Motion Transformation by Physically Based Spacetime Optimization

Zoran Popović

June 24, 1999

CMU-CS-99-106

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:
Andrew Witkin (chair)
David Baraff
Paul Heckbert
Matthew Mason
Michael Cohen, Microsoft Research

This research was sponsored by the Schlumberger Foundation Fellowship and National Science Foundation award IRI9502464. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors.

Keywords: computer animation, motion editing, spacetime constraints, motion capture, constrained optimization

ABSTRACT

Automatic generation of realistic human motion has been a long-standing problem in computer graphics. This thesis introduces a novel algorithm for transforming character animation sequences that preserves essential physical properties of the motion. The algorithm maintains realism of the original motion sequence without sacrificing the full user control of the editing process. We use the *spacetime constraints* dynamics formulation to manipulate the motion sequence.

In contrast to most physically based animation techniques that synthesize motion from scratch, we take the approach of *motion transformation* as the underlying paradigm for generating computer animations. In doing so, we combine the expressive richness of the input animation sequence with the controllability of spacetime optimization to create a wide range of realistic character animations. The spacetime dynamics formulation also allows editing of intuitive high-level motion concepts such as the time and placement of footprints, length and mass of various extremities, joint arrangement and gravity.

Our algorithm permits the reuse of highly-detailed captured motion animations. In addition, we describe a new methodology for mapping a motion to/from characters with drastically different number of degrees of freedom. We use this method to reduce the complexity of the spacetime optimization problems. Furthermore, our approach provides a paradigm for controlling complex dynamic and kinematic systems with simpler ones.

DEDICATION

To my parents, Branko and Ljiljana Popović

4

ACKNOWLEDGMENTS

First, I thank my advisor, Andy Witkin. Andy has been a mentor and a collaborator, and most of all a believer in the hard problem I have picked for my thesis, and in my abilities to solve it. He has thought me how to pick hard and interesting problems, how to think of the solution in a mathematically sound manner, and how to write coherent descriptions of my results. These three skills are the most important things I have learned in graduate school, and are the foundation of my future academic career.

My other four committee members have also been very helpful. David Baraff has spent a number of hours in front of the white board debating various aspects of constrained optimization with me. Many great ideas came out of those discussions. Paul Heckbert has been extremely helpful in the thesis writing stage. His detailed comments were instrumental in making this thesis a more complete and more understandable document (all remaining shortcoming are my fault). Upon Andy and David's departure for Pixar, Paul has in large part filled the role of an advisor. Matt Mason's doubts about my approach served as a great motivating challenge in the formative days of my thesis. His good, tough questions enabled me to think critically of my work and have made it stronger as a result. Michael Cohen has been a wonderful external committee member. He has followed my work from the very early stages and encouraged me along the way. In my final days at CMU I had great pleasure of spending time with Steve Seitz who was extremely helpful with my trials and tribulations about crafting the plans for the life after my Ph.D.

My graduate school experience would not have been the same without the Animation Lab and many of it's denizens: Will Welch, Michael Gleicher, Sebastian Grassia, Michael Garland, Andrew Wilmott, Tom Kang, Ari Rapkin, Jeff Smith. Special thanks also go to my officemates over the years, including Rich Goodwin, Jim Blythe, Dario Salvucci and Phillip Wickline.

I would also like to thank AOA Inc. for letting me use their optical motion capture equipment. I am also greatfull to my funders: Schlumberger Foundation and National Science Foundation.

* * *

On a more personal level, I want to thank my closest friends during my years at CMU: Justin Boyan, Jurgen Dingel, Joseph O'Sullivan, Xavier Pierron and Jovan Popović. I could simply not imagine surviving a number of my personal struggles during graduate school without their friendship. The entire graduate experience would also not be nearly as much fun without the moments of elation we have shared over the years. I would especially like to thank my brother Jovan who has perhaps seen the best and the worst of me during the precious time we lived together in Pittsburgh.

Dancing also kept me afloat during the past year, when my life was virtually subsumed by my thesis work. I would like to thank my dance partners for keeping me sane during those days: Susie Greco, Lynn Baumeister and Laurie Chern.

Finally, I would like to thank my parents, Branko and Ljiljana Popović for their love, selfless support and the value system they instilled in me early on. I devote this document to them.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Computer based methods have been used with great success in animation as well as to solve the motion generation problems occurring in robotics and biomechanics. For example, computers have become indispensable for menial tasks of animation in-betweening. Keyframing has become a *de facto* creative media for artist animators. Rigid body and cloth simulations have added realism to the secondary motion of animated characters and their environment.

There are, however, some animation problems where automatic methods fall far short. This work addresses one such problem — *realistic character animation*. Although people are generally quick at visually perceiving intricate subtleties of animal motion in nature, our perceptual understanding of natural movement helps us little with generating such motion. Synthesizing and analyzing high-quality motion of dynamic three-dimensional articulated characters proves to be an extremely difficult problem. The collective knowledge of biomechanics, control theory, robot path planning and computer animation indicates that the underlying processes that govern motion are complex and hard to control. This thesis deals with the problem of generating both controllable and realistic character animations.

For an animation to appear realistic, the motion of a character needs to be consistent with the laws of physics. This alone, however, is not sufficient. The entire musculo-skeletal structure of the character must be taken into account because the motion would not look natural otherwise. The ability to control this involved process adds further difficulties. We present a solution to the problem of generating both controllable and realistic character animations. Instead of motion synthesis, we take the approach of motion transformation. For example, we transform a human running sequence by restricting the range of motion for a knee joint to obtain a realistic run with a limp.

Any motion which in principle obeys Newton's Laws, such as a captured motion sequence or the result of a physical simulation, can be used as input to our transformation algorithm. The first step of our algorithm constructs a simplified character model and constructs the dynamics representation of motion which gives rise to an animation sequence which closely matches the input data. This dynamics representation includes the body's mass properties, various motion constraints (e.g. footstep positions), as well as the muscle representation and the objective function which describes the overall "feel" of the motion. To edit the animation we modify the constraints and physical or kinematic parameters of the model (e.g. , limb geometry, footprint positions, objective function, gravity). Finally, we map the *motion change* resulting from the modification of the dynamics representation onto the original motion to produce a final animation sequence.

Once the dynamics representation has been constructed from the input data, our algorithm can be turned into a motion library. Each consequent change in the physical formulation of the model produces a brand new motion sequence. For example, a captured motion sequence of a human run might be turned into a running motion library capable of generating a wide range of runs that fit the needs of the animator. Our hope is that these libraries will enable ordinary computer users to create complex animations that are currently created only by skilled animators.

Our algorithm presents a first solution to the problem of editing captured motion while taking dynamics into consideration. We introduce the process of character simplification and a way to correlate motion of drastically different characters. In addition, we introduce a method for simplification of complex dynamic systems without losing the fundamental dynamic properties of motion.

1.1 Motivation

The dominant computer animation technique, keyframing, consists of explicitly setting the "key" values of the character's degrees of freedom (e.g. elbow joint angle, hip ball-joint angles), at specific time instants. To create an animation, each degree of freedom (DOF) is then independently interpolated to pass through the specified "keys." These DOF functions are then used to create in-between animation frames. As direct control of DOFs proved to be somewhat tedious, various inverse kinematics techniques emerged. With these methods, the user can directly position arbitrary points on the character's body, and consequently position the character in less time. For example, when animating an arm motion, the user can specify the path of a hand, without having to explicitly adjust the shoulder and elbow joint angles.

Keyframing provides extreme flexibility, thus enabling full artistic expression. However, it also makes animations that much harder to create by the rest of the world's less talented animators. In fact, the task of appropriately positioning the character to a specific pose at the right time is painfully arduous even for simple animations. Instead of incrementally setting keyframes for various character poses, the user ideally wants to edit high-level motion constructs. For example the animator may wish to place one leg in cast and observe the resulting limping behavior. Alternatively, the animator may impose greater balance, or change the character's behavior by specifying that the walking surface be significantly more slippery.

Even better, the user should be able to reach for the human run library, and instantiate a specific run by demanding realism and specifying the character dimensions, foot placements and even the emotional state of the runner. By limiting the left knee's range of motion the library would produce a limping run that satisfies all previous requirements. Furthermore, the user can still specify the finer detail constraints on the bounce quality, air-time, or specific arm poses. This model of *reusable motion* has been the goal for much of recent research in computer animation. The *motion libraries* would make animations accessible to a much wider population of computer content providers. Any realization of such flexible motion libraries faces a very difficult problem: maintaining a level of realism in light of all other motion reparameterizations.

The *spacetime constraints* framework, introduced in 1988 by Witkin and Kass [Witkin 88], effectively addresses the need for both realism and controllability of character motion. Unfortunately, these methods have not been shown to successfully solve such non-trivial animation problems as the motion sequences of the human characters. The work in this thesis draws from the ideas of spacetime constraints, and extends this paradigm so that it can be applied to a much wider range of complex animations.

Another way to get realistic motion is to acquire it from the real world. Recent availability of real-time 3-D motion capture systems provides such an alternative. Motion capture systems use sensors to record absolute positions of key points on the character's body over a period of time (Figure 1.1).

Not surprisingly, the resulting clip motion is extremely believable, and full of expressive detail that is almost impossible to generate by any computer methods. Although this type of animation is quite realistic, it yields highly unstructured motion, even when converted to joint angles within a hierarchical character model (Figure 1.2). Such canonical data is very hard to edit or modify in any way.

FIGURE 1.1. The author in an *extremely tight* optical motion capture suit.

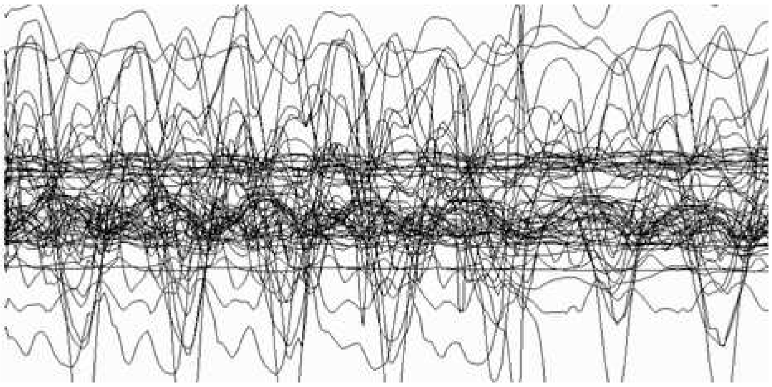

FIGURE 1.2. Some of the captured motion curves of human walking.

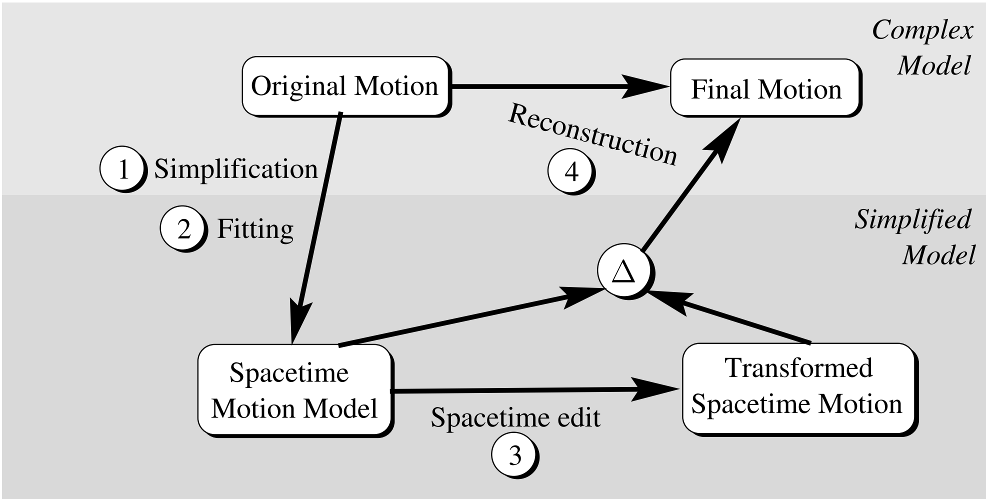

FIGURE 1.3. Algorithm outline.

Recently, a number of motion capture editing methods have been proposed [Witkin 95, Bruderlin 95, Gleicher 98a, Gleicher 97, Gleicher 98b, Rose 98]. These methods don't generate motion "from scratch" like other methods described earlier, but transform existing motion sequences instead (see Figure 2.1). Unfortunately, none of them include any notion of dynamics. As a result, a property of all motion editing methods that ignore inherent dynamics is that while they can effectively transform motion by small amounts, larger deformations reveal undesirable, unrealistic artifacts.

An alternative approach to editing realistic motion sequences is to extract a physical model from captured data, and perform all editing on the computer model instead. This thesis takes this approach. Spacetime optimization is a good candidate for this task. As we mentioned earlier, the primary problem of spacetime methods is that they do not guarantee a solution for motion problems of complex characters such as humans. We circumvent this issue by developing smaller, abstracted models motivated by biomechanics research [Blickhan 93].

1.2 Algorithm Outline

Much like motion capture editing methods, our algorithm does not synthesize motion from ground zero. Instead, it transforms the input motion sequence to satisfy the needs of the animation. Although our algorithm was motivated by the desire to enable realistic high-level control of high quality captured motion sequences, the same methods can be applied to motion from arbitrary sources.

Our algorithm uses spacetime optimization at its core because spacetime both maintains the dynamic integrity of motion and provides intuitive motion control. Because such methods have been shown to be infeasible for human motion models, we integrate model simplification ideas to reduce the complexity of the optimal motion model.

The entire process of transforming the input motion breaks down to four main stages (Figure 1.3):

1. Character Simplification

2. Spacetime Motion Fitting

3. Spacetime Edit

4. Motion Reconstruction

We briefly describe each stage.

1.2.1 Character Simplification

First, we manually create an abstract character model containing the minimal number of degrees of freedom necessary to capture the essence of the input motion. The reduced number of DOFs improves performance and facilitates convergence.

Furthermore, such an abstract model captures the more fundamental properties of motion, discarding the less important "high frequency" information specific to a given motion sequence. All our subsequent motion edits will be performed on this abstract model, leaving the high frequency characteristics of the animations unmodified throughout the motion transformation process. Although the character simplification process is performed manually, our experience suggests that the underlying principles are fairly straightforward and no particular expertise is required.

Once the simplified character model has been constructed, we map the input motion onto such a simplified model. The resulting motion is not physical; although, as we will show, it is very close to a dynamically sound motion sequence.

1.2.2 Spacetime Motion Fitting

The next step of our algorithm determines a realistic motion for the simplified character. This subproblem can be viewed as an inverse spacetime problem. More concretely, we attempt to find the spacetime optimization problem (including the constraints and the objective function) whose solution closely matches the simplified character motion. During this process, we rely heavily on the fact that the motion produced by the character simplification is relatively close to some spacetime problem solution. Once we have successfully determined the spacetime problem for the simplified character, we have correlated the dynamic model of the simplified character with the full model of the original animation.

1.2.3 Spacetime Edit

With a constructed spacetime problem we proceed to modify various spacetime formulation parameters. We introduce new pose constraints or modify the objective function, or change the environment by introducing additional obstacles. We can also change the character's shape, dimensions or DOFs as well as its mass distribution or muscle properties.

The ability to edit such high-level motion constructs allows easy creation of a wide variety of animation sequences that specifically meet the needs of an animator. Furthermore, only the first spacetime optimization performed during the spacetime fitting phase takes considerable time to solve. Since all subsequent spacetime optimizations of the modified spacetime problem start from a feasible nearby solution the optimizations take considerably less time. Shorter computation times, in turn, enable near real-time speeds during the spacetime editing.

1.2.4 Motion Reconstruction

Having completed the motion modification, we remap the change in motion introduced by the spacetime edit process onto the original model to produce the final animation. Effectively, we apply the change in motion between two spacetime motion sequences onto the original motion of the full character. Since the spacetime motion sequences contain only the "low frequency" information of the simplified character, so will the difference between these two sequences. Therefore, the resulting transformed motion will preserve the intricate details contained in the original motion sequence.

1.3 Contributions

This thesis introduces the first solution to the problem of editing captured motion which takes dynamics into consideration. We describe a procedure of extracting a physical motion model from canonic joint angle motion representation. During the process of model extraction we simplify the kinematic representation of the character, drastically reducing the number of DOFs in the process. This simplification significantly reduces the size of the resulting numerical problems. This in turn enables us to apply spacetime constraints optimization methods on human motion sequences for the first time. Previously, such complex motion models proved to be too large and non-linear for spacetime optimization methods.

Furthermore, spacetime constraints formulations provide intuitive high-level editing of motion sequences, such as foot placement and timing, changing the kinematic structure of the character, modifying the dynamic environment of the animation, or the objective function which the animation task optimizes.

In addition, the algorithms described in this thesis can also be used to map motion to/from characters with drastically different numbers of DOFs. Our retargeting method ensures that centers of mass of different body parts maintain their relation to each other, producing a more realistic remapped motion.

We also describe a methodology to control complex dynamic systems with simpler ones. We show that important dynamic properties can be preserved with a model of significantly fewer DOFs. These smaller systems can be solved much faster and more efficiently, enabling real time robotic motion planners, dynamically realistic interactive characters, realistic motion in video games etc.

Our algorithm can also be applied to non-physical motion to provide arbitrary levels of realism. In a way, such an application of our methods can be viewed as a realism filter. An animator could create an animation and pass it through such a filter to produce a more realistic version of the animation, much like an sharpen or blur filter would be applied to an image.

Finally, we also see our motion transformation method as a useful tool for analysis of natural motion. Empirical motion data of various animals can be analyzed for optimality or muscle usage, and help prove the very ideas that motivated this algorithm.

1.4 Thesis Layout

In the next chapter, we survey the state-of-the-art research which focuses on automatic character animation.

We then describe the mathematical underpinnings of the motion formulated in terms of the spacetime constraints optimization. It lays out the mathematical framework used in the subsequent chapters.

Chapter 4 describes the first two stages of the algorithm: character simplification and spacetime fitting. Together, these two stages create the physical model which can subsequently be modified to produce a variety of different transformed animations.

Chapter 5 addresses the last two stages of our algorithm: spacetime edit and motion reconstruction. We describe the power of editing the spacetime formulation, and how those changes are used to form the final animation sequence.

Chapter 6 describes the results of applying our algorithm on two captured motion sequences. We show a wide range of markedly different animations produced by our methods. We also discuss limitations of our approach and suggest the best-suited applications of our method.

Finally, Chapter 7 reviews all contributions of this thesis, and suggests a number of future research directions.

Chapter 2

BACKGROUND

In this chapter we describe the current state of research efforts focused on the problem of the automatic motion synthesis for articulated characters.

2.1 Basic Animation Methods

Currently, keyframing is by far the most prevalent computer animation tool. Animation by keyframing entails explicit definition of the "key" values of the character's degrees of freedom at specific time instants. Hermite or Catmull-Rom splines are often used to interpolate the values of DOFs so that they pass through the specified keys. Keyframing is often used together with the inverse kinematics (IK). With IK, the user can position arbitrary points on the character's body instead of controlling the DOFs directly. Along the same line of thought, instead of interpolating DOFs IK methods interpolate the explicit paths of the controlled point on the character's body. For example, when animating the motion of the arm, the user can specify the path of a hand, without having to explicitly adjust the shoulder and elbow joint angles.

Procedural animation is another frequently used motion synthesis technique. The animator specifies small procedures which programmatically describe the behavior of each DOF. The motion is then fine tuned by interactively "tweaking" the parameters of each procedure. Like keyframing procedural animation methods can also be augmented with IK.

The prevalent use of keyframing and procedural animation methods in computer animation stems mainly from the fact that these methods leave the full control of the resulting motion in the hands of the animator. The burden of animation quality rests entirely on the shoulders of the animator, much like a puppeteer has full control over the movement of the marionette by pulling on specific strings in accordance. Highly skilled animators and special effects wizards appreciate this extreme low-level controllability because it allows them to fully express their artistry.

Unfortunately, this absolute control can also be a curse. In many instances, the animators would much rather prefer to specify the overall appearance of motion containing only the relevant details allowing all other aspects of motion to be determined automatically. For example, an animator might want to create a karate kick animation where the character kicks the particular point in space, while jumping from one place to the other. Furthermore, the animator might require the character movement to appear realistic. Such small set of requirements presents a daunting task for even the most skilled animators. Realism of motion, in particular, an almost impossible to create from scratch without the help of some form of dynamics representation of motion.

2.2 Forward Dynamics

Forward dynamics methods present the most intuitive path to realism. These methods compute motion of objects which obey laws of physics. For inanimate objects such as the tumbling rigid objects [Baraff 91, Baraff 89, Baraff 90, Moore 88], or the secondary motion of cloth [Baraff 98, DeRose 98], forward dynamics techniques are ideal because for these problems obeying physical laws is synonymous with realism. In contrast, active characters create motion with their own muscles. The specific motion of real creatures depends on their intricate musculo-skeletal structure. Determining exact muscle forces which would make the animation look realistic is very difficult and requires the inclusion of accurate models of the muscular energy production. Therefore, the character animation which simply obeys the laws of physics is not necessarily a *realistic* animation.

Furthermore, dynamic forward simulation is very hard to control. With dynamics methods each animation frame depends on the previous frame (and consequently to all other preceding frames). As a result, the smallest change of dynamic properties of any single frame drastically affects all consecutive frames. This chained dependency results in lack controllability.

2.3 Spacetime Constraints Methods

Spacetime constraints [Witkin 88], provides both realism and control of character motion. In the spacetime framework the user first specifies *pose constraints* which must be satisfied by the resulting motion sequence (e.g. the character pose at the beginning and end of the animation). Additional *dynamics constraints* enforce the physical laws during the animation. In addition to these constraints, the user also specifies an *objective function* which is simply a metric of performance or style such as total power consumption of all of the character's muscles. The algorithm takes this spacetime specification and finds the motion trajectories which minimize the objective function while at the same time satisfying the constraints. High realism and intuitive control give this method a great appeal.

The downside, however, is that current algorithms do not scale up to the complexity of characters one would like to animate. Cohen developed a 3D animation system based on the spacetime paradigm [Cohen 92], and recently with Liu [Liu 94] proposed a multi-grid approach to spacetime constraints which somewhat improved the performance. Spacetime methods have also been used to create motion which ties in two separate motion sequences [Rose 96]. This work presents an alternate formulation of the dynamics constraints which improved the computation of the constraints and their derivatives by a constant. Still, the time complexity of the spacetime formulation remains a huge impediment for its application to complex characters.

An even more significant problem of these methods is that they are extremely sensitive to the starting position of the optimization process: if optimization starts far away from the solution, the optimization methods cannot converge to the solution. Unfortunately, there is no way to start relatively near the solution since the input motion specification contains only a few constraints. All current spacetime constraints methods suffer from this problem. As a result, spacetime optimization methods have not been successfully applied to automatic generation of human motion. The work in this thesis draws from the ideas of spacetime constraints and tries to address the issues that prevent this method from being applied to the motion of complex character.

2.4 Robot Controller Theory and Computer Animation

Robot controller design has also been applied to the domain of realistic computer animation [Raibert 91, van de Panne 94, van de Panne 93]. These methods use controllers which drive the actuator forces based on the current state of the environment. The forces, in turn, produce desired motion. Intuitively, controllers can be thought of as a set of instinctual reflexes which control muscles and collectively produce character's continuous motion. Once the controllers have been fine-tuned and synchronized between each other, this method can produce a wide range of expressive animations [Raibert 91, Hodgins 92]. Furthermore, a number of different animations can be created without any additional work, since the controllers adjust to the changes in the environment. Recently, van de Panne introduced an interesting method for generating motion from footsteps that include some rudimentary physical properties [van de Panne 97]. Although a controller transformation algorithm has been reported [Hodgins 97], determining controllers that produce realistic character motion is extremely difficult, and has not been formalized.

2.5 Robot Path Planning

A related motion generation task also occurs in robot path planning. In this framework, the problem is to find the trajectory of the robot that would avoid all the obstacles in the environment with a certain margin of error. In general, robot motion planning around obstacles is intractable. Most algorithms discretize both the space and time and apply heuristic search or dynamic programming techniques to find an optimal trajectory [Latombe 91]. To further reduce the time complexity, most algorithms consider only kinematic properties of robots when searching for optimal motion, although some theoretical results have been reported in optimal kinodynamic planning [Xavier 92, Donald 93]. Much work has also been done on computing locally time-optimal trajectories [Bobrow 85, Shin 85, Shiller 91], although other optimal criteria such as energy consumption have received little attention.

