Interactive Techniques for High-Level Spacetime Editing of Human Locomotion

by

Noah Lockwood

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Interactive Techniques for High-Level Spacetime Editing of Human Locomotion

Noah Lockwood

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2016

In recent years, computer animated characters have become commonplace, often appearing extremely lifelike in movies, television, and video games. Unfortunately, animating digital characters remains difficult and time-consuming, pursued mostly by dedicated professionals, because conventional animation methods require careful control and coordination of many different low-level parameters which affect the believability of a character's pose and timing.

In this thesis, we present techniques to allow high-level control of motions for animated human characters, by interactively editing existing locomotive motions. These techniques prioritize interactivity by allowing the user to manipulate more general or high-level aspects of the motion, such as a motion path. These techniques also have low data and computation requirements, as they operate on a single input motion without expensive optimization or simulation, allowing for instant feedback.

The foundation of this work is a novel path-based editing algorithm based on simple and well-established biomechanical and physical principles of motion. After user manipulation of a motion path, the poses of the character along the path are automatically adjusted to match the new path, and an automatic timewarping procedure adjusts the timing of the new motion depending on its context, in order to remain consistent with the original even after large changes.

Our two performance techniques utilize familiar input methods to control the path-based editing algorithm in different ways. First, based on a study exploring how users can express motion using their hands, our "finger walking" technique allows control of motion editing by mimicking full-body motion on a touch-sensitive tabletop. A further study was conducted to test automatic identification of the motion type of the performance, and to evaluate user satisfaction with the generated motions. The second technique uses handheld manipulation of

a mobile device, such as a tablet or smartphone, to mimic a motion through space. Using only the simple and standard motion sensors in these devices, we developed two gestural methods for editing motion, with both discrete, one-off gestures and continuous "steering wheel" style control of an ongoing motion.

Overall, the techniques presented in this thesis demonstrate that fast and interactive editing of human locomotion is possible using meaningful high-level controls, without requiring large amounts of additional motion data or computation.

Acknowledgements

I have had the great fortune over my graduate student career of working with and learning from many incredible professors. First and foremost, I am grateful for the support of my supervisor, Karan Singh. His encouragement, ideas, constructive criticism, and (most of all) patience have been essential to my success as an academic, a published researcher, and now a Ph.D. graduate. The other professors in the Dynamic Graphics Project lab, including the members of my advisory committee, Eugene Fiume and Ravin Balakrishnan, as well as Khai Truong, have also provided invaluable feedback and advice on a wide array of topics. I am also grateful to the other members of my Ph.D. examination committee, Daniel Wigdor and Michiel van de Panne, and its chair, Kaley Walker.

The students in DGP have been tremendous sources of knowledge, and motivation. My closest comrades Patricio Simari and Patrick Coleman inspired me with their work ethic, analytical skills, and good natures. Many fellow students shaped my work by freely sharing some of their expertise: Alex Kolliopoulos and C♯, Martin de Lasa's proofreading, Dustin Freeman and the Surface table, Ryan Schmidt and geometry, and Peter O'Donovan and machine learning. So many have been sources of help along the way: Michael Neff, Jacobo Bibliowicz, Evangelos Kalogerakis, Derek Nowrouzezahrai, the late Joe Laszlo, David Dearman, Jonathan Deber, Rorik Henrikson, Matthew O'Toole, James McCrae, and many others.

Many others from the University of Toronto have provided support. John Hancock, indefatigable systems administrator for DGP, who has always provided expert technical assistance. The staff of the graduate office, particular Vinita Krishnan and Sara Burns, who have helped me figure out the bureaucracy that I had to navigate from afar.

My friends have been constant sources of encouragement, humour, and good times, all of which has invaluable during my graduate career. The Institute is always ready for a gathering filled with wacky antics, and now the second generation is joining in on the fun. And the Random Dinner Night crew is a never-ending source of new adventures, culinary and otherwise.

My teaching experiences have been a truly enriching aspect of my time as a graduate student, and I am grateful for the opportunity to work as both a teaching assistant and course instructor while learning from so many great teachers. Diane Horton has been an inspiration and source of much great advice. Many thanks as well to Tom Fairgrieve, Paul Gries, Karen Reid, Michelle

Craig, Jennifer Campbell, Danny Heap, Daniel Zingaro, and Steve Engels.

Beginning with internships and continuing to the past four and a half years (and counting) of full-time employment, my experiences at Industrial Light & Magic as well as LucasArts have propelled me upwards as a thinker, engineer, and communicator. The "dream team" of my first internship - Ari Shapiro, David Lenihan, Jeff Smith, Brett Allen, and David Galles - treated me as an equal from day one and expected nothing but my best, which was a foundational experience of my adult and professional life. My internship experiences were also enriched by working with talented senior engineers and managers, including David Bullock, Nick Porcino, Charlie Hite, Ben Cooley, and Lucas Kovar.

As a Research and Development Engineer at ILM, I have continued to learn from incredible colleagues. I am grateful for the advice and encouragement about completing my Ph.D., from Steve Sullivan, Rony Goldenthal, Rachel Rose, Yuting Ye, and Kiran Bhat. I have worked with many talented and inspirational engineers, past and present: Mike Jutan, Matt Lausch, Kevin Wooley, Yoojin Jang, Sheila Santos, Mat Berglund, Eric Wimmer, Florian Kainz, Jeff Yost, and Philip Hubbard, and others. I am continually amazed by the talented artists I meet every day, and I'm especially grateful for the lessons and generous natures John Doublestein, Abs Jahromi, and Glen McIntosh. And the stage folks have welcomed me as I learned the ins and outs of motion capture: Mat Brady, Brooke Chapman, Mike Sanders, Rod Fransham, Lindsay Oikawa, and Sean Idol.

I am also grateful to many people who must, by necessity, remain anonymous. Many conference reviewers have provided constructive feedback on my research. And I have worked directly with thoughtful and generous folks who experienced my research hands-on and provided their thoughts, including my study participants and the expert animators.

I wouldn't be where I am today without the love and support of my family. Aunts, uncles, cousins, and my wonderful and now-departed grandparents, all were and are always in my corner. I am very fortunate to have joined my wife's equally loving family, including my parents-in-law who are always encouraging and interested in my work. My amazing sister and her family are always ready to lend a hand, or a hug. And my parents have always been there for me with lessons, encouragement, and support, for which I am eternally grateful.

Finally, and most importantly, I am who am I today because of the love of my life, my wife Isabelle. You never let me quit, never stopped believing in me and supporting me, and continue to be the best partner I could ever imagine. And our beautiful, incredible daughter, Amelia: you have brought so much joy into our lives, and so much more lies ahead. I treasure every moment I spend with the two of you. Thank you. I'm all yours.

Noah Lockwood
San Francisco, California, USA
September 2016

Contents

Chapter 1

Introduction

1.1 Motivation

For a long time, the most believable and compelling animated characters were traditionally animated, or hand-drawn. The development over time of principles of animation combined with the limitless potential of a blank page and the expertise of tireless animation artists allowed truly memorable characters and performances to be created. Now, computer animation technology and the associated skills have allowed the same level of performance to be achieved with computer animated characters, though the capabilities of animation to create memorable stories and characters remain the same. The great animators from Walt Disney Animation, Frank Thomas and Ollie Johnston, understood that the magic of animation is in the performance of a character, which makes animation "capable of getting inside the heads of its audiences, into their imaginations...Once the audience has become involved with your characters and your story, almost anything is possible." [109]

Computer animated characters have continued to become both increasingly complex and commonplace, leading to more nuanced character performances suitable to telling a variety of stories within any genre. Such characters populate an ever-increasing landscape of animated media, in television shows, movies, and videogames.

However, animation, and animated characters, are far less common outside of these types of linear and interactive stories. This is often because animation is a specialized professional pursuit, requiring large amounts of time for even the most skilled animators to produce a high-

quality result. As a result, endeavours which involve a large amount of character animation can require large budgets and long production timelines, with the biggest "blockbuster" games and animated movies taking years to complete, and costing tens of millions of dollars. While all aspects of these productions are time-consuming, animation is a significantly slow stage in a production pipeline.

Animating characters requires so much skill and time largely because the conventional approach to authoring animation is both *high-dimensional* and *low-level*. Spatially, creating an animation is high-dimensional since there are a large number of parameters which affect the character's pose, each of which must be specified explicitly. The timing of an animation increases the dimensionality of the task as well, since the paradigm of keyframes requires that some or all spatial parameter values are specified at multiple times.

Animation controls are also typically low-level, in that the parameters affect isolated, small parts of the character, such as the joints in a skeleton, and are specified at exact times. Extreme precision is required for these parameters, in units such as degrees, inches, and for timing, frames. And while the low-level parameters are generally specified independently, their values must be coordinated by the animator, to avoid creating impossible poses or unbelievable movement which doesn't appear consistent with laws of motion. This makes animation particularly "high stakes", since even the slightest changes in a small number of the many low-level parameters can stand between a believable animation and one which appears unconvincing or artificial.

However, these qualities which require large budgets and long timelines are not necessarily inherent, in general, to producing animation. The *conventional* method of animating characters is high-dimensional and low-level, but other approaches are possible, which could expand the ways in which animated characters are used if less time and precision was required.

One potential application of animated characters outside of entertainment media is the visualization of human motion. In team sports, for example, complex plays involving many co-operating players are visualized as a series of diagrams with a variety of icons, using arrows to indicate paths of motion. However, as static images, co-ordinated movement is difficult to illustrate. If animations of such plays could be produced as easily as sketching a diagram, more information could be communicated and further variations explored.

Visually planning the motion of characters is also useful for both live-action and animated productions. Before production begins, simple animations called "animatics" or "pre-visualization" are often used to explore character placement, movement, and timing, before committing to the costly and time-consuming process of or live action filming or polished animation. Unfortunately, the methods used to create animatics are the same as for complete animation. While animating for visualization purposes requires less time and effort to produce conventionally, any techniques which would allow the rapid production of simple animations would allow even more exploration of potential staging and movement.

The emerging field of virtual reality is another form of digital media where animated characters can have a significant impact. The movement of animated characters in virtual reality, including the user's own "avatar" representing their virtual body, must react believably to a user's behaviour and actions in order to preserve the immersiveness of the experience. Since it is prohibitive to produce animations for every possible situation that characters might encounter in an interactive virtual environment, techniques which can be used to automatically generate novel and believable motion in real-time would be invaluable.

Finally, professional animation of the highest quality could also benefit from different methods of controlling animated characters. Low-level control of animation is important for producing nuanced and detailed performances; however, as discussed earlier, this meticulous method of producing high-quality animation is extremely time-consuming. Unfortunately, this work must sometime be discarded entirely, as changes in staging, camera layout, or script can require re-animating a scene from scratch since hand-crafted animation cannot be easily modified. Techniques which could transform existing, high-dimensional animation to meet changing directorial intent would allow those animations to be re-used instead of manually re-created.

There is a large body of research which addresses some of the difficulties with traditional methods of animation by developing new techniques which are more accessible to users, while requiring less time and effort. There are many different approaches suited for many different animation situations, which require varying amounts of input data, user control, and computation time. Our work seeks to expand the research in this field by proposing and developing new algorithms and techniques in two main ways. First, by utilizing simple physical

and biomechanical rules to model the spacetime interdependence between the parameters of a motion, the user can modify solely what is most important, while the remaining parts of a motion are modified automatically to match. Second, by additionally adopting interfaces which involve simple physical mimicry, motions can be edited using that spacetime interdependence inherent in the user's input.

1.2 Terminology

This work examines the animation of *articulated characters*, which refers to characters with an underlying structural representation of their morphology, which is often used to separately deform or manipulate a representation of the character's outer surface. Therefore, the animation of articulated characters solely concerns the manipulation of this underlying structure, usually referred to as a *skeleton*. A skeleton is a directed acyclic hierarchy of rigid transformations, organized into a tree structure with a consistent *root* transformation. These transformations are often referred to as *joints* or *bones* with time-varying rotation and usually-constant translation. The root is an exception, with a variable translation representing the character's position in space. Non-root joints are often constrained to fewer than the full three rotational dimensions to model biomechanical limits of movement, such as the one–dimensional hinge joint of the human knee. A variable rotational or translational parameter of any joint in a character can be referred to as a *degree of freedom (DOF)* of the character.

A single complete set of character DOFs defines a character's *pose*, while time–varying DOFs generate animation. DOFs are often represented as continuous *motion signals*, which allow the evaluation of character pose at any arbitrary time within the defined time range of the DOF values. Alternately, DOFs can be represented discretely as a sequence of poses, or *frames*, usually sampled uniformly in time. Different animation techniques manipulate either the discrete or continuous motion representations, which can be converted between. Conventional animation approaches often involve the interpolation of an arbitrarily sparse set of *keyframes*, user–created partial or full poses, each with associated times.

1.3 Approach

As discussed in Section 1.1, the conventional method of creating animation with keyframes is an approach that is both high-dimensional and low-level, which results in animation requiring large amounts of professional time to create, and restricts its applications. Our goal in this work is to address these limitations by developing alternate *high-level* techniques for creating animation. As opposed to low-level control of individual joints in an articulated skeleton, high-level control is concerned with behavioural or overall aspects of animation, such as where a character is going, or what they are doing. This allows for the user to act more as a director, specifying goals or actions, with a character's low-level degrees of freedom controlled automatically to match the high-level control. While this approach means that not every aspect of a character's motion can be specifically controlled, our goal with high-level control is to allow animation to be used in a variety of novel situations where keyframing is not a viable solution.

Another fundamental element of our approach is the underlying process by which the input data is processed to generate the output animation. We refer to this as the animation operation. A variety of animation operations are discussed in-depth in Chapter 2 along with examples from related work. As there is a balance to be struck between the aforementioned requirements of an operation and the flexibility in results that it produces, we have chosen to develop techniques using the operation of combined spatiotemporal motion transformation (or simply *spacetime editing*), wherein the poses and timing of single pre-existing motion are modified to produce a new motion. Motion editing requires only a single input motion, as opposed to approaches which combine multiple motions or process a large corpus of motion data. Additionally, motion editing approaches can have relatively low computational requirements, allowing results to be generated quickly and using more readily available and even portable hardware.

A common approach in developing a motion editing technique is restricting its application to a particular type or class of motions. As illuminated by the overview of related work in Section 2.1.2 in particular, editing techniques can apply to motion classes as varied as those involving environmental contact, object manipulation, acrobatic maneuvers, or expressive gestures. For our editing techniques, we focus on a class of motions where the user's goal can potentially be clearly and quantitatively formulated: *locomotion*, where a character moves from location to

another. In editing locomotive motions, a user may have simple spatial goals, such as changing the direction or goal of a walk or the height of a jump. Our work aims to comprehensively and quickly edit motions to meet the user's goals, as well as explore different and accessible methods of specifying them.

In addition to a fundamental operation, a separate aspect of a motion generation technique is how the technique is utilized by a user. Many techniques do not require user input at all (as discussed in Section 2.2.1), however, in order to accommodate a variety of user goals, our locomotion editing techniques must be not automatic but *interactive*. In one sense of the term "interactive", we aim to enable the user to interact as closely with meaningful parts of the motion they are transforming, in a direct and accessible way, where the effect of a user's input is clear and simple, rather than manipulating more abstract or indirect controls. We also aim for the techniques to be "interactive" in the computational sense, to enable rapid user iteration by seeing the results as close to immediately as possible, if not in real-time while the input is occurring.

Finally, how this user interaction occurs is an important aspect of our techniques to formulate. Based on the assumption that many user goals for interactive locomotion editing are spatial in nature, enabling the direct specification of these goals through spatial user input is a straightforward but potentially effective approach. Strictly spatial input still allows for the transformation of locomotive motion in not just space but also time, since the spatial and temporal parameterizations of real motion are coupled by the laws of physics; therefore, changing the spatial or temporal parameters of a motion may require changing the other in a corresponding way in order to maintain realism or preserve qualities of the original motion.

This coupling of spatial and temporal parameters affects not just the motions to be edited, but can be accounted for in the user input as well. If a user's input is physically performed, then the spatial goals of the user can be potentially determined not just from the spatial aspects of the input, but inferred from the temporal aspects as well. Therefore, we also aim to explore techniques which utilize the relative timing of the user's input instead of the absolute timing. This allows our techniques to potentially treat performed input as representative of the final motion, rather than an exact specification of particular parameters.

By assuming that a user's performed input is internally consistent even if not absolutely

representative, its relative timing can be utilized in two ways. First, the relative timing within a single user input can utilized, such as the frequency of an event or characteristic. And second, the relative timing of multiple inputs can be examined to determine relative control, such as specifying that a motion should be edited to occur faster or slower.

1.4 Thesis Statement

As developed in the preceding section, our work presents an exploration of interactive techniques for high-level spacetime editing of human locomotion. Our thesis is the following:

> Creating and editing the motion of animated characters is a difficult and time-consuming task, requiring the specification of a large number of low-level parameters affecting a character's pose and timing. However, by utilizing the space-time information of a user's performance as well as well-established biomechanical and physical principles, high-level techniques can be developed to allow quick and effective editing of locomotive animations.

1.5 Contributions and Outline

The contributions of this work are:

- **A novel categorization and analysis of motion generation techniques.** Our categorization axes provide an abstract means of evaluating the entire field of motion generation research in terms of techniques' capabilities and requirements. This enables meaningful comparisons and connections to be drawn between related pieces of research, and allows the identification of a predominant shortcoming in how animation timing is controlled, which informs the remainder of this work.

- **Efficient algorithms for motion path generation and modification.** We enable meaningful user manipulations of motion paths based on novel criteria for identifying manipulation handles, as well as motion path-specific refinements to a standard curve deformation algorithm. Abstract or "overall" paths with desirable properties can also be determined from manipulation handles or from environmental contacts.

- **A flexible and automatic method for motion timewarping.** Our efficient timewarping algorithm automatically determines the timing of a modified motion using contextual rules based on physical principles and biomechanical observations, which generalize to many different types of locomotion. A general timewarping framework allows timing throughout a motion to be continuously adjusted based on criteria such as contact state, turn sharpness, and stride length.

- **An exploration of the capabilities of hand motion performance.** We present an exploration of types of hand performance and its limitations in a controlled study, with the results analyzed to explore the similarity between full-body and hand motions in a variety of ways as expressed through surface contact, such as the number of simultaneous contacts over time, as well as the frequency and mechanics of the contacts during a performance.

- **Algorithms for contact-based motion analysis.** Based on our analysis of hand performances, we define a concise set of descriptive features which can be used to reliably recognize finger motions based on surface contact over time. We also develop a heuristic for transforming full-body motions to match hand performance, and evaluate its performance based on user feedback.

- **Algorithms for gestural motion editing.** We utilize familiar and common mobile devices and their simple motion sensors to develop further techniques for editing motion based on user performance. After examining the data available and its quality, two techniques are developed for editing motions with single mimicked "discrete" gestures, as well as real-time steering control in the form of a "continuous" gesture.

In Chapter 2, we review and categorize previous motion generation research. In Chapter 3, we present a technique for interactive motion editing based on user manipulation of motion paths. In Chapter 4, we present a motion editing technique based on the novel input method of hand performance, and discuss the capabilities of this method. In Chapter 5, we present a method for gestural control of motion editing using handheld manipulations of a mobile device. Finally, in Chapter 6, we conclude with a summary of our research, a discussion of its capabilities and limitations, and its implications on future work.

