80–107. Proceedings of the 1996 Eurographics Worshop on Animation.

[64] Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. *Robust Statistics: The Approach Based on Influence Functions*. John H. Wiley, 1986.

[65] Handelman, D. A., and Lane, S. H. Human-to-machine skill transfer through cooperative learning. Submitted to Intelligent Control Systems, IEEE Press.

[66] Hars, A. Masters of motion. *Computer Graphics World* (Oct. 1996), 26–34.

[67] Hodgins, J. K., O'Brien, J. F., and Tumblin, J. Do geometric models affect judgements of human motion? In *Graphics Interface '97* (1997), pp. 17–25.

[68] Hodgins, J. K., and Pollard, N. S. Adapting simulated behaviors for new creatures. In *Computer Graphics* (Aug. 1997), pp. 153–162. Proceedings of SIGGRAPH 97.

[69] Hodgins, J. K., and Raibert, M. H. Biped gymnastics. *The International Journal of Robotics Research 9* (1990), 115–132.

[70] Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. Automated learning of muscle-actuated locomotion through control abstraction. In *Computer Graphics* (Aug. 1995), pp. 71–78. Proceedings of SIGGRAPH 95.

[71] Hogan, N. An organizing principle for a class of voluntary movements. *The Journal of Neuroscience 4* (1984), 2745–2754.

[72] Hogan, N. Control strategies for complex movements derived from physical systems theory. *International Symposium on Synergetics* (May 1985).

[73] Hogan, N. Impedance control: An approach to manipulation. *Journal of Dynamic Systems, Measurement, and Control 107* (1985), 1–24.

[74] Hogan, N. The mechanics of multi-joint posture and movement control. *Biological Cybernetics 52* (1985), 315–331.

[75] Holland, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[76] Hollerbach, J. M. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics SMC-10*, 11 (Nov. 1980).

[77] Houy, D. R. Range of motion in college males. Presented at the Conference of the Human Factors Society, Santa Monica, CA, 1983.

[78] Huang, P. S., and van de Panne, M. A planning algorithm for dynamic motions. In *Computer Animation and Simulation '96* (Sept. 1996), pp. 169–182. Proceedings of the 1996 Eurographics Worshop on Animation.

[79] Isaacs, P. M., and Cohen, M. F. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. In *Computer Graphics* (July 1987), pp. 215–224. Proceedings of SIGGRAPH 87.

[80] Koga, Y., Kondo, K., Kuffner, J., and Latombe, J.-C. Planning motions with intentions. In *Computer Graphics* (July 1994), pp. 395–408. Proceedings of SIGGRAPH 94.

[81] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[82] Koza, J. R. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, 1994.

[83] Lamouret, A., and van de Panne, M. Motion synthesis by example. In *Computer Animation and Simulation '96* (Sept. 1996), pp. 199–212. Proceedings of the 1996 Eurographics Workshop on Animation.

[84] Lane, S. H., and Gelfand, J. J. *Modulation of Robotic Motor Synergies Using Reinforcement Learning Optimization.* Kluwer Academic Publisher, 1992, pp. 521–538. G. Bekey and K. Goldberg, eds.

[85] Lasseter, J. Principles of traditional animation applied to 3d computer animation. In *Computer Graphics* (July 1987), pp. 35–44. Proceedings of SIGGRAPH 87.

[86] Laszlo, J., van de Panne, M., and Fiume, E. Limit cycle control and its application to the animation of balancing and walking. In *Computer Graphics* (Aug. 1995), pp. 155–162. Proceedings of SIGGRAPH 96.

[87] Latombe, J.-C. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[88] Lee, P., Wei, S., Zhao, J., and Badler, N. I. Strength guided motion. In *Computer Graphics* (Aug. 1990), pp. 253–262. Proceedings of SIGGRAPH 90.

[89] Liu, Z., and Cohen, M. F. An efficient symbolic interface to constraint based animation systems. In *Proceedings of the 5th EuroGraphics Workshop on Animation and Simulation* (Sept. 1995).

[90] Liu, Z., and Cohen, M. F. Keyframe motion optimization by relaxing speed and timing. In *Proceedings of the 5th EuroGraphics Workshop on Animation and Simulation* (Sept. 1995).

[91] Liu, Z., Gortler, S. J., and Cohen, M. F. Hierarchical spacetime control. In *Computer Graphics* (July 1994), pp. 35–42. Proceedings of SIGGRAPH 94.

[92] Maciejewski, A. A., and Reagin, J. M. A parallel algorithm and architecture for the control of kinematically redundant manipulators. *IEEE Transactions on Robotics and Automation 10* (1994), 405–414.

[93] Maestri, G. Capturing motion. *Computer Graphics World* (1995), 47–51.

[94] Maestri, G. *[Digital] Character Animation.* New Riders Publishing, 1996.

[95] Magnenat-Thalmann, N., and Thalmann, D. Complex models for animating synthetic actors. *IEEE Computer Graphics & Applications* (Sept. 1991).