2.6 Artificial Life

The ideas from the spacetime constraints formulation and the robot control theory have taken a new direction in [Ngo 93, Sims 94, van de Panne 93, van de Panne 94]. These methods solve for the feedback control strategies instead of the DOF trajectories. At the core of each of these methods is a simulation engine. The character with an initial control strategy is placed in the dynamic simulation and its performance is measured against a fitness function (e.g. how far it moved within a given amount of time). Depending on this score, control strategy is modified so that consecutive simulations would hopefully produce a higher score.

Once the controllers have been computed, these methods have the advantage of being able to produce a number of different animations by without any additional computations.

This comes at the cost of computing the controllers which is considerably more complicated than solving directly for the trajectories. While these methods produce quite interesting and often unexpected animations especially from the perspective of artificial life research, they do not provide the animator with the overall control of the resulting motion. They also take prohibitively long time to compute making them ill-suited for the interactive motion synthesis process.

2.7 Motion Editing

A number of researchers have attacked the problem of realism from a completely different angle. They obtain realistic motion by way of motion capture systems and then proceed to edit such motion sequences in order to meet the constraints of the animation. Recently a number of such methods have been proposed [Witkin 95, Bruderlin 95, Gleicher 98a, Gleicher 97, Gleicher 98b]. These methods use a common strategy to transform the animation sequence:

1. Introduce constraints that the transformed motion needs to specify.
2. Transform the motion curves so that all constraints are satisfied while trying to be as close to the original motion as possible.

Rose *et al.* take a somewhat different approach. They compile a number of captured motions which they interpolate in order to arrive at the desired motion [Rose 98]. Effectively, this method samples a large space of all possible motion sequences that an animator might require with a small number of captured motion sequences. Such vast space is approximated by an interpolation strategy which determines the closest samples constructs the approximation space from these samples.

Some of the methods [Gleicher 97, Gleicher 98b] solve for the entire transformed motion sequence at once. This allows each constraint to influence the animation curves way before and after it's enforced interval. Even though these methods *do not* include any notion of dynamics, they're still refered to, perhaps erroneously, as spacetime constraints.

Recently Gleicher [Gleicher 98b] introduced a method for remapping captured motion onto drastically different characters. While his method is capable of producing many interesting motions, it has no means of making the retargeted motion physically realistic. For

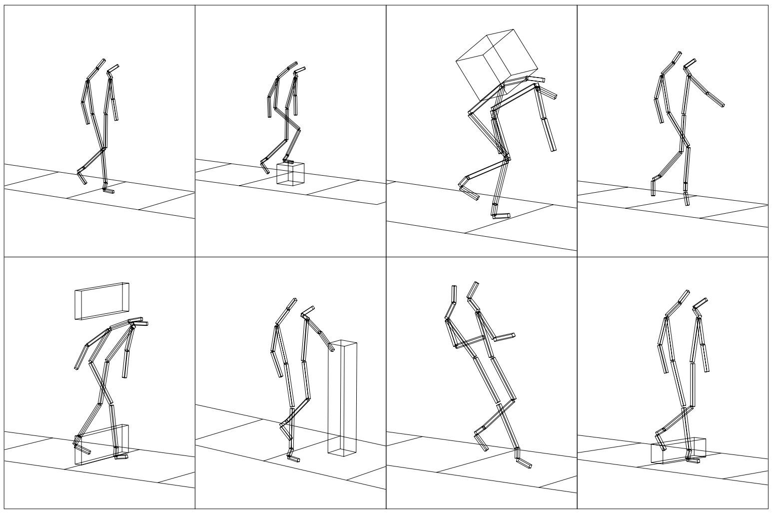

FIGURE 2.1. A frame from the original walking sequence, and the corresponding frames from a number of transformed sequences. Clockwise from upper left: The original sequence; stepping onto a block; carrying a heavy weight; walking on tiptoe; bending through a doorway; stepping around a post; trucking; stepping over an obstacle

example, a tall lanky character would utilize his muscles (and therefore move) in very different ways than a more compact character. By ignoring the dynamic aspects of motion, current methods can edit captured motion by relatively small amounts. Any significant modification produces noticeably unrealistic animations.

2.8 Biomechanics

The biomechanics literature looks at the motion from the perspective of analysis rather than synthesis. Analysis of motion has lead to a abstract perspective of motion. Our character simplification approach is motivated by such abstract views of motion in biomechanics [Blickhan 93]. Blickhan and Full demonstrate the similarity in the multi-legged locomotion of kinematically different animals. They show striking similarities between a human run, a horse run and the monopode bounce (i.e. pogostick). This similarity motivates our approach to reducing the DOF count of complex kinematic structures such as humans.

The biomechanics community has also developed a number of complex muscle models, which closely match empirical data [Pandy 92, Seireg 75, Crownninshield 81, Dul 84, An 84]. These models are quite accurate since they include specific data sampled from the real-world muscles. They are also very complex, which complicates their use in the forward simulation and robot control frameworks.

The same research also tries to address the *muscle indeterminacy problem*. This problem stems from the fact that there are a number of different muscle actuation histories which could result in the same motion sequence. Animals contain a redundant system of muscles which improves performance in light of potential failures. Some of this work could potentially lead to a more useful muscle model which would account for entire muscle groups instead for the individual muscles.

Biomechanics also studies the postulated optimality of motion in nature [Alexander 80, Alexander 89, Alexander 90, Alexander 91, Pedotti 78]. There has been a considerable amount of work in the area of human performance in sports [Pandy 90, Pandy 91]. This work also reaffirms that spacetime optimization is a good choice for realistic motion synthesis.

2.9 Where does Spacetime Motion Transformation Fit?

Our approach draws primarily from the spacetime constraints formulation. In fact, our method uses spacetime constraints formulation to represent all dynamic properties of motion. Unlike other spacetime constraints methods, we adopt the paradigm of motion transformation rather than motion synthesis. Thus, instead of generating realistic motion, we transform realistic motion which is either gathered from the real world (motion capture) or precomputed (forward simulation). From that perspective, our method is similar to other motion editing approaches such as motion warping and motion retargeting.

By taking the approach of applying spacetime constraints to motion capture sequences, we get the best of both worlds. We attain the expressiveness of the captured motion sequence and dynamics and controllability from the spacetime constraints.

Our approach also draws from the biomechanics research. In order to make spacetime problems more tractable we simplify the animated character in accordance to a few biomechanics principles (Figure 2.2).

The resulting method described in the following chapters presents a significant step towards the problem of automatic generation of expressive/realistic motion sequences that achieve a specific set of tasks.

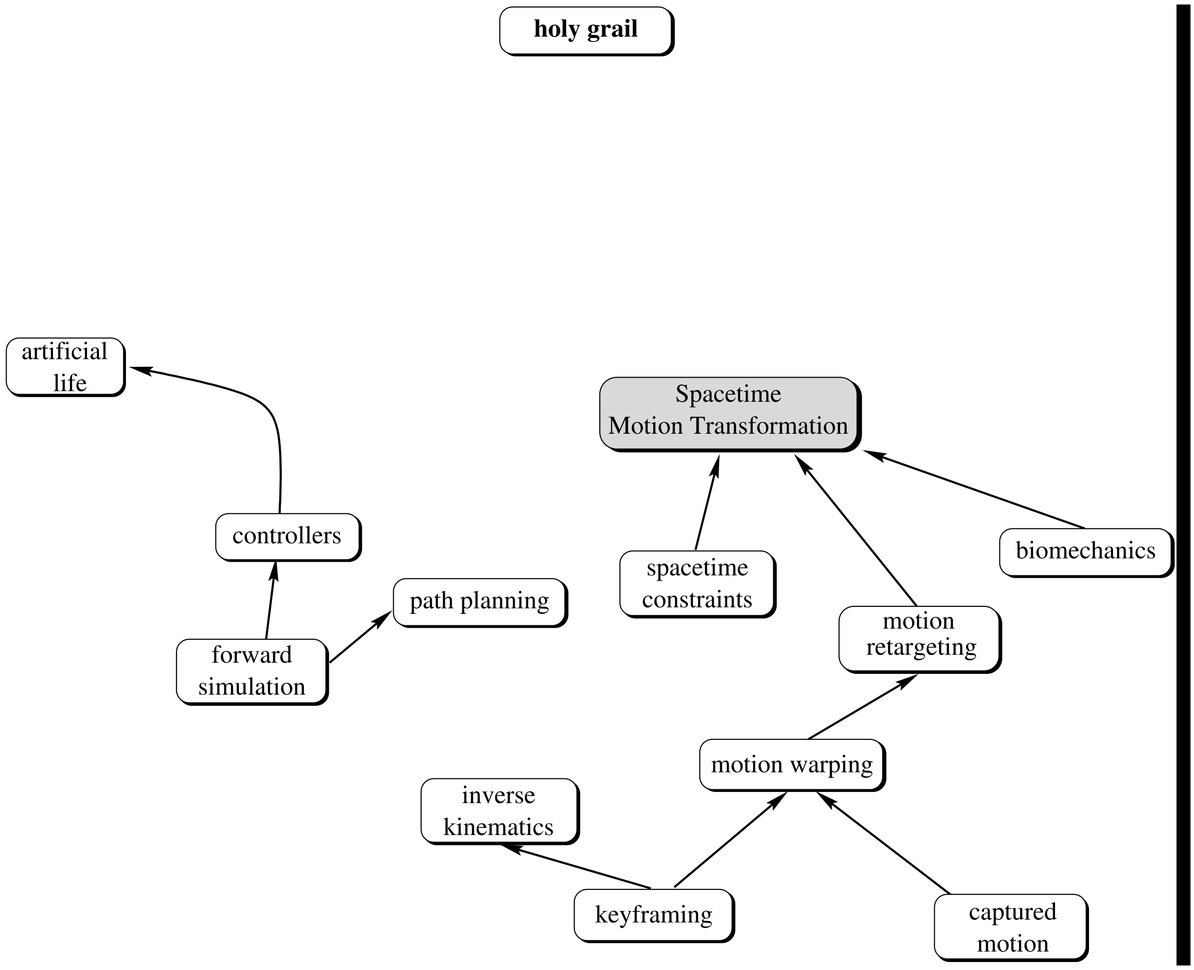

FIGURE 2.2. The flow of ideas towards the ultimate goal of automated synthesis of expressive/realistic character animation.

Chapter 3

SOLVING FOR CHARACTER MOTION WITH SPACETIME OPTIMIZATION

As described in Section 1.2 spacetime optimization is at the foundation of our approach to transforming motion. This chapter will describe in detail the mathematical underpinnings of spacetime optimization, and will lay out all necessary terminology used throughout this thesis.

3.1 Motion Synthesis in Terms of Constrained Optimization

All concepts of spacetime optimization map quite easily to their real-life counterparts. In this sense, spacetime methods tend to be quite intuitive.

A *character* is an object performing motion of its own accord. It has a finite number of kinematic DOFs and a number of *muscles*. DOFs usually represent joint angles of the character's extremities, while muscles exert forces or torques on different parts of the body, thus providing locomotion. Given that both body and muscle DOFs change through time we refer to them collectively as $\mathbf{q}(t)$, or separately as kinematic $\mathbf{q}_k(t)$, and muscle DOFs $\mathbf{q}_m(t)$. This motion is fundamentally different from that of an object simply obeying physical laws – a character is alive by virtue of its ability to locomote using it's own muscles.

The task of computer animation is, then, to find the desired motion of a character. This "goal motion" is rarely uniquely specified; rather, one looks for a motion that satisfies some set of requirements. Generally these requirements are represented either through constraints, external forces or through some objective function.

For instance, requirements of a sequence which animates a person getting up from the chair would include the fact that the person is sitting in the chair at time t_0 and standing up at final time t_1. The character must use its own muscles to satisfy these constraints. In the framework of this thesis, such requirements are referred to as *pose constraints* ($\mathbf{C}_p$).

In addition to pose constraints, the environment imposes a number of *mechanical constraints* ($\mathbf{C}_m$) onto the body. For example, in order to enforce the upright position of a human, we need to constrain both of her feet to the floor. The floor exerts forces onto the feet ensuring that they never penetrate the floor surface. All mechanical constraints provide external forces necessary to satisfy the constraints. There may also be other external *forces* within the environment such as gravity and wind.

Finally, we also need to ensure physical quality of motion. We do this by constraining the acceleration of each degree of freedom to be proportional to the generalized force exerted on that DOF. In other words, we make sure that "$F = ma$" holds for all degrees of freedom at all times. In this document we call such constraints *dynamics constraints* ($\mathbf{C}_d$). As long as these constraints are satisfied, we know that the resulting motion is physically possible.

When motion is defined in this way, it straightforwardly maps onto a non-linearly constrained optimization problem: we optimize the objective function parameterized in space and time, subject to the specified constraints:

$$\min_{\mathbf{q}(t)} E(\mathbf{q}(t), t) \qquad \text{subject to} \qquad \begin{cases} \mathbf{C}_p(\mathbf{q}(t), t) = 0 \\ \mathbf{C}_m(\mathbf{q}(t), t) = 0 \\ \mathbf{C}_d(\mathbf{q}(t), t) = 0 \end{cases} \tag{3.1}$$

$E(\mathbf{q}(t), t)$ is a functional making this a typical calculus of variations problem — we solve for functions, not values. Such problems are solved by continuous optimization methods, which require computation of first derivatives of all constraints and the objective function. The following sections describe differentiable mathematical representations of all spacetime optimization concepts.

3.2 Variational Description of Degrees of Freedom

To avoid the difficult problem of finding an arbitrary function $\mathbf{q}(t)$, numerical methods for solving variational problems often represent $\mathbf{q}(t)$ with the coefficients of arbitrary basis functions $\mathbf{q}(\mathbf{c}, t)$. Instead of solving for $\mathbf{q}(t)$, we are solving for coefficients $\mathbf{c}$.

In addition to evaluating $\mathbf{q}$ we also need to evaluate $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, $\frac{\partial \mathbf{q}}{\partial \mathbf{c}}$, $\frac{\partial \dot{\mathbf{q}}}{\partial \mathbf{c}}$ and $\frac{\partial \ddot{\mathbf{q}}}{\partial \mathbf{c}}$[1]. This implies that the basis should be at least C^1 continuous. We would also prefer that coefficients have localized influence in time. That way we can easily fix certain time values of $\mathbf{q}(t)$. Localized influence of coefficients also ensures that we can exploit the inherent sparsity of

[1] "Doted" quantities represent the time derivatives (e.g. , $\frac{\partial \mathbf{q}}{\partial t} = \dot{\mathbf{q}}$)

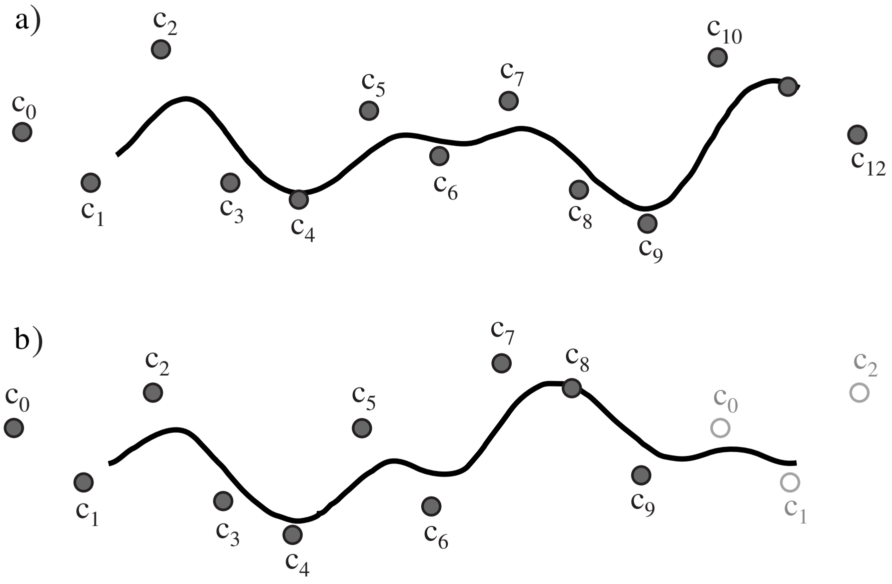

FIGURE 3.1. Representation of a) ordinary B-spline DOFs and b) cyclical B-spline DOFs.

matrices during the optimization process. We have found that a cubic B-spline basis is well suited for spacetime optimization problems: it is continuous, and derivative evaluation is constant time. B-spline wavelet basis has better optimization convergence properties, but it has two major drawbacks for our purposes: the derivative computation is not constant time, and more importantly coefficients don't have localized influence.

When optimization is solving for cyclical gait motion such as human running or walking, the representation of DOFs has implicit constraints that value, first and second derivatives at the initial and final time be the same. With the B-spline basis, this is achieved by making the first three coefficients identical to the last three coefficients (see Figure 3.1).

3.3 Character Transformation Hierarchy

We represent the animated character as a transformation hierarchy. Each node of the hierarchy contains a transformation $\mathbf{R}_i(\mathbf{q})$ (e.g. rotation, translation, scale) containing all kinematic DOFs and an optional primitive P_i (e.g. sphere, cylinder) which can be thought of as the character's "flesh". Primitives contain the character's mass.

3.3.1 Transformations

Transformations are represented as the usual 4×4 matrices, restricting transformations to the affine space. For example, an Euler rotation around the z-axis and translation along the y-axis are represented as

$$\mathbf{R}_z(q_\alpha) = \begin{bmatrix} \cos(q_\alpha) & -\sin(q_\alpha) & 0 & 0 \\ \sin(q_\alpha) & \cos(q_\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R}_{ty}(q_{ty}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{ty} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

respectively.

In the spacetime optimization framework each DOF is actually a function of time $q_i(t)$ so $\dot{q}_i(t)$ and $\ddot{q}_i(t)$ are well defined. Therefore, in addition to computing the transformation matrix $\mathbf{R}$ we can also compute the time derivatives $\dot{\mathbf{R}}$ and $\ddot{\mathbf{R}}$, as well as corresponding partial derivatives $\frac{\partial \mathbf{R}_i}{\partial q}$, $\frac{\partial \dot{\mathbf{R}}_i}{\partial q}$, $\frac{\partial \ddot{\mathbf{R}}_i}{\partial q}$. We show example derivatives for the y-axis translation.

$$\dot{\mathbf{R}}_{ty}(q_{ty}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dot{q}_{ty} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \ddot{\mathbf{R}}_{ty}(q_{ty}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddot{q}_{ty} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{R}_{q_{ty}}}{\partial q_{ty}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \frac{\partial \dot{\mathbf{R}}_{q_{ty}}}{\partial q_{ty}} = \frac{\partial \ddot{\mathbf{R}}_{q_{ty}}}{\partial q_{ty}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

3.3.2 Body Point

We denote a parent of node transformation $\mathbf{R}_i$ as $\mathbf{R}_{i-1}$. If two nodes $\mathbf{R}_i$ and $\mathbf{R}_j$ have the same parent then $\mathbf{R}_{i-1} = \mathbf{R}_{j-1}$. To better understand the composition of transformations within a hierarchy, we compute the world position $\mathbf{p}$ of a body point $\mathbf{x}$ within the node N_i and show how to compute position derivatives. The position of the body is a product

$$\mathbf{p} = \mathbf{R}_0 \mathbf{R}_1 \cdots \mathbf{R}_{i-1} \mathbf{R}_i \mathbf{x} \tag{3.2}$$

Also, we can also evaluate the partial derivative p with respect to some DOF q_j contained within $\mathbf{R}_k$

$$\frac{\partial \mathbf{p}}{\partial q_j} = \mathbf{R}_0 \mathbf{R}_1 \cdots \frac{\partial \mathbf{R}_k}{\partial q_j} \cdots \mathbf{R}_{i-1} \mathbf{R}_i \mathbf{x} \tag{3.3}$$

To avoid writing long matrix product chains we introduce some additional notation. We refer to a chain of transformations starting from $\mathbf{R}_j$ down to the transformation $\mathbf{R}_i$ as $\mathbf{W}_i{}^j$ and compute it as a product of transformations along the path from $\mathbf{R}_j$ to $\mathbf{R}_i$: $\mathbf{W}_i{}^j = \mathbf{R}_j \cdots \mathbf{R}_{i-1} \mathbf{R}_i$. We further reduce notation clutter by defining the transformation path from the root to node $\mathbf{R}_i$ as $\mathbf{W}_i = \mathbf{W}_i{}^0$. Note that $\mathbf{W}_i$ depends on all DOFs along the transformation path. The formulas 3.2 and and 3.3 are now much simpler

$$\mathbf{p} = \mathbf{W}_i \mathbf{x} \tag{3.4}$$

$$\frac{\partial \mathbf{p}}{\partial q_j} = \mathbf{W}_{k-1} \frac{\partial \mathbf{R}_k}{\partial q_j} \mathbf{W}_i{}^{k+1} \mathbf{x} \tag{3.5}$$

For the purposes of spacetime optimization we need to evaluate $\mathbf{W}_i$, $\dot{\mathbf{W}}_i$, $\ddot{\mathbf{W}}_i$, $\frac{\partial \mathbf{W}_i}{\partial q_j}$, $\frac{\partial \dot{\mathbf{W}}_i}{\partial q_j}$, $\frac{\partial \ddot{\mathbf{W}}_i}{\partial q_j}$ for each i and j. We compute these recursively to minimize the number of multiplications:

$$\mathbf{W}_i = \mathbf{W}_{i-1} \mathbf{R}_i \tag{3.6}$$

$$\dot{\mathbf{W}}_i = \dot{\mathbf{W}}_{i-1} \mathbf{R}_i + \mathbf{W}_{i-1} \dot{\mathbf{R}}_i \tag{3.7}$$

$$\ddot{\mathbf{W}}_i = \ddot{\mathbf{W}}_{i-1} \mathbf{R}_i + 2\dot{\mathbf{W}}_{i-1} \dot{\mathbf{R}}_i + \mathbf{W}_{i-1} \ddot{\mathbf{R}}_i \tag{3.8}$$

$$\frac{\partial \mathbf{W}_i}{\partial q_j} = \mathbf{W}_{k-1} \frac{\partial \mathbf{R}_k}{\partial q_j} \mathbf{W}_i{}^{k+1} \tag{3.9}$$

$$\frac{\partial \dot{\mathbf{W}}_i}{\partial \mathbf{q}_j} = \frac{\partial \dot{\mathbf{W}}_{i-1}}{\partial q_j} \mathbf{R}_i + \dot{\mathbf{W}}_i \frac{\partial \mathbf{R}_{i-1}}{\partial q_j} + \frac{\partial \mathbf{W}_{i-1}}{\partial q_j} \dot{\mathbf{R}}_i + \mathbf{W}_i \frac{\partial \dot{\mathbf{R}}_{i-1}}{\partial q_j} \tag{3.10}$$

$$\frac{\partial \ddot{\mathbf{W}}_i}{\partial \mathbf{q}_j} = \frac{\partial \ddot{\mathbf{W}}_{i-1}}{\partial q_j} \mathbf{R}_i + \ddot{\mathbf{W}}_i \frac{\partial \mathbf{R}_{i-1}}{\partial q_j} + 2\frac{\partial \dot{\mathbf{W}}_{i-1}}{\partial q_j} \dot{\mathbf{R}}_i + 2\dot{\mathbf{W}}_{i-1} \frac{\partial \dot{\mathbf{R}}_i}{\partial q_j} + \frac{\partial \mathbf{W}_{i-1}}{\partial q_j} \ddot{\mathbf{R}}_i + \mathbf{W}_{i-1} \frac{\partial \ddot{\mathbf{R}}_i}{\partial q_j} \tag{3.11}$$

3.3.3 Primitives

Primitives represent the "flesh and blood" of the character. Aside from the geometric representation used for rendering the primitive P_i has three additional properties: mass m_i,

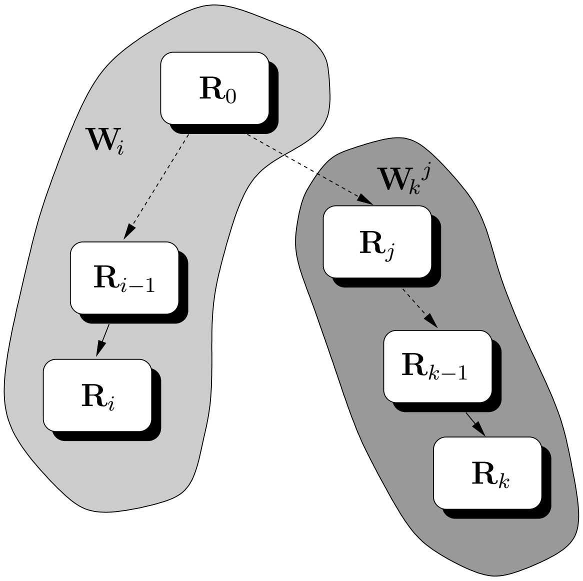

FIGURE 3.2. Character transformation hierarchy notation.

center of mass $\mathbf{c}_i$, and mass matrix $\mathbf{M}_i$. Even though mass and mass center are intuitive entities, for the sake of completeness we represent all three quantities as volume integrals over the primitive.

$$m_i = \iiint\limits_V \rho_i \, dx \, dy \, dz \tag{3.12}$$

$$\mathbf{c}_i = \frac{1}{m_i} \iiint\limits_V \mathbf{p} \, \rho_i \, dx \, dy \, dz \tag{3.13}$$

$$\mathbf{M}_i = \iiint\limits_V \mathbf{p}\mathbf{p}^T \, \rho_i \, dx \, dy \, dz \tag{3.14}$$

where $\mathbf{p} = [x \, y \, z]^T$, ρ_i is the density of the primitive P_i at point $\mathbf{p}$ and the outer product

$$\mathbf{p}\mathbf{p}^T = \begin{bmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{bmatrix}$$

Naturally, these values are precomputed for each primitive so that no integration is done during the optimization process. Section 3.8 describes the use of these terms in spacetime optimization.