Chapter 2

Related Work

There is a large body of research which addresses the problem of creating novel motion on an animated character, while requiring less time and effort. We examine these works, and the algorithms they present, by categorizing them from the perspective of a potential user along two axes: the type of *animation operation* the work presents, and the type of *user-specified parameters* which affect the algorithm. Table 2.1 presents an overview of these works, classified by the specific categories within each axis.

While all of the work we examine shares a common *task* – the generation of novel animation – different approaches can be categorized by their *animation operation*, which is defined by the type of input data they utilize and how that data is processed to produce novel animation. *Posing* is a common operation in producing animation by hand using traditional animation software, where poses are interpolated using simple procedures to form motion. *Motion transformation* generates animation by modifying a single input animation; transformations can affect a motion spatially, temporally, or both. *Motion blending* involves the combination of multiple animations to create a novel animation; motion splicing and interpolation are two forms of simultaneously blending motions, motion sequencing is the sequential blending of motions, and blending approaches which combine both are also possible. Finally, *motion synthesis* creates novel animation using rule-based systems or models which are either manually specified or generated from a set of input animations; synthesis can encompass kinematic or dynamic processes, or techniques which use both kinematic and dynamic rules.

Another important axis along which we can categorize these works is by their type of *user-*

specified parameters. Some approaches are entirely *automatic* and require no user input, or require only *abstract parameters* that do not directly correspond to character pose or timing. A *single spacetime value* can be used to specify one spatial or temporal "goal" or constraint, such as a target position, a new pose at a particular time, or conversely, a new time for a particular pose. Slightly more complex is specifying a *discrete spacetime set* of values, which indicates multiple goals to be satisfied, such as a set of new poses ("keyframes") either relatively ordered or each associated with absolute times. Finally, user input can also consist of *continuous spacetime* values, where the user performs input in real–time, such as "videogame–style" interactive control of a moving character.

			User Parameters			
			None or Abstract	Single Spacetime Value	Discrete Spacetime Set	Continuous Spacetime
Animation Operation	Posing			Davis03 [19], Grochow04 [33]	Igarashi05 [41]	Öztireli13 [83]
	Transformation	Spatial	Gleicher03 [32], Ikemoto06 [42], Kovar02 [55], Shin03 [97]	Gleicher97 [29], Popović99 [88], Tak05 [107]	Coleman12 [14], Gleicher01 [31], Ho10 [37], Jain09 [49], LeCallennec04 [62], Lee99 [64], Li06 [67], Shapiro07 [96], Witkin95 [121]	Abe04 [1], Dontcheva03 [23], Shin01 [98]
		Temporal	Hsu05 [40], Menardais04 [75]		Mukai09 [78]	Terra04 [108]
		Combined	Wang06 [119]	Coleman08 [15], Majkowska07 [71], Neff03 [79], Sok10 [103]	Jain09 [48], Kim09 [50]	Yin03 [125]
	Blending	Splicing or Interpolation	Heck06 [36], Ikemoto04 [44], Ikemoto07 [43], Majkowska06 [70]	Kovar04 [53]	Mukai05 [77]	Park02 [84]
		Sequencing	Ren10 [90]		Arikan03 [6], Arikan05 [7], Lee06 [65], Lin06 [68], Sung05 [106]	Gleicher03 [32], Kovar02 [54], Kwon05 [56], Lai05 [57], Lee02 [63], Levine09 [66], McCann07 [74], Treuille07 [113], Wang06 [120]
		Combined		Cooper07 [16], Rose98 [91], Shin06 [99]	Bruderlin95 [12], Perlin96 [87], Pullen02 [89]	Heck07 [35], Kovar03 [52], Safonova07 [93], Shum08 [101]
	Synthesis	Kinematic	Brand00 [11], Hsu05 [40], Lau09 [61], Shapiro06 [95]	Feng12 [27]	ElKoura03 [24], Perlin95 [86], Yamane04 [122]	Barnes08 [9], Chai05 [13], Min09 [76], Seol13 [94], Thorne04 [110]
		Dynamic	Nunes12 [82], Sims94 [102], Wampler09 [116], Yin07 [124]	Hodgins95 [38], Laszlo96 [59]	Allen07 [4], Coros08 [17], Coros11 [18], Faloutsos01 [25], Ha14 [34], Kokkevis96 [51], Tsang05 [114]	deLasa10 [21], DiLorenzo08 [22], Laszlo00 [60], Wang09 [118], Zhao05 [126]
		Combined	Rose96 [92], Wampler14 [117], Zordan02 [127]	Neff05 [80], Zordan05 [128]	Fang03 [26], Liu02 [69], Neff08 [81]	Abe06 [2], Ishigaki09 [47], Tournier12 [112], Yang04 [123]

Table 2.1: An overview of the works we will discuss, classified with respect to their animation operation and user–specified parameters.

2.1 Animation Operations

2.1.1 Posing

While current animation software requires the use of partial or complete poses in keyframes to create animated characters, mainstream tools for creating and manipulating poses are often simplistic. As a response, some research has investigated novel posing techniques, which can be separately used to create animation by either interpolation over time, by real–time interactive re–posing.

The approach of Davis et al. [19] uses hand–drawn pose sketches and, due to the ambiguity of the user's two–dimensional input, their system provides a number of possible three–dimensional skeleton poses. They cull poses which violate typical ranges of motion and rank the remaining poses using pre–set criteria, such as preferred joint angles and relationships between joints which preserve balance. The top–ranked pose is selected by default, and the user can incorporate pose corrections from any of the other ranked poses.

Style–based Inverse Kinematics by Grochow et al. [33] learns a probability distribution over the space of all possible poses, based on the poses from an input animation. The user can then set a number of constraints on the character's pose, and the system determines the most likely pose satisfying those constraints. Pose constraints in image space are represented as three-dimensional constraint rays, and pose style is determined by the style of the input animation, such as baseball motions or walking.

Igarashi et al. [41] present a novel method for generating poses based on a small set of user–specified poses. Each pose is associated with a position in a low–dimensional manifold, and the algorithm they present uses radial basis function interpolation and a novel orthonormalization technique to interpolate and even extrapolate the pre-set poses. This allows for interactive control of spaces of poses with user–defined parameterizations, controlling behaviours such as character gaze directed at the control position, or walking.

Öztireli et al. [83] present a novel blending algorithm to deform a character's surface mesh around its skeleton, and a sketch-based interface for posing. While single bones can be bent and deformed with a sketched curve, a "line of action" curve can also control a set of selected bones. By determining the bone which best fits the line of action curve, a parameterization is

determined which drives the matching deformation of all other selected bones, yielding a viable and expressive pose from a single curve.

2.1.2 Motion Transformation

Spatial Transformation

Spatial transformation modifies the poses of a character but not their timing; this is equivalent to modifying the pose of each frame in a discrete (frame–based) representation, or modifying amplitude in a continuous (motion signal) representation. There are a number of applications, and both dynamic and kinematic approaches.

Automated motion *cleanup* is a common application of motion transformation, to correct visual artifacts in an animation. Both Ikemoto et al. [42] and Kovar et al. [55] present complimentary methods to address "footskate", where characters' feet do not remain stationary during apparent ground contact. Ikemoto et al. present an "oracle" system trained on annotated animations which can robustly detect ground contacts, while Kovar et al. present a method which uses pre–detected ground contacts and a novel geometric re–posing algorithm to ensure stationary feet during ground contact and smooth limb motion.

A similar cleanup problem is addressed by Shin et al. [97], who present a transformation technique to increase the physical plausibility of motions. With an estimated mass distribution of the character, they enforce the preservation of proper balance and torque during ground contact phases of the motion, and ballistic trajectories during flight phases, by applying a set of predetermined parameterized transformations.

Motion retargeting is an application of motion transformation which involves transferring a motion from its original skeleton to a new one, while maintaining the content of the motion as well as skeleton–appropriate plausibility. Gleicher [30] approaches this problem by allowing low–level user–defined constraints such as joint positions or separations. Spacetime optimization is used to solve the non–linear problem of determining an animation on the new skeleton which meets the constraints, which can be easily transferred in an absolute or relative sense; for example, retargeting a walk to a smaller character can maintain the absolute footplants with larger steps, or take smaller, scale–appropriate steps.

Tak and Ko [107] approach retargeting by incorporating similar kinematic constraints (which

can be user–specified), but also physical constraints such as balance, momentum, and realistic torque limits. Their approach operates in an on–line manner as a sequential filter, modifying successive frames of an animation in sequence by sampling and combining different solutions in their nonlinear solution space, instead of solving for the entire transformation simultaneously.

Shin et al. [98]also present their retargeting algorithm as a filter, to allow performers to control a virtual puppet character in real–time using motion capture. Their algorithm determines a number of constraints which may not all be satisfiable by the new character. They determine the "importance value" of each constraint heuristically, taking the character's actions and environment into account, and the most important constraints are then satisfied using a novel analytic inverse kinematics solution.

A common motivation for motion transformation is to satisfy one or more user–specified pose changes, while maintaining the smoothness and characteristics of the original motion. Witkin and Popović [121] satisfy these user–specified poses with a signal processing approach, directly modifying each degree of freedom's values over time by applying limited scaling and offsetting. Gleicher [29] accomplishes the same task using spacetime optimization, applied to single pose adjustments which are specified interactively.

Lee and Shin [64] also present a transformation method which responds to interactive pose changes, but also allow modification of a character's skeleton. They satisfy single pose changes by maintaining positional constraints using a novel inverse kinematics algorithm, and incorporate any number of changes into the overall motion by using multi–level B splines fitted to the character's degrees of freedom. They then solve for displacement maps to these splines in a coarse–to–fine fashion which generate the changed poses while matching sampled values of the original motion.

Coleman [14] also presents a method for transforming a motion given changes to key poses, but the poses are automatically identified as extreme or significant by one of a number of metrics. Once the poses have been adjusted, a series of prioritized constraints are applied which transform the motion to - in order of precedence - preserve ground contacts and avoid ground penetration, match the modified poses, and preserve end effector and root motion. This allows the approach to, in essence, treat the modified poses as soft constraints which may be only partially applied in order to preserve various aspects of the original motion, to generate

plausible results.

A more indirect approach to transformation is presented by Gleicher [31], who uses a path–based parameterization of character movement to allow motion transformation driven by path editing. The technique represents the root trajectory as a displacement map applied to a spline representing the overall path, resulting in the preservation of high–frequency motion elements when the spline is changed. Contact constraints are then detected and satisfied using conventional inverse kinematics.

Transformation can also be based on introducing or modifying how the character interacts with external phenomena. Jain and Liu [49] modify the motion of a character to maintain physical plausibility of human–object interaction by extending previous work on optimal control of a rigid body simulation to include kinematic parameter's of a character's motion. By coupling the rigid simulation and kinematic edit at predefined points of impact, a single optimization allows the modification of one to affect the other.

Shapiro et al. [96] transform motions to avoid or enforce character collisions with the environment. They present a novel motion planner that incorporates pre–existing body motion as constraints, which results in transformed motions that avoid collisions but retain maximum similarity to the original motion. A simple extension of the planner can enforce collisions at particular times, generating a motion in which new object manipulations occur seamlessly.

Ho et al. [37] present a representation of character pose called an "interaction mesh" which can be used to retarget the motion of one or multiple characters, or a character interacting with the environment. At each frame, a Delaunay tetrahedralization of the point cloud formed by bone and object positions is used to obtain a volumetric mesh. Driven by changes in skeleton configuration or user-specified position constraints, a Laplacian deformation of the mesh, combined with collision detection, determines new poses which maintain the implicit spatial relationships of the original motion.

Physical calculations can also be used for motion transformation. Abe et al. [1] present a method which determines time–varying curves describing a character's linear and angular momentum during a motion. To satisfy constraints such as a new landing position of a jump, the motion is transformed with spacetime optimization to satisfy the new constraint while maximizing similarity to the previous poses and momentum curves.

Le Callennec and Boulic [62] combine physical and kinematic calculations to transform a motion. They present a framework for detecting or specifying constraints from the original motion, including kinematic constraints such as trajectories of skeleton parts, and physical constraints such as center of mass. A novel inverse kinematics algorithm applies the constraints in user–specified priority order on a per–frame basis, and smooths the results if necessary.

Popović and Witkin [88] use a simplified version of the character to apply physically–based transformations. For any input motion, they determine a simplified skeleton containing the minimal number of DOFs necessary to express the coarse actions of the motion. This reduced–dimensionality model can then be transformed more efficiently through spacetime optimization to maintain its physical features while satisfying new constraints, and the resulting motion is mapped back onto the full character.

Finally, motions can also be transformed for a expressive effect rather than to meet explicit spatial goals. Dontcheva et al. [23] allow a user to create motions in layers by performing the motion of character DOFs separately. Performance can also be applied relatively, to exaggerate pre–existing components of a motion. A novel correspondence algorithm allows an initial performance to select and determine mappings for a number of (potentially offset in time and magnitude) DOFs which the exaggeration will be applied to.

Temporal Transformation

Temporal transformation, also called "timewarping", modifies strictly the time parameterization of motions. In a continuous setting, motion signal frequency is modified, but every pose from the original motion still occurs, though at a potentially different time. For discrete frames, some form of interpolation may be necessary to maintain uniform frame spacing in time. In either case, timewarping is often represented as a one–dimensional mapping of original time to new time.

Mukai and Kuriyama [78] allow interactive timeline–based timewarps of arbitrary segments of a motion, which can also be applied to specific parts of the character. They use a novel matching technique to identify simultaneous coordinated aspects of a motion which should be timewarped together, as well as similar parts of the motion across the entire timeline which can be modified by automatically propagating a timewarp, instead of the user performing similar

manipulation repeatedly.

Hsu et al. [39] present a discrete technique which uses the timing of a "guide" motion to determine the timewarp of an input motion. They compress motions by selecting a subset of the input motion frames, such that the approximate acceleration and velocity of the remaining frames locally matches velocity and accelerations from a corresponding frame of the guide motion.

Terra and Metoyer [108] determine a timewarp to match a real–time user sketch of the motion. They allow an approximate sketch, and correspond the sketched motion and input motion by matching geometric features of their paths through space. Once corresponded, the relative timing of each segment of the user's sketch is used to generate a piecewise timewarp of the user motion, and the timewarp is approximated with spline segments for smoothness.

Combined Transformation

There are some transformation approaches which work in both the spatial and time domains; some can be applied to either domain separately, while others affect both domains sequentially or simultaneously.

Wang et al. [119] automatically transform a motion both spatially and temporally according to a number of traditional animation principles. They formulate a simple filter based on Gaussian convolution which can be applied to individual motion signals to exaggerate their anticipation and follow–through of large changes.

Majkowska and Faloutsos [71] present a technique to spatially and temporally transform acrobatic motions. They identify ballistic phases in a motion and use those to estimate the character's overall physical parameters and per–frame linear and angular momentum. This allows a physically-valid spatial transformation by adding rotation around a character's angular momentum vector, and uses the properties of uniform timewarping to ensure continuity of linear momentum.

Kim et al. [50] allow interactive path modifications to modify both the spatial and temporal aspects of multi-character motion. Multiple user-specified constraints in either time or space can be satisfied by applying Laplacian edits to the motion signals, which maintains high–frequency aspects of the motion but meets constraints precisely. Constraints may also be automatically

detected, such as characters interacting at a particular time, which results in simple edits being smoothly propagated to multiple characters to maintain their synchronization.

Jain et al. [48] use a sequence of hand-drawn keyframes to drive the transformation of a single motion capture segment matched from a database. A standard timewarping approach is first applied, using a novel distance metric between the motion capture motion and two–dimensional keyframe sequence, after which the timewarped poses are spatially modified to match the 2d sketches by inferring their missing depth.

Yin and Pai [125] use a similar approach, by taking the user input through a foot pressure sensor pad to select and modify a clip from a motion database. The user's real-time input drives a novel inverse kinematics technique to synchronize the user and character footplant positions, and time-invariant motion feature correspondence between the two motions determines a piecewise linear timewarping at runtime.

Neff and Fiume [79] present a number of related transformations which can be used to alter character expressiveness rather than meet specific spatial goals. They enable the temporal transformation of motions by modeling succession, or the time–delayed progression of motion down a chain a connected joints. Spatial transformations presented include controlling the amplitude of a motion compared to a pre–defined reference pose, and a number of "extent" parameters which control how connected joints move relative to each other (e.g., hand and shoulder, arms and body).

Coleman et al. [15] present a new type of motion representation that identifies "staggered poses", related extrema in motion signals which may not occur at precisely the same time. These identified poses can be used for both spatial transformation, such as exaggerating a motion, or temporal transformation, such as modifying the absolute time of a pose or applying a generalized succession edit to a chain of joints included in the pose.

Sok et al. [103] enable indirect control of a motion's spatial extent and timing through manipulation of linear and angular momentum. Selecting a particular axis from a motion allows the user to modify that axis' momentum magnitude over time; for example, a jumping motion can go higher by increasing its vertical linear momentum. Additional spatial or temporal constraints maintain important motion features such as landing position.

2.1.3 Motion Blending

There are two ways to combine multiple motions simultaneously into a single new motion: splicing motions involves combining unmodified and mutually exclusive sets of DOFs from different motions together, while interpolating motions combines the same DOFs of different motions to form novel motion signals.

Regardless of the particular method, a common requirement for combining multiple motions simultaneously is to synchronize the timing of motions, so that similar spatial features occur at the same time. Menardais et al. [75] augment the motion signal matching used traditionally by examining a motion's support phases, i.e., the sequence of character ground contacts. They use a number of heuristics to determine which support phase orderings are compatible for matching, and determine a piecewise linear timewarping to align the matched support phases of their two input motions, which allows splicing or interpolation to occur without introducing motion artifacts.

For splicing, Heck et al. [36] use a number of techniques to increase the quality of motion generated by splicing upper and lower body motions together. A novel timewarping technique synchronizes the spliced motions, and a broad spatial alignment and more refined "posture transfer" modify the final motion to preserve aspects of the original motion while retaining the high frequency details of the new, spliced–in motion.

Majkowska et al. [70] focus particularly on the splicing of body and hand motions, since detailed motions of both are difficult to capture simultaneously using motion capture, due to the difference in scale. The technique corresponds the timing of hand and body motions through a standard timewarping algorithm, using a number of novel distance metrics based on the motion capture markers in common between the motions, which then allows straightforward splicing of the timewarped hand motions onto the original body motion.

Ikemoto and Forsyth [44] present a method that generates spliced motions in an undirected fashion to provide variety in a collection of motions. A number of rules can be applied to splice pre–determined parts of annotated character motions together, and a specially trained classifier is used to automatically determine which splices are successful.

Interpolation is often used to determine intermediates between complex or long motions whose parameter space is difficult to sample fully. Kovar and Gleicher [53] use a novel motion

search method to find "families" of similar motions in a large motion database which can be interpolated via parameterizations defined by the user at runtime. They exploit their motion search algorithm to easily synchronize the motions before interpolation, and to determine a well–distributed set of motions to interpolate.

Mukai and Kuriyama [77] present a new method of motion interpolation using the statistical technique of geostatistics, which predicts continuous distributions of interpolating variables from given samples. Corresponding motions are synchronized through timewarping, and a standard pose distance metric is used on all poses against a reference pose to determine a single distribution among poses, corresponding to low–dimensional parameters like the target position of a reaching motion

Park et al. [84] present a novel incremental method for determining a monotonic timewarp to synchronize locomotive motions before interpolation. They automatically determine the high–level motion parameters of speed and turning angle, and use a robust angle interpolation technique to generate interpolated poses as determined by the runtime user control of the locomotion parameters.