[96] Maiocchi, R. 3-D character animation using motion capture. In *Interactive Computer Animation*, N. Magnetat-Thalmann and D. Thalmann, Eds. Prentice-Hall, London, 1996, pp. 10–39.

[97] Maurel, W., Thalmann, D., Hoffmeyer, P., Beylot, P., Gingins, P., Kalra, P., and Thalman, N. M. A biomechanical musculoskeletal model of human upper limb for dynamic simulation. In *Computer Animation and Simulation '96* (Sept. 1996), pp. 121–136. Proceedings of the 1996 Eurographics Workshop on Animation.

[98] McGeer, T. Passive bipedal running. *Proceedings of the Royal Society of London 240* (1990), 107–134.

[99] McGeer, T. Dynamics and control of bipedal locomotion. *Journal of Theoretical Biology 163* (1993), 277–314.

[100] McKenna, M., and Zeltzer, D. Dynamic simulation of autonomous legged locomotion. In *Computer Graphics* (Aug. 1990), pp. 29–38. Proceedings of SIGGRAPH 90.

[101] Metaxas, D. Articulated figure dynamics, behavior, and control. In *Virtual Humans: Behaviors and Physics, Acting and Reacting.* SIGGRAPH, Aug. 1997. SIGGRAPH '97 Course Note Series.

[102] Metaxas, D., and Terzopoulos, D. Dynamic deformation of solid primitives with constraints. In *Computer Graphics* (July 1992), pp. 309–312. Proceedings of SIGGRAPH 92.

[103] Micchelli, C. A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation 2* (1986).

[104] Milenkovic, V. J. Position-based physics: Simulating the motion of many highly interacting spheres and polyhedra. In *Computer Graphics* (Aug. 1996), pp. 129–136. Proceedings of SIGGRAPH 96.

[105] Molet, T., Boulic, R., and Thalmann, D. A real time anatomical converter for human motion capture. In *Computer Animation and Simulation '96* (Sept. 1996), pp. 79–94. Proceedings of the 1996 Eurographics Workshop on Animation.

[106] Mussa-Ivaldi, F. A., Hogan, N., and Bizzi, E. Neural, mechanical, and geometric factors subserving arm posture in humans. *The Journal of Neuroscience 5* (1985), 2732–2743.

[107] Mussa-Ivaldi, F. A., Morasso, P., and Zaccaria, R. Kinematic networks: A distributed model for representing and regularizing motor redundancy. *Biological Cybernetics 60* (1988), 1–16.

[108] Musse, S., and Thalmann, D. A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation* (1997), pp. 39–51.

[109] Nedel, L. P., and Thalmann, D. Modeling and deformation of the human body using an anatomically-based approach. In *Computer Animation '98* (June 1998), pp. 34–40.

[110] Ngo, J. T., and Marks, J. Spacetime constraints revisited. In *Computer Graphics* (Aug. 1993), pp. 343–350. Proceedings of SIGGRAPH 93.

[111] Pentland, A., and Williams, J. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics* (July 1989), pp. 215–222. Proceedings of SIGGRAPH 89.

[112] Perlin, K. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics 1*, 1 (Mar. 1995), 5–15.

[113] Perlin, K., and Goldberg, A. Improv: A system for scripting interactive actors in virtual worlds. In *Computer Graphics* (Aug. 1996), pp. 205–216. Proceedings of SIGGRAPH 96.

[114] Phillips, C. B., and Badler, N. I. Interactive behaviors for bipedal articulated figures. In *Computer Graphics* (Aug. 1991), pp. 359–362. Proceedings of SIGGRAPH 91.

[115] Pighin, F., Hecker, J., Laschinski, D., Szeliski, R., and Salesin, D. H. Synthesizing realistic facial expressions from photographs. In *Computer Graphics* (July 1998), pp. 75–84. Proceedings of SIGGRAPH 98.

[116] Potel, M. J. On the trail of the shadow woman: The mystery of motion capture. *IEEE Computer Graphics and Applications 18*, 5 (Sept. 1998), 14–19.

[117] Powell, M. J. D. Radial basis functions for multivariable interpolation: A review. In *Algorithms for Approximation*, J. C. Mason and M. G. Cox, Eds. Oxford University Press, Oxford, UK, 1987, pp. 143–167.

[118] Raibert, M. H. *Legged Robots that Balance*. The MIT Press, 1986.

[119] Raibert, M. H., and Hodgins, J. K. Animation of dynamic legged locomotion. In *Computer Graphics* (Aug. 1991), pp. 349–348. Proceedings of SIGGRAPH 91.

[120] Reynolds, C. W. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics* (July 1987), pp. 25–34. Proceedings of SIGGRAPH 87.

[121] Reynolds, C. W. Competition, coevolution, and the game of tag. In *Artificial Life IV: Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems* (1994), pp. 59–69.

[122] Reynolds, C. W. Evolution of corridor following behavior in a noisy world. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior* (1994), pp. 402–410.