3.4 Forces

Various forces act on the character during the animation and thus change its dynamic behavior. The most common force, gravity, is a constant $\mathbf{g}$ and acts on the character's center of mass. It often represented in dynamics equations through *potential energy* $U = m\mathbf{g}\mathbf{c}$ as we will see in Section 3.8. Forces also emerge from enforcing mechanical constraints (see Section 3.7).

3.4.1 Point Forces

Each point force $\mathbf{F}$ acts on the specific character body point $\mathbf{p}_f$. The point of force action $\mathbf{p}_f$ differentiates just like any other body point as described in Section 3.3.2. Each force $\mathbf{F}$ can also possibly depend on kinematic DOFs. An example of such a force is a spring force $F = k_s[(\mathbf{p}_1 - \mathbf{p}_0) - l]$ between two body points $\mathbf{p}_0$ and $\mathbf{p}_1$ with rest length l and spring constant k_s. Assuming that k_s and l are constant, each derivative of F by chain rule reduces to expressions of point derivatives. For example, a derivative with respect to two kinematic DOFs is

$$\frac{\partial^2 \mathbf{F}}{\partial q_i \, \partial q_j} = k_s \left(\frac{\partial^2 \mathbf{p}_1}{\partial q_i \, \partial q_j} - \frac{\partial^2 \mathbf{p}_0}{\partial q_i \, \partial q_j} \right)$$

.

3.5 Muscles

Muscles are a primary source of character locomotion. In nature, muscles generate equal and opposite forces between two or more body points. They contract, and cannot apply force by expanding. Consequently, most organisms have elaborate muscle networks which enable them to maximize the range of motion of their extremities. The amount of exerted force as well as the force derivatives are bounded by the chemical energy release mechanisms of muscles. These bounds not only restrict the strength of a muscle but also the speed at which a muscle can be activated. Each muscle attaches to the rigid body structure by a *ligament*, which has the physical properties of a very stiff highly-damped spring. Ligaments pull body limbs towards the rest position, as other muscles or environment forces act on it. This rather intricate structure of muscles in animals greatly affects their movement. The biomechanics community has developed a number of complex muscle models, which closely match empirical data. [Pandy 92, Seireg 75, Crownninshield 81, Dul 84, An 84]. While these models tend to be very accurate, their complexity makes them very hard to differentiate and use in full body optimizations. Also, a realistic model of a human muscle system needs a prohibitively large number of muscles. Instead, we would like to use simple structures which account for entire muscle groups, yet still produce forces acting on DOFs similar to those of real muscles.

On the other side of the complexity spectrum, generalized muscle forces represent the most abstract muscle. They apply accelerations directly onto DOFs, much like robotic servo-motors positioned at joints apply forces on robotic limbs. Having a generalized muscle at each DOF presents the minimum set of muscles that preserves the full range of character motion. This of course is very good for reducing the size of the optimization problem. Unfortunately, the ability to apply arbitrary generalized force onto each joint is a poor model of natural muscles. For example, sudden non-smooth muscle forces generate extremely jerky unnatural motion much like motion generated by the bang-bang controllers [Pontryagin 62]. In fact, in my experience the realistic representation of muscles is the *most important* determining factor of the resulting motion.

In addition, arbitrary impulse muscle forces tend to produce highly unstable spacetime optimization problems, with bad convergence properties, because the problem becomes extremely badly scaled. This is easy to see if we compare the relative change in motion resulting from changing a single coefficient of a kinematic DOF $q(t)$ and a generalized muscle DOF $q^m(t)$ by the same fixed amount δq. Naturally, motion changes are orders of magnitude more drastic when we displace $q^m(t)$ coefficients, since muscles directly affect accelerations of many kinematic DOFs. This imbalance in sensitivity between the coefficients of kinematic and generalized muscle DOFs makes it very hard for any optimization methods to converge to a solution.

To circumvent problems of simple generalized force muscles described above, yet still maintain a simple and differentiable muscle model we use a damped servo model often used in robotic simulations [Raibert 91, Hodgins 92]. Each kinematic DOF q_i has a corresponding damped generalized muscle force

$$Q_i = k_s(q_i - q_i^m) - k_d(\dot{q}_i - \dot{q}_i^m) \tag{3.15}$$

where q_i^m is the additional muscle DOF. Differentiation with respect to DOFs and their velocities is straightforward. For example,

$$\frac{\partial Q_i}{\partial q_i} = k_s$$
$$\frac{\partial Q_i}{\partial \dot{q}_i^m} = -k_d$$

This formulation does not have the scaling problem since q_i^m is of the same scale as q_i. The velocity dependent damping ensures smoothness. To further ensure smooth muscle forces akin to those in nature, we always include a muscle smoothness metric within the objective function (see Section 3.9).

3.6 Pose Constraints

Pose constraints specify certain invariants of the modeled motion. Foot placements, specific torso orientation or full body pose are all examples of pose constraints. Constraints consist of three basic components:

- algebraic expression dependent on DOFs
- whether the constraint is equality or inequality ($=, >, <$)
- time interval over which the constraint must hold

.

During the optimization process the pose constraints must be satisfied for the entire duration of their active interval. Therefore, in constructing a constrained optimization problem, the constraint with a large active interval would produce a larger set of time instantiated constraints than the constraint which is active for a relatively small time interval.

Most common constraint expressions contain certain body point positions as subexpressions. Foot placement at a specific floor location $\mathbf{p}_{floor}$ can be expressed as

$$\mathbf{C}_{foot} = \mathbf{p}_{foot}(\mathbf{q}_k) - \mathbf{p}_{floor} = \mathbf{0}.$$

Since it's only DOF dependency is through the foot body point, derivatives are straightforward

$$\frac{\partial \mathbf{C}_{foot}}{\partial \mathbf{q}_k} = \frac{\partial \mathbf{p}_{foot}}{\partial \mathbf{q}_k}$$

In practice, we often use 2 foot point constraints to avoid foot twisting while on the ground. A floor penetration constraint is similar to the foot placement constraint. The main difference is that it is a one dimensional inequality constraint:

$$C_{floor} = \mathbf{p}_{foot}(\mathbf{q}_k)_y - floor_y > 0.$$

In some cases constraints depend on certain body orientations rather than positions. Suppose that we want to constrain a facing direction of the character to be pointing in the direction of the unit vector $\mathbf{d}$. We use 2 body points which define the body direction p_0, p_1 to specify the constraint

$$\mathbf{C}_{orient} = \frac{\mathbf{p}_1(\mathbf{q}_k) - \mathbf{p}_0(\mathbf{q}_k)}{\|\mathbf{p}_1(\mathbf{q}_k) - \mathbf{p}_0(\mathbf{q}_k)\|} - \mathbf{d} = 0.$$

3.7 Mechanical Constraints

Mechanical constraints are identical to pose constrains as they are specified in the same manner. The only difference is that they also contain forces which act to satisfy these constraints. Examples include a foot on the floor, or a hand being physically attached to a fixed object. In the example of the character foot being on the ground, there is always an equal and opposite force that the floor exerts on the foot. Regardless of the amount of gravity force, and the force that muscles exert on the floor through the foot contact, the floor will always counteract these forces so that the sum of all forces exerted on the foot is zero. In the classical dynamics parlance we say that the mechanical constraint force does zero work. In other words, this force can never add extra energy to the character. Specifically, given a mechanical constraint $\mathbf{C}_m$ the corresponding generalized force acting on a specific kinematic DOF q_i is

$$\mathbf{F}_i = \mathbf{q}_\lambda \frac{\partial \mathbf{C}_m}{\partial q_i} \tag{3.16}$$

where $\mathbf{q}_\lambda$ is a vector of DOFs of the same dimension as $\mathbf{C}_m$. In other words, each mechanical constraint has a corresponding DOF which it uses to generate the force necessary to maintain the constraint. Mechanical constraint DOFs $\mathbf{q}^\lambda$ are closely related to Lagrangian multipliers. We can easily attain derivatives of this generalized force:

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{q}_m^\lambda} = \frac{\partial \mathbf{C}_m}{\partial q_i} \tag{3.17}$$

$$\frac{\partial \mathbf{F}_i}{\partial q_j} = \mathbf{q}^\lambda \frac{\partial^2 \mathbf{C}_m}{\partial q_i \, \partial \mathbf{q}_j} \tag{3.18}$$

Section 3.8 shows how this generalized force component factors in the definition of Newtonian constraints. Note that since pose constraints have no generalized force associated with them, they must be satisfied with other forces in the system (e.g. muscles, external environment forces).

3.8 Newtonian Constraints

Newtonian constraints ensure that the laws of dynamics are satisfied. A violation of these constraints implies that the current motion is physically impossible. Naturally they are the most complicated constraints; they are highly non-linear. In fact, they contain by far the most complex expressions of the entire spacetime optimization problem. A great deal of difficulty with finding the spacetime solution stems directly from the fact that it is very hard to find a feasible starting point which satisfies all Newtonian constraints. Furthermore, it is almost as hard to remain feasible with respect to these constraints, while searching for the optimal solution.

The complexity of Newtonian constraints also makes them very hard to evaluate efficiently. For completeness, we derive these constraints from first principles. Our formulation of the Newtonian constraints is an extended version of the one found in [Liu 96].

The fundamental principle of analytical mechanics, Hamilton's principle [Lanczos 70], leads to a system of simultaneous differential equations of the second order, the Lagrangian equations of motion

$$\sum_i \frac{d}{dt}\frac{\partial T_i}{\partial \dot{q}_j} - \frac{\partial T_i}{\partial q_j} - \frac{\partial U_i}{\partial \dot{q}_j} = 0 \tag{3.19}$$

where T_i and U_i are kinetic and potential energy of the body primitive P_i.

We start our derivation of T_i and U_i by treating the character as a collection of points $\mathbf{x}_i = [x\,y\,z\,1]^T$ in body space each with an infinitesimal mass τ_i. We define points as homogeneous 4-vectors in order to be able to apply 4×4 transformation matrices defined in Section 3.3.1. The world position of the body point $\mathbf{x}_i$ is then

$$\mathbf{p}_i = \mathbf{W}\mathbf{x}_i$$

where $\mathbf{W}$ is the transformation path to the primitive in which $\mathbf{x}_i$ is contained. By definition, the kinetic energy of the body primitive P_i is

$$
\begin{aligned}
T_i &= \frac{1}{2} \int_i \dot{\mathbf{p}}_i^T \dot{\mathbf{p}}_i \, \tau_i \, dx \, dy \, dz && (3.20)\\
&= \frac{1}{2} \int_i \mathbf{x}_i^T \dot{\mathbf{W}}^T \dot{\mathbf{W}} \mathbf{x}_i \, \tau_i \, dx \, dy \, dz && (3.21)\\
&= \frac{1}{2} \int_i \operatorname{tr}\left(\dot{\mathbf{W}} \mathbf{x}_i \mathbf{x}_i^T \dot{\mathbf{W}}^T \right) \, \tau_i \, dx \, dy \, dz && (3.22)\\
&= \frac{1}{2} \operatorname{tr}\left(\dot{\mathbf{W}} \left[\int_i \mathbf{x}_i \mathbf{x}_i^T \, \tau_i \, dx \, dy \, dz \right] \dot{\mathbf{W}}^T \right) && (3.23)\\
&= \frac{1}{2} \operatorname{tr}\left(\dot{\mathbf{W}} \mathbf{M}_i \dot{\mathbf{W}}^T \right) && (3.24)
\end{aligned}
$$

where $\mathbf{M}_i$ is the primitive mass tensor defined in Section 3.3.3.

Taking various derivatives of the primitive kinetic energy we have

$$
\begin{aligned}
\frac{\partial T_i}{\partial q_j} &= \frac{1}{2} \operatorname{tr}\left(\frac{\partial \dot{\mathbf{W}}_i}{\partial q_j} \mathbf{M}_i \dot{\mathbf{W}}_i^T + \dot{\mathbf{W}}_i \mathbf{M}_i \frac{\partial \dot{\mathbf{W}}_i^T}{\partial q_j} \right)\\
\frac{\partial T_i}{\partial \dot{q}_j} &= \frac{1}{2} \operatorname{tr}\left(\frac{\partial \dot{\mathbf{W}}_i}{\partial \dot{q}_j} \mathbf{M}_i \dot{\mathbf{W}}_i^T + \dot{\mathbf{W}}_i \mathbf{M}_i \frac{\partial \dot{\mathbf{W}}_i^T}{\partial \dot{q}_j} \right)\\
&= \frac{1}{2} \operatorname{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \dot{\mathbf{W}}_i^T + \dot{\mathbf{W}}_i \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_j} \right)\\
\frac{d}{dt}\frac{\partial T_i}{\partial \dot{q}_j} &= \frac{1}{2} \operatorname{tr}\left(\frac{\partial \dot{\mathbf{W}}_i}{\partial q_j} \mathbf{M}_i \dot{\mathbf{W}}_i^T + \frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^T + \ddot{\mathbf{W}}_i \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_j} + \dot{\mathbf{W}}_i \mathbf{M}_i \frac{\partial \dot{\mathbf{W}}_i^T}{\partial q_j} \right)
\end{aligned}
$$

So

$$
\begin{aligned}
\frac{d}{dt}\frac{\partial T_i}{\partial \dot{q}_j} - \frac{\partial T_i}{\partial q_j} &= \frac{1}{2} \operatorname{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^T + \ddot{\mathbf{W}}_i \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_j} \right) && (3.25)\\
&= \operatorname{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^T \right) && (3.26)
\end{aligned}
$$

The potential energy due to gravity is $U_i = m_i \mathbf{g} \mathbf{W}_i \mathbf{c}_i$ and

$$
\frac{d}{dt}\frac{\partial U_i}{\partial \dot{q}_j} - \frac{\partial U_i}{\partial q_j} = m_i \mathbf{g} \frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{c}_i \qquad (3.27)
$$

The generalized force acting on the DOF q_j due to the point force $\mathbf{F}_k$ acting on the body point $\mathbf{p}_k$ is

$$
\frac{\partial \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j}
$$

Similarly, the generalized force for the DOF q_j due to the mechanical constraint $\mathbf{C}_{m_l}$ is

$$\mathbf{q}_l^\lambda \frac{\partial \mathbf{C}_{m_l}}{\partial q_j}.$$

$\mathbf{q}_l^\lambda$ can be intuitively thought of as additional force DOFs used to satisfy the mechanical constraint $\mathbf{C}_{m_l}$ without adding additional energy to the dynamic system.

By combining all of the force contributions, we determine the total generalized force with respect to DOF q_j, which is also the Newtonian constraint for q_j

$$C_{n_j} = \sum_i \left[\mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^T \right) + m_i \mathbf{g} \frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{c}_i \right] + \sum_k \left[\frac{\partial \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j} \right] + \sum_l \left[\mathbf{q}_l^\lambda \frac{\partial \mathbf{C}_{m_l}}{\partial q_j} \right] \tag{3.28}$$

where i ranges over all primitives, k over all point forces, and l over all mechanical constraints.

The formulation of the Newtonian constraint in Equation 3.28 contains a large number of redundant operations. An alternative recursive definition would be drastically more efficient. Since the formulation of $\mathbf{F}_k$ and $\mathbf{C}_{m_l}$ can be arbitrarily complex and could ultimately depend on all kinematic DOFs in the hierarchy, the point force and mechanical constraint contributions to the generalized force cannot be recursively reformulated. However, the kinetic and potential energy contributions to the Newtonian constraints have natural recursive definitions. First let us define n_j to be the index of the transformation node which contains DOF q_j. Also let $S(i)$ be the set of all node indices in the subtree rooted at node i. We rearrange the Equation 3.28

$$\sum_i \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^T \right) + m_i \mathbf{g} \frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{c}_i \tag{3.29}$$

$$= \sum_{i \in S(n_j)} \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^T \right) + m_i \mathbf{g} \frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{c}_i \tag{3.30}$$

$$= \mathrm{tr}\left(\frac{\partial \mathbf{W}_{n_j}}{\partial q_j} \sum_{i \in S(n_j)} \mathbf{W}_i^{n_j} \mathbf{M}_i \ddot{\mathbf{W}}_i^T \right) + \mathbf{g} \frac{\partial \mathbf{W}_{n_j}}{\partial q_j} \sum_{i \in S(n_j)} m_i \mathbf{W}_i^{n_j} \mathbf{c}_i. \tag{3.31}$$

We introduce two new recursively defined variables

$$\hat{\mathbf{c}}_i = m_i \mathbf{c}_i + \sum_{j \in K(i)} \mathbf{R}_j \hat{\mathbf{c}}_j \tag{3.32}$$

$$\widehat{\ddot{\mathbf{M}}}_i = \mathbf{M}_i \ddot{\mathbf{W}}_i^T + \sum_{j \in K(i)} \mathbf{R}_j \widehat{\ddot{\mathbf{M}}}_j \tag{3.33}$$

which leads us to the recursive definition of the Newtonian constraint

$$C_{n_j} = \mathrm{tr}\left(\frac{\partial \mathbf{W}_{n_j}}{\partial q_j}\widehat{\ddot{\mathbf{M}}_{n_j}}\right) + \mathbf{g}\frac{\partial \mathbf{W}_{n_j}}{\partial q_j}\hat{\mathbf{c}}_{n_j} + \sum_k \left[\frac{\partial \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j}\right] + \sum_l \left[\mathbf{q}_l^\lambda \frac{\partial \mathbf{C}_{m_l}}{\partial q_j}\right]. \tag{3.34}$$

3.8.1 Derivatives

Since Newtonian constraints depend on $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ we calculate all derivatives of the Newtonian constraint

$$\begin{aligned}\frac{\partial C_{n_j}}{\partial q_k} = &\sum_{i \in S(n_j)\cap S(n_k)} \mathrm{tr}\left(\frac{\partial^2 \mathbf{W}_i}{\partial q_j\,\partial q_k}\mathbf{M}_i \ddot{\mathbf{W}}_i^T + \frac{\partial \mathbf{W}_i}{\partial q_j}\mathbf{M}_i\frac{\partial \ddot{\mathbf{W}}_i^T}{\partial q_k}\right) + m_i \mathbf{g}\frac{\partial^2 \mathbf{W}_i}{\partial q_j\,\partial q_k}\mathbf{c}_i \\ &+ \sum_k \frac{\partial^2 \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j\,\partial q_k} + \sum_l \mathbf{q}_l^\lambda \frac{\partial^2 \mathbf{C}_{m_l}}{\partial q_j\,\partial q_k}\end{aligned} \tag{3.35}$$

$$\frac{\partial C_{n_j}}{\partial q_l^\lambda} = \frac{\partial \mathbf{C}_{m_l}}{\partial q_j} \tag{3.36}$$

$$\frac{\partial C_{n_j}}{\partial \dot{q}_k} = \sum_{i \in S(n_j)} \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j}\mathbf{M}_i\frac{\partial \ddot{\mathbf{W}}_i^T}{\partial \dot{q}_k}\right) + \sum_k \frac{\partial^2 \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j\,\partial \dot{q}_k} + \sum_l \mathbf{q}_l^\lambda \frac{\partial^2 \mathbf{C}_{m_l}}{\partial q_j\,\partial \dot{q}_k} \tag{3.37}$$

$$\frac{\partial C_{n_j}}{\partial \ddot{q}_k} = \sum_{i \in S(n_j)} \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j}\mathbf{M}_i\frac{\partial \ddot{\mathbf{W}}_i^T}{\partial \ddot{q}_k}\right) + \sum_k \frac{\partial^2 \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j\,\partial \ddot{q}_k} + \sum_l \mathbf{q}_l^\lambda \frac{\partial^2 \mathbf{C}_{m_l}}{\partial q_j\,\partial \ddot{q}_k} \tag{3.38}$$

Again we compute the recursive formulations of the above expressions for minimal computation cost. Assuming that $u_j \in S(u_k)$ (i.e. u_k is in the path from the root to u_j, or u_j is the descendant of u_k) we have

$$\sum_{i \in S(n_j)\cap S(n_k)} \mathrm{tr}\left(\frac{\partial^2 \mathbf{W}_i}{\partial q_j\,\partial q_k}\mathbf{M}_i \ddot{\mathbf{W}}_i^T + \frac{\partial \mathbf{W}_i}{\partial q_j}\mathbf{M}_i\frac{\partial \ddot{\mathbf{W}}_i^T}{\partial q_k}\right) + m_i \mathbf{g}\frac{\partial^2 \mathbf{W}_i}{\partial q_j\,\partial q_k}\mathbf{c}_i \tag{3.39}$$

$$= \sum_{i \in S(n_j)} \mathrm{tr}\left(\frac{\partial^2 \mathbf{W}_{n_j}}{\partial q_j\,\partial q_k}\mathbf{W}_i^{n_j}\mathbf{M}_i \ddot{\mathbf{W}}_i^T + \frac{\partial \mathbf{W}_{u_j}}{\partial q_j}\mathbf{W}_j^{n_j}\mathbf{M}_i\frac{\partial \ddot{\mathbf{W}}_i^T}{\partial q_k}\right) + m_i \mathbf{g}\frac{\partial^2 \mathbf{W}_{u_j}}{\partial q_j\,\partial q_k}\mathbf{W}_i^{n_j}\mathbf{c}_i \tag{3.40}$$

$$\begin{aligned}= &\mathrm{tr}\left(\frac{\partial^2 \mathbf{W}_{n_j}}{\partial q_j\,\partial q_k}\sum_{i \in S(n_j)} \mathbf{W}_i^{n_j}\mathbf{M}_i \ddot{\mathbf{W}}_i^T + \frac{\partial \mathbf{W}_{u_j}}{\partial q_j}\sum_{i \in S(n_j)} \mathbf{W}_j^{n_j}\mathbf{M}_i\frac{\partial \ddot{\mathbf{W}}_i^T}{\partial q_k}\right) \\ &+ \mathbf{g}\frac{\partial^2 \mathbf{W}_{u_j}}{\partial q_j\,\partial q_k}\sum_{i \in S(n_j)} m_i \mathbf{W}_i^{n_j}\mathbf{c}_i\end{aligned} \tag{3.41}$$

By introducing a new recursive variable

$$\widehat{\frac{\partial \ddot{\mathbf{M}}_i}{\partial q_k}} = \mathbf{M}_i \frac{\partial \ddot{\mathbf{W}}_i^T}{\partial q_k} + \sum_{j \in K(i)} \mathbf{R}_j \widehat{\frac{\partial \ddot{\mathbf{M}}_j}{\partial q_k}} \tag{3.42}$$

we arrive at the recursive definition for $\frac{\partial C_{n_j}}{\partial q_k}$

$$\begin{aligned} \frac{\partial C_{n_j}}{\partial q_k} = &\operatorname{tr}\left(\frac{\partial^2 \mathbf{W}_{n_j}}{\partial q_j\, \partial q_k} \widehat{\ddot{\mathbf{M}}_{n_j}} + \frac{\partial \mathbf{W}_{u_j}}{\partial q_j} \widehat{\frac{\partial \ddot{\mathbf{M}}_{u_j}}{\partial q_k}} \right) + \mathbf{g} \frac{\partial^2 \mathbf{W}_{n_j}}{\partial q_j\, \partial q_k} \hat{\mathbf{c}}_{n_j} \qquad \text{if } n_j \in S(n_k) \\ &+ \sum_k \frac{\partial^2 \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j\, \partial q_k} + \sum_l \mathbf{q}_l^\lambda \frac{\partial^2 \mathbf{C}_{m_l}}{\partial q_j\, \partial q_k} \end{aligned} \tag{3.43}$$