Another application of interpolation is to generate motion transitions. Ikemoto et al. [43] present a method to transition between arbitrary frames of an animationby using a pre-computed table of valid transitions between motions to generate a longer but more visually plausible transition by interpolating through one or more "intermediary" motions.

Finally, Bruderlin and Williams [12] focus on interpreting motions as time–varying signals, and apply the signal processing technique of multiresolution filtering to create new motions. Single motions can be filtered at multiple spatial resolutions, and each resulting band can have its gain adjusted individually, to emphasize higher– or lower–frequency aspects of a motion, and such adjusted motions can be straightforwardly interpolated, yielding more control than simple linear motion interpolation. Timewarping is also used as a preprocess when the motions to be interpolated don't align temporally.

Sequencing

Motion Sequencing creates a new motion by concatenating multiple motions sequentially in time; usually, algorithms are also concerned with segmenting input motions into smaller

sequences (usually referred to as "clips"), and determining where transitions between clips or frames can occur. These methods determine the output sequence of motions in various ways, depending on their application.

Motion Graphs by Kovar et al. [54] is a foundational work in motion sequencing. The technique identifies transitions between a source and target motion (which may be the same) by finding the local minima of a similarity metric among all pairs of both motion's frames. This leads to a graph structure, where transitions are edges between vertices representing motion subsequences uninterrupted by transitions. Once the graph is pruned, the authors present a search strategy to find graph walks which meet arbitrary criteria, which they specifically apply to path specification. Work developed in parallel by Lee et al. [63] uses a similar function based on figure pose and velocity to find transitions within a motion database, but builds a higher–level graph structure using cluster analysis to group strongly related motion clips together.

Gleicher et al. [32] present a technique for generating a particular type of motion graph called a "move tree", aimed specifically at real–time applications such as games. Their technique allows the user–directed formation of such a structure, which detects motion clips that can be transitioned to and from a central "hub" pose, and finds transitions which can implemented as straightforward cuts from one clip to another without interpolation, for run–time efficiency.

Kwon et al. [56] present another sequencing algorithm for real–time applications. Focusing on locomotion, they segment motions using rules based on biomechanical observations, and classify these segments into preset groups corresponding to difference locomotion phases. They construct a two–level hierarchical motion graph where the different motion phases and transitions subsume the groups of motion segments, which is used to determine the coarse– and then fine–level nodes which will provide the next frame of animation.

Treuille et al. [113] also approach the problem of on–line sequencing, but without constructing an explicit graph structure. Instead, their algorithm considers all blended transitions between manually segmented motion clips, each representing a character's single step of various types of motions. They use a linear programming approach to determine a near–optimal control policy in a lower-dimensional space, consisting of weights of a small number of basis functions over the character's state state. This policy includes motion transition costs over time, allowing their algorithm to choose high–cost transitions in the short term which

minimize longer–term error better than a greedy approach.

McCann and Pollard [74] present an on–line approach which also considers the cost of motion transitions over time, but examine the often–opposing goals of generating motions which are both smooth as well as reactive to changes in user control. Sample user input is used to build a table of the conditional probabilities of transitioning between input states over time, which allows the selection of subsequent clips to be based on a weighted combination of how smoothly the character motion changes and how well the evident control of a clip matches the expected upcoming changes in user control.

Levine et al. [66] generate a motion sequence in real–time using solely voice input. They train a system on synchronized voice and motion data, which classifies the motion into general gestures, and identifies prosody (rhythm) features in the voice data based on pitch and intensity. A Hidden Markov Model is then optimized to associate gestures with their driving prosody features, which is used in real–time to predict and potentially correct upcoming gestures based on a voice stream, which are interpolated to maintain continuity during transitions.

Arikan et al. [7] use motion sequencing to generate realistic character reactions, or response motions, to user–controlled impulses (pushes). They train an "oracle" with random combinations of pre–impact motion, response motions, and impulses, and for each, the oracle generates a candidate motion sequence by applying a novel deformation algorithm to the response motion to generate a reaction to the impulse. User–approved combinations are incorporated into the oracle, which uses scattered data interpolation in real–time to determine the appropriate response motion to transition to and deform.

Removing the constraint of on–line motion generation allows sequencing algorithms to incorporate more complex types of input, such as work by Arikan et al. [6] which allows a user to generate a motion sequence by specifying motion annotations over time. A dynamic programming algorithm is used to sequence representative multi–frame "blocks" from a pre–annotated motion database, progressively refined by using shorter blocks similar to the enclosing block of the previous pass. The sequence is optimized to maximize similarity to the timing of the provided annotations while maintaining frame–to–frame smoothness.

Lin [68] uses a sequence of sketched 2D poses to generate a motion sequence. Once sketched poses have been aligned with poses from segmented motion data and the best matches identified

as "key poses", a sequence must be generated between each pair of key poses. Unmodified motion segments are used when two key poses belong to the same segment, and the best pair of transition frames is automatically determined between two key poses in different segments.

Wang and van de Panne [120] present a voice–directed motion sequencing system. A pre–specified set of voice commands is used to select or refine different types of motions, which can be directed by setting a destination using a conventional GUI. The system uses a pre–authored motion graph to allow meaningful semantic motion categories of interruptible continuous motions, uninterruptible discrete actions, and composable motions. A path planning algorithm allows motion sequences to occur while a character travels to a destination or follows another character, while avoiding obstacles.

Lee et al. [65] allow indirect control of a motion sequence through a character's environment. Pre-specified "unit objects" in the environment, such as steps or slides, allow the system to automatically segment input motion data into "motion patches", where each patch consists of similar clips which occur on or between the same unit objects. The patches are then assembled into a motion–graph like structure, where transitions can occur between two patches with similar enough beginning/ending boundaries. Motion sequences can then be created by assembling a novel environment from the unit objects, and then directing a character through the environment.

Beyond single characters, Sung et al. [106] use a motion graph to quickly generate motion sequences for crowds of characters who avoid each other. Given the initial and final positions of a character during a timestep, which can be generated by a path planner or crowd simulator, the algorithm greedily finds two graph walks called "seed motions", each of which generate a sequence that begins or ends with the initial or final constraint, respectively. The seed motions are merged by randomly perturbing clips in each with valid alternative clips, and then attempting to find similar poses between the two perturbed sequences where they can be merged.

Lai et al. [57] also use motion graphs for animated groups of characters, but represent the movement of the entire group. The input is group animation generated by simulation and flocking rules, augmented with a local "configuration space" of the group at every frame, based on average character velocity. This allows representative configurations of the group (the

equivalent of "key poses" for a single character) to be determined via k–means clustering. A novel and flexible correspondence algorithm can then be combined with constrained simulation to generate aligned transitions between different pairs of configurations in varying spatial arrangements, finally generating a conventional motion graph structure.

Ren et al. [90] present a semi–autonomous approach for constructing a motion graph to generate transitions between motions. An optimization process is used to determine a sequence of pre–existing frames which will form each transition (i.e., edge) in the final motion graph. Since these optimizations are computationally expensive and may not always produce satisfactory results, a selection policy for the optimizations is determined by adapting to user input of accepting or rejecting potential transitions, reducing the number of attempted optimizations to generate the final motion graph.

Combined Blending

Combined blending techniques generate novel motion by incorporating both the simultaneous combination of motion splicing or interpolation, and the sequential combination of motion sequencing.

A common approach in combined blending is to augment motion graphs with a splicing or interpolation technique. Heck and Gleicher [35] present a method to create "parametric motion graphs", which allow motion sequencing similar to regular motion graphs, but of parameterized motion spaces (identified using the technique of Kovar and Gleicher [53]) instead of single motion clips. Each directed edge between nodes contains a set of random parameter vectors sampled from the source motion space, with each vector corresponded to subspace in the parameter space of the target motion which will yield good transitions. Interpolating the subspaces between samples allows target motion space parameters to be determined that yield a good transition, from any set of parameters in the source motion space.

Shin and Oh [99] present a similar method using motion graphs and parameterized motion spaces. The authors focus on a novel parameterization technique by augmenting the user–directed, hub–based graphs of Gleicher et al. [32] with "fat edges", formed by merging normal edges which represent similar motion sequences transitioning between the same poses. Similarities in the fat edge motions are detected, and a best–fit two–dimensional manifold is

used to spatially parameterize the motions (e.g., by a target position for kicking motions, and so on).

Shum et al. [101] use a single-character motion graph to generate multi–character animations by forming "interaction patches" describing how two character's motions can interact, and detecting possible transitions between these patches. Interaction patches can be combined sequentially, allowing two characters to interact in multiple ways or for a character to interact with multiple other characters in turn. The patches can also be spliced simultaneously, allowing a normal two–character interaction patch to involve additional characters through compatible patches.

Safonova and Hodgins [93] present a technique which generates motion by interpolating two motion sequences from the same motion graph. They generalize their graph into an interpolated motion graph, in which each node represents an interpolable pair of nodes from the original graph. This graph is "unrolled" into absolute coordinates so that graph traversals leading to constraints can be detected. This massive graph is culled, and a constraint–satisfying traversal of the graph is determined using A$*$, informed by some novel heuristics which evaluate the progress of both the character position and motion graph state toward the goals.

Cooper et al. [16] present a algorithm to generate parameterized motion tasks that uses a motion graph–like structure to interpolate and sequence motions in real–time. They use an active learning approach where the system generates "pseudoexamples" for particular parameter values; user–approved pseudoexamples are incorporated, while rejected pseudoexamples are replaced by a user–performed correct motion. Similar motions are clustered, and given real–time parameter changes, the algorithm selects a weighted interpolation of a motion cluster to meet the parameters.

Some combined blending approaches focus on motion interpolation, but allow motion sequencing at runtime by interpolating one motion to the next, while time progresses. The system of Rose et al. [91] allows a sparse sampling of motion variants ("verbs", such as walking) to drive a user–defined parameterization for motion interpolation in intuitive terms ("adverbs", such as happy). The system determines timewarps to correspond motions of the same verb, and radial basis functions allow for the interpolation of each motion's degrees of freedom, weighted by their similarity to the user–specified adverb parameters. A user–constructed "verb graph"

determines how specified verb combinations can be transitioned between at runtime, generating a motion sequence.

Kovar and Gleicher [52] present a novel method of animation interpolation which provides a more robust method to handle motions that differ in timing, direction and path, and contact. They construct a "timewarp curve" which maps between frames of each motion, and from that, an "alignment curve" which dictates 2D rigid transformations at each frame to align the motions. Two or more motions can then be interpolated by interpolating the curves and applying the resulting transformations.

Pullen and Bregler [89] use a signal processing approach to sequence and interpolate motion capture data. Given a simple keyframed animation of a subset of a character's degrees of freedom (DOFs), their technique matches fragments of the animated DOFs to a small set of similar motion fragments from motion capture data, each with its own corresponding motion fragment for the unspecified DOFs. The smoothest possible sequence of those fragments is selected for the unspecified DOFs, which are aligned and smoothed (interpolated). The resulting signals are then spliced to complete the original partial animation.

Finally, Perlin [87] uses an animation engine capable of splicing, interpolation, and sequencing to assist authors in creating interactive behaviour–based characters. After the user defines small character actions such as gestures with an "action script", the system automatically sequences or splices actions depending on whether they can occur simultaneously or not. A simple motion graph structure can also be introduced by specifying "buffer actions" which must occur in between two actions which aren't directly interpolable.

2.1.4 Motion Synthesis

Kinematic

Kinematic motion synthesis creates novel motion through rules–based systems that operate purely on kinematic data, without considering physical factors involving masses or forces.

Data–driven approaches are common in kinematic synthesis; these approaches analyze sets of similar input motions and extract and modify particular learned motion parameters. Lau et al. [61] analyze similar motions and model their spatial and temporal variations using a Dynamic Bayesian Network. This model represents a probability distribution from which new

variations in spatial and temporal parameters can be sampled, and the corresponding motions synthesized, with the addition of a novel inverse kinematics formulation which is incorporated into their distribution model.

Brand and Hertzmann [11] approach the problem of synthesizing motions in particular styles, by learning patterns of a set of input motions using a "style machine". Their approach develops a Hidden Markov Model with states that model the "structural" segments of a motion with probability distributions in pose space. They determine viable transitions between states, and augment the model with a learned low–dimensional style vector which accounts for how the structures are performed. This allows the modification of a motion's style by varying the style vector of a pre–set sequence of states, or the formulation of a new style by interpolating between machines with particular styles.

Similar "style transfer" can be synthesized using just two input motions which are the same in content but differ in performance. Hsu et al. [40] approach this problem by determining separate linear transformations in space and time to correspond the motions, and then optimize a simple linear model to account for the two corresponded motions' variations of the same content. Shapiro et al. [95] use Independent Component Analysis to identify and extract "style components" expressing the difference in motion signals, which can be applied to any motion which is timewarped for correspondence.

Min et al. [76] also present a statistical approach to motion synthesis, but by modeling the variation in both the spatial and temporal parameters in a set of related motions. After semi–automatically determining timewarps to align all motions with a representative example, they determine separate low–dimensional models of the pose sequences and timewarps. They then use a Maximum A Posteriori framework to determine the most likely low–dimensional pose sequence and timewarp which satisfies user–specified constraints, and these low–dimensional parameters are then used to reconstruct the full–dimensional animation.

Kinematic synthesis can also be useful to generate complete motions from high–level or incomplete user input. Perlin [86] presents a system which allows the creation and combination of character actions based on periodic noise functions. Each action is defined as a weighted sum of a fixed number of time–varying noise functions, to form the motion signals for each character DOF. This allows sequenced actions to be automatically aligned in time, and transitions involve

simple interpolation of the two actions as well as their respective tempos.

Yamane et al. [122] utilize the simple user input of starting and ending object positions to synthesize an animation of a character manipulating the object. A randomized path planner is used to find a path for the object which avoid obstacles, and each point along the path is used as input to an inverse kinematics solver that matching frames from a motion database to resolve ambiguity in character pose. Finally, the manipulation is post–processed to maintain smoothness, and the temporal parameterization of the movement is optimized to resemble typical human arm motion.

Feng et al. [27] also present a system to generate motions which fulfill user goals such as goal positions of a manipulated object. A layered approach utilizes motion blending to create a basis for the final motion, by combining pre-existing motion clips to generate basic locomotion and reaching. Parts of the body in these motions are selectively overridden by more specialized controllers, such as determining correct hand and finger placement, or modifying the upper body and head to maintain gaze tracking on a specified target.

ElKoura and Singh [24] present a technique to synthesize the motion of a hand playing the guitar based on input musical chords. They determine a greedy solution within a small time window (adjustable, to model playing expertise) which satisfies the required finger positions, minimizing a cost function which incorporates measures such as finger reach and preference, and wrist movement. Heuristics transform this solved pose to maintain realistic wrist extension and avoid finger collisions, and the results are interpolated with a matching hand pose from a database, to model the natural sympathetic movement of unoccupied fingers.

Finally, incomplete or low–dimensional user input can form the basis of kinematic synthesis in real–time settings as well. Thorne et al. [110] allow the user to sketch a motion path in real–time, recognizing different path "gestures" which indicate the type of motion to be used for path traversal. The path is not applied to the character directly, but several spatial and temporal features of the path are extracted, and used to generate the final motion from a parameterized model of the recognized motion type.

Barnes et al. [9] use a real–time video tracking system to allow a user to perform parts of an animation by manipulating paper cutouts. While the recorded motion for the character root is recorded directly by the system, the animation of further parts of the skeleton can be performed

separately in a layered fashion, where only their rotation is applied about a pre–determined anchor point (i.e., a joint) on their parent bone to generate the final motion.

Chai and Hodgins [13] present a method to allow incomplete input from only part of a user's full body motions to control a complete character in real time. At run time, their system finds samples from a motion database which resemble the user's current pose and the previous synthesized poses. These samples are used to determine a local linear model used to reconstruct the user's full pose, subject to a smoothness constraint.

Seol et al. [94] also control characters in real-time based on a user's full-body performance. The motion of non–human characters is driven by sparse motion capture of a user's full body in two ways. First, user motion is directly applied to parts of the character by mapping functions learned during a mimicked performance. Second, more complex motions are applied by classifying the user's motion and selecting and modifying a corresponding pre–existing motion (a "coupled motion") for the character. The final character motion is a combination of the directly mapped and coupled motions.

Dynamic

Dynamic motion synthesis uses physical simulation to create motion, often involving controllers which determine simulated torques to propel and pose a simulated character.

A natural application of dynamic synthesis is to automatically generate motion through simulation for characters which would be difficult to animate traditionally. Sims [102] presents an approach which evolves both the control system ("brain") and morphology ("body") of virtual creatures, selected over multiple generations to perform at motion tasks. Each brain is represented by a graph of connected behaviour rules relating virtual sensors and motors, and characters performing best after short simulations are mutated to spawn subsequent generations.

Wampler and Popović [116] also address the virtual locomotion of characters with arbitrary morphology, but focus on generating stable gaits for characters with fixed skeletons. The authors combine optimization techniques, one to generate a stable gait without initialization but requiring a specification of cyclic foot contact timings, alternating with the other, which determines the timings themselves. Terms may also be included which allow the optimization to modify bone lengths and masses for better performance at the locomotion tasks.

Nunes et al. [82] also generate stable locomotion controllers by optimizing the dynamic parameters of user–selected modal vibrations of the character. A rough "sketch" of the desired motion is created by combining multiple modes, which then constrain an optimization to determine a stable controller which resembles the initial sketch but is physically valid.

Similar optimization approaches are often used to generate simulated human walking, which is traditionally difficult to generate robustly and realistically. Laszlo et al. [59] address this problem by generating "limit cycles", circuitous paths through character state–space. They use a cycle of states, each with an associated pose and timing difference, and perform short simulations at each timestep to determine a simple model of how control perturbations affect the current pose, which are then used to estimate the control parameters to achieve the next pose in the cycle.

Yin et al. [124] present "SIMBICON" (Simple Biped Locomotion Control), which combines a number of controls to generate a robustly walking character. They use a simple and symmetric state machine to model the different phases of walking, with each phase having a target pose and duration, and unanticipated ground/foot contacts triggering automatic state transitions. Separate controllers for absolute torso orientation and torso–relative swing–leg position combine with a simple balance feedback law to continually drive the character forward, robust to distubances.

Wang et al. [118] present a method to optimize a modified SIMBICON [124] controller for varying character types. Their objective function includes a number of terms which encode intuitive locomotion features, such as left–right symmetric timing, as well as biomechanically motivated terms such as torque minimization and passive arm and leg swinging. The optimized controllers generate motions which exhibit a number of features observed in natural human walking, and are robust to interactive perturbations.

Coros et al. [17] also develop a real–time walking controller, but specifically geared toward traversing variable gaps during locomotion. They begin with an offline synthesis stage in which they generate a variety of control inputs and record the resulting steps, and then analyze the results and inputs to build a low–dimensional model of how varying control inputs affects the steps in different situations. At runtime, a planner examines the upcoming gaps to be traversed and generates the appropriate control using the low–dimensional model.

de Lasa et al. [21] present a framework where multiple high–level objectives can be combined to form controllers for different types of motion. A variety of features within a motion, such as a character's center of mass, can be controlled by simple objective functions which are solved in a priority order, with the solutions of successive objectives determined in the null space of previous ones. Without any pre-computation, small sets of these prioritized objectives can be combined to form controllers for real–time motions such as walking and jumping, for a variety of character types.