[123] Rijpkema, H., and Girard, M. Computer animation of knowledge-based human grasping. In *Computer Graphics* (Aug. 1991), pp. 339–348. Proceedings of SIGGRAPH 91.

[124] Rose, C. F., Guenter, B., Bodenheimer, B., and Cohen, M. Efficient generation of motion transitions using spacetime constraints. In *Computer Graphics* (Aug. 1996), pp. 147–154. Proceedings of SIGGRAPH 96.

[125] Rose, J., and Gamble, J. G. *Human Walking*. Williams & Wilkins, 1994.

[126] Ruprecht, D., and Müller, H. Image warping with scattered data interpolation. *IEEE Computer Graphics and Applications 15*, 2 (Mar. 1995), 37–43.

[127] Scheepers, F., Parent, R. E., Carlson, W. E., and May, S. F. Anatomy-based modeling of the human musculature. In *Computer Graphics* (Aug. 1997), pp. 163–172. Proceedings of SIGGRAPH 97.

[128] Schröder, P., and Zeltzer, D. The virtual erector set: Dynamic simulation with linear recursive constraint propagation. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics* (Mar. 1990).

[129] Shoemake, K. Animating rotation with quaternion curves. In *Computer Graphics* (July 1985), pp. 245–254. Proceedings of SIGGRAPH 85.

[130] Shoemake, K., and Duff, T. Matrix animation and polar decomposition. In *Graphics Interface* (1992), pp. 258–264. Proceedings of Graphics Interface '92.

[131] Sims, K. Evolving 3d morphology and behavior by competition. In *Artificial Life IV Proceedings* (1994), pp. 28–39.

[132] Sims, K. Evolving virtual creatures. In *Computer Graphics* (Aug. 1994), pp. 15–22. Proceedings of SIGGRAPH 94.

[133] Steketee, S. N., and Badler, N. I. Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control. In *Computer Graphics* (July 1985), pp. 255–262. Proceedings of SIGGRAPH 85.

[134] Stephenson, N. *Snow Crash.* look it up and check publication year at home, 1994.

[135] Terzopoulos, D., Rabie, T., and Grzeszczuk, R. Perception and learning in artificial animals. In *Artificial Life V: Proceedings of the Fifth International Conference on the Synthesis and Simulation of Living Systems* (1996).

[136] Terzopoulos, D., Tu, X., and Grzeszczuk, R. Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life 1* (1994), 327–351.

[137] Thalmann, N. M., and Thalmann, D. *Synthetic Actors in Computer-Generated 3D Films.* Springer-Verlag, Berlin, 1990.

[138] Thomas, F., and Johnston, O. *Disney Animation– The Illusion of Life.* Abbeville Press, New York, 1981.

[139] Tu, X., and Terzopoulos, D. Artificial fishes: Physics, locomotion, perception, and behavior. In *Computer Graphics* (July 1994), pp. 43–50. Proceedings of SIGGRAPH 94.

[140] Unuma, M., Anjyo, K., and Tekeuchi, R. Fourier principles for emotion-based human figure animation. In *Computer Graphics* (Aug. 1995), pp. 91–96. Proceedings of SIGGRAPH 95.

[141] van de Panne, M., and Fiume, E. Sensor-actuator networks. In *Computer Graphics* (Aug. 1993), pp. 335–342. Proceedings of SIGGRAPH 93.

[142] van de Panne, M., Fiume, E., and Vranesic, Z. Reusable motion synthesis using state-space controllers. In *Computer Graphics* (Aug. 1990), pp. 225–234. Proceedings of SIGGRAPH 90.

[143] Watt, A., and Watt, M. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison-Wesley / ACM Press, 1992.

[144] Wiley, D. J., and Hahn, J. K. Interpolation synthesis for articulated figure motion. In *Proceedings of the Virtual Reality Annual International Symposium* (Mar. 1997), IEEE Computer Society Press, pp. 157–160.

[145] Wilhelms, J., and Gelder, A. V. Anatomically based modeling. In *Computer Graphics* (Aug. 1997), pp. 173–180. Proceedings of SIGGRAPH 97.

[146] Witkin, A., Gleicher, M., and Welch, W. Interactive dynamics. *Computer Graphics* (Mar. 1990).

[147] Witkin, A., and Kass, M. Spacetime constraints. In *Computer Graphics* (Aug. 1988), pp. 159–168. Proceedings of SIGGRAPH 88.

[148] Witkin, A., and Popović, Z. Motion warping. In *Computer Graphics* (Aug. 1995), pp. 105–108. Proceedings of SIGGRAPH 95.

[149] Witkin, A., and Welch, W. Fast animation and control of nonrigid structures. In *Computer Graphics* (Aug. 1990), pp. 243–252. Proceedings of SIGGRAPH 90.

[150] Zhao, X., Tolani, D., Ting, B.-J., and Badler, N. I. Simulating human movements using optimal control. In *Computer Animation and Simulation '96* (Sept. 1996), pp. 109–120. Proceedings of the 1996 Eurographics Workshop on Animation.