Assuming that $n_k \in S(n_j)$ leads to an analogous equation

$$\begin{aligned} \frac{\partial C_{n_j}}{\partial q_k} = &\operatorname{tr}\left(\frac{\partial^2 \mathbf{W}_{n_k}}{\partial q_j\, \partial q_k} \widehat{\ddot{\mathbf{M}}_{n_k}} + \frac{\partial \mathbf{W}_{u_k}}{\partial q_j} \widehat{\frac{\partial \ddot{\mathbf{M}}_{u_k}}{\partial q_k}} \right) + \mathbf{g} \frac{\partial^2 \mathbf{W}_{n_k}}{\partial q_j\, \partial q_k} \hat{\mathbf{c}}_{n_k} \qquad \text{if } n_k \in S(n_j) \\ &+ \sum_k \frac{\partial^2 \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j\, \partial q_k} + \sum_l \mathbf{q}_l^\lambda \frac{\partial^2 \mathbf{C}_{m_l}}{\partial q_j\, \partial q_k} \end{aligned} \tag{3.44}$$

Combining the Equations 3.43 and 3.44 we get

$$\begin{aligned} \frac{\partial C_{n_j}}{\partial q_k} = &\sum_k \frac{\partial^2 \mathbf{F}_k \mathbf{p}_{F_k}}{\partial q_j\, \partial q_k} + + \sum_l \mathbf{q}_l^\lambda \frac{\partial^2 \mathbf{C}_{m_l}}{\partial q_j\, \partial q_k} \\ &+ \begin{cases} \operatorname{tr}\left(\frac{\partial^2 \mathbf{W}_{n_j}}{\partial q_j\, \partial q_k} \widehat{\ddot{\mathbf{M}}_{n_j}} + \frac{\partial \mathbf{W}_{u_j}}{\partial q_j} \widehat{\frac{\partial \ddot{\mathbf{M}}_{u_j}}{\partial q_k}} \right) + \mathbf{g} \frac{\partial^2 \mathbf{W}_{n_j}}{\partial q_j\, \partial q_k} \hat{\mathbf{c}}_{n_j} & \text{if } n_j \in S(n_k), \\ \operatorname{tr}\left(\frac{\partial^2 \mathbf{W}_{n_k}}{\partial q_j\, \partial q_k} \widehat{\ddot{\mathbf{M}}_{n_k}} + \frac{\partial \mathbf{W}_{u_k}}{\partial q_j} \widehat{\frac{\partial \ddot{\mathbf{M}}_{u_k}}{\partial q_k}} \right) + \mathbf{g} \frac{\partial^2 \mathbf{W}_{n_k}}{\partial q_j\, \partial q_k} \hat{\mathbf{c}}_{n_k} & \text{if } n_k \in S(n_j), \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{3.45}$$

We also derive recursive formulations for the derivatives of the Newtonian constraint with respect to the DOF velocity and acceleration

$$
\begin{aligned}
\frac{\partial C_{n_j}}{\partial \dot{q}_k} &= \sum_{i \in S(n_j)} \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \frac{\partial \ddot{\mathbf{W}}_i^T}{\partial \dot{q}_k} \right) \\
&= \mathrm{tr}\left(\sum_{i \in S(n_j)} 2 \frac{\partial \mathbf{W}_{u_j}}{\partial q_j} \mathbf{W}_i^{n_j} \mathbf{M}_i \frac{\partial \dot{\mathbf{W}}_i^T}{\partial q_k} \right) \\
&= \mathrm{tr}\left(\frac{\partial \mathbf{W}_{u_j}}{\partial q_j} \sum_{i \in S(n_j)} 2 \mathbf{W}_i^{n_j} \mathbf{M}_i \frac{\partial \dot{\mathbf{W}}_i^T}{\partial q_k} \right) \\
&= \mathrm{tr}\left(\frac{\partial \mathbf{W}_{u_j}}{\partial q_j} 2 \frac{\widehat{\partial \dot{\mathbf{M}}_{n_j}}}{\partial q_k} \right)
\end{aligned} \tag{3.46}
$$

$$
\begin{aligned}
\frac{\partial C_{n_j}}{\partial \ddot{q}_k} &= \sum_{i \in S(n_j)} \mathrm{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i \frac{\partial \ddot{\mathbf{W}}_i^T}{\partial \ddot{q}_k} \right) \\
&= \mathrm{tr}\left(\sum_{i \in S(n_j)} \frac{\partial \mathbf{W}_{u_j}}{\partial q_j} \mathbf{W}_i^{n_j} \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_k} \right) \\
&= \mathrm{tr}\left(\frac{\partial \mathbf{W}_{u_j}}{\partial q_j} \sum_{i \in S(n_j)} \mathbf{W}_i^{n_j} \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_k} \right) \\
&= \mathrm{tr}\left(\frac{\partial \mathbf{W}_{u_j}}{\partial q_j} \frac{\widehat{\partial \mathbf{M}_{n_j}}}{\partial q_k} \right)
\end{aligned} \tag{3.47}
$$

where

$$
\frac{\widehat{\partial \dot{\mathbf{M}}_i}}{\partial q_k} = \mathbf{M}_i \frac{\partial \dot{\mathbf{W}}_i^T}{\partial q_k} + \sum_{j \in K(i)} \mathbf{R}_j \frac{\widehat{\partial \dot{\mathbf{M}}_j}}{\partial q_k} \tag{3.48}
$$

$$
\frac{\widehat{\partial \mathbf{M}_i}}{\partial q_k} = \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_k} + \sum_{j \in K(i)} \mathbf{R}_j \frac{\widehat{\partial \mathbf{M}_j}}{\partial q_k}. \tag{3.49}
$$

3.9 Objective Function

Constraints determine a large portion of the motion specification. The fact that there are significantly fewer constraints than unknowns in the spacetime optimization specification, allows the objective function to control the behavior of the remaining degrees of freedom.

The spacetime optimization finds values of $\mathbf{q}(t)$ and t for which the objective is minimized. In the spacetime framework an *objective function* (E) provides the means to modify the quality of motion. Ideally, one would want to control things like agility or emotional state, or other entities difficult to specify mathematically.

We represent the objective function as a weighted sum of a number of different smaller sub-objectives measuring different qualities of motion.

$$E = \sum_i w_i E_i(\mathbf{q}(t)) \tag{3.50}$$

The user controls the quality of motion by selecting the appropriate weights of different sub-objectives. We use a number of standard sub-objectives which we describe in detail.

3.9.1 Power Consumption

Objective functions that measure energy consumption are by far the most common in the spacetime optimization literature. We approximate the energy consumption measurement by the muscle force output as defined in Section 3.5)

$$E_{\text{power}} = \int_t \mathbf{Q}(\mathbf{q}(t), \mathbf{q}^m(t))^2 \tag{3.51}$$

3.9.2 Muscle Smoothness

The muscle smoothness sub-objective plays an instrumental role in maintaining the realistic muscle force output. We measure the muscle smoothness by the change in accelerations for each generalized muscle force

$$E_{\text{musc-smooth}} = \int_t \ddot{\mathbf{Q}}(\mathbf{q}(t), \mathbf{q}^m(t))^2 \tag{3.52}$$

3.9.3 Kinematic Smoothness

Although not useful during the dynamics optimizations, an objective measuring the kinematic smoothness is essential in both the spacetime fitting and reconstruction phases of the algorithm. It ensures frame coherence and avoids "glitches" in the animation. Again we measure smoothness with the magnitude of the DOF acceleration

$$E_{\text{kin-smooth}} = \int_t \ddot{\mathbf{q}}(t)^2 \tag{3.53}$$

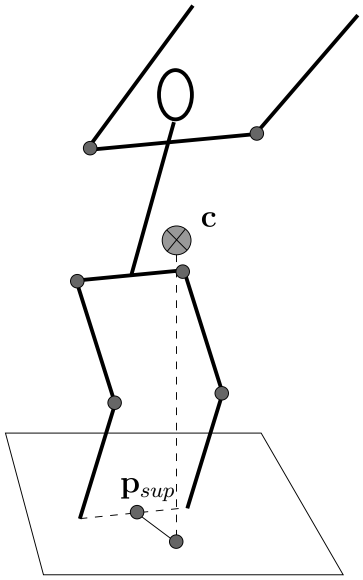

FIGURE 3.3. Static Balance measurement.

3.9.4 Static Balance

Spacetime optimization ensures dynamic balance simply by the virtue of the character not falling down during the motion sequence. The solution however can at times "coast" along the very edge of failure. In nature, such motions are rare since the uncertainty of the environment dictates locomotion with greater stability. We can improve the overall stability by introducing a sub-objective which measures the static balance of the character.

We measure the static balance as the distance between two points projected onto the floor surface: the body center of mass $\mathbf{c}$ and a centroid of the current floor contact points $\mathbf{p}_{sup}$. Naturally, the static balance can be measured only when the character is in the contact with the ground. When both legs are on the floor, the floor contact centroid is measured as the midpoint between the two footprints (Figure 3.9.4). Assuming that the floor is in the xz plane at $y = 0$ we measure the static balance as

$$E_{\text{bal}} = \int_t ([\mathbf{c}(\mathbf{q}(t))]_x - [\mathbf{p}_{sup}(\mathbf{q}(t))]_x)^2 + ([\mathbf{c}(\mathbf{q}(t))]_z - [\mathbf{p}_{sup}(\mathbf{q}(t))]_z)^2 \tag{3.54}$$

3.9.5 Floor Impact

By controlling the various characteristics of the forces exerted on the floor surface we can affect various aspects of the overall motion. Forces exerted by the floor, or any other obstacle for that matter, are modeled with extra degrees of freedom $\mathbf{q}^{\lambda}$. These DOFs appear in the mechanical constraint contribution to the Newtonian constraints (see Section 3.8). We can try to conserve the amount of force exerted by the obstacle with a sub-objective which would measure the magnitude of such forces

$$E_{\text{tot-impact}} = \int_t \mathbf{q}^{\lambda}(t)^2 \tag{3.55}$$

Conversely, we could minimize the maximum force exerted during the contact. For example, "stomping" in a walking/running sequence manifests itself with a large spike in the impact forces at the beginning of the floor contact. This stomping effect can be reduced by including a sub-objective that measures the maximum exerted mechanical force.

$$E_{\text{max-impact}} = \max_t \mathbf{q}^{\lambda}(t)^2 \tag{3.56}$$

A similar effect can be achieved by smoothing the impact forces

$$E_{\text{smooth-impact}} = \int_t \ddot{\mathbf{q}}^{\lambda}(t)^2 \tag{3.57}$$

3.9.6 Friction Stability

In naturally occurring motion, friction properties of the obstacles we encounter affect the way we move. For example, a slippery floor would modify our walking/running gait in order to reduce the likelihood of a fall. We include such considerations in the objective function by measuring the deviation of the normalized impact forces with respect to the floor surface unit normal $\mathbf{n}$

$$E_{\text{frict}} = \int_t \sum_i \|\mathbf{q}_i^{\lambda}(t)\| \cdot \mathbf{n} \tag{3.58}$$

3.9.7 Animator Specific Objective Functions

Naturally, the user can design her own sub-objective specific to the type of motion she wants to create. As an example, suppose we wanted to modify the running sequence in such a way that the knees are lifted significantly higher than in the original sequence. We can introduce a new objective which measures the elevation of two body points located at the knees $\mathbf{p}_{lk}$ and $\mathbf{p}_{rk}$. Assuming that the floor is in the xz plane at $y = 0$ such an sub-objective would have the following form

$$E_{\text{knee}} = \int_t [\mathbf{p}_{lk}(\mathbf{q}(t))]_y + [\mathbf{p}_{lk}(\mathbf{q}(t))]_y \tag{3.59}$$

Chapter 4

CONSTRUCTING THE SPACETIME MOTION MODEL

In this chapter we describe the methodology of constructing the dynamic model which closely correlates to the input motion. We have tried to apply the spacetime optimization algorithm to the full human character, and were not able to converge to a solution. In fact, there was no convergence of any kind mainly for the reason that the starting point of the optimization was not close enough to the solution, and the dynamic constraints simply could not be satisified no matter what optimization approach we took.

These convergence problems motivated the character simplification and consequently the need to map motion onto the more tractable solution space. When constructing the simplified spacetime motion model we drastically simplify the character and determine the spacetime constraints minimization problem whose solution closely matches the original motion.

4.1 Creating the Simplified Model

Instead of solving spacetime constraint optimizations on the full character, we construct a simpler character model which we then use for all spacetime optimizations. There are two important reasons for character simplification:

- reducing DOFs improves performance and facilitates convergence
- creation of an abstract model that contains only DOFs essential for the given motion captures a "low frequency" information of body movement. As a result, detailed motion in the input sequence will be preserved during the transformation process.

Simplified models capture the minimum amount of structure necessary for the input motion task, and therefore capture the "essence" of the input motion. Subsequent motion transformations modify this low-frequency representation while preserving the specific feel and uniqueness of the original motion. Our simplification process draws from ideas of biomechanics research [Blickhan 93]. We take the view that, abstractly speaking, natural

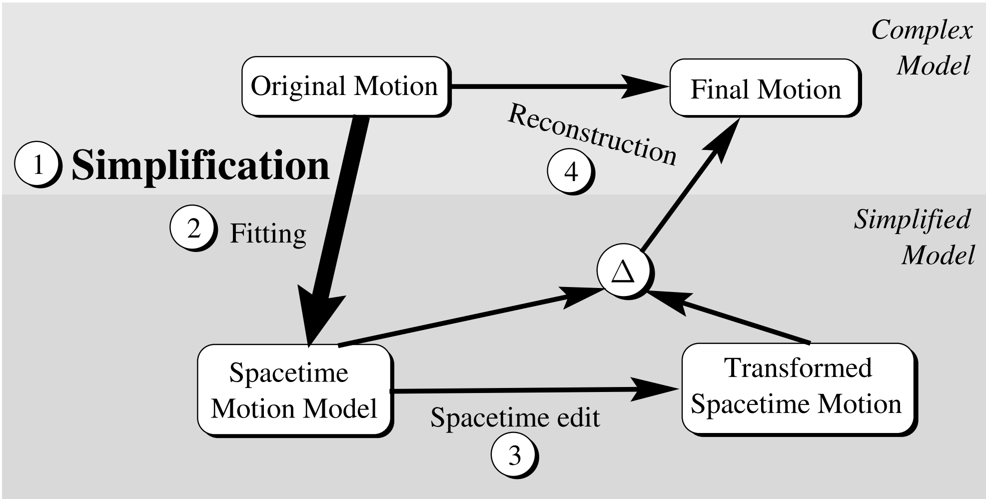

FIGURE 4.1. The Character Simplification Algorithm Stage.

motion is created by "throwing the mass around," or changing the *relative position* of body mass. With this in mind, it is easy to see how a human arm with more than 10 DOFs can be represented by a rigid object with only three shoulder DOFs without losing much of the "mass throwing" ability. Simplification of certain body parts also depends on the type of the input motion. For example, while the above mentioned arm simplification may work well for the human run motion, it would not make much sense for the ball-throw motion.

Simplification can reduce the number of kinematic DOFs, as well as muscle DOFs by a factor of two to five. Since each DOF is represented by hundreds of unknown coefficients during the optimization, simplification can reduce the size of the optimization by as many as 1000 unknowns. Furthermore, a character with fewer DOFs also creates simpler constraints which are significantly less nonlinear. In practice, the optimization has no convergence problems with the simplified character models.

Character simplification is performed manually. Fortunately, performing simplification does not require significant expertise. We abstract the parts of the body which do not seem to be extremely relevant to the motion.

The simplification process ensures that the overall mass distribution is preserved. So if a number of nodes are represented with a single object an attempt is made to match the mass, center of mass and moments of inertia of the new structure to be as close to the original structure as possible.

We apply three basic principles during this process:

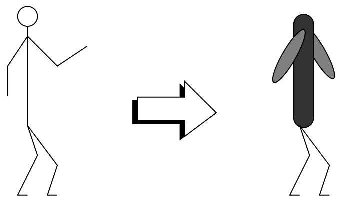

FIGURE 4.2. Simplification by removal of elbow and spine DOFs.

1. DOF removal
2. Node subtree removal
3. Exploit symmetric movement

4.1.1 DOF Removal

The most straightforward character simplification occurs when certain character DOFs have apparent marginal or even irrelevant function in the given motion sequence. For example, wrists and even elbows have minor influence on the dynamic properties of the running or walking motion. Consequently, we fix those DOFs to a constant value creating a single rigid structure for the entire arm (Figure 4.2).

When removing a DOF, body parts connected by that DOF are fused as if that part of the body is placed in a cast. The most straightforward way to fix the DOF value is to average the values over the entire animation sequence.

We obtain a more dynamically accurate measurement of the fixed DOF value, by minimizing the deviation of the relative center of mass of the character subtree affected by the given DOF. For example, when considering the removal of q_j which connects hierarchy primitives P_i and P_{i+1} we look for the value of q_j which has the smallest deviation of the relative center of mass of the subtree rooted at P_i.

4.1.2 Node Subtree Removal

In some cases of high-energy motion the entire subtree of the character hierarchy can be replaced with a single object, usually a mass point with three translational DOFs. Naturally, the mass of the point would equal the total mass contained within the node subtree.

For example, the upper body of a human character can be reduced to a mass point for various jumping motion sequences where the upper body catapults in the direction of the

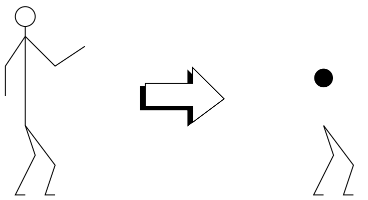

FIGURE 4.3. Simplification by reducing the upper body to a mass point.

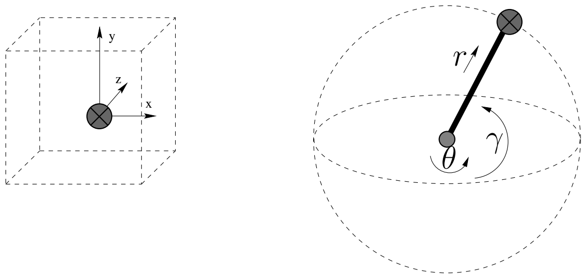

FIGURE 4.4. Translational and polar representation of the mass point range of motion.

jump. This type of abstraction assumes that all fine movements of the subtree have the goal of displacing the mass.

In the jump example, the arms usually have a large swooping motion which rapidly moves the mass distribution of the upper body from one side of the body to the other, and provides added momentum for displacing the body torso which contains most of the upper body mass. When we represent the entire upper body with a single point of mass (Figure 4.3), we establish that the overall motion of the arms will not be fundamentally changed by any subsequent transformation. Any displacement in the center of mass introduced by the motion transformation process will be reflected by a differential displacement of the upper body pose that will match the appropriate center of mass and minimally deviate from the original arms and torso movement characteristics. Thus, if both arms were moving forward and backward in unison in the original motion sequence (e.g. broad jump), and we reduced the upper body to a single point, we cannot produce a transformed motion sequence which would have arms moving in separate directions.

Since the range of motion of the upper body center of mass is restricted by its inherent kinematic structure, we must impose the same restriction on the mass point abstraction. Without such bounds, the mass point could freely move into configurations which could not be correlated to any feasible configuration of the upper body. Often, a crude bound on

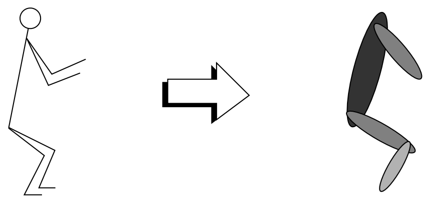

FIGURE 4.5. Simplification by exploiting symmetric motion.

each of three mass point DOFs suffices to alleviate this problem. Such bounds create a 3D box and keep the mass point within it. In rare cases when finer bounds are necessary, we can choose different DOFs to represent the freedom of movement for the mass point. For example, we can use 2 rotational DOFs and one telescoping translational DOF to have a polar coordinate representation for the motion of the mass point (Figure 4.4). This representation would allow us to apply more natural bounds on the mass point range of motion since this representation closely relates to pivoting around the waist motion of the upper body center of mass.

4.1.3 Exploit Symmetric Movement

In some motion sequences certain body parts move in similar ways such as both arms swinging in the same direction or a jump with both legs pushing off at the same time. This symmetric movement of different node subtrees allows us to apply yet another form of abstraction: we can represent two subtrees with just one. In a broad jump animation, for example, both legs move in unison. We reduce the two legs representation of the full character with a single leg, essentially turning the human character into a monopode (Figure 4.5). A newly created leg is placed at the center of the body and its foot is guided by the midpoint between the two feet of the original character. Again, we ensure that each primitive of the symmetric abstraction subtree weighs as much as the sum of the corresponding primitives in the full character. For example, a monopode thigh weighs twice as much as a human thigh.

Although it would appear that a character with one leg would have greater difficulty balancing than the original bipedal character, we have not found that such a change affects our transformation algorithm in any way. Naturally, this form of abstraction ensures that subsequently transformed motion will always have symmetric subtrees moving in unison.

4.2 Mapping the Motion to a Simplified Character

Once the character has been simplified the original motion can be mapped onto the simplified model. Since the simplified character has significantly fewer DOFs, this problem is over-determined. We define *handles* to aid us in the motion transfer process by correlating essential properties between complex and simplified motion sequences.

Handles are multi-valued time-varying functions that can be evaluated on both complex and simplified character models. All handles depend on the character pose defined by the vector of values for each DOF $\mathbf{q}(t_i)$. We already mentioned that each handle can be applied to the domain of the original character DOFs $\mathbf{q}_0$, as well as to the domain of the simplified character DOFs $\mathbf{q}_s$. To avoid confusion, when a handle is evaluated on the original character we refer to it as $\mathbf{h}(\mathbf{q}_0)$, whereas when the same handle is applied to the simplified character we use a primed function $\mathbf{h}'(\mathbf{q}_s)$.

Handles often represent intuitive measurements of various body properties such as 3D point positions, 3D directions, distance between two assigned body points, etc. Although our algorithm does not in any way depend on the specifics of handle description, in practice handles generally fall into several basic groups:

- body point handle
- center of mass handle
- direction handle
- distance handle
- expression handle

We will describe the computation of the values and derivatives for each handle group.

4.2.1 Body Point Handle

Body point handles are fixed points on the character's body. One such handle is the foot—ground contact point. On the full character this point is usually placed at the ball of the foot. Since simplified characters often do not have the foot primitive, the foot-ground contact point is the lowest body point on the leg.

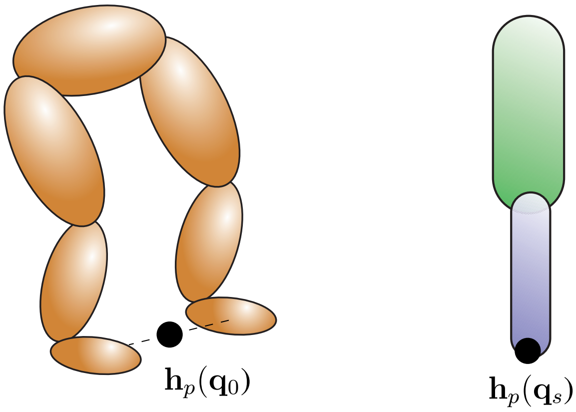

FIGURE 4.6. Body point handle correlates the foot contact points on the original and simplified model.

We define body point handles mathematically just like any other point on the body (Section 3.3.2)

$$\mathbf{h}_p = \mathbf{W}_i\mathbf{x} \tag{4.1}$$

where $\mathbf{x}$ is the body point in the local coordinate frame of the i-th node. This definition also shows that body point handles depend on all DOFs in the transformation path of the i-th node. So it's derivative with respect to q_k is

$$\frac{\partial \mathbf{h}_p}{\partial q_k} = \frac{\partial \mathbf{W}_i}{\partial q_k}\mathbf{x}. \tag{4.2}$$

4.2.2 Center of Mass Handle

Another widely used handle correlates centers of mass of various subtrees within the original and the simplified character hierarchy. For example, during the simplification process we may choose to reduce the entire arm down to a single rigid object. Subsequently, we use the arm's center of mass to guide the motion of the simplified arm.