Ha and Liu [34] also present a system for creating motion controllers using high-level objectives, however, their approach is based on the methodologies that humans use to learn motor skills. This is accomplished by introducing intermediate "control rigs" which take higher-level pose and motion objectives as input, and control the corresponding lower-level degrees of freedom in the character. These objectives are then used as the controller "practices", in which a novel optimization scheme considers not only successful iterations, but unsuccessful attempts at the goal as well, in order to refine the new control variables more efficiently.

Dynamic synthesis can also be used to generate animations from input that is traditionally processed kinematically, such as Allen et al.'s [4] method to interpolate user–specified keyframes. Their solution uses traditional Proportional Derivative (PD) control to determine joint torques, but automatically and continuously adjusts the controller parameters to match the keyframes, avoiding the tedious task of hand–tuning parameters. A two–pass algorithm computes per–frame inertia and torque for each bone in a hierarchy–aware fashion, after which the controller parameters can be determined analytically.

Kokkevis et al. [51] also dynamically synthesize motion to satisfy kinematic constraints, but address full body motion synthesis constrained by timed trajectories of some skeleton parts. They use a single–pass adaptive control scheme to generate motion which follows the given trajectories, but also incorporate a scheme to recover from unexpected collisions when following an impossible trajectory. Finally, they also simulate the passive motion of "secondary" degrees of freedom affected by the primary motion.

Naturally, controlling motions which involve significant dynamic components is a problem well–suited to these types of approaches. Hodgins et al. [38] generate full human animations for running, cycling and vaulting by presenting specifically constructed control laws for each

task. State machines and parameterized, pre–specified and highly tuned behaviours for each task generate motion to satisfy high–level user constraints on velocity and facing direction.

Coros et al. [18] present a system for controlling simulated quadrupeds by combining a number of specialized control components, such as a graph to provide the stance/flight rhythm for each leg in a variety of gaits, and a flexible spine which joins the front and rear leg pairs. The control parameters are optimized to generate gaits which are stable, natural, and similar to reference motion. Interactive controllers are developed for a number of motions, including gaits such as trotting and galloping, leaping, and sitting.

Faloutsos et al. [25] present a framework to compose multiple black–box controllers, each of which control a human character for a particular physical task. Each controller can have manually–specified or automatically–determined "preconditions" which dictate when they are useful, and controllers can detect when they have succeeded or failed, so that the "supervisor controller" can determine the next controller to transition to. Some transitions occur often while others are rare, which results in an emergent motion graph–like organization of the controllers.

Dynamic synthesis can also be used to augment low–dimensional user input to control virtual characters in real time. Laszlo et al. [60] use either continuous input from a mouse or discrete keypresses to control and trigger character actions to traverse a virtual environment. Simple mappings of input to character degrees of freedom and tuned PD controllers, combined with a simple method to switch control between DOFs (e.g., alternating legs during a walk) allow easy control of a variety of characters.

Zhao and van de Panne [126] use a similar method to control characters who launch themselves into aerial maneuvers such as diving or snowboarding. They present a number of two–dimensionally parameterized actions in an "action palette", each of which determines PD control at specific times, and allow the keyframe–like arrangement and refinement of actions with specific timings. A real-time input scheme is also presented, augmented by a simple automatic balance controller.

Finally, dynamic synthesis can also take place when accurately simulating specific parts of the body. Tsang et al. [114] present a detailed model of the human hand and forearm, incorporating skeleton, muscles, and tendons. Accurate muscle simulations can determine hand movement through forward simulation based on timed muscle activations. They also present

the inverse process, determining the muscle activations to recreate input hand movements.

DiLorenzo et al. [22] also present a hybrid simulation of rigid and flexible components, but simulate the various torso components involved in laughter (diaphragm, ribs, etc.). Based on biomechanical research, they develop a model to correspond audio laughter with lung pressure. By adding a pressure model to their simulation, they can optimize their anatomical controls to match an input laughter recording.

Combined

Combined synthesis utilizes both the physical simulation of dynamic synthesis, as well as kinematic techniques which may include transformation, blending, or synthesis. The dynamic and kinematic operations may be applied sequentially, such as using a kinematically synthesized motion as input to a simulation, or simultaneously, such as optimizing an error function that includes both kinematic and dynamic terms.

An obvious application of combined synthesis is to achieve the "best of both worlds" in generating animation to satisfy a user's kinematic constraints, combining the realism of dynamics with the accuracy of kinematic techniques. Rose et al. [92] consider the problem of generating transitions between motions, and develop a spacetime optimization scheme that includes both an inverse kinematics solution for ground–contacting "support limbs" and torque minimization terms for the remainder of the character's body.

Liu and Popović [69] also present a combined approach to satisfying kinematic constraints, but use a coarse keyframed animation as a "suggestion" for their synthesized animation. They automatically detect environmentally constrained and unconstrained stages of the input motion, and optimize their output motion using a spacetime error formulation which incorporates pre–specified poses for transitioning between stages, constraint maintenance, and momentum conservation.

Fang and Pollard [26] use coarse kinematic input to generate a physical motion. They present novel constraint formulations of a variety of physical factors, such as momentum conservation and ground contact, which allows efficient evaluation of motion characteristics used for optimization. They then use a kinematic optimization procedure to generate smooth motion, subject to the physical constraints detected from the input, expressed in their efficient

formulation.

Tourner and Reveret [112] determine a kinematic parameterization of input motion and which is then driven by reformulated equations of motion within the parameterized space. Principal geodesic analysis, a rotation-friendly analogue of principal component analysis, is used to reduce the dimensionality of the input data. Configurations in this reduced model are treated as generalized coordinates in a novel integrator which incorporates Lagrangian dynamics, generating simulated motions which are similar to the input motions, while matching user-specified position constraints.

Instead of specific keyframes, higher–level kinematic or behavioural control can be used as input in combined synthesis. Neff and Fiume [81] allow the rapid exploration and refinement of character gesture performance through a variety of user–specified motion properties. These properties range from partial or complete pose specification to rhythm or timing parameters, and a higher–level "character sketch" can modify property sequences with global performance modifications. The resulting motion can be synthesized either kinematically or dynamically.

Neff et al. [81] also allow the kinematic or dynamic synthesis of a motion sequence to match the gesture characters of a real person during speech. A model is built from video performance annotated with specific gesture types and timing, as well the corresponding dialogue. Input text is then used to generate a stream of corresponding gestures, which are kinematically enhanced using observed motion properties such as the circularity or linearity of particular gestures, and the final sequence can be interpolated kinematically or used as input to a simulation.

Combined approaches can also be used to generate motion which responds accurately but realistically to interactive user input. Zordan et al. [128] uses simulation, interpolation and sequencing to generate novel character reactions to impact. A passive "ragdoll" simulation provides an initial guess to the reaction, which is used to select a farther–along motion clip from a database for the end of the reaction. A guided simulation is then used to transition between the pre– and post–impact animations.

Ishigaki et al. [47] present a method to allow a user's full–body motion to control a character in a virtual environment. Because the user cannot fully perform some motions due to differences in their environment and the character's, the system recognizes user "intention" and generates environment–appropriate character motions by combining motions from a database. A simple

optimization procedure generates output motion that obeys physical constraints but also retains similarity to the combined target motion.

Yang et al. [123] present a method to interactively control the goal position of a swimming character. They determine a target pose by interpolating a hand–animated sequence of poses of a stroke sequence, which is kinematically modified in a number of parameterized layers to more specifically control rotation and forward displacement. This target pose is then used as input to a simulated character in a simplified fluid simulation, and the target pose is continually updated as the character progresses toward the target.

Abe and Popović [2] use kinematic data to drive the simulation of a character manipulating virtual objects. They formulate manipulations in pose–relative "task–space" and optimize the torques of the simulated character to generate the desired task–space displacements. A small set of example motions allows the identification of a task–space–similar guide motion, which is also used as input to be imitated during the optimization.

Wampler et al. [117] use a combination of example motion and dynamics to generate a gait for a given animal skeleton. First, video data of bipedal and quadrupedal animals was tracked, and a spacetime constraints optimization is used to reconstruct the corresponding animal skeleton and motion. Then, using a generative model for producing new motions given an extensive set of physical parameters, an inverse optimization problem is solved jointly across all of the example motions to produce an interpolating set of parameters suitable for the new animal skeleton.

Finally, autonomous character interactions are also possible with combined synthesis approaches. Zordan and Hodgins [127] use on–line kinematic techniques to generate a stream of motion that controls a physically simulated character. They use a simple motion graph–like structure to model the states of characters who hit and react, transforming individual motions to meet spatial targets, and transitioning between states by interpolating the transformed motions. The resulting animation is used as a target by the simulated character, with the gains of each simulated joint adjusted in real–time to allow for realistic reaction and recovery when the character is hit, while an additional controller adds additional torques to maintain balance.

2.2 User-Specified Parameters

2.2.1 None or Abstract

Very few animation techniques operate with no user input whatsoever; a few notable exceptions involve fully autonomous characters who can retain stability while moving continually [116,124], or can physically interact with each other [127]. However, there are many approaches which operate automatically once a user has selected one or more input animations, and such coarse control represents an extremely limited and potentially unintuitive method of controlling the resulting animations.

Some techniques automatically modify a selected single input motion to meet kinematic criteria such as footskate correction [42, 55] or animation principles [119], or dynamic criteria to ensure physical realism [97]. Single selected animations can also be used to automatically generate variations [61], or have their motion retargeted to a selected new skeleton [30]. The motion of multiple example characters can also be generalized to create motion on a new character [117].

Multiple user–selected animations can also be processed automatically, allowing techniques to synchronize or correspond the motions to allow interpolation [75] or timewarping [39]. Animations individually selected for particular roles can be automatically spliced [36, 44, 70] or interpolated between [43, 92].

A system that processes multiple animations can also present the user with discrete, abstract parameters that don't correspond directly to either spatial or temporal features of the generated motion. A common method of abstract control is to allow the user to apply different learned "styles" (which can express character intent or emotion) to modify an existing animation [11, 40, 95]. Another type of abstract control allows the development of autonomous characters who succeed at selected tasks [102]. Vibrational modes of a character's motion can be selected to provide rough guidelines for simulated motions [82]. Potential motions can also be accepted or rejected, to train a policy for determining the best motion transitions [90].

Most abstract control is discrete, involving a selection from a finite set of options. However, it is possible to present the user with abstract continuous control, such as adjusting the frequency band gains of motions [12]. In this case, the effect of the user–specified parameters is

mathematically direct, but unintuitive in its effect on the spatial and temporal features of the resulting animation, and as such is suitable only for exploration of a space of motions rather than producing a desired result.

2.2.2 Single Spacetime Value

Often it is sufficient for a user to directly control a single spatial or temporal feature of the generated animation. A straightforward way to provide solely spatial input to an algorithm is to manipulate a part of the character skeleton directly. This can involve directly manipulating or "dragging" a part of the character to an arbitrary position [29,33] or within a pre–determined manifold [53, 99]. Specific points of character/environment interaction such as footplants can also be repositioned [88, 107], as can spatial representations of physical parameters like a character's center of mass [71,88]. Image–space (and thus ambiguous) specifications of character pose [19, 33] are also possible.

Factors determining the overall motion of the character but external to their skeleton can also be modified, such as goal positions for manipulations [16,27,91,93] or locomotion [59]. Some approaches allow higher–level control of how the character moves, such as character direction or velocity [38, 128], or turning rate [91].

Momentum over a character's entire motion can also be scaled to allow indirect control of the distance and rotation of motions [103].

Finally, solely temporal input can be used to affect the temporal parameterization of an animation. Selected poses can be re–assigned to different times [15], or single time–based features such as speed and succession of connected limbs can be adjusted [15, 79].

2.2.3 Discrete Spacetime Set

While some degree of control is possible by specifying a single spacetime feature, some approaches allow multiple values associated with spatial and/or temporal positions to be satisfied simultaneously. By far, the most common way this is accomplished is through user–specified keyframes: a sequence of poses, each with an associated absolute time.

A user can specify keyframes which must be precisely satisfied by an algorithm, which can either modify a pre–existing animation [64, 121], or generate a novel motion through physical

simulation [4] or motion sequencing [106].

Keyframes can also act as partial constraints, or starting points, for a generated animation. Sparse keyframes can supply overall body movement, while the fine motions of extremities can be added by simulation [26, 69] or matching from a motion capture database [89]. Two–dimensional drawn keyframes [48], which are inherently ambiguous, can partially constrain the generated three–dimensional poses as well.

Spatial data beyond just a character's pose can also be specified with absolute times. Multiple goal positions can be specified in time and space for a character's overall motion path [31, 50], footplants [62], or hand positions [96]. Motion signals can also be modified and constrained directly, to control specific degrees of freedom in a character [51, 86].

Precisely timed sets of data need not have direct spatial effects. A common approach is to allow the user to supply a sequence or "script" of timed high–level actions, such as full–body motions like running or jumping [6, 25, 87], and the system determines the best way for a character to perform the actions. Other approaches with timed sequences of data include annotated text to determine character gestures [81], muscle activations to drive a simulated limb [114], or even music to determine fingerings of a guitar–playing hand [24]. Alternately, sets can be unordered and affect a variety of temporal factors over an entire animation, such as gesture follow–through [80].

Timing information in a set of parameters can be relative instead of absolute, by specifying only an ordering without durations. An animation of automatically–determined length can be generated by specifying only start and end poses or positions [67, 122], an ordered set of any number of two–dimensional pose sketches [68], or a sequence of pose and motion goals [34]. Control of spatial data external to the character is also possible by manipulating a linear sequence of obstacles to be traversed in order, such as gaps [17] or steps [77].

Finally, discrete sets of spacetime parameters can consist solely of spatial or temporal parameters. Some techniques allow the user to modify the duration, speed, or other temporal aspects of multiple pre–existing actions [49, 78]. Untimed spatial data can include a two–dimensional environment grid [65], arbitrarily–arranged spatial positions associated with user–defined poses [41], or changes to skeleton bone size [37]. Motions can also be controlled by a set of both spatial and temporal parameters, such as the height, distance, and velocity of a

jump [18].

2.2.4 Continuous Spacetime

The most complex type of user–specified parameters occurs continuously over time; namely, they involve some form of "real–time" or "performance" input from the user. These can take a variety of forms, and be utilized by the animation algorithm in a number of ways.

Some types of input consist of continuous spatial values without a strict temporal parameterization, in the form of a path through space (i.e., a space curve). Since some of the earliest pioneering animation systems [8], the sampled positions and timing of a user's sketched path have been used to control animated performances. The timing of the user input can also be ignored, allowing an algorithm to determine how long the resulting animation should take to follow the path through space. Paths can determine the progression of a character's general position along the ground plane [54, 93], the specific position of a part of a character [76], or the motion of a group [57]. Sketched curves can also be used to determine a character's pose without any timing [83].

Many methods do utilize the timing of a user performance; some, however, are delayed in their feedback, providing their results once the user has completed the entire input. These "high latency" user performances can include a timed sequence of preset character actions for which other interacting characters are generated [101], drawn paths with their performed timing affecting the character [110], a mimicked motion to determine modifications to a character's motion through space [23] or time [108], or even an audio recording to control a character's laughter [22].

While such delayed methods of continuous input allow user *iteration* to refine an animation, "on–line" algorithms which provide results continuously during input allow true user *interaction*, where the results of the user's input are immediately apparent. A common method of interactive control is "videogame–style" input through a low–dimensional input device to control overall character direction [1], which can be augmented with specified speed [52, 74] or motion type [32,35,56,113]. Particular degrees of freedom in a character can also be controlled directly with such low–dimensional devices [60, 126], or by manipulating a camera–tracked proxy object [9].

Such low–dimensional interactive input can also control factors external to the character.

Interactively controlled goal positions can be tracked by a character's limbs or center of mass [21, 112], or used as targets for walking [84] or swimming [123] motions. Physical interactions with the character can also be controlled, by moving an object the character is holding [2], or by applying arbitrary forces to a moving character [7, 118].

Finally, another possibility for on-line input is literal user performance, by passively measuring a user's physical actions. Users' full body movements can be used to "suggest" control of a character's motion which may be difficult to perform exactly, such as expert sport maneuvers [13], interactions with a virtual environment [47, 63], or the motion of non–human characters [94]. The "intent" of a user performance can also be retargeted to a differently–sized character [98]. Specific parts of a performance can also be used to determine the complete animation of a character, such as a user's footsteps [125]. Vocal performances can also interactively control a character's gestures [66] or locomotion [120].

2.3 This Thesis in Context

While the categorization and coverage of related work presented in this chapter is useful for understanding the field of motion generation research in general, it also enables a discussion of how the work in this thesis relates to the field. As an initial means of comparison to related work, the location of the work in this thesis along both of the categorization axes can be used to find similar approaches. The path-based editing algorithm presented in Chapter 3 uses a sparse set of positional handles to modify both the path of a single motion and its timing; thus, its animation operation is *combined transformation*, and its user-specified parameters are a *discrete spacetime set*. The performance techniques presented in Chapters 4 and 5 use the path-based editing technique to edit a single motion, but control the editing handles automatically based on real-time user performance; thus, their animation operations are also *combined transformation*, and their user-specified parameters are *continuous spacetime*.

The two relevant cells of Table 2.1 show that while these combinations of operation and user-specified parameters are not novel, they are shared with very few other works. The work of Jain et. al [48] uses hand-drawn and annotated keyframes to timewarp and then adjust the poses within a clip of motion capture data, which broadly resembles the approach of our

path-based editing algorithm. However, this work is in some sense orthogonal to ours, because in that work, the user's control is focused on fine pose adjustments, rather than the coarser but more impactful positional control afforded by our path-based editing. Our performance techniques are also similar to the work of Yin and Pai [125], which uses the performance of a user's footsteps to select and modify both the poses and timing of an input motion. Again, however, this approach is complementary to ours in that its effect on the input motion is focused on pose adjustments, and its application on locomotive motions is not demonstrated.

The most related work with this combination of operation and user-specified parameters is the path-based editing technique of Kim et. al [50]. Our path-based editing algorithm presented in Chapter 3 is inspired by that work, and uses the same basic approach of modifying a motion path using an as-rigid-as-possible deformation. There are some significant differences, however. Our algorithm generalizes that work's path deformation by correcting artifacts caused by uneven path sampling and large deformations (Section 3.2.2), expanding the deformation to three dimensions by appropriately deforming vertical motion (Section 3.3.1), and by applying a different type of deformation to ballistic portions of a motion (Section 3.3.2). The work of Kim et. al also allows the user to manually specify timewarping throughout a motion arbitrarily; our approach uses a variety of criteria to automatically timewarp the motion to maintain plausibility (Section 3.2.4). Beyond the specific spatial and timing deformation algorithms, the works differ significantly their applications: our work modifies the motion of a single character, while one of the main contributions of the work of Kim et. al is that it detects and preserves the spatial and timing relationships between multiple interacting characters.

Aside from the previous work which shares the categorization with our work, there is also a larger trend among this related work which is not adequately captured by our categorization axes. In those works which produce motion that varies in its timing, the user's control of that timing is generally one of two extremes: the user either has no control over the motion timing, or the timing of the user's real-time input controls the output motion's timing absolutely. Usage of absolute timing control is potentially problematic because of the precision required in the user's performance; however, the other extreme solution of not allowing any control over motion timing can be too restrictive.