Let $\mathbf{c}_i$ be the local center of mass of the node i. We also define two index sets $S(i)$ and $K(i)$ that contain all node indices in the subtree rooted at node i, and all node indices parented by node i, respectively. The center of mass handle of the subtree rooted at i-th node is

$$\mathbf{h}_{com} = \sum_{j \in S(i)} \frac{m_j \mathbf{W}_j \mathbf{c}_j}{\sum_{k \in S(i)} m_k}. \tag{4.3}$$

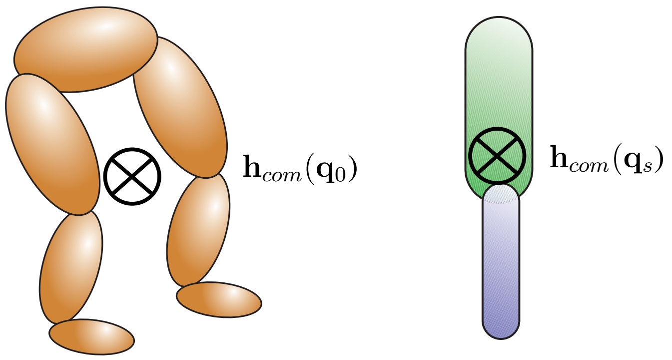

FIGURE 4.7. Center of mass handle correlates the human lower body and it's simplified counterpart.

In order to compute center of mass handles efficiently, we represent the same quantity as

$$\mathbf{h}_{com} = \|\mathbf{W}_i \hat{\mathbf{c}}_i\| \tag{4.4}$$

where

$$\hat{\mathbf{c}}_i = m_i \mathbf{c}_i + \sum_{j \in K(i)} \mathbf{R}_j \hat{\mathbf{c}}_j. \tag{4.5}$$

$\hat{\mathbf{c}}_i$ is a homogeneous vector representing a recursively defined local center of mass of the subtree rooted at node i.

It is easy to see that since $\mathbf{c} = [c_x \, c_y \, c_z \, 1]^T$ the fourth component of $\hat{\mathbf{c}}_i$ by definition contains the total mass under the subtree rooted at node i. Thus, the normalization operation in the equation 4.4 performs the division by the total mass evident in the non-recursive definition in the equation 4.3.

This recursive formulation allows us to efficiently evaluate the derivatives by propagating the values of $\hat{\mathbf{c}}_i$ from the leaves upwards.

Similar recursive definition allows us to efficiently compute the derivatives of the center of mass handle

$$\frac{\partial \hat{\mathbf{c}}_i}{\partial q_k} = \|m_i \mathbf{c}_i + \sum_{j \in K(i)} \frac{\partial \mathbf{R}_j}{\partial q_k} \hat{\mathbf{c}}_j + \mathbf{R}_j \frac{\partial \hat{\mathbf{c}}_j}{\partial q_k}\| \tag{4.6}$$

$$\frac{\partial \mathbf{h}_{com}}{\partial q_k} = \|\frac{\partial \mathbf{W}_i}{\partial q_k} \hat{\mathbf{c}}_i + \mathbf{W}_i \frac{\partial \hat{\mathbf{c}}_i}{\partial q_k}\| \tag{4.7}$$

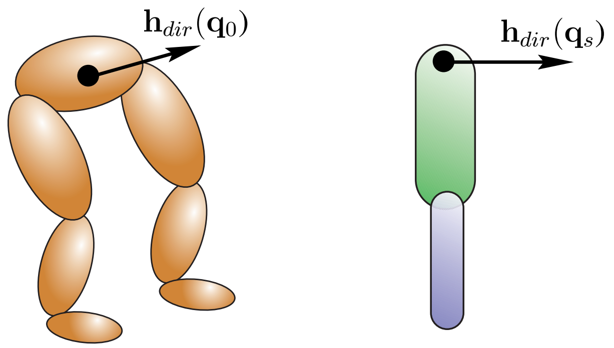

FIGURE 4.8. Direction handle correlates the torso orientation of the original and simplified model.

Note that since the fourth component of $\mathbf{c}_i$ (total subtree mass) does not depend on any DOFs the derivatives can freely propagate through the normalization operator

$$\frac{\partial \|\mathbf{W}_i \hat{\mathbf{c}}_i\|}{\partial q_k} = \|\frac{\partial \mathbf{W}_i \hat{\mathbf{c}}_i}{\partial q_k}\| \tag{4.8}$$

During the hierarchy traversal we precompute the values of $\hat{\mathbf{c}}_i$ and the sparse vector $\frac{\partial \hat{\mathbf{c}}_i}{\partial q_k}$ for each node i. Once those values are computed, the evaluation of $\mathbf{h}_{com}$ values and derivatives for any node in the hierarchy is constant time.

4.2.3 Direction Handle

Direction handles are often used to constrain or measure the direction of a particular body primitive. They are also used to maintain the relative direction between two body parts. For example, we use direction handles to measure the direction in which the shoulders are pointing during the running sequence. We specify the direction by two body points $\mathbf{p}_i$ and $\mathbf{p}_j$. If we define the direction as a vector pointing from $\mathbf{p}_j$ to $\mathbf{p}_i$

$$\mathbf{d} = \mathbf{p}_i - \mathbf{p}_j \tag{4.9}$$

then the direction handle is the unit direction between points $\mathbf{p}_i$ and $\mathbf{p}_j$

$$\mathbf{h}_{dir} = \frac{\mathbf{d}}{\|\mathbf{d}\|}. \tag{4.10}$$

With a few chain rule applications we arrive at the direction handle derivative

$$\frac{\partial \mathbf{h}_{dir}}{\partial q_k} = \frac{\frac{\partial \mathbf{d}}{\partial q_k}}{\|\mathbf{d}\|} - \frac{(\mathbf{d} \cdot \frac{\partial \mathbf{d}}{\partial q_k})\mathbf{d}}{\|\mathbf{d}\|^3} \tag{4.11}$$

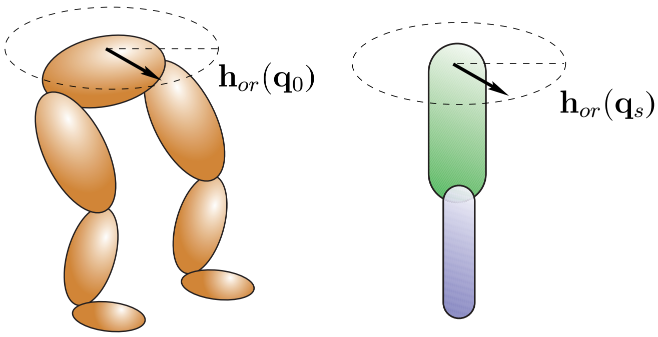

FIGURE 4.9. Orientation handle correlates the human lower body orientation with it's simplified counterpart.

where

$$\frac{\partial \mathbf{d}}{\partial q_k} = \frac{\partial \mathbf{p}_i}{\partial q_k} - \frac{\partial \mathbf{p}_j}{\partial q_k}. \tag{4.12}$$

This formulation contains an inherent singularity which occurs when two points defining the direction coincide. In practice, however, such a configuration never occurs. Given that two points are both part of the character body (often internal to the body), their coincidence would imply an unnatural self-intersecting body configuration.

4.2.4 Orientation Handle

Orientation handles control the specific direction of the body movement. Unlike a direction handle, an orientation handle is a one dimensional handle that can be intuitively thought of as a direction vector projected onto a plane; more specifically, the angle of such a projection.

We may want, for example, to enforce a certain turning motion on the walking human sequence. In this case we are interested in the torso orientation with respect to the floor plane. Similarly, we may want to constrain the final body orientation in a jump sequence.

We can design the orientation handle as a projection of an expression defining the direction handle. Unfortunately, this formulation introduces singularities. For example a direction handle pointing along the normal of the plane would produce an undefined orientation.

In order to avoid singularities and significantly simplify the evaluation of values and derivatives we allow orientation handles only on quaternion nodes within the hierarchy.

We use the fiber bundle twist theory [Shoemake 94] to define the twist angle α around an axis aligned along one of the 3 principal quaternion directions $q_i \in \{q_x, q_y, q_z\}$

$$\tan(\alpha/2) = q_i/q_w. \tag{4.13}$$

Therefore we define the orientation handle to represent the twist angle

$$h_{or} = 2\arctan(q_i/q_w)$$

which has the following derivatives:

$$\frac{\partial h_{or}}{\partial q_i} = \frac{1}{q_w\left(1 + \frac{q_i^2}{q_w^2}\right)} \tag{4.14}$$

$$\frac{\partial h_{or}}{\partial q_w} = -\frac{q_i}{q_w^2\left(1 + \frac{q_i^2}{q_w^2}\right)} \tag{4.15}$$

Alternatively, we can define the orientation handle to represent the angle tangent which leads to much simpler expressions without any loss of generality

$$h_{or} = q_i/q_w \tag{4.16}$$

$$\frac{\partial h_{or}}{\partial q_i} = \frac{1}{q_w} \tag{4.17}$$

$$\frac{\partial h_{or}}{\partial q_w} = -\frac{q_i}{q_w^2}. \tag{4.18}$$

4.2.5 Distance Handle

A distance handle is another scalar handle. It measures the distance between two body points, although there is nothing in the definition which would preclude one of the points not being on the body.

When kinematic topology has been drastically changed during the simplification, it is often useful to use distance handles to correlate various body points. For example, suppose we have changed the usual kinematics of a leg in such a way that the knee is no longer represented by the angular joint, but by a prismatic joint instead. This change drastically modifies the movement of the lower leg. We can, however, still keep a rough measurement of the leg extension by tracking the hip-to-foot distance with the distance handle. The definition of the distance handle is straightforward

$$h_{dist} = \|\mathbf{p}_i - \mathbf{p}_j\| \tag{4.19}$$

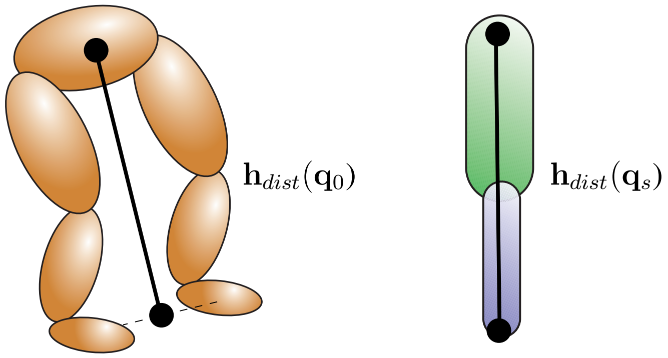

FIGURE 4.10. Distance handle correlates the human lower body length with it's simplified counterpart.

4.2.6 Expression Handle

We can also use an arbitrary expression of various body properties as a handle. In fact, distance and direction handles are expressions of body points and can consequently be thought of as expression handles. Various expression operators are provided for this purpose including vector addition, subtraction, scaling, normalization, dot product, norm-2 vector distance, etc. The automatic differentiation mechanism ensures that all derivatives can be computed with existing body properties and expressions.

4.2.7 Constructing the Simplified Character Motion

In order to match two animations, we ensure equality between the corresponding handles. For example, if we reduced the human upper body down to a mass point, we would use the center of mass handle to correlate two animations. When two legs are reduced to one, we would use the foot—floor contact point handle, defined as the midpoint between two foot contact points and equate it with the foot point handle of the monopode.

Let us define the collection of all handles of the complex motion as $\mathbf{h}_0(\mathbf{q}_0(t))$, and let $\mathbf{h}_s(\mathbf{q}_s(t))$ be the corresponding simplified motion handles. We find the motion of the simplified character by solving

$$\min_{\mathbf{q}_s(t_i)} (\mathbf{h}_0(\mathbf{q}_0(t_i)) - \mathbf{h}_s(\mathbf{q}_s(t_i)))^2 \tag{4.20}$$

for each frame t_i. This process is equivalent to solving an inverse kinematics problem for each time frame of the animation. Naturally, there should be *at least* as many handles

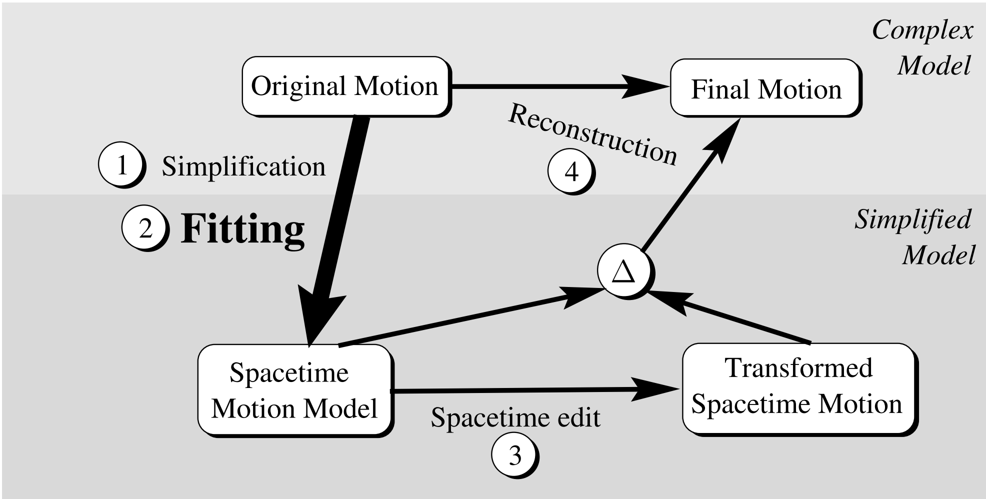

FIGURE 4.11. The Motion Fitting Algorithm Stage.

as there are DOFs in the simplified character. Under these conditions way the simplified motion is fully determined by $\mathbf{h}_s(\mathbf{q}_s(t))$.

4.3 Spacetime Fitting

Handles help us map the original motion onto the simplified character. However, this motion is no longer dynamically correct. Before we can edit motion with spacetime constraints we need to create not only a dynamically correct but also realistic motion for simplified model. In other words, we need to find the spacetime optimization problem whose solution comes very close to the simplified model motion we computed in section 4.2. Subsequent sections describe our approach to finding the appropriate muscles, spacetime constraints and the objective function which would yield the motion closely matching the input sequence.

4.3.1 Constraints

Most of the pose and mechanical constraints fall out of the nature of the input motion. For example, in a run or walk sequence we specify mechanical point constraints during each period the foot is in contact with the floor. Similarly, a leg kick animation defines a pose constraint at the time the leg strikes the target. We avoid specifying extraneous constraints that are not essential for the input motion, since they reduce the flexibility of the subsequent spacetime editing process.

The model simplification process may also introduce additional constraints. For example, if the upper body was reduced to a mass point, the mass point DOFs need to be restricted to stay within the bounds of the upper body center of mass. This ensures that movement of mass points can never cause an improper human configuration.

Additional pose constraints can be introduced for further control during motion editing. For example, we can introduce the obstacle hurdle into the human jump motion environment which forces the character to clear a certain height during flight.

4.3.2 Muscles

As described in Section 3.5 Each kinematic DOF q_i has a corresponding damped generalized muscle force

$$Q_i = k_s(q_i - q_i^m) - k_d(\dot{q}_i - \dot{q}_i^m) \tag{4.21}$$

where q_i^m is the additional muscle DOF that is often interpreted as the desired value of q_i. Each muscle an input motion sequence has different coefficients k_s and k_d. We treat these coefficients as unknown during the spacetime fitting stage. In some cases a better motion fit can be achieved by allowing k_s and k_d to vary through time, even though we use significantly fewer coefficients compared to regular DOFs $\mathbf{q}(t)$. This prevents the scaling and damping coefficients from varying wildly over the coarse of the animation.

4.3.3 Objective Function

There has been much research into the optimality of motion in nature [Alexander 80, Alexander 89, Alexander 90, Alexander 91, Pedotti 78]. We, however, avoid guessing the right objective function altogether. Instead, we rely on the fact that the starting motion is very close to the optimum. At first, our objective measures the deviation from the original motion (E_d) as described by Equation 4.20. We also include the muscle smoothness objective component $E_m = \int_i \ddot{\mathbf{q}}^{m^2}$ at all times. The spacetime objective is a weighted sum of the two objective components.

$$E = w_d E_d + E_m \tag{4.22}$$

Once Newtonian constraint residuals become small, we gradually decrease w_d all the way to zero. The existence of the E_d component early in the optimization process prevents the optimization method from diverging from the initial motion until the spacetime constraints are satisfied (i.e. until the dynamics integrity of the motion has been established). This approach ensures that the spacetime minimization process stays near the input motion, while at the same time keeping the muscle forces smooth.

Upon convergence, we end up with a spacetime problem definition whose solution is very close to the original motion. With the spacetime optimization problem successfully constructed, the intuitive "control knobs" of the spacetime constraints formulation can be edited to produce a nearly inexhaustible number of different realistic motion sequences.

Chapter 5

MOTION TRANSFORMATION AND RECONSTRUCTION

This chapter describes the process of modifying the motion sequence and producing the final transformed animation. This process can be viewed as a two step procedure. The first step involves modification of the spacetime constraints formulation. The second step reconstructs the final motion sequence from the results of the modified spacetime solution.

5.1 Spacetime Edit

A spacetime constraints parameterization provides a powerful and intuitive control of all aspects of the dynamic motion sequence. In fact, the ability to edit motion with high-level controls is the main reason for creating of the spacetime constraints formulation from the input motion. All possible modifiable spacetime properties can be classified into the following groups: pose and environment constraints, explicit kinematic and dynamic properties of the character, and the objective function.

5.1.1 Modifying Constraints

By changing existing constraints the user can rearrange foot placements both in space and time. For example, a human run sequence can be changed into a zig-zag run on an uphill slope by moving the floor contact constraints wider apart and progressively elevating them. Similarly, a karate kick motion sequence can reposition the kick placement pose constraint. Changing the initial or final pose constraint is another example of constraint modification.

The constraint timing can also be changed: extending the floor contact time duration of one leg creates the animation which gives the appearance of favoring one leg.

We can also introduce new obstacles along the running path, producing new constraints that, for example, require legs to clear a certain height during the flight phase of the run.

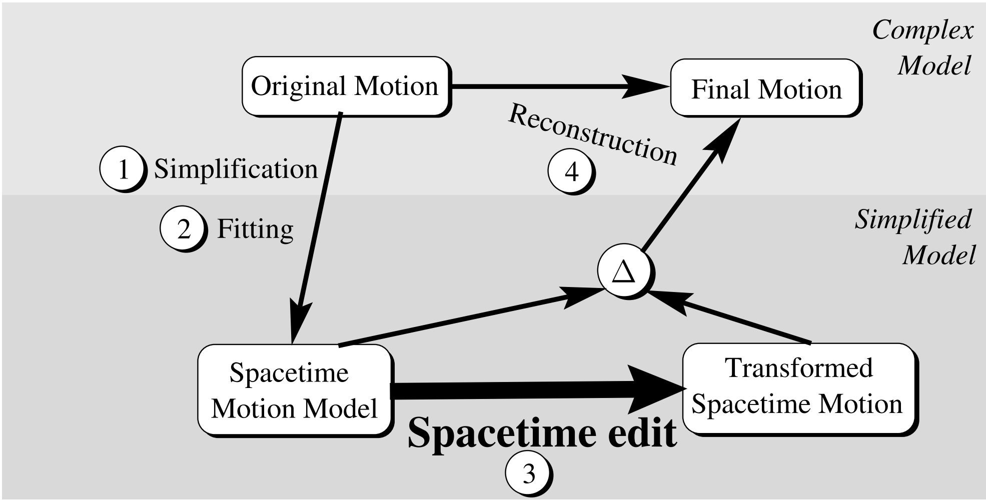

FIGURE 5.1. The Spacetime Edit Algorithm Stage.

5.1.2 Modifying Character Kinematics

In addition to changing constraints, we can also retarget motion to a different character. We achieve this by modifying the character kinematics. The simplest form of change to the character is modifying the dimensions of various limbs. For example we can retarget the walking sequence of a man into a walking sequence of a child whose limb proportions are significantly different.

We are also free to remove degrees of freedom, such as making the character's leg stiff by removing the knee joint. Alternatively, we can just reduce the range of motion of the knee by placing bounds on the joint movement. We can also change the type of DOF altogether. For example, we can change the joint angle DOF into a prismatic joint, or a universal joint into a ball joint. Such changes of a DOF type can produce significantly different dynamics of the character movement.

A more drastic change of the character kinematics can remove or introduce new limbs. This sort of topological change also changes a number of DOFs.

5.1.3 Modifying Character Dynamic Properties

Mass distribution, and muscles represent the dynamic properties of the character. Changing either of these properties affects the resulting motion. For example, making one leg heavier can produce a leg dragging walking gait.

Various muscle properties of the character can also affect the look of transformed motion. We can limit the force output of the muscles, forcing the character to compensate by using other muscles. Alternatively we can remove the activation ability of the muscle, turning a muscle into a passive spring, which would also force the character to use different sources of energy to achieve a given task. We can further tweak the muscle properties by changing the active and the damping coefficients of of the muscle representation.

We can also affect the environment of the run by changing the gravity constant, producing a human running sequence on the moon surface.

5.1.4 Modifying Objective function

Finally, the overall "feel" of the motion can be changed by adding additional appropriately weighted objective components. Section 3.9 describes a number of possible objective function components that can be used. For example, we can produce a softer looking run by adding an objective component that minimizes floor impact forces. Or we can make the run look more stable by including a measure of static balance in the objective.

After each edit we re-solve the spacetime optimization problem and produce a new transformed animation. Since the optimization starting point is near the sought solution, and all dynamic constraints are satisfied at the outset, optimization converges rapidly. In practice, while the initial spacetime optimization may take more than 10 minutes to converge, spacetime optimizations during the editing process take less than a minute.

5.2 Motion Reconstruction

In order to create the transformed animation of the full character model, we reconstruct the final motion from the input motion and two simplified spacetime motions. Intuitively, we modify the low-frequency dynamic properties of the motion contained within the simplified spacetime motion representation, while we preserve the high-frequency details contained in the input motion data.

The reconstruction process relies heavily on both spacetime constraints and motion handles described in Section 4.2.

Any change in the pose or mechanical constraints introduced during the spacetime editing stage are mapped to their full character equivalents. All new constraints which were

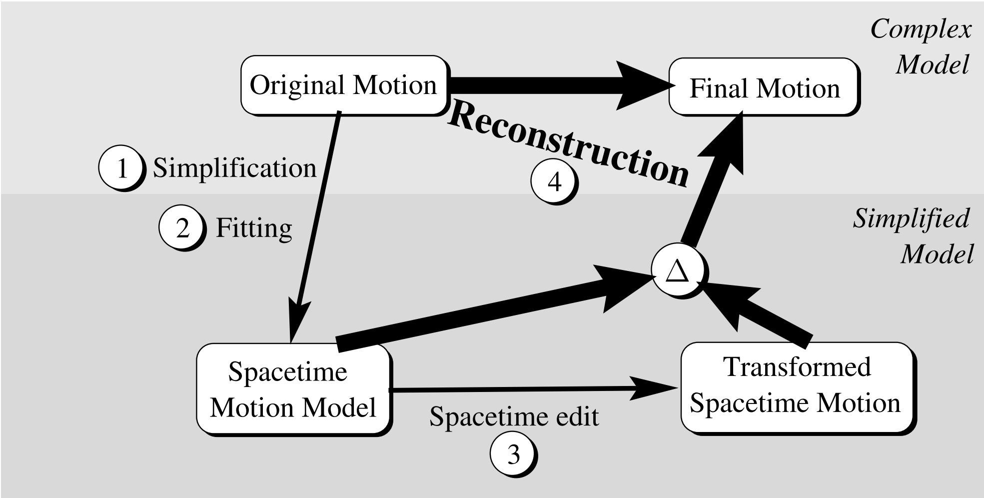

FIGURE 5.2. The Motion Reconstruction Algorithm Stage.

created during the editing process are also created for the final motion sequence. For example, a change in the foot placement constraints is mapped onto foot constraints of the full character.

Once all constraints have been put in place, we displace the original motion by the *difference* between the transformed and initial spacetime motion sequences. Since these motion sequences vary in the number of DOFs we use handles to apply the displacements. Having completed the spacetime editing stage, we have three distinct sets of handles[1]

- input motion handles $\mathbf{h}_0(\mathbf{q}_0)$
- spacetime fit handles $\mathbf{h}_s(\mathbf{q}_s)$
- transformed spacetime handles $\mathbf{h}_s(\mathbf{q}_t)$

We define the final motion handles as

$$\mathbf{h}_0(\mathbf{q}_f) = \mathbf{h}_0(\mathbf{q}_0) + (\mathbf{h}_s(\mathbf{q}_t) - \mathbf{h}_s(\mathbf{q}_s)) \tag{5.1}$$

Since the right side of the equation is known, it would seem that solving for the inverse-kinematics-like problem of finding $\mathbf{q}_f$ that satisfies the Equation 5.1 would complete the

[1]For clarity, we omit the explicit time dependency of handles and DOFs.

reconstruction. Unfortunately, the number of handles is *considerably smaller* than the number of DOFs in the full character, so this problem is highly under-determined. Consequently, we cannot directly solve for $\mathbf{q}_f$ without accounting for the extra DOFs.

We formulate the reconstruction process as a sequence of per frame subproblems:

$$\begin{array}{ll} \min_{\mathbf{q}_f} & E_{dm}(\mathbf{q}_0, \mathbf{q}_f) \\ \text{subject to} & \begin{cases} \mathbf{C}(\mathbf{q}) = 0 \\ \mathbf{h}_0(\mathbf{q}_f) = \mathbf{h}_0(\mathbf{q}_0) + (\mathbf{h}_s(\mathbf{q}_t) - \mathbf{h}_s(\mathbf{q}_s)) \end{cases} \end{array} \tag{5.2}$$

Simply stated, we follow the transformed handles and satisfy all constraints ($\mathbf{C}(\mathbf{q})$) while we try to be as close as possible to the original motion.

Of course, we still need to formulate "the closeness to the original motion." A simple objective function that measures the deviation of each DOF $E_{dd} = \int_t (\mathbf{q}_f - \mathbf{q}_0)^2$ produces undesirable results. Each DOF needs to be carefully scaled both with respect to what it measures (joint angles measure radians, translational DOFs measure meters), and with respect to its importance within the character hierarchy. For example, the change of the hip joint DOF affects the overall motion more significantly than the same amount of change applied to the ankle joint.

At first, we tried to use a simple extention to the joint angle metric. We scaled the DOF vector $\mathbf{q}$ by the diagonal matrix $\mathbf{D}$ which contained diagonal componenets of the mass metric.

$$E_D = \mathbf{q}^T \mathbf{D} \mathbf{q}$$

Although, an improvement over the unscaled DOF values, this approach still failed to capture the pose comparison metric. In order to get the accurate metric for pose compariso, we designed a completely new objective that measures the amount of displaced mass between the two poses.

5.2.1 Minimum Displaced Mass

Given two character poses described by the DOF values $\bar{\mathbf{q}}$ and $\mathbf{q}$ we compute the total amount of displaced mass $E_{dm}(\bar{\mathbf{q}}, \mathbf{q})$ when transforming from pose $\bar{\mathbf{q}}$ to pose $\mathbf{q}$. This metric is loosely analogous to the measurement of the dynamic power consumption, with the exception that we compare two kinematic (not dynamic) states. Total displaced mass is the sum of mass displacements for each node k in the hierarchy $E_{dm} = \sum_k E_k$.

We compute the node mass displacement E_k as a body point (p_i) displacement scaled with its mass (μ_i) integrated over all body points of the node k

$$E_k = \int_i (\mathbf{p}_i - \bar{\mathbf{p}}_i)^2 \mu_i \, dx \, dy \, dz,$$

where each "bar"-ed symbol refers to quantities computed at pose $\bar{\mathbf{q}}$.

Since we are only interested in the *relative* mass displacement, we compute the body positions invariant of the global rotation and translation. In other words, if

$$\mathbf{p}_i' = \mathbf{R}_0 \mathbf{R}_1 \cdots \mathbf{R}_{j-1} \mathbf{R}_j x_i$$

is the world space position of the body point x_i in the node j of the character hierarchy, and transformation $\mathbf{R}_0$ contains the global rotation and translation of the hierarchy, we define

$$\mathbf{p}_i = \mathbf{R}_1 \cdots \mathbf{R}_{j-1} \mathbf{R}_j \mathbf{x}_i = \mathbf{W}_j \mathbf{x}_i$$

This notation allows us to simplify E_k

$$\begin{aligned} E_k &= \int_i \mu_i (\mathbf{W}_i \mathbf{x}_i \mathbf{W}_i \mathbf{x}_i - 2 \mathbf{W}_i \mathbf{x}_i \overline{\mathbf{W}}_i \mathbf{x}_i) \;\; dx \, dy \, dz \\ &= \operatorname{tr}\Big(\mathbf{W}_i \mathbf{M}_i \mathbf{W}_i^T - 2 \mathbf{W}_i \mathbf{M}_i \overline{\mathbf{W}}_i^T + \overline{\mathbf{W}}_i \mathbf{M}_i \overline{\mathbf{W}}_i^T \Big) \\ &= \operatorname{tr}\Big(\mathbf{W}_i \mathbf{M}_i (\mathbf{W}_i - 2\overline{\mathbf{W}}_i)^T + \overline{\mathbf{W}}_i \mathbf{M}_i \overline{\mathbf{W}}_i^T \Big) \end{aligned}$$

where mass matrix tensor $\mathbf{M}_i$ is computed as an integral over all body points $\mathbf{x}_j$ of outer products scaled by node mass (m_i) for each node i.

$$\mathbf{M}_i = m_i \iiint_j \mathbf{x}_j \mathbf{x}_j^T \, dx \, dy \, dz$$

When using E_k while solving the optimization problem in 5.2 the only unknowns in E_k are contained within transformation paths $\mathbf{W}$ while the initial transformation paths $\overline{\mathbf{W}}$ are considered to be known. Therefore, the term $\overline{\mathbf{W}}_i \mathbf{M}_i \overline{\mathbf{W}}_i^T$ does not depend on any unknowns and can be avoided altogether.

We also compute derivatives with respect to kinematic DOFs

$$\begin{aligned} \frac{\partial E_k}{\partial q_j} &= \operatorname{tr}\left(\frac{\partial \mathbf{W}_i}{\partial q_j} \mathbf{M}_i (\mathbf{W}_i - 2\overline{\mathbf{W}}_i)^T + \mathbf{W}_i \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_j} \right) \\ &= \operatorname{tr}\left(2(\mathbf{W}_i - \overline{\mathbf{W}}_i) \, \mathbf{M}_i \frac{\partial \mathbf{W}_i^T}{\partial q_j} \right) \end{aligned}$$

Both E_k and $\frac{\partial E_k}{\partial q_j}$ can be computed efficiently by recursively computing the subexpressions $\mathbf{M}_i\mathbf{W}_i{}^T$ and $\mathbf{M}_i\frac{\partial \mathbf{W}_i{}^T}{\partial q_j}$ for each node in the hierarchy. We find that E_{dm} performs extremely well as a measure of closeness between two motions.

Since the reconstruction process is performed separately for each frame the resulting motion may on occasion appear non smooth. We correct this by defining intervals of animation where improved smoothness is required. The problem defined in Equation 5.2 is then solved collectively over the entire interval with the added smoothness objective $E_{\text{smooth}} = \ddot{\mathbf{q}}^2$

$$\begin{array}{ll} \min_{\mathbf{q}_f(t)} & \int_i E_{dm}(\mathbf{q}_0(t), \mathbf{q}_f(t)) + \int_i E_{\text{smooth}} \\ \text{subject to} & \left\{ \begin{array}{l} \mathbf{C}(\mathbf{q}(t)) = \mathbf{0} \\ \mathbf{h}_0(\mathbf{q}_f(t)) = \mathbf{h}_0(\mathbf{q}_0(t)) + (\mathbf{h}_s(\mathbf{q}_t(t)) - \mathbf{h}_s(\mathbf{q}_s(t))) \end{array} \right. \end{array} \tag{5.3}$$

We also have control over the trade-off between following handle constraints and minimizing the mass displacement objective. For example, the user can increase the importance of the mass displacement objective, to produce motion which has closer resemblance to the input animation at the expense of a slight violation of handle constraints. In this case, we include the handle constraint in the objective function itself

$$\begin{array}{ll} \min_{\mathbf{q}_f} & w_{dm}E_{dm}(\mathbf{q}_0, \mathbf{q}_f) + w_h(-\mathbf{h}_0(\mathbf{q}_f) + \mathbf{h}_0(\mathbf{q}_0) + (\mathbf{h}_s(\mathbf{q}_t) - \mathbf{h}_s(\mathbf{q}_s)))^2 \\ \text{subject to} & \left\{ \mathbf{C}(\mathbf{q}) = 0 \right. \end{array} \tag{5.4}$$

and adjust the relative importance by modifying the weights w_{dm} and w_h.

Chapter 6

RESULTS

We have applied our algorithm to two very different motion capture sequences (human run and human jump) in an attempt to showcase the versatility of the underlying framework. This chapter describes the specific details of creating a large spectrum of animations from a single motion capture sequence. We follow with a discussion on the limitations of this approach.

All of the resulting transformed motion sequences would be difficult to create with existing motion editing tools. While it is conceivable that a number of constraints could be used with the existing motion editing tools to enhance realism, for some sequences it would require a significant amount of work, on a par with creating the realistic motion sequence from scratch with keyframing. We would like to emphasize the minimal number of intuitive changes we performed for each generated motion sequence.

6.1 Processing Captured Motion Data

We used the high detail optical motion capture data as the input sequences for our algorithm. However, before we could apply our algorithm to this data we performed several pre-processing steps which ensured the maximum veracity of the resulting captured animation. Although most of the currently available motion capture data comes in already preprocessed form, we outline our processing technique for the sake of completeness.

The optical system used seven cameras which tracked the motion of 27 cat's-eye markers placed on the body (Figure 6.1). The output of the optical system contained intermittent 3D trajectories of the markers. There were a number of instances where the markers disappeared and new markers appeared. A number of "ghost" markers was also apparent.

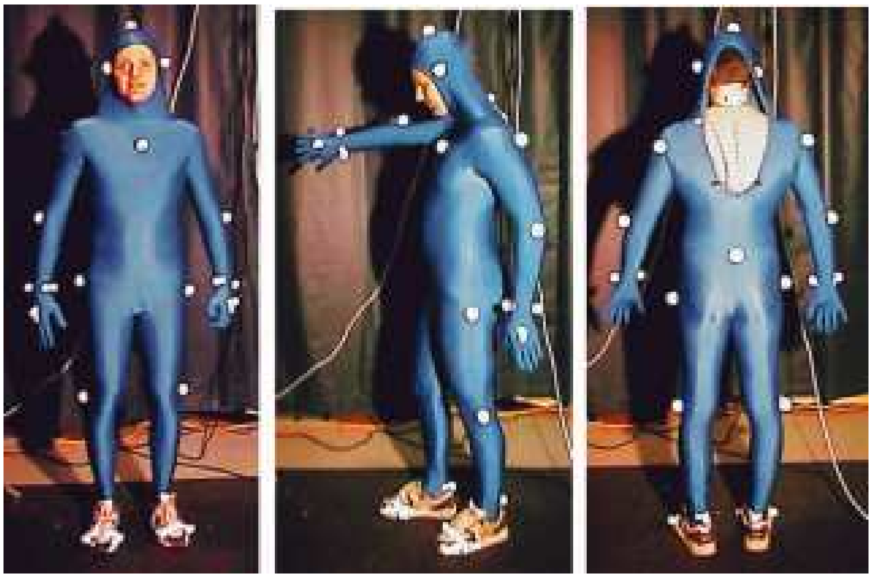

FIGURE 6.1. The placement of the cat's-eye markers on the optical motion capture suit.

6.2 Marker Trajectory Cleanup

The first step of the motion capture cleanup entailed the construction of *exactly* 27 marker trajectories. A number of statistical correlation comparisons were performed in order to determine which trajectories corresponded to the same physical marker. We used a number of heuristics in order to determine this trajectory correspondence:

Temporal Coherence. The disappearance of one trajectory is often followed or slightly preceded by the appearance of a new trajectory. This temporal closeness of two trajectories is a good indicator that the two trajectories should be merged as they belong to the same marker.

Neighbor Proximity. Each physical marker has at least two and often more neighboring markers which due to the rigid limb structure of the character preserve the relative distances between each other. For example, the elbow marker has two neighboring wrist markers and a neighboring marker on the shoulder. During the motion sequence the distances between the elbow marker and all of its neighbors is invariant. We use this information to determine the appropriate marker for the given trajectory.

Velocity Statistics. If two trajectories belong to the same marker they often have the same velocity statistics. We can use these statistic to determine the likely marker-trajectory matches. Also, in addition to the temporal coherence we can use the velocity direction and magnitude as additional hints when determining which two trajectories should be assigned to the same marker. This heuristic is considerably less accurate than the previous two.

When all heuristics fail, we have manually enforced assignment of trajectories to markers. A user interface which displays the temporal "trails" for each trajectory proved extremely useful for this purpose. We used the Hermite spline curves to fill the temporal gaps in the marker trajectories. If gaps prove to be too large we leave the missing interval unspecified, treating it as if we had no information of the marker's position during that time.

At the end of the cleanup stage we are left with exactly 27 marker trajectories. We use these trajectories in the next step of our algorithm in order to determine the character DOF values $\mathbf{q}(t)$.

6.3 Determining the Joint Angles

In order to compute the joint angle values through time, first we need to create the realistic character model that will appropriately represent all the human DOFs. We chose a model which has 59 DOFs (Figure 6.2). Appropriate limb measurements were gathered from the motion capture animation performer. In our case this was not a problem since the author of this thesis also performed the captured motion sequences. We also use images (e.g. Figure 6.1) taken during the motion capture session to determine the approximate fixed locations of the markers on the character itself.

In order to compensate for errors in marker trajectory acquisition, character model inaccuracies, and marker placement measures we use a least squares minimization to determine the joint angle values from the marker trajectories. Let $\mathbf{m}_i(t)$ be the i-th marker position and let $\mathbf{p}_i(\mathbf{q}(t))$ be the corresponding body point. Then for each time step t we solve the minimization

$$\min_{\mathbf{q}} \sum_i c_i(\mathbf{m}_i - \mathbf{p}_i(\mathbf{q}))^2 + \frac{\mathbf{q} - \mathbf{q}(t - \Delta t)^2}{\Delta t} \tag{6.1}$$

Under normal circumstances c_i is 1 for each marker. However, if some markers have smaller confidence assign to them the c_i coefficients for those markers could be reduced

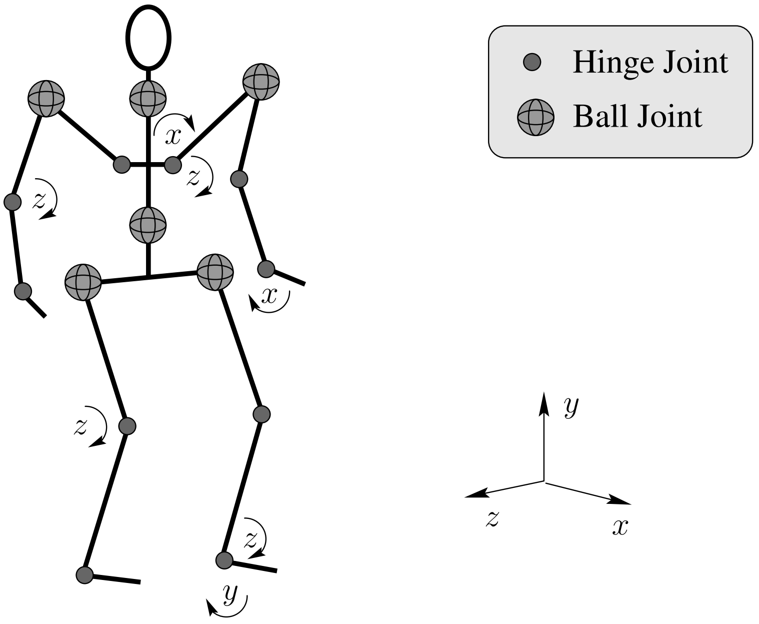

FIGURE 6.2. DOFs for the human character representation.

appropriately. The part of the expression that minimizes the DOF value deviation from its previously computed value ensures the smoothness and time coherence of DOF values at the neighboring time steps.

In the event that some markers contain time intervals with unknown position values, we perform the spacetime minimization for the that specific time interval $t \in \{t_0, t_{end}\}$

$$\min_{\mathbf{q}(t)} \sum_i c_i(\mathbf{m}_i(t) - \mathbf{p}_i(\mathbf{q}(t)))^2 + \ddot{\mathbf{q}}^2 \tag{6.2}$$

This formulation ensures that DOF values vary minimally during the time period where the marker information is lacking. Once the minimizations are completed, we are left with the motion capture character animation.

6.4 Human Run

We started with the captured motion sequence of a human running at a moderate pace. From this motion we extracted a cyclical gait (two steps) so that we could seamlessly concatenate the gait animation into a continuous run of arbitrary length.

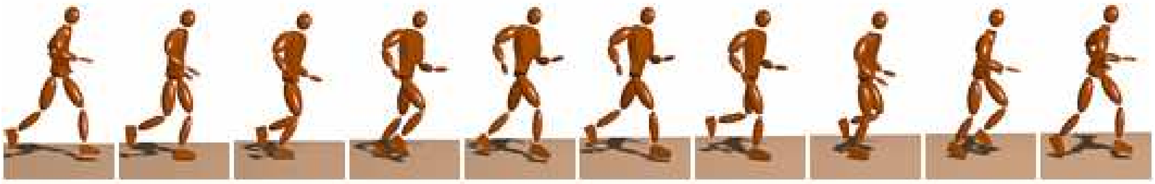

FIGURE 6.3. Frames from the original run animation.

6.4.1 Creating Cyclic Gaits

For a cyclical motion sequence to be continuous each DOF needs to have the coinciding start- and end-point. Also, the first and second derivatives should be identical. The easiest way to achieve the coincidence of the endpoints is to blend the beginning and the end of the DOF values. We use the simple ease-in-ease-out blending function

$$w(u) = u^2(-2u + 3) \qquad \text{where } u \in [0, 1] \tag{6.3}$$

scaled over a small time interval at the beginning and at the end of the gait motion sequence. Let the blending time period be Δt. Then for each $t \in [0, \Delta t]$ we compute

$$q(t) = q(t_{end} - t) = w\left(\frac{t}{\Delta t}\right) q(t) + \left[1 - w\left(\frac{t}{\Delta t}\right)\right] q(t_{end} - t). \tag{6.4}$$

After this cyclification process we represent each DOF with a cyclical B-spline, where the first three coefficients influence the beginning and the end of the DOF values (see Section 3.2). The entire gait cycle animation lasts for 1.1 seconds. We used 70 coefficients to represent each $q(t)$ function. The same coefficient density was used for muscle $\mathbf{q}^m(t)$ and mechanical constraint $\mathbf{q}^\lambda(t)$ DOF representation.

6.4.2 Character Simplification

During the simplification process we removed all hand, foot and elbow DOFs. This effectively reduces the entire arm into one rigid object. Similarly, the foot and the shin are also represented as a single object.

Furthermore, we preserved very few of the upper body DOFs. The torso, head and shoulders have been fused into a single object that has a quaternion ball joint at the waist.

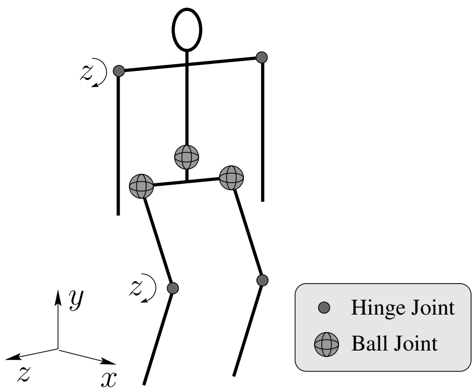

FIGURE 6.4. Biped: Simplified character for the human run motion sequence.

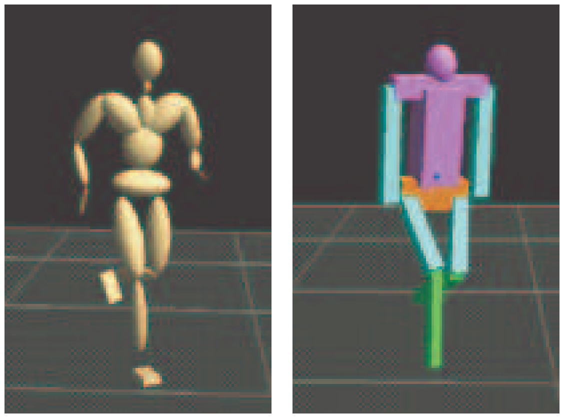

FIGURE 6.5. Full and simplified characters for the human run motion sequence.

Each shoulder has a single Euler hinge joint that allows an arm to move back and forth. This is a significant reduction from the original configuration of a ball joint at each shoulder. Hip joints are the only ball joints that have been preserved during the simplification. (Figure 6.4).

We should note that by reducing the arms to a single object which has only one DOF we have significantly restricted the amount of change each resulting motion can undertake. If we wanted to allow greater ability to alter the movement of arms, we would allow for additional DOFs at each shoulder.

6.4.3 Motion Fitting

In order to map the human motion onto the biped we introduce a number of handles:

Foot contact points. The floor contact points for each foot are located at the ball of the foot on the full character. On the biped these points are shin endpoints. The handles are tracked throughout the entire animation, not just while the foot is on the ground.

Character mass center. We ensure that mass centers for the human and biped follow the same trajectories.

Upper body mass center. We define the upper body as the character subtree rooted at the waist joint.

Arm mass centers. The character subtree rooted at the shoulder joint is a three link chain for the human character and a single object for the biped. We correlate the mass centers for both the left and the right arm subtree.

Torso orientation. We preserve the body orientation by tracking the orientation of the lower abdomen character node.

Shoulder orientation. Shoulder alignment handle is defined as the unit direction handle defined by two points located at the left and right shoulder.

We also define two mechanical constraints corresponding to the foot-ground contact events. These constraints need to be mechanical since the floor exerts forces onto the body in order to keep the foot in place. We also constrain the forces produced by the floor mechanical constraints so that they are non-negative in the y direction. This effectively indicates that the floor provides forces which prevent floor penetration, but not floor separation.

In addition, we provided a muscle for each kinematic DOF of the biped character. The muscle coefficient were identical for each muscle: $k_s = 4.0$ and $k_d = -0.04$. Our experiments showed that different coefficients affected the speed of convergence for the optimization. In fact, the convergence properties of the solution were significantly more sensitive to these coefficients than the actual resulting motion.

Optimizations during the spacetime model fit stage took about 5 minutes to complete on a SGI Octane. The deviation between the final spacetime optimal solution and the starting point generated with handles was hard to spot visually although numerical examination showed changes on all DOFs.

Once the spacetime model was complete we created a number of realistic animations by editing various parameters of the spacetime formulation.

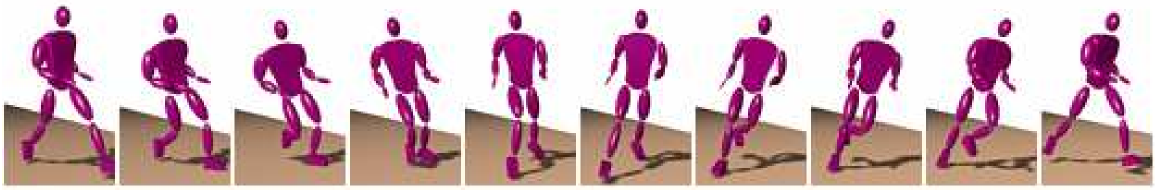

FIGURE 6.6. Frames from the wide footsteps run animation.

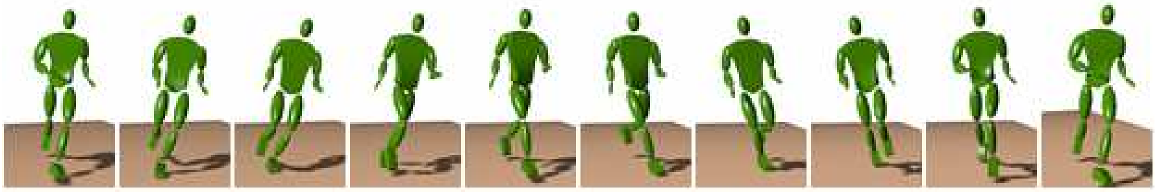

FIGURE 6.7. Frames from the crossed footsteps run animation.