Very few works differ from this trend, and those that do are techniques which focus

exclusively on temporal transformation. Hsu et. al [39] produce a continuous and continually-varying timewarp of an input motion based on the timing of a second "guide" motion, but this timewarp is not constrained to the absolute timing of the guide motion, as it relies on smoothly maintaining similar velocity and acceleration between corresponding keyframes in the two motions. Terra and Metoyer [108] also present a technique to generate a timewarp based on correspondences, where similarities between the motion path of an object and a user's sketched path are used to transfer timing aspects of the user's input to the object's motion.

Similar in spirit to those two works, and as discussed in Section 1.3, the techniques in this thesis aim to provide accessible high-level control of animation editing by avoiding the restrictive requirement of absolute timing. This is accomplished with two alternative methods of controlling the new motion's timing. First, as presented in our path-based editing algorithm in Chapter 3, a user's *spatial input* is used to determine the timing of the new motion. Since the spatial and timing parameters of real-world motion are coupled by the laws of motion, our algorithm attempts to preserve that relationship so that, for example, modifying a jump to be higher also results in the jump taking more time. Second, as presented in our performance techniques in Chapters 4 and 5, the *relative timing* of a user's input can also be used to control a new motion. By utilizing aspects of the user's input such as the frequency of contacts or the similarities between two successive inputs, a user's internally consistent timing can be accommodated without assuming that the timing of their input resembles real-world motion in the absolute sense.

Chapter 3

Biomechanically-Inspired Motion Path Editing

This chapter presents a method for generating novel motions through interactive editing of a single initial motion. This is accomplished through an analytic approach which incorporates simple physics approximations and well-established biomechanical relationships between motion extent and timing. Since this approach is not data–driven by a large set of example motions or the result of complex simulation or optimization, it enables interactive editing of a single motion without significant precomputation. Motions are edited by interactive manipulation of a sparse but meaningful set of controls, or handles, which are automatically identified along the character's motion path. This method is generic and applicable to many different types of locomotive motions, as well as motions which contain a variety of different actions.

Biomechanics Inspiration

The scientific field of biomechanics is extremely relevant to work relating to the creation or editing of animation. Biomechanics can be broadly defined as the application of mechanical principles to the study of the structure and function of biological systems. As much work within biomechanics concerns understanding the systems contributing to body motion, as well as trends or behaviours during motion, there are a wide variety of applications of biomechanics research to animation, depending on the both the approach and goals of an animation system.

Much of the work within biomechanics which is related to motion concerns the forces applied

by muscles, and their interaction with tendons, ligaments, and bones in order to effect motion in particular body parts, or at particular joints. As discussed in Section 2.1.4, there are a variety of animation techniques which utilize physical simulation, varying in complexity from simple masses and joint motors to detailed muscle and tissue simulation. Especially for those more detailed simulations, the work within biomechanics which focuses on the lower-level mechanical aspects of bodies in motion is very relevant. For our purposes, however, this type of work is less useful, as our editing techniques may not have a detailed model of a character's physiology provided as input, and the computation time for such simulations run counter to our goal of generating results interactively.

However, there is also work within biomechanics which aggregates the action of individual muscles and joints by examining higher-level aspects of body motion, such as overall behaviour, motion strategies, and the timing and extent of events during motion which are more semantic than strictly mechanical, such as a footplant. It is this work which we are inspired by, which identifies trends in locomotive behaviours and often provides simple but powerful kinematic models for describing specific aspects of motion, without incorporating dynamics, but based instead on empirical observation and testing.

Our technique is inspired by a few particular phenomena studied in biomechanics. Our path-based editing technique restricts how the user can manipulate the path shape, similar to how studies have shown that turns can only be introduced at particular points during a gait (Section 3.2.1). The timing of the animation along the path is also automatically modified based on observed relationships between velocity and stride length (Section 3.2.4), as well as between velocity and path curvature (Section 3.3.3).

3.1 Overview

This method uses a modified as-rigid-as-possible deformation technique to modify motion paths. Instead of allowing arbitrary constraints and manipulations of the path, which can result in unrealistic motions, user manipulation of the path is restricted to a sparse set of handle points along the path. These handles are automatically detected by identifying geometric features along the path which correspond to changes in the character's ground contact, which ensures

that any significant path changes occur when the character has leverage to make such changes. Once a user has manipulated the handle positions to determine a new path, all poses of the motion are modified accordingly, and an automatic timewarping step modifies the timing along the path to preserve the relationship between the stride length and velocity of the original motion.

In addition, all stages of this method are automatically augmented and restricted by a set of biomechanically and physically-inspired heuristics. Horizontal and vertical motion editing are decoupled, to better reflect biomechanical motion rather than orientation-invariant geometric relationships. Segments of the motion path without environmental contact are identified, and the spatial deformation and timewarping of these segments are restricted to preserve the shape and timing of ballistic motion. Finally, turning also automatically affects motion timing by using a well-established relationship between path curvature and velocity.

This system can be used to quickly and easily modify a variety of locomotive motions, and can accurately reproduce recorded motions that were not used during the editing process.

This work makes a number of contributions. An improved as-rigid-as-possible deformation algorithm corrects the artifacts introduced when applying the original algorithm to the editing of motion paths. A general kinematic path editing framework identifies useful manipulation handles based on motion features, and automatically timewarps deformed motion based on the original motion's timing. Finally, a number of motion components based on biomechanical observations and simple physics are presented, which augment and restrict both the spatial and temporal editing procedures to preserve natural motion.

3.2 Basic Motion Editing Procedure

This method operates on the animation of a single character skeleton, in the form of a standard hierarchy of joint transformations. Character motion is represented as a discrete sequence of *keys*, which describe the rotation of every skeleton joint as well as the translation of the root joint at an associated time; the sequence of 3D root joint positions defines a character's *motion path*. This discrete representation, rather than the the motion curve information present in modern animation software, allows for the editing of sampled animation from physical simulation or

motion capture. Sparsely keyframed animation is also permitted, as keys are not required to be uniformly arranged in time.

The method operates on an animation and motion path in four stages. In the first stage, which occurs only when an animation is initially loaded, the animation is processed to detect contact constraints and determine path manipulation handles. In the second stage, the user manipulates the handles to deform the path into a new shape. In the third stage, the character's poses are automatically adjusted at every key while conforming to the user's path modification. Finally, in the fourth stage, an automatic timewarp adjusts the timing of the animation.

3.2.1 Motion Preprocessing

The goal of the preprocessing stage is to identify points along the motion path to serve as user-manipulable *handles* which constrain the deformed path. While it is possible to allow user-selected handles [50], this can easily generate unrealistic motions; for example, such motions could violate the observation that humans are incapable of turning suddenly during a step and instead require appropriate preparation during the step preceding a turn [85]. Therefore, the goal of this stage is to automatically determine handles which only enable natural-looking changes in the motion path. To avoid confusing the user, handles should not be added or removed during interaction. Having a constant set of handles implies that handles should occur not only where a motion changes, but where a motion *could* change.

Handles are determined according to the hypothesis that the most meaningful path changes during locomotion can occur at times corresponding to vertical minima of a character's root position. As this method applies to characters who propel themselves by exerting force against a static environment in the presence of gravity, vertical minima naturally correspond to the inflection point at which a character stops traveling downwards with gravity and is actively opposing it – indicating an exertion which could, but does not always, change the path significantly. However, many local minima can occur during a motion due to high-frequency motion characteristics or motion capture noise, not all of which correspond to potential motion changes (Figure 3.1, top).

To address this, handles are identified from the minima during specific periods of the motion identified from the character's environmental contact. Using the technique of Coleman et

al. [15], contacts are detected for pre-defined parts of the character skeleton ("end effectors"). Based on the timing of these contacts, time periods of a locally maximum number of contacts are identified. These periods occur in any type of locomotive motion; for example, the double-stance periods of a human walk which occur in-between single contact periods of a stance foot, or the single-contact periods of a run which occur in-between airborne periods. Within each period of maximum contact, the vertically minimum path point is retained as a handle. This results in one handle for each step of periodic motions, in addition to handles enforced at the first and last path points (Figure 3.1).

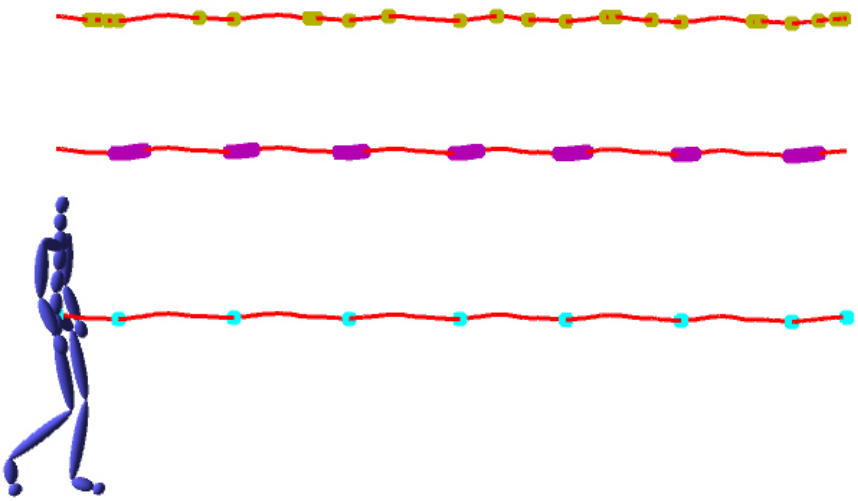

Figure 3.1: User-manipulated handles (bottom) are detected by finding local vertical minima along the path (top) during each period of locally maximal contact (middle).

3.2.2 Path Manipulation and Deformation

The interactive path manipulation within this method is designed to fulfill the goals of a direct manipulation system [100]. Chief among these in a motion editing context is that operations should be easily incremental and reversible, which is accomplished by determining each deformation relative to the starting configuration, rather than the previous deformation result.

Path manipulation is accomplished by interactively manipulating the three-dimensional positions of only the handle points identified by the precomputation stage. The deformation algorithm takes as input the original motion path as well as the new handle positions, and

generates as output a full new motion path which satisfies the specified handle positions while retaining a path shape similar to the original.

This path deformation algorithm utilizes a modified version of the as-rigid-as-possible deformation algorithm of Igarashi et al. [41], which was previously used with slight modification by Kim et al. [50] for motion path editing in two dimensions. In the algorithm's first stage, a "scale-free deformation" of the initial points $\mathbf{p} = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is determined by defining a relative position (u_i, v_i) for each point $\mathbf{p}_i$ in a local coordinate system determined by the neighbouring points:

$$\mathbf{p}_i = \mathbf{p}_{i-1} + u_i(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) + v_i R_{90}(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) \tag{3.1}$$

where R_{90} is a 90-degree rotation matrix:

$$R_{90} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{3.2}$$

The "desired" position of each point in the scale-free path $\bar{\mathbf{p}} = \{\bar{\mathbf{p}}_0, \bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_n\}$ is the reconstructed position from these local coordinates:

$$\bar{\mathbf{p}}_i^{desired} = \bar{\mathbf{p}}_{i-1} + u_i(\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_{i-1}) + v_i R_{90}(\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_{i-1}) \tag{3.3}$$

The scale-free path $\bar{\mathbf{p}}$ minimizes the weighted sum of squared distances between the desired and final positions of each point:

$$\underset{\bar{\mathbf{p}}}{\operatorname{argmin}} \sum \bar{w}_i \|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_i^{desired}\|^2 \tag{3.4}$$

where $\bar{w}_i$ weights the per-vertex error by inverse edge length of the original path:

$$\bar{w}_i = \frac{1}{\|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\|} \tag{3.5}$$

and the handle positions are enforced as as hard linear constraints on the scale-free path $\bar{\mathbf{p}}$:

$$\mathbf{H}\bar{\mathbf{p}} = \mathbf{h} \tag{3.6}$$

The second stage of the algorithm generates the final "scale-adjusted" path $\hat{\mathbf{p}} = \{\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_n\}$ based on $\bar{\mathbf{p}}$. Let $\bar{\mathbf{e}}_i$ be the edge between two adjacent scale-free points $\bar{\mathbf{p}}_{i+1}$ and $\bar{\mathbf{p}}_i$ scaled to the length of the corresponding original edge:

$$\bar{\mathbf{e}}_i = (\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_i)\frac{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{\|\bar{\mathbf{p}}_{i+1} - \bar{\mathbf{p}}_i\|} \tag{3.7}$$

Then $\hat{\mathbf{p}}$ minimizes the weighted sum of squared distances between the final and scaled edges:

$$\underset{\hat{\mathbf{p}}}{\operatorname{argmin}} \sum \hat{w}_i \|\hat{\mathbf{e}}_i - \bar{\mathbf{e}}_i\|^2 \tag{3.8}$$

where $\hat{\mathbf{e}}_i$ is the edge between two final vertex positions:

$$\hat{\mathbf{e}}_i = \hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_i \tag{3.9}$$

$\hat{w}_i$ weights the per-vertex error by inverse edge length of the scale-free path:

$$\hat{w}_i = \frac{1}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|} \tag{3.10}$$

and $\hat{\mathbf{p}}$ is subject to the same hard handle position constraints of Equation 3.6, in the form of $\mathbf{H}\hat{\mathbf{p}} = \mathbf{h}$.

The linear systems for both stages, expressed in Equations 3.4 and 3.8), are sparse and can be efficiently computed using a sparse LU solver [20]. Both systems only require factorization once for a particular path and set of handles, allowing the final path to be computed very quickly as handle positions change during interactive manipulation.

This algorithm has been modified in two ways. The original deformation algorithm operates on a triangulated mesh, but artificial triangle edges can cause kinks during deformation (Figure 3.2b). While Kim et al. [50] modified the scale adjustment term to not require additional edges, the explicit edge difference formulation of this method (Equation 3.8) is simpler. Both stages

of the deformation also incorporate inverse edge length weighting ($\bar{w}_i$ and $\hat{w}_i$) to compensate for uneven path sampling. In the second stage in particular, weighting by $\hat{w}_i$ has the desirable effect of modifying deformations with two handles exactly into a nonuniform scale in the handle displacement direction (compare Figure 3.2c and Figure 3.2d).

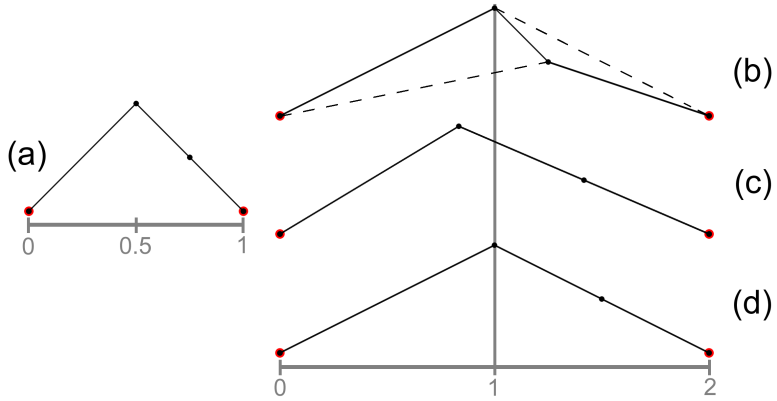

Figure 3.2: An initial curve (a) being deformed under three formulations: with an artificial triangulation (b), uniform edge weighting (c), and inverse edge length weighting (d).

Automatic Scale Factor

A significant disadvantage of this algorithm as utilized by previous work is that as deformations become larger, increasingly large derivative discontinuities are introduced at point constraints, causing a loss of smoothness (Figure 3.3). This behaviour occurs in the original triangulated formation of the deformation algorithm as well, and is not improved by supersampling the initial curve. The discontinuities are due to the scale adjustment error term (Equation 3.8), which results in the path segments between point constraints being "flattened" to more closely match the original edge lengths.

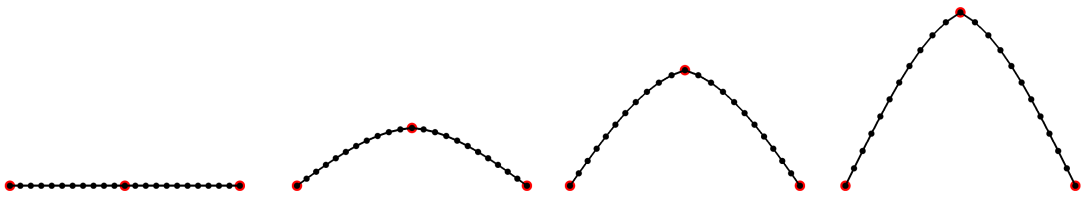

Figure 3.3: The original algorithm introduces increasingly large discontinuities in smoothness with larger deformations.

The intuition behind the solution to these artifacts is that since relatively small deformations

introduce small artifacts and the final handle positions constrain the absolute scale of the deformation, the scale of the deformation relative to the path can still be adjusted by pre-scaling the initial path shape. That is, large deformations of a particular path could be replaced by small deformations of a larger copy of that path. Figure 3.4 shows the same handle positions applied to a shape at scales of 50%, 100%, and 200%, which shows that for compressive deformations, the path shapes are "inflated", the opposite behaviour of the flattening which occurs in expansive deformations.

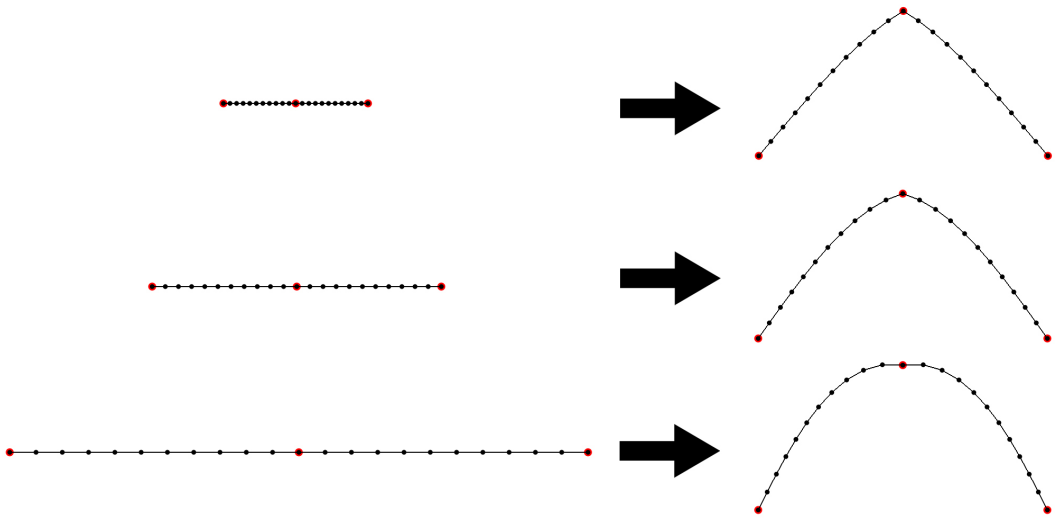

Figure 3.4: Varying the scale of the original shape (left) prior to deformation affects the smoothness of the final shape (right).

To incorporate this pre-scaling, an extra "scale factor" parameter s is introduced which uniformly scales the path prior to deformation. The first stage of the algorithm (Equation 3.4) is scale-independent and is unaffected by any uniform scaling introduced by s. The second stage of the algorithm is modified to penalize deviations from a variable scale rather than the original scale, and is expressed by modifying equation 3.8 to:

$$\hat{\mathbf{p}}\,(s) = \underset{\hat{\mathbf{p}}}{\operatorname{argmin}} \sum \hat{w}_i \|\hat{\mathbf{e}}_i - s \cdot \bar{\mathbf{e}}_i\|^2 \tag{3.11}$$

Because this formulation is quadratic with respect to the scale factor, the solution $\hat{\mathbf{p}}\,(s)$ is

linear with respect to s (Figure 3.5), i.e.,

$$\hat{\mathbf{p}}(s) = \hat{\mathbf{p}}(s) + s\left(\hat{\mathbf{p}}(1) - \hat{\mathbf{p}}(0)\right) \tag{3.12}$$

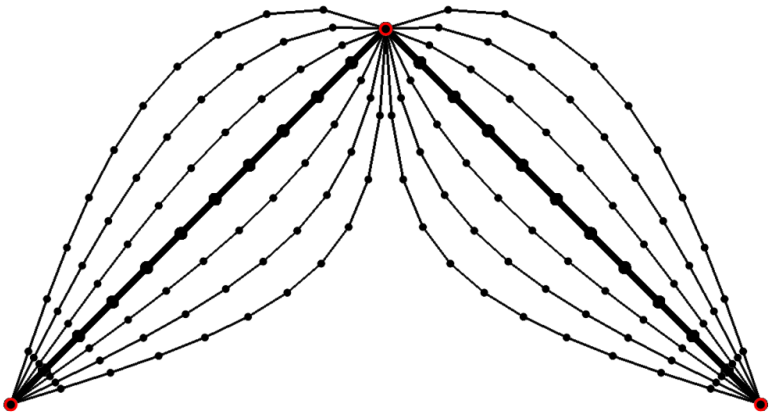

Figure 3.5: A deformation at integer scale factors between -3 and 3, with the zero-scale curve (center) in bold.

The "zero-scale path" of $\hat{\mathbf{p}}(0)$ minimizes the weighted sum of edge lengths of the final path, resulting in a path which linearly interpolates the point constraints (centre path in Figure 3.5) which can be computed in closed form. Therefore, combined with the scale-adjusted path for any pre-determined non-zero s (e.g., $\hat{\mathbf{p}}(0)$), the full linear parameterization of Equation 3.12 can be computed with only a single solution to the linear system in Equation 3.11.

Smoothness discontinuities in the final path are almost always eliminated by choosing the scale factor s which minimizes the absolute difference between the arclength of the final path and of the scaled initial path; i.e., the scale factor which approximately results in only bending and not stretching to deform the scaled initial path into its final shape:

$$\underset{s}{\operatorname{argmin}} \| s \cdot \sum \mathbf{e}_i - \sum \hat{\mathbf{e}}_i \| \tag{3.13}$$

These final scale factors are determined automatically through a line search which generally converges within 2-5 iterations, and each iteration is simple to compute using the linear scale parameterization. Figure 3.6 shows the automatically-determined scale factors for the same series of increasingly large deformations shown in Figure 3.3, but without the smoothness discontinuities.

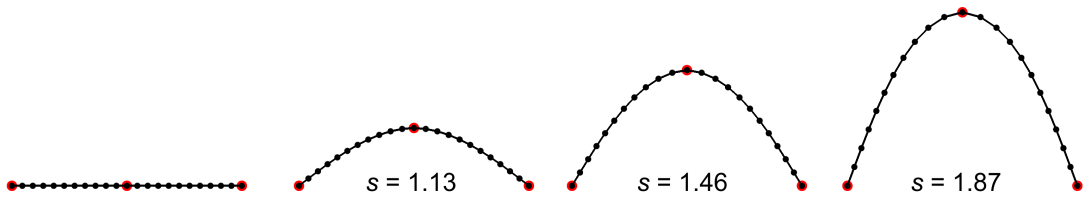

Figure 3.6: Pre-scaling the initial shape corrects smoothness discontinuities during deformation.

However, for asymmetric deformations, a single scale factor can still introduce discontinuities; instead, separate scale factors can be determined independently for each path segment (Figure 3.7).

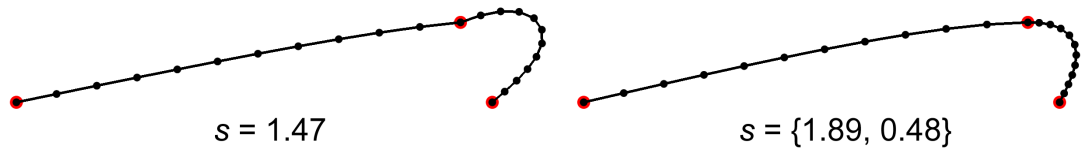

Figure 3.7: A single scale factor results in discontinuities in highly asymmetric deformations (left). Independent scale factors for each segment maintain smoothness (right).

3.2.3 Pose Transformation

The transformation of the root path determines the transformation of all other aspects of the animation. First, the character's rotation at each key must be adjusted to maintain its heading relative to the deformed path. This is determined from the instantaneous horizontal direction at each key of the path, calculated with central differences from the preceding and subsequent key positions. For each key, any rotation of this direction is applied to the character, thus preserving the character's orientation relative to their instantaneous horizontal direction.

Any end effectors of the character which are constrained from environmental contact must also be adjusted to maintain smooth motion without introducing sliding. This is accomplished by applying the same motion path deformation algorithm to the end effector paths, with the contact keys as handles. Gleicher [31] noted that constrained foot positions during ground contact must be rigidly transformed together, and while this transformation should correspond to *some* root key from the contact period, the most appropriate key is unclear. However, based on informal observation of recorded motion data, human footplants appear to be often nearly parallel with the tangent of the root path at its closest point (Figure 3.8). Thus, the

associated key for a contact is determined not temporally but spatially, by choosing the key which minimizes the horizontal distance to the contact.

For any edit, the horizontal transformation of the associated key is applied to the contact, which maintains horizontal foot placement relative to the root but avoids ground interpenetration. The entire contact period of the end effector path is rigidly transformed by the single transformation of the associated key, preserving the intricate motion of the footplant. End effector positions on the path in-between contact periods are automatically determined by the deformation algorithm, using additional handles to maintain relative foot positions where a swing leg passes a footplant, with new rotations determined using the same method as the root. Given these new end effector transformations, the intervening joint angles are determined using the closed–form inverse kinematics solution of Tolani et al. [111].

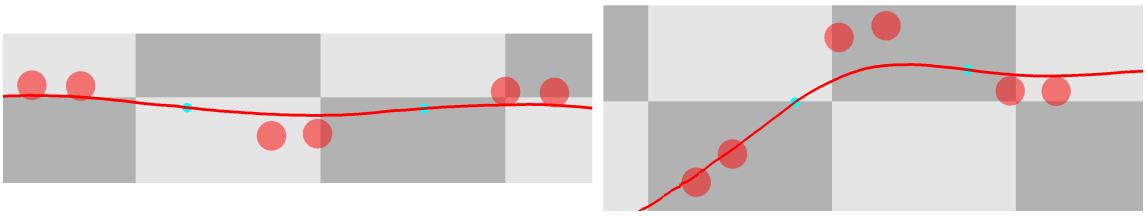

Figure 3.8: Heel and toe ground contacts from motion capture data, viewed top-down orthographically with the root path. Foot orientation seems to remain roughly tangent to the root path at its nearest point.

3.2.4 Timewarping

Once a path manipulation is complete, the new keys viewed with the original timing may appear unnatural. While it is possible for a user to manually manipulate the timing of a deformed motion [50], determining a natural timing is a difficult task, and especially undesirable when a user may want a deformed motion to simply "look correct" relative to the manipulated path, without having to specify motion speed or duration.

In addition, the spatial and temporal attributes of a motion are tightly coupled and arguably should not be modified independently. Such a relationship can be described using the Froude number, a quantity used to establish dynamic similarity between phenomena at different scales, which was originally developed to aid in 19th century ship design [28] (Vaughn and O'Malley [115] provide a complete historical and contemporary context of the Froude number). The

dimensionless Froude number is defined as:

$$Fr = \frac{v^2}{gL} \tag{3.14}$$

where v is velocity, g is gravity, and L is a characteristic length. Alexander [3] used the Froude number to describe a consistent relationship in many different animals, including humans, between velocity and stride length λ, where L is leg length:

$$\frac{\lambda}{L} = 2.3 \left(\frac{v^2}{gL} \right)^{0.3} \tag{3.15}$$

This relationship between stride length and velocity is used to automatically timewarp a motion. Equation 3.15 can be rearranged to describe the ratio between new and old velocities caused by a change in stride length:

$$\frac{v_{new}}{v_{old}} = \left(\frac{\lambda_{new}}{\lambda_{old}} \right)^{\frac{5}{3}} \tag{3.16}$$

which allow a new "target velocity" to be determined from the old velocity and the old and new stride lengths:

$$v_{Fr} = \left(\frac{\lambda_{new}}{\lambda_{old}} \right)^{\frac{5}{3}} \cdot v_{old} \tag{3.17}$$

Since only a stride length *ratio* is required for this calculation, actual stride length need not be measured from the motion at all. Instead, changes in stride length are approximated at any key by the proportional change in path edge length at that key; these per-key changes in edge length are closely approximated by the scale factors which are already calculated for the path deformation as described in Section 3.2.2. However, since the scale factors are computed independently to each path segment between a pair of adjacent path handles, the ratios of new-to-old path edge length can be discontinuous. To obtain a continuous and smooth edge length ratio, the segment scale factors are assigned to the arclength midpoint of each segment, and the intervening values are computed through cubic spline interpolation.

Using this continuous scale factor as an approximation of the stride length ratio, a new target velocity can be calculated for each key. Since the starting time of the animation remains

at time 0, these target velocities overconstrain the new key times $\hat{\mathbf{t}} = \{\hat{t}_0, \hat{t}_1, \ldots, \hat{t}_n\}$, which are calculated by minimizing the squared error between the target and new velocities (calculated through finite differences):

$$\underset{\hat{\mathbf{t}}}{\operatorname{argmin}} \sum \left(v_{Fri} - \frac{\|\hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_{i-1}\|}{\hat{t}_{i+1} - \hat{t}_{i-1}} \right)^2 \tag{3.18}$$

3.3 Motion Components

For anything but the simplest motions, applying the approach from the previous section uniformly to all parts of the motion is unsuitable, because different parts of a motion should not be freely deformed or timewarped. For example, Kim et al. [50] identified interactions between multiple characters and constrained any path edits to maintain the relative character positions during interaction. This notion is extended to a variety of other *motion components* for single characters, which are easily identified automatically and can occur at any time as well as simultaneously.

3.3.1 Vertical Motion

Previous path deformation approaches [31, 50] as well as the basic algorithm described in Section 3.2 operate solely horizontally, in two dimensions, which greatly restricts the user's editing capability. While it may be possible to adapt as-rigid-as-possible deformation to work in three dimensions simultaneously, a general approach treating all dimensions similarly would not generate realistic deformations resulting from vertical manipulations. This is due to the rotation-invariant local coordinate system of the method (Equation 3.1): while motion paths in a horizontal plane can be easily rotated around a vertical axis without affecting their plausibility, rotating motion paths in a vertical plane around a horizontal axis would not result in a realistic new path (Figure 3.9).

Instead of a rotation, a more appropriate transformation of a motion in a vertical plane is a shear, which retains the vertical orientation of a motion. This is accomplished by applying the as-rigid-as-possible algorithm a second time, to the horizontal and vertical components of a motion path, where the horizontal coordinates are fully constrained after being determined

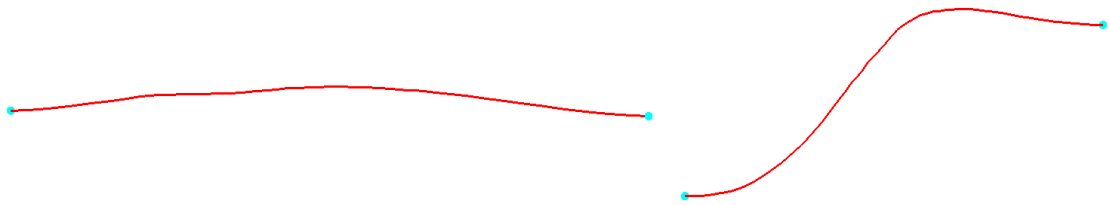

Figure 3.9: Side views of a path during a forward and upward step. Actual vertical adjustments of a motion path do not induce rotation of the path, but a different transformation.

from the ground-plane-parallel two-dimensional deformation. In this vertical deformation, a different local coordinate system can be defined which uses a local semi-horizontal axis, and the global vertical axis:

$$\mathbf{p}_i = \mathbf{p}_{i-1} + u_i(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) + v_i(0, 1) \tag{3.19}$$

This formulation results in local shears instead of rotations, as intended (Figure 3.10). However, deforming part of a path by a shear while adjacent path segments are deformed normally can result in tangent changes around path handles; this is undesirable since the handles were selected due to their local vertical minimality, and as such should have horizontal tangents. This is addressed by adding a single hard tangent constraint to the least squares system for each handle originally selected as a local minima.

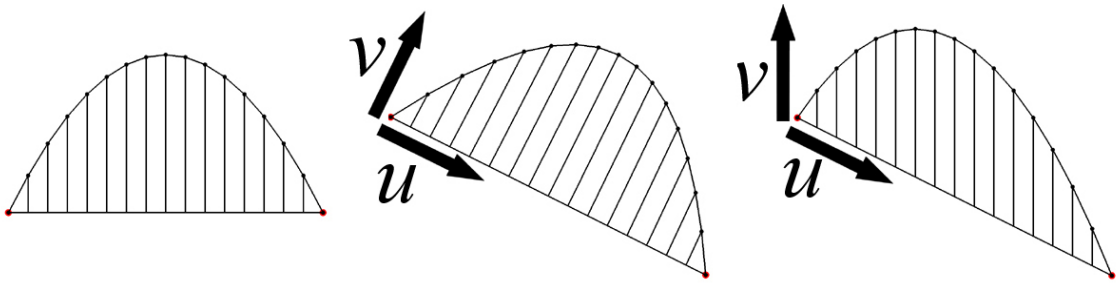

Figure 3.10: Deforming an initial curve (left) using a fully local coordinate system produces rotations (middle). A mixed local/global coordinate system produces shears (right), which is more appropriate for gravity-oriented vertical motion.

3.3.2 Ballistic Motion

Applying a deformation algorithm uniformly to a motion path can result in unrealistic deformations to non-contact or airborne segments of motion (Figure 3.11, left). To address

this, the system considers all motion without any ground contact as ballistic motion, which should result in restricting a character's centre of gravity to a parabolic arc. The character's root provides a sufficient approximation to the centre of gravity without requiring the dynamics calculations to compute the character's mass distribution (e.g., as in Shin et al. [97]). However, as an approximation, the root does not necessarily follow a parabolic arc. To address this, the transformation of ballistic path segments is restricted to those affine transformations which *would* preserve parabolas. During horizontal deformation (Section 3.2.2), simple hard constraints restrict the ballistic segment to a similarity transform (Figure 3.11, right). Since ballistic segments generally contain no side-to-side motion, this is roughly equivalent to allowing these segments to stretch and rotate, but not bend. During vertical deformation (Section 3.3.1), the entire ballistic segment is restricted to a single shear.

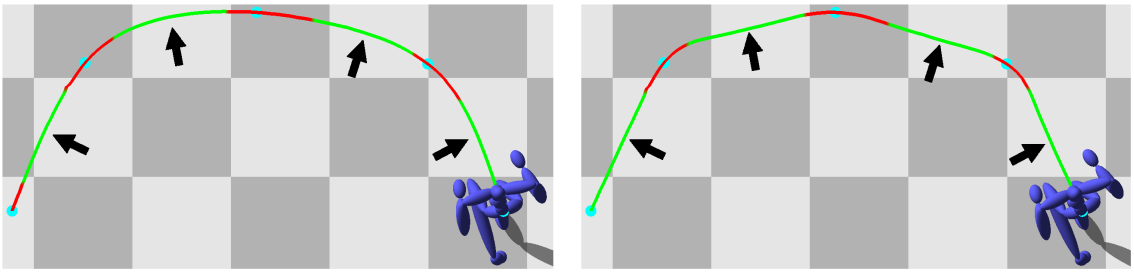

Figure 3.11: Without appropriate constraints, ballistic motion (green path segments) can be deformed into impossible shapes (left). Constraining the deformation (right) allows the path to bend during contact (red path segments), but only scale/stretch during ballistic motion.

While realistic ballistic motions should not be deformable horizontally, vertical deformations should be permitted: specifically, modifying the height of a jump without modifying the vertical positions of its enclosing path handles. This is accomplished by adding a vertical-only handle which uniformly scales the vertical displacement of each constrained key from the plane connecting the first and last unconstrained keys.

Following a constrained deformation of a ballistic path segment, the pose adjustment stage of the system (Section 3.2.3) works without modification. However, the velocity-based timewarping produces an unrealistic result on deformed ballistic segments, which is unsurprising as it is based on the behaviour of stride-based ground motion. Because ballistic segments are affected primarily by gravity, the system determines a timewarp for these segments which preserves the magnitude of each key's acceleration (i.e., gravity) rather than velocity. However,

the same least squares formulation is not adaptable because this condition cannot be expressed in terms which are linear with respect to key times, and nonlinear solutions for an acceleration-based timewarp may not obtainable at interactive rates.

To quickly determine an acceleration-based timewarp, the full timewarp is formed by combining local timewarps calculated for each key. First, the new acceleration at key i can be defined by finite differences in terms of the preceding and subsequent keys:

$$accel_i = \frac{1}{\Delta\hat{t}_{i-1} + \Delta\hat{t}_i} \cdot \left(\frac{1}{\Delta\hat{t}_i} \cdot \Delta\hat{\mathbf{p}}_i - \frac{1}{\Delta\hat{t}_{i-1}} \cdot \Delta\hat{\mathbf{p}}_{i-1} \right) \tag{3.20}$$

where time and position deltas $\Delta\hat{t}_i$ and $\Delta\hat{\mathbf{p}}_i$ are used for shorthand, since only relative values affect the calculation:

$$\Delta\hat{t}_i = \hat{t}_{i+1} - \hat{t}_i \tag{3.21}$$

$$\Delta\hat{\mathbf{p}}_i = \hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_i \tag{3.22}$$

For each key, the local uniform timewarp of magnitude m_i is determined which minimizes the squared distance between the new timewarped acceleration and the original acceleration $\mathbf{a}_i$:

$$\underset{m_i}{\operatorname{argmin}} \left|\left| \frac{1}{{m_i}^2} accel_i - \mathbf{a}_i \right|\right|^2 \tag{3.23}$$

Each m_i is calculated in closed form. Applying the local timewarps to ballistic keys preserves accelerations correctly, but retaining the velocity-based timewarp for all contact keys leads to velocity discontinuities at the contact/ballistic transition. Therefore, the as-rigid-as-possible formulation is adapted to determine the final timewarp, similar to Kim et al. [50], though this process is fully automatic.