6.4.4 Wide Footsteps

We repositioned the footprint mechanical constraints wider apart so that the character would have to move significantly farther to the side at each step. We kept the constraint intervals unchanged. Since each step now covered more distance, the overall resulting motion has leaps of smaller height. The appropriate change in the upper body orientation was apparent in the resulting motion.

6.4.5 Crossed Footsteps

We moved foot prints to the opposite side of the body, forcing the character to twist at each step, criss-crossing the feet. Again we kept the constraint intervals unchanged. We also introduced the "slippery floor" objective component, which penalized the component of the floor reaction forces in the floor plane (see Section 3.9.5).

6.4.6 Moon Run

In order to create a human run sequence on the moon, we reduced the gravity constant down to $1.6m/s^2$, and we allowed for more time to elapse by applying a global time warp. The resulting run was slower and had much higher leaps appropriate for low gravity environment.

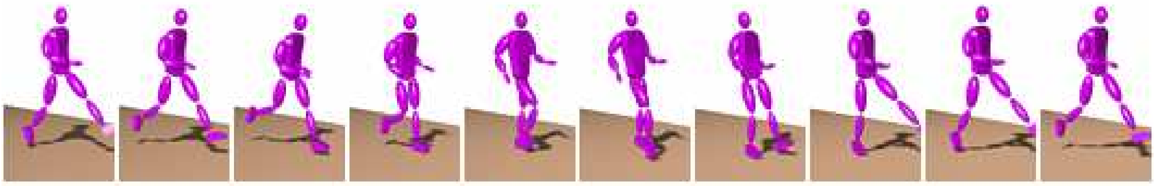

FIGURE 6.8. Frames from the original limp run animation.

6.4.7 Neptune Run

We also increased the Earth's gravity by tenfold to see how the running sequence would adjust to such extreme gravitational field[1]. Naturally, no human muscle forces could possibly produce a running motion under such extreme gravitational field. Consequently we removed any existing bounds on the muscle actuation forces.

The resulting flight phase of the run was so low to the ground that the running character had the appearance of speed walking. This animation produced another interesting discovery: by simply changing the gravity constraint we were able to approach the transition point between running and walking human gaits. This leads us to hypothesize that the hyper-space of all possible runs is smooth even around gait transition points.

6.4.8 Run with a Limp

We removed the left knee DOF, creating the appearance of the entire leg being put in a cast. We also reduced the duration of the left footstep mechanical constraint, while we increased the duration of the right footstep. The advent of a straight leg introduces two new problems.

First, the stiff leg is now more likely to penetrate the ground during the flight phase. We circumvent this by adding the simple inequality constraint which prevents the foot position from lowering below the floor surface.

Secondly, the stiff leg can move either to the outside or to the inside during the flight phase in order not to hit the ground. These are effectively two distinct local minima. If a leg were to move towards the inside, which is a more stable option, it would inevitably collide with the other leg. In order to prevent this we include the objective component which penalizes for the closeness of the two foot points during the small time period just as the stiff leg leaves the ground. Effectively, we are biasing the solution towards the one

[1]Neptune's gravitational acceleration is $88.98 m/s^2$

where the legs do not go through each other. We do not need to include this objective for the entire animation since we only need to intervene at the specific point of the solution space bifurcation. We should note that this problem would not be solved by including the collision detection in our model since the choice to go inside or outside occurs way before the actual collision occurs.

In the final motion sequence the character leans to the side and swings the right leg in a more dramatic fashion, creating a realistic (albeit painful) limp run.

6.4.9 Short-Legged Run with a Limp

In order to test the limits of the drastic character model modifications, we shortened the shin of the right leg, as well as fixed the left knee DOF as in the *limp run*. We also kept the non-penetration inequality constraint for the feet during flight. The output running sequence now has an extreme limp, with the leg in the cast swinging more to the side due to the shorter right leg. The motion has an extreme lean towards the shorter leg. In fact, the lean appears to be right on the edge of falling. The motion maintains the dynamic balance by significantly increasing the push-off forces of the shortened leg.

6.5 Broad Jump

We created the broad jump library to explore how far we can simplify the character without losing the dynamic essence of the jump. Implicit symmetry of the broad jump allows us to drastically simplify the character model (Figure 6.10.) The entire upper body structure is reduced to a single mass point allowed to move with three prismatic muscles that push off from the rest of the body. Since legs move together during the broad jump, we turned them into a single leg. Although it was not necessary, we also turned the knee hinge joint into a prismatic joint, to show that even with this completely changed character model the dynamic properties of the broad jump are preserved. The simplified character has *ten* DOFs of which six are the global position and orientation of the model.

We define the foot handle as the midpoint between the two balls of the feet. We also have the full body and upper body center of mass handles, as well as the torso orientation handle. We specify the mechanical constraint at the point of the floor contact. We also specify the initial and final upright pose constraints.

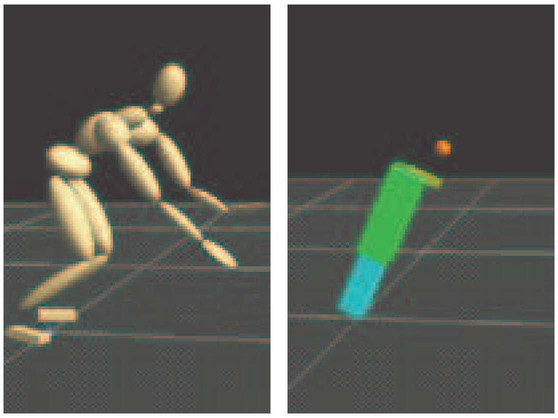

FIGURE 6.9. Full and simplified characters for the broad jump.

The entire broad jump motion sequence lasts for more than two seconds, which is more than twice as long as the run gait cycle. Since the sequence duration is linearly proportional to the size of the problem it would appear that optimization would take considerably longer. However, due to a much smaller DOF count, the spacetime fitting optimization converges within 6 minutes. Subsequent transformation optimizations all finish within 2 minutes on an SGI Octane machines.

6.5.1 Character Simplification

During the fitting stage, we use a drastically different simplification approaches for the two input motion sequences in order to demonstrate the power and flexibility of the simplification tools. During a broad jump both arms and legs move in unison. We can exploit this inherent symmetry by simplifying the character significantly beyond what we had to do for the running sequence (Figure 6.9). In fact, the simplification process for the broad jump library served as an exercise in how far we can simplify the character without losing the dynamic essence of the jump.

The entire upper body structure is reduced to a single mass point. The mass point moves with three prismatic muscles that push off from the rest of the body (Figure 6.10). Since legs move together during the broad jump, we turned them into a single leg. Although it was not necessary, we also turned the knee hinge joint into a prismatic joint, to show that even with this completely changed character model the dynamic properties of the broad jump are preserved. The simplified character (hopper) has *ten* DOFs of which six are the

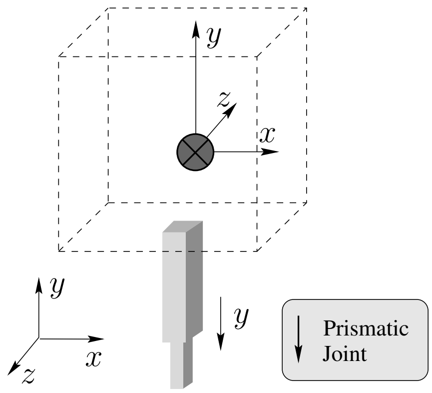

FIGURE 6.10. Hopper: Simplified character for the broad jump motion transformation.

global position and orientation of the model. It does not contain any angular joints.

The broad jump motion sequence lasts for more than two seconds, which is more than twice as long as the run gait cycle. The total animation contains 140 frames.

6.5.2 Motion Fitting

In order to fit the human broad jump motion onto the hopper we define a number of handles:

Foot position. For the human character we define the foot handle as the midpoint between the two balls of the feet. The hoppers foot is located at the shin's endpoint.

Leg extension. For the human character the leg extension is measured as a distance handle between the foot handle and the midpoint between the two hips. For the hopper this handle is measured as a distance between the foot handle and the location of the prismatic hip.

Body mass center. For both the human and the hopper the body mass center point is the center of mass for the entire hierarchy.

Upper body mass center. For the human character the upper body mass center is the center of mass of the subtree rooted at the lower abdomen. The hopper's upper body mass center is identical to the location of the upper body mass point.

Torso orientation. We preserve the body orientation by tracking the orientation of the lower abdomen character node. For both the human and the hopper character this is

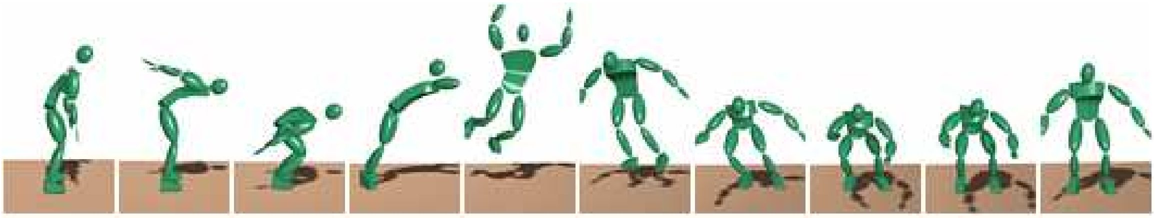

FIGURE 6.11. Frames from the twist jump animation.

a direction of the unit vector pointing down the x-axis in the local frame of the lower abdomen node.

In addition to handles, we also define a number of constraints. Again, we use mechanical constraints for the foot-ground constraints. We also specify the initial and final upright pose constraint. These constraints are needed to make the endpoints of the animation invariant. We also restrict the freedom of movement for the upper body mass point, with simple bound constraints in each dimension. Without such constraints, the upper body mass point would be free to move outside of the range of motion for the human upper body center of mass.

The broad jump motion sequence lasts for more than two seconds, which is more than twice as long as the run gait cycle. The total animation contains 140 frames. Since the sequence duration is linearly proportional to the size of the problem it would appear that optimization would take considerably longer. However, due to a much smaller DOF count spacetime fitting optimization converges within just 6 minutes. Subsequent transformation optimizations all finish within 2 minutes on SGI Octane machines. The rest of this section describes each transformed jump in more detail.

6.5.3 Twist Jump

We introduced the torso orientation pose constraint at the end of the animation, which mandates a 90 degrees turn. The constraints controls only the position of the last frame. We also applied the same rotation to the foot positions in the reconstruction phase so that the line between the feet remains perpendicular to the torso orientation.

The output motion clearly shows the change in the anticipation and the introduction of the body twist during the flight stage. The landing stage has also changed to accommodate for the sideways landing position. Since the input jump represented the maximum length

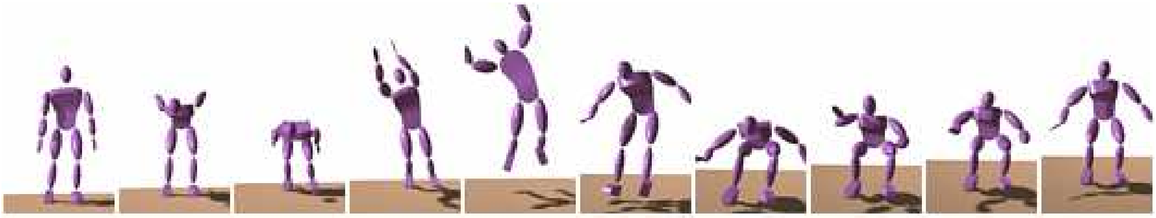

FIGURE 6.12. Frames from the diagonal jump animation.

FIGURE 6.13. Frames from the obstacle jump animation.

jump for the human actor it would haven been impossible to capture the jump of the same length with the 90 degree twist.

6.5.4 Diagonal Jump

We moved the landing position to the side and constrained the torso orientation to point straight ahead at all times. This change realigned the push-off and anticipation stages in the direction of the jump. Since the jump length was increased, the entire resulting motion looks more impulsive. The constraint torso direction forced the character to bend sideways around the waist during flight in order to safely land at the specified position.

6.5.5 Obstacle Jump

We raised the mechanical constraint controlling the landing position with intention of generating a jump onto a higher plateau. In addition we introduced the inequality pose constraint which forced the legs to raise higher during flight in order to clear the obstacle. As a result, the character push-off is more vertical, and the legs tuck in during the flight in order to avoid the hurdle. The landing phase also has a different feel which results from a more vertical jump.

FIGURE 6.14. Frames from the unbalanced jump animation.

6.5.6 Unbalanced Jump

We removed the final pose constraints that impose the upright position. In the resulting sequence, the character never uses its muscles to stand up upon landing since this would add unnecessary non-smoothness of the muscles. Instead of straightening up, the character tumbles forward giving the appearance of poor landing balance.

6.6 Example Summary

Our experiments show that despite the extreme simplification and turning angular joints into linear joints, the hopper spacetime model still encapsulates the realistic properties of the broad jump with surprising accuracy.

We have also demonstrated a wide spectrum of animations produced from a single motion capture sequence. The power of our transformation algorithm stems directly from the underlying spacetime constraints formulation. We also draw attention to the minimal set of intuitive modifications that were performed in order to produce each motion sequence. This relative ease of creating transformed character animations suggests that our methodology can be used for the creation of reusable motion libraries. Such libraries would enable ordinary non-skilled users to create expressive animations.

6.7 Validation of Realism for the Transformed Motions

Under visual inspection the generated transformed motion sequences appear realistic. Unfortunately, there are very few deterministic claims that we can make about the physical correctness of the resulting motion. In fact, since the forces and accelerations are never computed for the full human character, we cannot provide any guarantee the generated animations are realistic.

We can, however, compare the transformed motion sequence with the real-life captured motion sequence that performs the same task. Using this comparison we can measure the relative differences between the synthesized and real-life motion sequences.

Such a comparison sequence should, of course, be realistically possible. Most of the generated animations described in this chapter are not easily replicated in reality. For example, it would be really hard to capture the running sequence on the moon surface, the running with a shortened leg, or a jump where the character exceeds the ability of its real-life counterpart. Although such impossible or hard to replicate motion is one of the main appeals of our method, it is not good for our realism validation purposes.

We used two broad jump motion capture sequences for our comparisons:

- maximum length broad jump
- 45 degree diagonal broad jump of smaller distance

The maximum length broad jump sequence was the original motion for all of the transformed broad jump sequences. The start and the end pose are identical for the two captured motion sequences, with the exception that the character is displaced to the new position in the diagonal jump motion sequence.

When creating the transformed motion sequence, we started with the maximum length broad jump as the original jump sequence. We specified the start and the end character pose constraints, and introduced mechanical constraints for the start and end floor-contact periods. We proceeded to fit the original motion to the spacetime simplified model described in Section 6.5.1. During the spacetime edit stage of our algorithm, we applied the appropriate change to the final pose constraint, and then resolved the spacetime constraints problem. After the reconstruction stage we are left with the transformed motion sequence that accurately matches the diagonal broad jump motion. To demonstrate the accuracy of the match between the generated and real motion, we compare the DOF curves for the three representative joint angles:

lknee Euler angle representing the left knee joint (Figure 6.15).

lshoulderz The z component of the quaternion representing the left shoulder joint (Figure 6.16).

lhipz The z component of the quaternion representing the left hip joint (Figure 6.17).

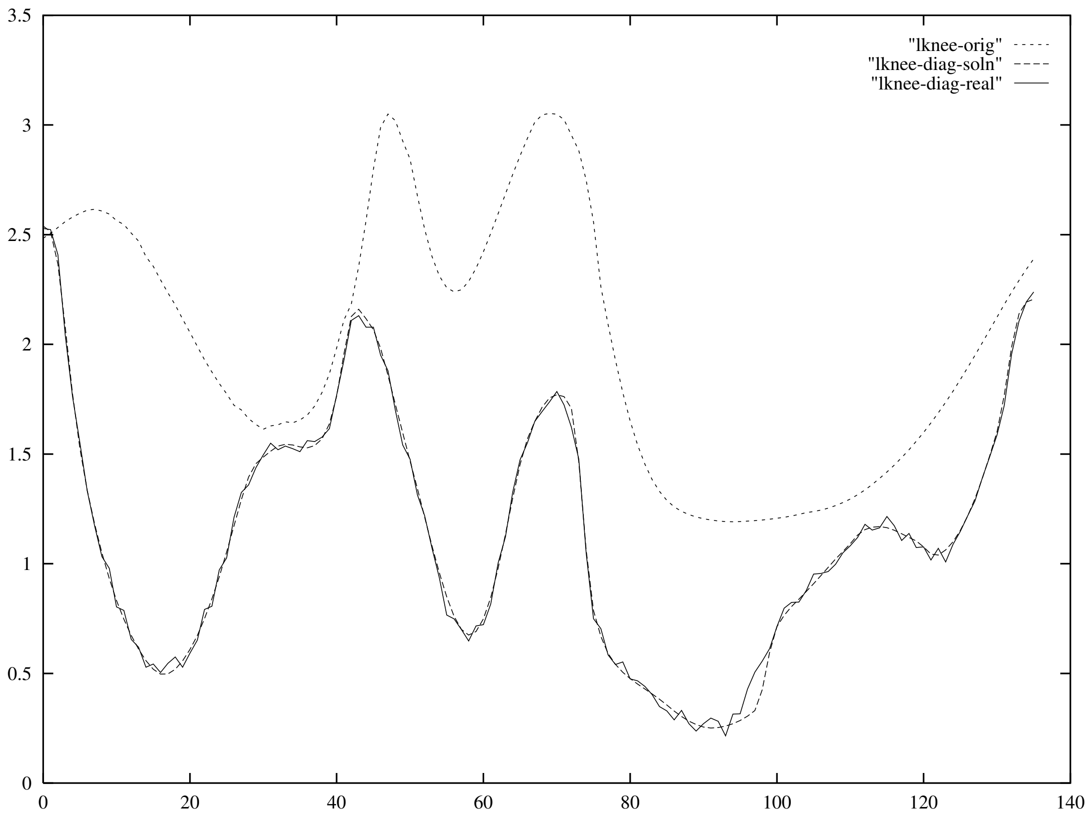

FIGURE 6.15. Left knee DOF functions for the original jump, computed diagonal jump, and the real-life diagonal jump.

Each figure contains three curves which depict the DOF function in the original, real-life diagonal, and the transformed diagonal motion sequence. Unlike the original motion capture sequence, the real-life diagonal motion was not smoothed and the noise is clearly present in all DOF curves.

Both knee and shoulder DOFs show a very good match (Figures 6.15,6.16). Aside from the motion capture noise in the real-life output the real-life and synthetic diagonal jump curves are virtually identical. The only difference appears to be at the extrema points, and even at those points the error is of the same scale as the motion capture noise.

The hip joint appears to show greater deviation between the real-life and the transformed motion (Figure 6.17). Still, the error appears to be less than twice as much as the motion capture noise. We draw attention to the emergent inflection points which are present in both synthetic and real-life curves.

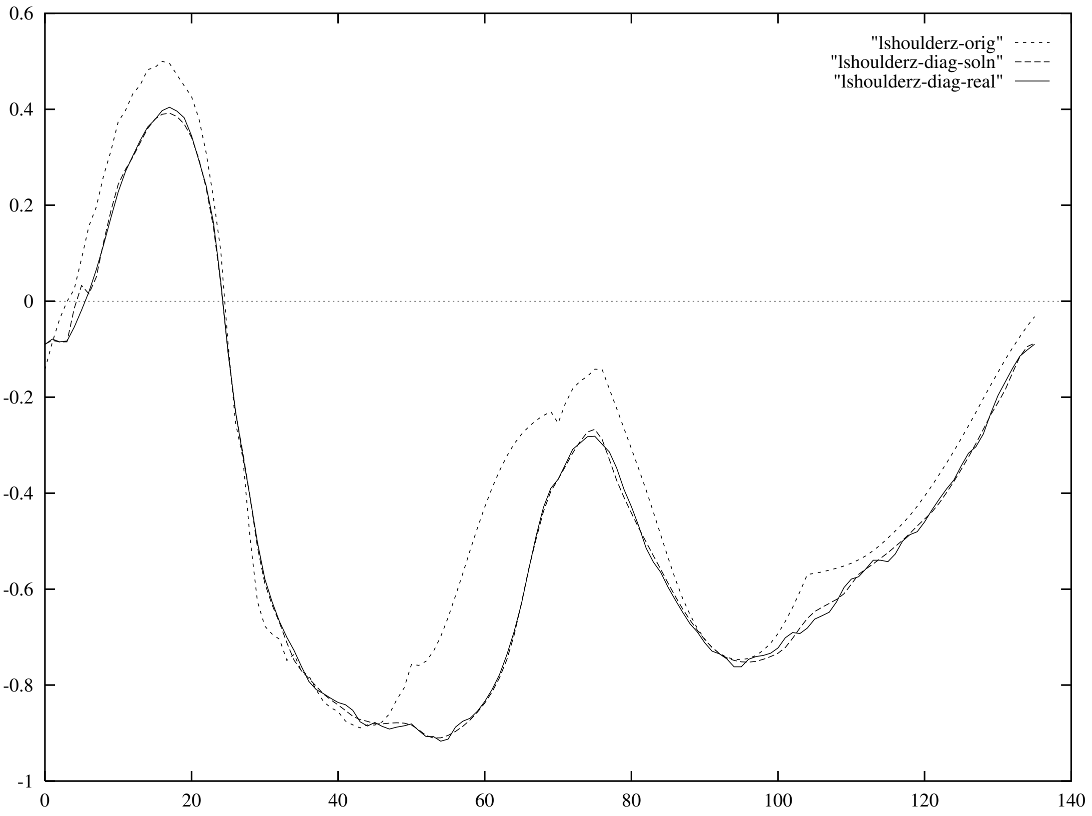

FIGURE 6.16. Left shoulder z quaternion component DOF functions for the original jump, computed diagonal jump, and the real-life diagonal jump.

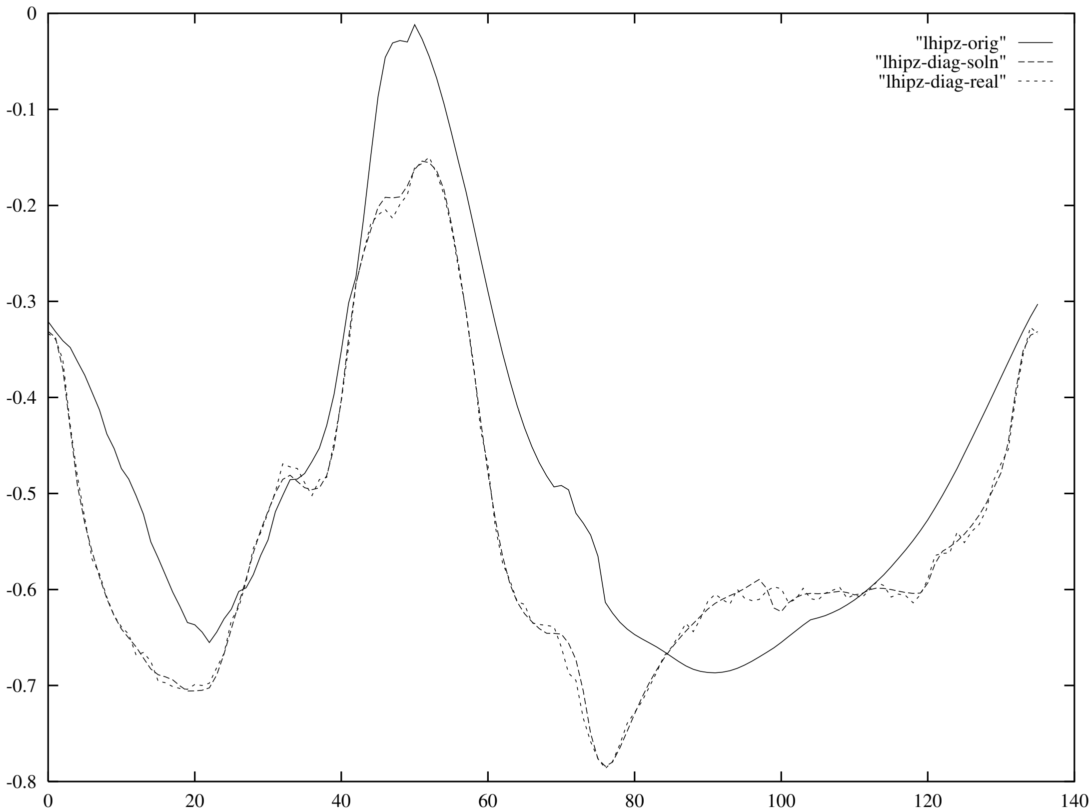

FIGURE 6.17. Left hip z quaternion component DOF functions for the original jump, computed diagonal jump, and the real-life diagonal jump.