All keys in a fully in-contact path segment between two handles have their relative times from velocity-based timewarping (Equation 3.17) set as hard constraints. Ballistic keys also have their relative times set as hard constraints from the acceleration-based timewarps m_i as follows, for each time delta $\Delta\hat{t}_i$ between keys j and k inclusive, based on the original time deltas

Δt_i:

$$\Delta \hat{t}_i = \begin{cases} m_i \cdot \Delta t_i & \text{if } i = j \\ 0.5\,(m_i + m_{i+1}) \cdot \Delta t_i & \text{if } j < i < k-1 \\ m_{i+1} \cdot \Delta t_i & \text{if } i = k-1 \end{cases} \tag{3.24}$$

The times of "transition keys" preceding and following a ballistic segment up to the nearest handle are not constrained; therefore, a one-dimensional application of the as-rigid-as-possible algorithm determines a smooth timing transition between the constrained time deltas from the velocity-based contact timewarps and the acceleration-based ballistic timewarps.

3.3.3 Turns

While the stride length-based timewarping described in Section 3.2.4 is effective in many situations, it can yield unrealistically fast motion during turns. To address this, an additional timewarping criteria is used, based on the "one-third power law", a commonly-used model in biomechanics which relates instantaneous velocity v and path curvature κ:

$$v = c \cdot \left(\frac{1}{\kappa}\right)^{\frac{1}{3}} \tag{3.25}$$

where c is often referred to as an empirically-determined "velocity gain factor". This law intuitively results in an inverse relationship between path curvature and velocity, i.e., sharper turns occur slower. To evaluate path curvature in practice, this law is generally applied not to the high-frequency actual path of the kinematic root or centre of mass, but to an ideal path which is being actively followed.

To approximate this ideal path, an abstraction of the root path is desirable; an "overall path" which incorporates only the lower-frequency aspects of the root's motion. Directly accomplishing this by frequency filtering the root path would require parameter tuning which may need to be motion-specific, and as such is undesirable. As well, an overall path that is curvature continuous would be consistent with recent biomechanics research which indicates that human walking paths are well-described by curvature continuous paths [5].

A simple but effective solution to meet these criteria is a natural C2 cubic spline which interpolates only the path manipulation handles. This results in a path which ignores the

higher-frequency oscillations like hip sway between handles but still describes the complete root path well, is curvature continuous, and is efficient to compute (Figure 3.12). The common approach in one-third power law literature is followed for determining curvature at a key by evaluating the curvature at the nearest point on the overall path.

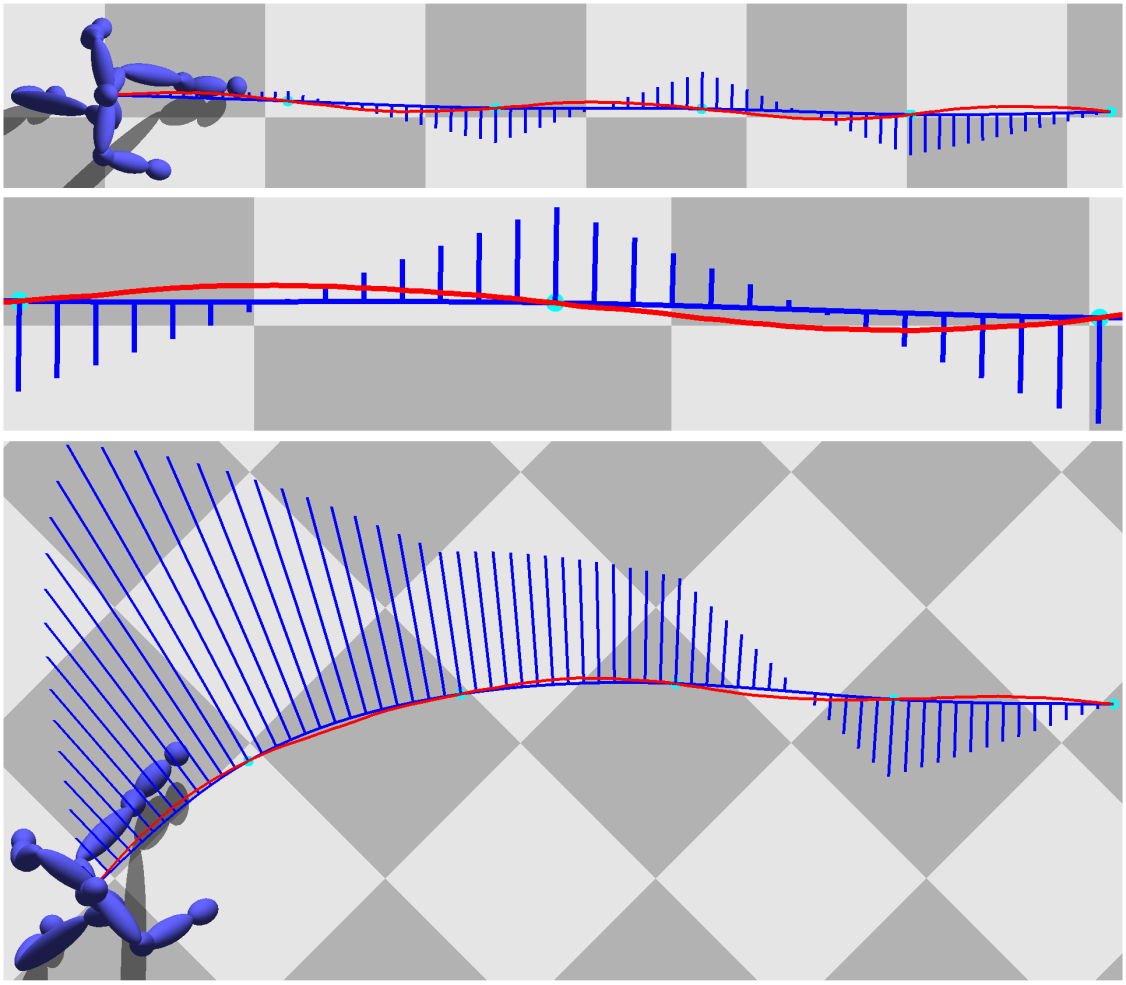

Figure 3.12: Top: The overall path (blue) fits the actual root path (red) well and has low curvature (blue normal lines) for forward motion. Middle: close up of higher-frequency hip sway ignored by the overall path. Bottom: overall path and curvature for a turning walk.

The one-third power law is generally applied in a piecewise fashion to ranges of curvatures, and clearly contains a singularity at $\kappa = 0$ which prevents the law from applying in all situations. However, by approximating κ with $\epsilon + \kappa$ where $\epsilon > 0$, and assuming that path deformations do not affect the velocity gain factor, the ratio between old and new velocities can be expressed as

a function of curvature with no singularities:

$$\frac{v_{new}}{v_{old}} = \left(\frac{\epsilon + \kappa_{old}}{\epsilon + \kappa_{new}}\right)^{\frac{1}{3}} \tag{3.26}$$

This, in turn, allows a new target velocity to be determined from this modified curvature ratio and the old velocity:

$$v_\kappa = \left(\frac{\epsilon + \kappa_{old}}{\epsilon + \kappa_{new}}\right)^{\frac{1}{3}} \cdot v_{old} \tag{3.27}$$

The two power laws presented in Equations 3.17 and 3.27 are generally mutually exclusive, as the Froude number-based and curvature-based velocity laws describe straight-ahead and turning movements, respectively. However, recent work by Bennequin et al. [10] presented a theory of movement timing which combines multiple velocity predictions based on different geometric properties. Therefore, the two different target velocities can be combined with the following product:

$$v_{new} = (v_{Fr})^{\beta_{Fr}} \cdot (v_\kappa)^{\beta_\kappa} \tag{3.28}$$

where $\beta_{Fr} + \beta_\kappa = 1$ and $\beta_{Fr}, \beta_\kappa \geq 0$. Bennequin et al. present a similar multiplicative form of velocity combination rather than an additive form, with exponents which sum to 1. While their model's weights vary continuously with geometric properties, the adaptation of such a derivation is left to future work, as constant values of $\beta_{Fr} = \beta_\kappa = 0.5$ work well and generate realistic results that are also very similar to ground truth comparisons (see Section 3.4).

3.4 Results

This motion editing system has been tested on a variety of human locomotive motions recorded by optical motion capture at 120 frames per second. Most motions are loaded in under one second, an interactive framerate is maintained during editing without subsampling motion, and the pose adjustment and timewarping are performed quickly enough to be imperceptible. All of the following edits were performed interactively in single sessions of less than 30 seconds each.

The system can easily edit motions which contain a mix of motion components. Figure 3.13 shows a motion containing a sequence of running, a jump, standing, and running, modified in a number of ways: the path has deformed into a zigzag pattern, the jump has been made shorter

and higher, and the landing has been lowered.

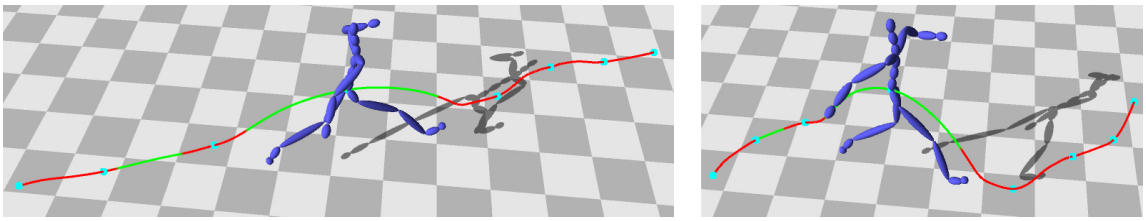

Figure 3.13: A mixed motion of running, jumping, and standing, before (left) and after (right) a variety of path edits.

In most cases, end effector constraints are constrained to horizontal translation to prevent ground interpenetration, and to rotation about a vertical axis to prevent tilting or rolling the foot into an unfeasible position. However, removing the constraints on translation allows footplants to be automatically elevated along with a motion path. Figure 3.14 shows a level walking motion modified into a stair climbing motion.

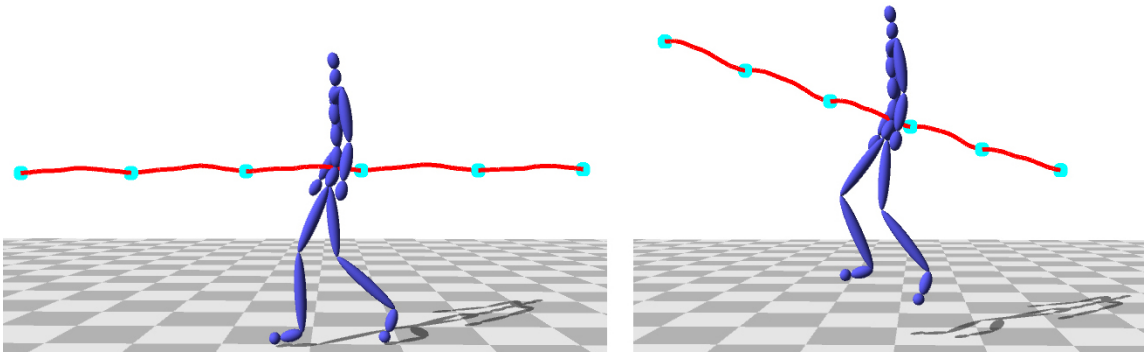

Figure 3.14: Vertical translation of end effector constraints allows a walk (left) to be modified into a stair climb (right).

Significantly different variations on a single motion can be quickly generated with this system. Figure 3.15 shows a single jumping motion which has been modified in both distance and height, with additional modifications to the crouching height before and after the jump to properly anticipate and follow-through on the exaggerated jump. The acceleration-based timewarping speeds up the lengthened ballistic motion but slows down the heightened motion to maintain gravity. The original motion took 3.45 seconds; the longer jump requires a higher forward velocity which results in a shorter duration of 3.14 seconds, while the higher jump necessarily requires more "hang time" and is 3.88 seconds in duration.

Non-human character motion can also be edited using the system, which requires only small

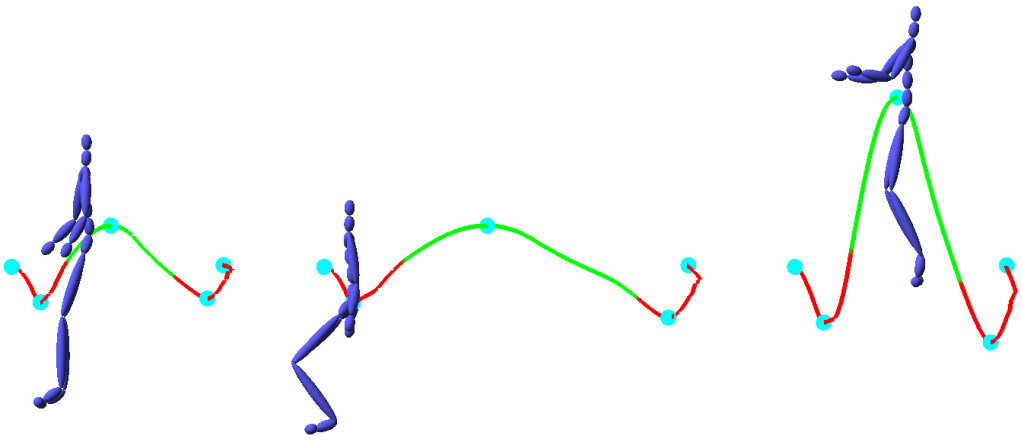

Figure 3.15: A jumping motion shown unedited (left), lengthened (middle), and heightened (right). All motions are shown 2.3 seconds from their start.

modifications to the criteria for identifying manipulation handles on the motion path. For a forward walk cycle of a horse (Figure 3.16, left), double-stance periods of solely the rear legs are used to identify handles, since the skeleton root is located at the hips. Once these handles are identified, no further modifications to the system are necessary to edit the motion (Figure 3.16, right).

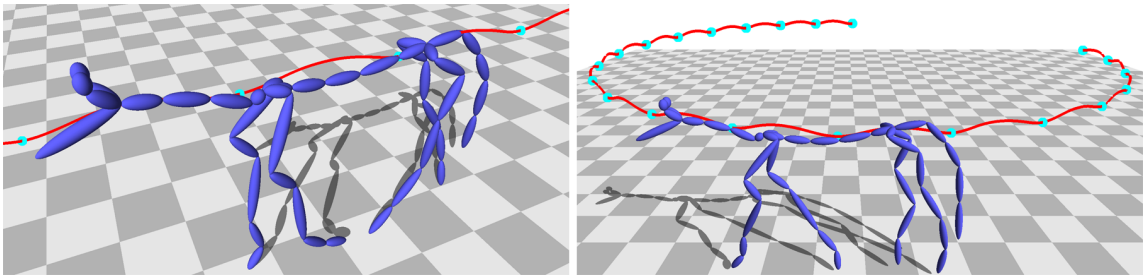

Figure 3.16: A forward walk cycle of a horse (left) edited into a circular walk (right).

Editing completely non-biomechanical character motion is also possible. Motion sampled from a simulated bouncing ball can be edited by identifying manipulation handles at all local minima without considering contact, i.e., at each impact. Once this is completed, each bounce of the ball is treated as a separate ballistic motion period (Section 3.3.2). Figure 3.17 shows a bouncing ball before and after a number of edits to the height and length of each bounce.

Finally, the path deformation algorithm is able to accurately reproduce motion paths with very low error by modifying another motion from the same subject. Figure 3.18 shows two motions manually truncated to contain the same number of path handles, with the same foot

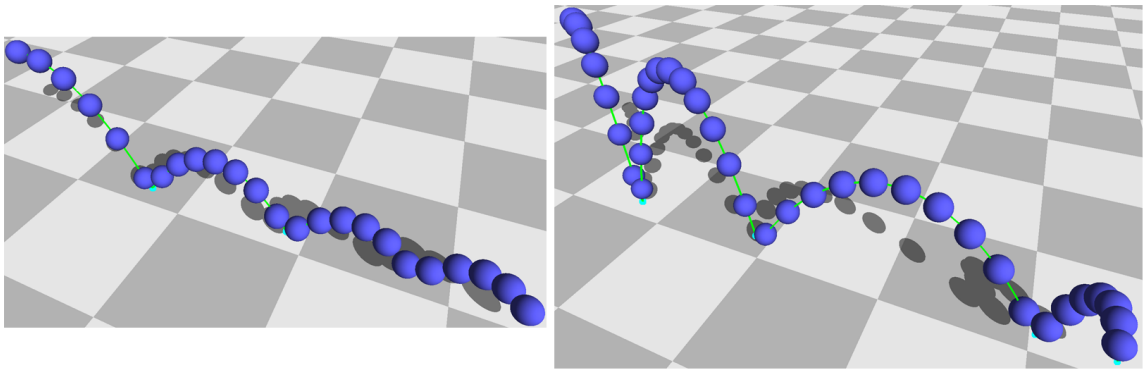

Figure 3.17: A bouncing ball animation before (left) and after (right) several edits. Each motion is "strobed" at 0.1 second intervals; the original motion is 2.32 seconds in duration, and the edited motion is 3.55 seconds.

forward at each corresponding handle. An automated edit was used to align the path handles of the source motion, a forward walk, with the corresponding handles of the target motion, a turn. Using the real-world scale from the motion capture data, the average distance between each key in the modified path and the nearest position along the target path has been calculated to be 0.5cm, and the maximum distance is 1.5cm.

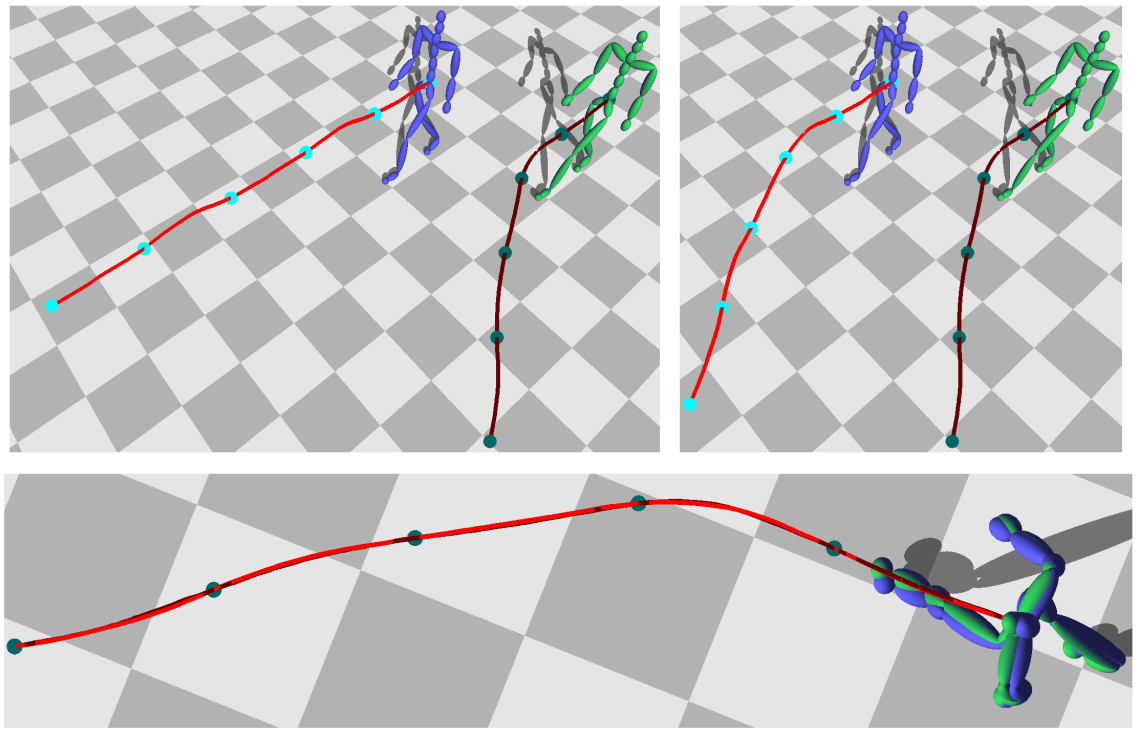

Figure 3.18: A forward walk and turn compared before (top left) and after (top right) an automated alignment edit. The modified and target motion are overlaid (bottom) to compare their paths.