The comparisons show that for this specific example our motion transformation algorithm accurately models motion modifications introduced by changing spacetime constraints. It is quite possible that for other motions the reality match would not be so favorable. There are a number of reasons why such discrepancies might occur:

- The simplification process might abstract away some aspects of the character which could turn out to be crucial in the transformed motion. For example, if we represent the upper body with the single mass point and use such a simplified character for the ice skating motion, we cannot accurately represent the torque change that occurs when arms are further away from the body.

- The muscle properties are not represented accurately enough. For example, the generalized muscle DOFs could end up producing forces which would not be in accordance with the natural processes in real muscles. In particular, natural muscles have very specific energy buildup patterns which are not modeled in our representation.

- The minimum displaced mass objective measures the static mass displacements and does not take into the account the velocity and acceleration of the body mass when comparing the two body poses.

6.8 Comparisons with Simple Motion Warping

Motion warping [Witkin 95] approaches require not only the modification of the pose constraints, but also additional "pin-down" constraints which essentially restrict the DOF curve deformation to the specific time range. These constraints need to be placed carefully because they greatly affect the motion warping results.

During the motion warping process, we often have to know a priori the realistic pose of the desired motion sequence at some key points in time. Sometimes these poses are easy to determine, but not always. For example, if we wanted to motion warp the broad jump into the broad jump with the 90 degree turn, it would not be sufficient to specify only the endpoint constraints (as in the spacetime formulation). We would also need to know the particular pose of the body while in the air and upon contact with the ground (Figure 6.11). It would be really hard to come up with those realistic poses without having seen the particular twisting body movement which is characteristic for such a turning jump.

The advantages of taking dynamics into the account when transforming motion are apparent in our examples. We demonstrate the power of dynamics constraints in the transformation process by showing the results of excluding dynamics constraints from the algorithm formulation. Although not identical to any of the existing motion warping methods, this non-dynamic algorithm has many of the same properties.

Again, we applied the transformations to the broad jump motion sequence. This time our goal was the obstacle broad jump which has the elevated destination constraint (Figure 6.13). Figure 6.18 shows the global translational DOF in the upward (y) direction for the original, dynamic and non-dynamic motion transformation sequences.

In order to help out the non-dynamic solution we introduced a number of additional constraints during the time when the feet are on the ground before the take-off, and after the landing (take-off and landing occur at the time samples 20 and 44 respectively). These additional constraints essentially made the portion of the curve after the landing identical for both dynamic and non-dynamic transformation. Note that without this a priori knowledge about the exact poses during those intervals we would not be able to pin down these portions of the curve; so, in essence, we are partially including the dynamic solution through these constraints.

The most apparent discrepancy between the dynamic and non-dynamic curve occurs during the flight phase. Since the non-dynamic algorithm knows nothing about gravity the parabolic path of the character is much lower and looks very unrealistic. Also, since the character jumps higher in the air the dynamic transformation has a greater dip during the pre-takeoff phase which allows the character to collect greater upwards momentum before it jumps higher. This dip is completely absent in the non-dynamic version.

This example, shows that even if we try to help out the standard warping method beyond what would be possible under normal circumstances of performing motion warping, we still endup with grossly unrealistic motion.

6.9 Limitations

Although our algorithm has been proven effective for a wide range of input motion sequences and for an even wider spectrum of transformed motion animation sequences a number of deficiencies and possibilities for improvement remain. In this section we describe a number of limitations of our technique.

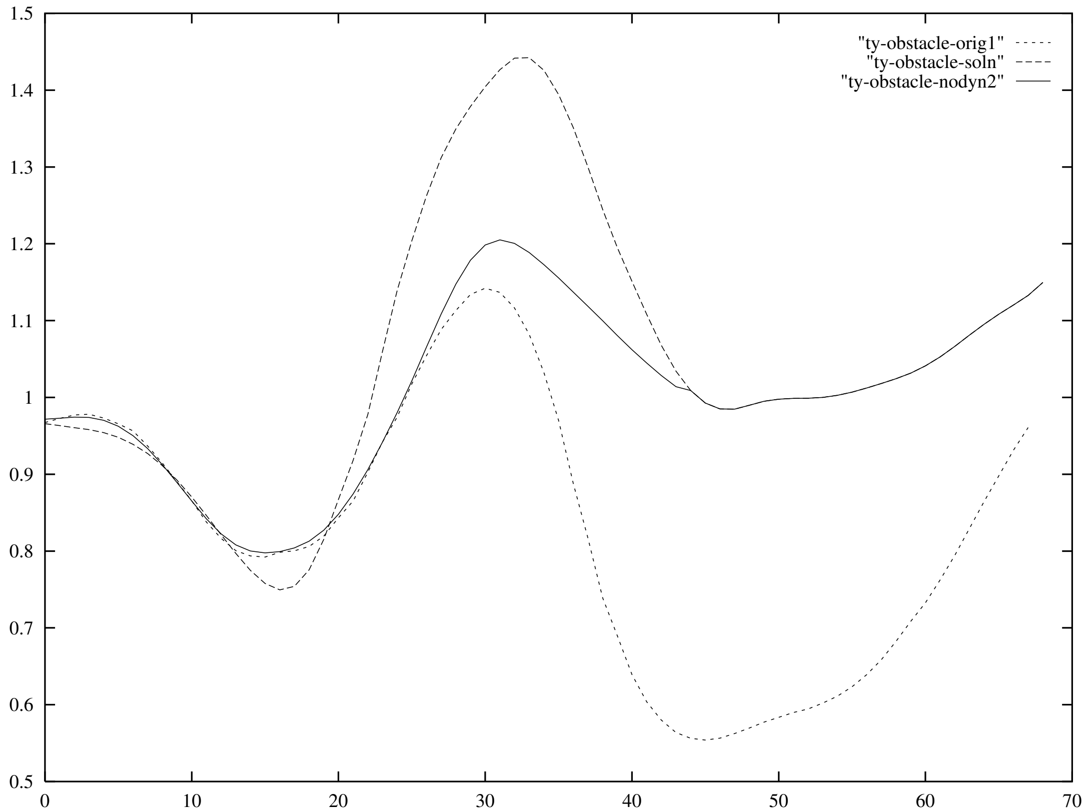

FIGURE 6.18. y component of the global translation shown for the original jump, jump onto the elevated obstacle with and withouth dynamics constraints.

It appears that our methodology is best suited for the animation sequences containing high-energy, very dynamic character movement. Other less animated and more kinematic motion sequences such as picking up an object, climbing a ladder, rowing are not well suited for our dynamics transformation framework. Of course, it is always possible to apply our technique to such animations. However, the benefits of a full-blown dynamics representation would not be large since much simpler methods could be used for transformation without a significant loss of the output motion quality. Most overly constrained motions where both feet and hands are constrained for a large portion of the animation (e.g. climbing a ladder) display very few dynamics properties and are consequently not a good candidate for the input to our algorithm. Other "lethargic" motions such as transferring the weight from one leg to the other, raising a hand while standing, or extremely slow walking would also not be good candidate input motions for our method. In general, the problem of non-realism for such motions is much less of an issue. The more a particular motion contains visible dynamic properties, the more suitable it is for our motion transformation algorithm.

During the development of our method, very little time was devoted to the concerns of speed. Our method is not interactive. Each change in the spacetime formulation produces a new animation in 2-3 minutes. This time lag would prove to be quite tedious for most animators. However, it is our belief that with a few adjustments and code optimizations this delay could be reduced significantly, at least down to the 30 second range. This problem can be further alleviated by interactively displaying the current state of the solution during the optimization. A mistake or an undesirable result often becomes apparent to the animator long before the optimization has converged. With interactive display the user can spot the mistake early and modify the motion again.

Another shortcoming of our approach is that large portions of the motion fitting algorithm stage are performed manually. This stage is performed only once per input motion sequence, so the effort spent by the motion library creator is amortized over the large number of possible transformed animation sequences. In addition, unlike the robot controller based methods for character animation synthesis, the expertise required to perform these manual stages of the motion fitting is small. Many simplification decisions as well as handle selections appear to be quite intuitive. Nevertheless, automating this manual decision making process would enable on-the-fly construction of the physically based spacetime formulation from the input animation.

Furthermore, specific decisions in the motion fitting stage directly affect the types of modifications that can be performed to the motion. For example, if we wanted to add the waving gesture to the human running sequence we cannot simplify the waving arm down to a single rigid object as in the biped model (see Section 6.4.2). Consequently, a modification of the simplification process might be necessary in order to achieve a certain motion transformation which was unforeseen during the motion fitting process. These modifications break down the motion library concept.

Since all dynamics computations are done on the simplified model, there is no guarantee that the reconstruction stage of the algorithm would preserve the dynamics properties. In fact, the final motion sequence is *not* physically realistic in the absolute sense, simply due to the fact that no dynamics computations are done on the full character model. Our algorithm preserves the *essential* physical properties of the motion. This makes our algorithm ill-suited for applications which require that a resulting motion contains all the forces involved in the character locomotion.

Simplifications which reduce the large portions of the body down to a single mass point effectively ignore the inertia moments which are present in the original motion, but are completely ignored in the simplified model representation. As a result, such simplifications cannot be used if moments of inertia are important for the given motion sequence (e.g. any motion involving twisting or rotating with large angular velocities). An alternative simplification which would account for the moments of inertia could include a larger single objects such as a "stick" which could generate the moments of inertia around the relevant axes of rotation.

During the spacetime optimization we ignore self-collision of character's extremities. For example, there is nothing in our dynamics representation that would prevent an arm going through the torso or a leg freely passing through the other leg. These problems can be partially addressed, for example, by introducing the "ankle closeness" penalties in the objective function. Of course, such penalties can also alter the motion appearance by keeping the ankles unnaturally far apart. A better solution would involve collision detection for the various body parts that are likely to collide. For example, we could check for collision between the left and right shin, as well as the hand and forearm intersection with the body torso. Each time the collision occurs, we could introduce the mechanical constraint which would prevent inter-penetration of the body parts.

Chapter 7

CONCLUSION

This thesis describes a novel algorithm for generating realistic animation sequence from a single motion capture sequence. In this chapter we enumerate the main contributions of this work. We proceed to describe what animation problems are best suited for our algorithm. Finally, we suggest a number of interesting and challenging directions for extending our work.

7.1 Contributions

Our algorithm presents the first solution to the problem of editing captured motion taking dynamics into consideration. Dynamic properties of motion are taken into the account by constructing the physically based spacetime constraints formulation of motion. During the editing process various aspects of the dynamic representation are modified. The resulting change in motion is, consequently, mapped back into the space of the input motion in order to produce the final animation. Our algorithm provides a large number of dynamics related motion modifiers to be applied to the original sequence. No current motion editing methods consider preserving the physical properties of motion.

Furthermore, we provide a rich set of motion modification operators which provide an intuitive control interface to all aspects of the motion sequence. Previously, setting constraints at different times was the only tool for motion transformation available to the animators.

The ability to preserve dynamics of the motion, and the existence of a rich set of motion controls enables the creation of motion libraries from a single input motion sequence. Once the original motion is fitted onto the spacetime constraints motion model, our motion transformation model can be presented to the user as tool which can generate the motion that meets the exact specifications of the given animation. So an animator can be presented with a few motion libraries such as human run, human jump, karate leg kick, soccer ball kick, tennis serve. Based on the need, the animator can pull the appropriate library "off the shelf" and edit the motion so that it meets the needs of the animation.

By way of character simplification, we also present the first method which successfully uses spacetime constraints formulation to generate motion sequences for non-trivial characters — characters with complexities comparable to that of a human actor.

We also describe a novel methodology for mapping motion to/from characters with drastically different kinematic structures. We show how a collection of handles and the displaced mass metric can be used to correlate motion between characters of different dimensions, mass properties and even different kinematic structure. These ideas could broaden the applicability of motion capture data to animation of characters that do not exist in nature.

This thesis also demonstrates that complex dynamic systems can be successfully controlled with simpler ones. This realization could potentially have significant impact outside of computer graphics, in particular, in fields which are concerned with real-time control of dynamic systems (e.g. robotics).

7.2 Applications

This method is best suited for the transformation of motion sequences that are highly dynamic:

- jumps, leaps, hops
- runs, skips, trots, high-energy modern dance elements
- karate leg kicks, soccer ball kicks, football punts
- ball throws, tennis serve, shot put, punches and arm kicks
- gymnastic disciplines, board dives

Other less energetic motions such as a slow arm movements in ballet, slow walking, reaching and picking up an object appear to have little use for dynamic analysis. Consequently our method would most likely be an overkill for such input motion. However, other aspects of our method such as retargeting motion to different characters would still be just as useful.

The physically based motion transformation methods have a natural fit with the realistic motion for video games, virtual avatars, telepresence applications and human and animal motion simulations. In such environments, the realism of the motion is important. Furthermore the motion is governed by a certain number of real-time input parameters. These parameter, in turn, specify a number of mechanical and pose constraints. Thus, a number of high-detail captured motion sequences could be transformed to meet the constraints introduced by the real-time input of the user producing a seamless perception of simulated reality that responds to the human input.

7.3 Future Directions

The results described in this thesis open new avenues towards reusable animation that can be utilized not only in the film and video game industry, but more importantly on every home PC. It won't be long before every PC will have sufficient 3D rendering capabilities to enable each one of us to use computer animation as a expressive medium, much like web pages are an ubiquitous form of expression today. I see reusable motion as a crucial concept that will enable most of us, non-skilled animators, to become capable, expressive storytellers. The approach we described lays down the foundation for creating such motion libraries. To this end, additional work needs to be done in order to take our "proof of concept" and turn it into a usable motion editing tool.

Large portions of the motion fitting algorithm stage are performed manually which creates the need for certain amount of human preprocessing for each input sequence. There are a number of possible ways to automate this process by analyzing the structure of the input motion in order to determine which aspects of the motion are more important than the others. In addition, a number of mechanical constraints can be postulated by analyzing the absolute movement of various character body points. While it is unlikely that the entire motion fitting process can be fully automated, the human intervention can be reduced to a few decision approvals suggested by the algorithm.

So far our user interface consists of the animation viewer which contains various UI tools for setting up the optimization problem, for interactive manipulation of $\mathbf{q}(t)$ curves and for visually examining the motion sequences. The character transformation hierarchy, as well as specifics of certain constraints and objectives are specified within C++ files which are dynamically loaded into the system. The motion sequences are stored in files which contain coefficients for each DOF. A number of additional features need to be made accessible to the animator through the UI. Currently, a significant knowledge of C++ character and constraint building blocks is required in order to edit a given captured motion sequence. All of these aspects should be eventually transfered into the UI. Although these issues do not involve creating new technologies, some software engineering effort is required.

Additional efforts need to be made towards the interactive speed of the motion transformation process. The current system was designed as an experimentation tool with little concern towards efficiency. Additional exploitation of sparsity within the optimization problem definition can drastically reduce the computation times. A non-uniform representation of the DOFs can significantly reduce the size of the problems as well.

The methods of physically based spacetime transformation can also be applied to non-realistic motion data. For example, our transformation framework can be modified to allow motion editing of the arbitrary keyframed character animation. In this setup, our motion transformation methods can be thought of as a "beleiveabllity filter" which would improve the physical plausibility of the motion sequence while still preserving the exaggerated non-physical gestures frequently used in character animation.

Further exploration in the use of this method for motion retargeting for the characters with different kinematic and dynamic properties is needed. In particular, significant changes in the kinematic structure and the mass distribution of the character can render the minimum displaced mass objective inapplicable. Certain augmentation of the displaced mass metric should be developed to deal with the significant structural changes of the character.

Earlier we mentioned that this work suggests a methodology for controlling the complex dynamic systems with significantly simpler models. These ideas can be further developed into a real-time optimal robot control planner. Such a planner would internally keep the simplified representation of the robot dynamics. It would, then, predict and plan the outcome of the motion within a small window of time by rapidly solving the spacetime optimization problem for that time period. The model simplification reduces the complexity of the optimizations and enables such real-time computations. Such controllers would give raise to robots moving in very natural ways.

Finally, these methods can be used for motion analysis in biomechanics and help prove the very ideas of common principles of natural motion that motivated this algorithm.

References

R. M. Alexander. *Optimum walking techniques for quadrupeds and bipeds*. J. Zool., London, 192:97–117, 1980. (pp. 29, 64)

R. M. Alexander. *Optimization and gaits in the locomotion of vertebrates*. Physiol. Rev., 69:1199–1227, 1989. (pp. 29, 64)

R. M. Alexander. *Optimum take-off techniques for high and long jumps*. Phil. Trans. R. Soc. Lond., 329:3–10, 1990. (pp. 29, 64)

R. M. Alexander. *Optimum timing of muscle activation for simple models of throwing*. J. Theor. Biol., 150:349–372, 1991. (pp. 29, 64)

K.N. An, B.M. Kwak, E.Y. Chao, and B.F. Morrey. *Determination of muscle and joint forces: A new technique to solve the indeterminate problem*. J. of Biomech. Eng., 106:663–673, November 1984. (pp. 28, 38)

David Baraff. *Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies*. In Computer Graphics (SIGGRAPH 89 Proceedings), volume 23, pages 223–232, July 1989. (p. 24)

David Baraff. *Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation*. In Computer Graphics (SIGGRAPH 90 Proceedings), volume 24, pages 19–28, August 1990. (p. 24)

David Baraff. *Coping with friction for non-penetrating rigid body simulation*. In Computer Graphics (SIGGRAPH 91 Proceedings), volume 25, pages 31–40, July 1991. (p. 24)

David Baraff and Andrew Witkin. *Large Steps in Cloth Simulation*. In Michael Cohen, editor, *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 43–54, July 1998. (p. 24)

R. Blickhan and R. J. Full. *Similarity in multilegged locomotion: bouncing like a monopode*. J Comp. Physiol. A, 173:509–517, 1993. (pp. 17, 28, 51)

J.E Bobrow, S. Dubowsky, and Gibson J. S. *Time-optimal control of robotic manipulators along specified paths*. International Journal of Robotics Research, 4(3):3–17, 1985. (p. 26)

Armin Bruderlin and Lance Williams. *Motion Signal Processing*. In Computer Graphics (SIGGRAPH 95 Proceedings), pages 97–104, August 1995. (pp. 17, 27)

Michael F. Cohen. *Interactive spacetime control for animation*. In Computer Graphics (SIGGRAPH 92 Proceedings), volume 26, pages 293–302, July 1992. (p. 25)

Roy D. Crownninshield and Richard A. Brand. *A physiologivcallly based criterion of muscle force prediction in locomotion*. J. Biomechanics, 14(11):793–801, 1981. (pp. 28, 38)

Tony DeRose, Michael Kass, and Tien Truong. *Subdivision Surfaces in Character Animation*. In Michael Cohen, editor, *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 85–94, July 1998. (p. 24)

B. Donald, P. Xavier, J. Canny, and J. Reif. *Kinodynamic motion planning*. Journal of the ACM, 40(5), 1993. (p. 26)

J. Dul, M.A. Townsend, R. Shiavi, and G.E. Johnson. *Muscular Synergism — I. On criteria for load sharing between synergistic muscles*. J. Biomechanics, 17(9):663–673, 1984. (pp. 28, 38)

Michael Gleicher. *Motion Editing with Spacetime Constraints*. In Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 139–148. ACM SIGGRAPH, April 1997. ISBN 0-89791-884-3. (pp. 17, 27)

Michael Gleicher. *Retargeting Motion to New Characters*. In Computer Graphics (SIGGRAPH 98 Proceedings), pages 33–42, July 1998. (pp. 17, 27)

Michael Gleicher and Peter Litwinowicz. *Constraint-based Motion Adaptation*. The Journal of Visualization and Computer Animation, 9(2):65–94, 1998. (pp. 17, 27)

Jessica K. Hodgins, Paula K. Sweeney, and David G. Lawrence. *Generating natural-looking motion for computer animation*. In Proceedings of Graphics Interface 92, pages 265–272, May 1992. (pp. 26, 38)

J. K. Hodgins and N. S. Pollard. *Adapting simulated behaviours for new characters.* SIGGRAPH 97, pages 153–162, 1997. (p. 26)

Cornelius Lanczos. *The Variational principles of Mechanics*. University of Toronto Press, 1970. (p. 41)

J.C. Latombe. *Robot motion planning*. Kluwer Academic Publisheres, 1991. (p. 26)

Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. *Hierarchical Spacetime Control*. In Computer Graphics (SIGGRAPH 94 Proceedings), July 1994. (p. 25)

Zicheng Liu. *Efficient Animation Techniques Balancing Both User Control and Physical Realism*. PhD thesis, Princeton University, November 1996. (p. 41)

Matthew Moore and Jane Wilhelms. *Collision Detection and Response for Computer Animation*. In Computer Graphics (SIGGRAPH 88 Proceedings), volume 22, pages 289–298, August 1988. (p. 24)

J. Thomas Ngo and Joe Marks. *Spacetime Constraints Revisited*. In Computer Graphics (SIGGRAPH 93 Proceedings), volume 27, pages 343–350, August 1993. (p. 26)

M.G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine. *An optimal control model of maximum-height human jumping*. J. Biomechanics, 23:1185–1198, 1990. (p. 29)

M.G. Pandy and F. E. Zajac. *Optimum timing of muscle activation for simple models of throwing*. J. Biomechanics, 24:1–10, 1991. (p. 29)

M.G. Pandy, F. C. Anderson, and D. G. Hull. *A Parameter Optimization Approach for the Optimal Control of Large-Scale Musculoskeletal Systems*. J. of Biomech. Eng., pages 450–460, November 1992. (pp. 28, 38)

A. Pedotti, V. V. Krishnan, and L. Stark. *Optimization of muscle-force sequencing in human locomotion*. Math. Biosci., 38:57–76, 1978. (pp. 29, 64)

L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. John Wiley and Sons, New York, N.Y., 1962. (p. 38)

Marc H. Raibert and Jessica K. Hodgins. *Animation of dynamic legged locomotion*. In Computer Graphics (SIGGRAPH 91 Proceedings), volume 25, pages 349–358, July 1991. (pp. 25, 26, 38)

C. Rose, B. Guenter, B. Bodenheimer, and M. Cohen. *Efficient Generation of Motion Transitions using Spacetime Constraints*. In Computer Graphics (SIGGRAPH 96 Proceedings), pages 147–154, 1996. (p. 25)

C. Rose, M. F. Cohen, and B. Bodenheimer. *Verbs and Adverbs: Multidimensional Motion Interpolation*. IEEE Computer Graphics & Applications, 18(5), September – October 1998. (pp. 17, 27)

A. Seireg and R. J. Arvikar. *The prediction of muscular load sharing and joint forces in the lower extremities during walking*. J. Biomechanics, 8:89–102, 1975. (pp. 28, 38)

Z. Shiller and Dubowsky S. *On Computing the Global Time-Optimal Motions of Robotic Manipulators in the Presence of Obstacle*. IEEE Transactions on Robotics and Automation, 7(6), 1991. (p. 26)

K. G. Shin and N. D. McKay. *Minimum-time control of robotic manipulators with geometric path constraints*. IEEE Transactions on Automatic Control, AC-30(6):531–541, 1985. (p. 26)

K. Shoemake. *Fiber bundle twist reduction*. In P. Heckbert, editor, *Graphics Gems IV*, pages 230–236. Academic Press, 1994. (p. 61)

Karl Sims. *Evolving Virtual Creatures*. In Computer Graphics (SIGGRAPH 94 Proceedings), July 1994. (p. 26)

Michiel van de Panne and Eugene Fiume. *Sensor-actuator Networks*. In Computer Graphics (SIGGRAPH 93 Proceedings), volume 27, pages 335–342, August 1993. (pp. 25, 26)

Michiel van de Panne and Eugene Fiume. *Virtual Wind-up Toys*. In Proceedings of Graphics Interface 94, May 1994. (pp. 25, 26)

M. van de Panne. *From Footprints to Animation*. Computer Graphics Forum, 16(4):211–224, 1997. (p. 26)

Andrew Witkin and Michael Kass. *Spacetime Constraints*. In Computer Graphics (SIGGRAPH 88 Proceedings), volume 22, pages 159–168, August 1988. (pp. 15, 24)

Andrew Witkin and Zoran Popović. *Motion Warping*. In Computer Graphics (SIGGRAPH 95 Proceedings), August 1995. (pp. 17, 27, 94)

P. Xavier. *Provably-good approximation algorithms for optimal kinodynamic robot motion plans*. PhD thesis, Cornell University, April 1992. (p. 26)