3.5 Conclusions

This chapter has presented a path-based kinematic motion editing system which extends the path deformation capabilities of previous work by identifying meaningful handles to restrict path deformations realistically and modifying the as-rigid-as-possible deformation algorithm to deform motion paths more realistically. The method uses a timewarping technique which allows the user to concentrate on meaningful spatial edits, while motion timing is automatically modified to preserve the biomechanical relationship between path shape and velocity. A basic editing procedure is extended by a number of motion components which are automatically identified and augment or restrict all stages of the editing process to maintain gravity-relative motion during vertical manipulations, preserve the shape and timing qualities of ballistic motion, and adjust turning speeds in a biomechanically correct way.

There are some limitations to this system. The velocity-based timewarping strictly enforces a general model for natural timing, but in reality, there is a range of plausible velocities for any path, with more extreme variations possible with deliberate effort. Allowing the user to modify timing within this feasible range would allow meaningful timing control without resorting to specifying absolute timing.

There are many alternative methods for deforming a motion path to fit user-specified constraints which could potentially produce more realistic results or allow for the incorporation of more desirable behaviours. For example, Solomon et al. [104] presented a two-dimensional technique similar to as-rigid-as-possible deformation which computes "approximate killing vector fields" to define a series of near-isometric direction fields to warp a shape to meet user constraints. The generalization of such approaches by Martinez Esturo et al. [73] also demonstrated such a method's feasibility in three dimensions at interactive rates, which could allow for a path deformation algorithm which incorporates appropriate constraints while acting in all dimensions simultaneously, unlike our two-stage horizontal-vertical approach (Section 3.3.1).

There most likely are limits to how aspects of motion can be edited without incorporating dynamics calculations. More complicated acrobatic maneuvers would require a centre of mass to maintain realism. Attempts within this method have been unsuccessful at modeling foot

placement during turns kinematically using path curvature and velocity, while even simple dynamics calculations can modify foot placement accurately to maintain apparent balance [97].

An alternative to incorporating real-world dynamics, either by explicit force-based calculations or our biomechanical and physics rules-based approach, could be to use "cartoon physics" for a different type of performance. Lasseter [58] has previously explored how to apply principles of traditional animation to computer-animated characters, and approaches such as the signal filtering technique of Wang et al. [119] have shown that these principles can be applied automatically to character motion; by using these rules instead, our system could potentially modify motion paths and timing after a user edit to appear more "cartoonish".

This approach cannot edit aspects of a motion which are not significantly expressed by the root motion path. For example, the same motion in a variety of styles would differ in performance aspects such as extreme poses and limb timing, but each could have similar motion paths. As well, static motions containing gestural components do not have a root motion path that can be meaningfully edited; however, perhaps this editing approach could be adapted to limb paths as well.

The system also does not currently modify joints within the character skeleton except to satisfy end effector constraints. Allowing the upper body of a character to be adjusted would allow, for example, a walking character to swing their arms further as their strides lengthened, or a horse to bend its spine to lean towards the direction of a turn.

Chapter 4

Finger Walking: Motion Editing By Contact–Based Hand Performance

This chapter presents a method for generating full-body animation from the performance on a touch-sensitive tabletop of "finger walking", where two fingers are used to pantomime leg movements. Data collected from finger walking performances in an exploratory user study was analyzed to determine which motion parameters are most reliable and expressive for the purpose of generating corresponding full-body animations. Based on this analysis, a compact set of motion features is presented for classifying the locomotion type of a finger performance. A prototype interactive animation system was implemented to generate full-body animations of a known locomotion type from finger walking by estimating the motion path of a finger performance, and editing the path of a corresponding animation to match. The classification accuracy and output animation quality of this system was evaluated in a second user study, demonstrating that satisfying full-body animations can be reliably generated from finger performances.

4.1 Overview

Creating full-body animations can be an imposing task for novice animators, since animation software often requires a large number of parameters such as positions and joint angles to be specified at precise times. One approach to address this problem is *performance interfaces*,

which utilize the timing of a user's actions or physical movements. Performance interfaces have the potential to be particularly accessible, since realistic motion parameters can be generated not only from the intentional aspects of a performance, but also implicitly by the physical constraints on a user's performance.

An exploratory user study was conducted to examine how motion can be communicated through hand motions, by having users perform a number of motion types in whatever manner they preferred. A majority of users chose to use "finger walking", where the middle and index fingers of the dominant hand are used to pantomime the leg motion of a full body (Figure 4.1). However, analysis of the touch-sensitive tabletop data collected during this study shows that seemingly-precise and expressive finger motions can not only possess inconsistent motion parameters, but can differ significantly from the corresponding full-body motions.

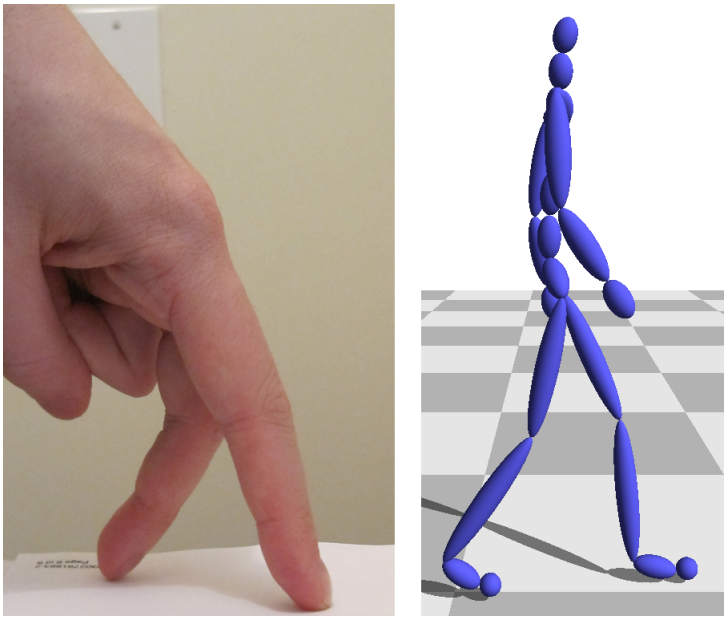

Figure 4.1: Finger walking (left) is a natural and expressive way to communicate full-body locomotion (right).

Based on this first study, an interactive animation system that was developed to be controlled by finger walking, with specific handling of the particularities of how users perform finger walks and adopting a gestural approach to match the general parameters of a user's performance without too closely replicating unrealistic motion. The motion type of a user's finger performance can be determined by summarizing the performance in a compact feature vector, and comparing it to the feature vectors of other performances with known types. Given

a particular locomotion type, the system estimates the central path of the motion from its "footprints" and edits a full-body animation of the corresponding type to match this path.

A second performance study was conducted to evaluate the accuracy of the motion type classification as well as user satisfaction with the motions generated from their finger performances. The system can reliably classify the locomotion type of a new user's input, as well as generate satisfying animations matching the intent of the user's performance.

This work makes a number of contributions. An analysis of how users express full-body motion with hand motions, particularly finger walking, informs the design of a gestural interface. This finger walking interface, using feature-based motion classification and path-based editing, generates full-body animation which match a user's gestural input. Finally, a novel kinematic path estimation algorithm is presented which determines a smooth motion path based only on a sequence of footprint positions.

4.2 Exploratory Study

An initial exploratory study was conducted to identify ways that users would choose to communicate full-body motion using their hands. The goal was to identify a preferred and comfortable method of interaction, to inform the design of an animation interface.

4.2.1 Methodology

The study was designed to passively measure hand performances with minimal intrusion to the participants, in order to maintain all possible performance strategies or techniques. To accomplish this, a Microsoft Surface touch-sensitive table was used to record any surface contact that occurred, and a video camera recorded above-surface motions and gestures (Figure 4.2). As the display functionality of the Surface was inactive to avoid encouraging surface-based interaction, a secondary monitor was placed beyond the Surface to display instructions. Participants were presented with a scripted introduction which explained the purpose of the study and the equipment being used. They were also informed that they could express motion in any manner they chose within the "capture volume" above the Surface, from any direction, as there was enough space to interact with the Surface from any side.

The study consisted of two stages. In the first "freeform" stage, names and brief descriptions of locomotion types (such as walking or running) were presented, and the participants were asked to perform these motions without any spatial instructions, such as directionality. In the second "mimicked" stage, short motion-captured animations varying in locomotion type and direction (such as walking and turning or running forward) were played, instead of presenting written descriptions. The participants were asked to communicate the general type and direction of the animation, without specifically trying to capture particular details, such as the number of steps. Participants could restart their performances and replay animation clips at any time. After deciding when a performance was complete, participants rated their satisfaction with their performance on an integer scale of 1 ("very unsatisfied") to 5 ("very satisfied"). After the final performance, an image of a participant's hands placed palm-down was captured using the Surface, and a short survey of follow-up questions was administered.

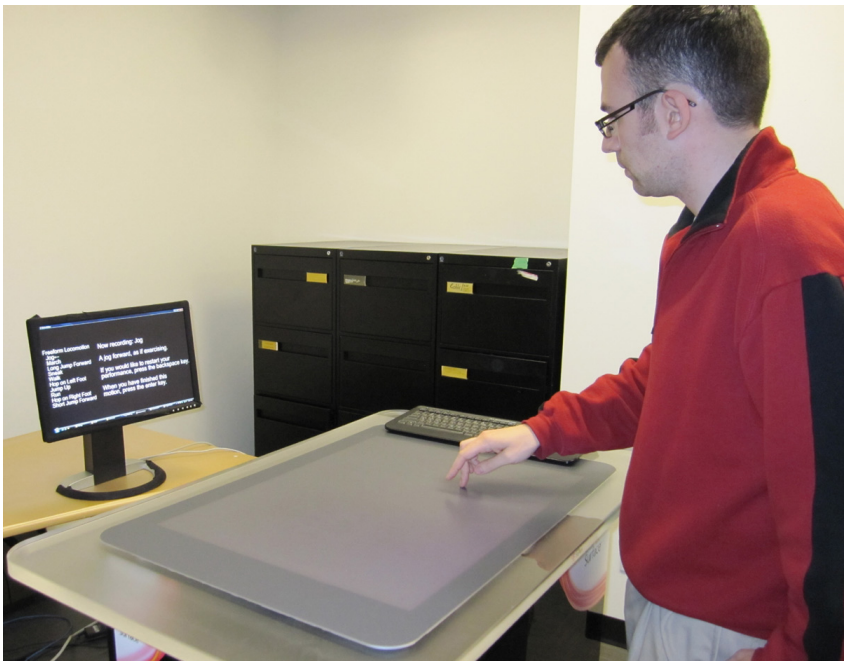

Figure 4.2: Equipment used to study hand performances. Directed by instructions on the external monitor, participants were free to move around the Microsoft Surface, which captured contact data, while their above-surface motions were video recorded.

4.2.2 Results and Observations

The study was performed by 12 volunteer participants (10 male, 2 female), each performing 28 motions (10 in the first stage, 18 in the second stage). The methods of performance were as follows:

- 73 percent of all performances employed "classic" finger walking on the Surface, with the middle and index fingers of the dominant hand pantomiming the leg movement of the corresponding full-body motion.
- 12 percent were by pantomiming upper-body motion above the Surface, such as hands swinging back and forth during a walk.
- 10 percent used both hands to pantomime foot movement, with either fists or open palms alternately "stepping" on or above the Surface.
- 5 percent used a single abstraction of solely full-body position, such as a single finger or clenched fist, on the Surface.

While some participants experimented with multiple techniques, 11 out of 12 used finger walking at least once. Overall, participants were more satisfied with the finger walking performances; the average rating of finger walking motions was 3.7 out of 5, compared to 3.3 out of 5 for all other types of performance. During follow-up questions, most users indicated that their choice of finger walking was because motion of the lower body was more important to their performance, and they did not need to incorporate upper body motion to communicate locomotion type.

Therefore, finger walking seems to be both a natural and comfortable choice for communicating full-body motion through hand performance, and the contact data gathered during only the finger walking performances was analyzed.

4.2.3 Data Analysis

The first method of examining finger walking contact data was to statically visualize all of the contacts which occurred during a single performance. Contact information is provided by the Surface as ellipses with associated times; a single contact consists of a sequence of ellipses,

representing contact shape from initial contact to lift-off. Since this contact data may be provided at irregular rates, for the purposes of visualization, linear interpolation was used to re-sample all ellipse parameters (position, rotation, major and minor axes) at 30Hz; an example of this visualization for a finger walk, with a detail of a single contact, is shown in Figure 4.3.

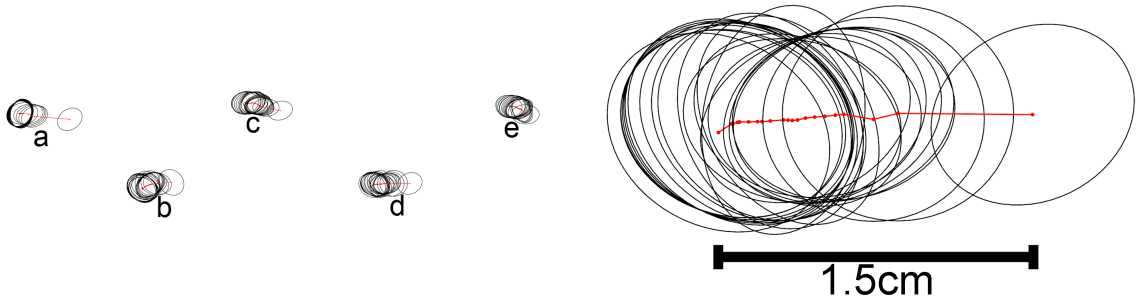

Figure 4.3: Contact shapes for a forward finger walk in order a-e, sampled at 30Hz (left). An enlarged view of contact d (right), with ellipse center path in red, shows how contacts generally begin with a stomp, followed by an accelerating roll forward.

Most finger walking contacts during forward motion do not consist of a smooth and constant roll in the direction of the motion, but instead generally begin with a "stomp" down with low forward velocity, and roll forwards while accelerating. It was not unusual to observe finger contacts which slip just before liftoff, most likely due to reduced friction when a user's fingernail is the predominant contact. Finger contact shape also appears to be unreliable and does not have a consistent orientation, such as elongation relative to the direction of travel, and the path of each contact's center is generally forward, but noisy.

There are other expressive parameters of finger motions which can be compared to the equivalent parameters of corresponding full-body motions. For example, the number of active contacts - i.e., fingers or feet in contact with the ground - can be examined as it changes over time. During a full-body walk, the majority of time is spent on one foot, with regular brief periods of double-stance after the swing foot lands but before the stance foot lifts off. However, since a finger walking hand is not propelled by ground contact, but rather by the movement of the arm or upper body of the performer, the number of active contacts over time can be significantly inconsistent even for finger motions which appear "correct". Figure 4.4 compares the active contacts over time of a motion-captured full-body walk and a freeform finger walk which was rated 5 out of 5 by its performer.

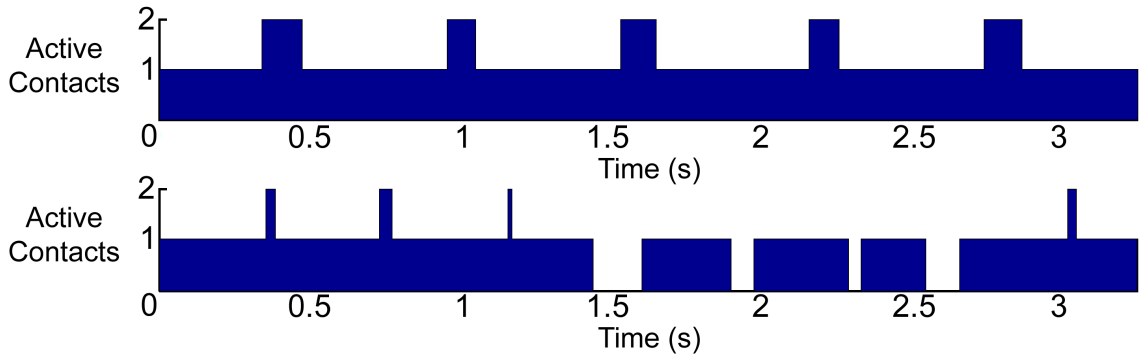

Figure 4.4: The number of active contacts over time during a full-body walk (top) is consistent, with regular, brief periods of double-contact. The number of active contacts during a finger walk (bottom), however, can be significantly inconsistent even though the motion may still appear appropriate.

Examining the number of active contacts *overall* throughout an entire motion, rather than moment-to-moment, can be more robust to temporary variations. Figure 4.5 shows ternary plots of the proportion of time spent during a number of locomotion types with zero, one, or two active contacts, for the freeform performances, mimicked performances, and the example motion capture clips. The full-body motions almost entirely consist of two contact states, for example, either zero or one contact for running, and one or two contacts for walking. The finger motions, however, usually contain a more significant proportion of the third contact state, such as zero contacts during walking. It also appears that finger motions are biased towards single-contact states relative to the full-body motions; however, the magnitude of this bias seems to be locomotion type-dependent.

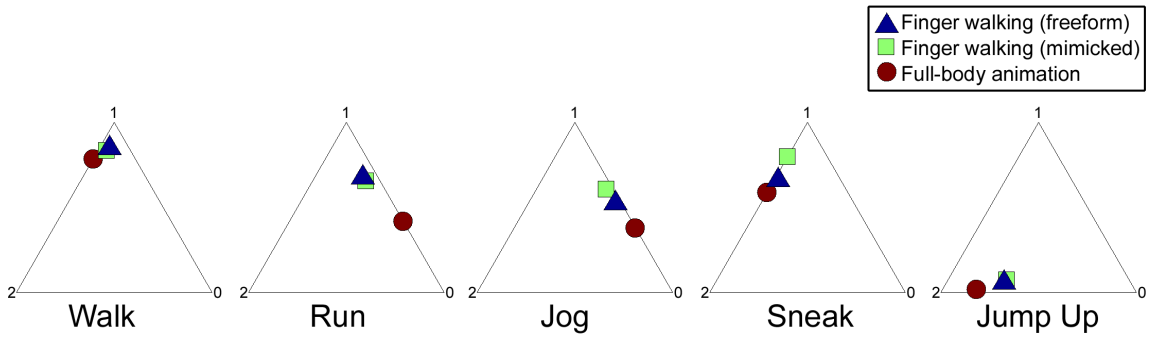

Figure 4.5: Average proportion of time spent during a motion with 0,1, or 2 active contacts, plotted by barycentric coordinates, for a variety of locomotion types. The relationship between freeform and mimicked finger walking and a corresponding full-body motion differs depending on locomotion type.

There are many other parameters which can be used to compare finger motions to full-body motion. Figure 4.6 shows the average contact frequency for a number of locomotion types, for the performed and example motions. For some types, the contact frequency is similar. However, running contacts were performed significantly more rapidly than full-body running, even during the mimicked motions where an example had been played. The frequency of jogging contacts was initially under-performed and then over-performed during the mimicked motions.

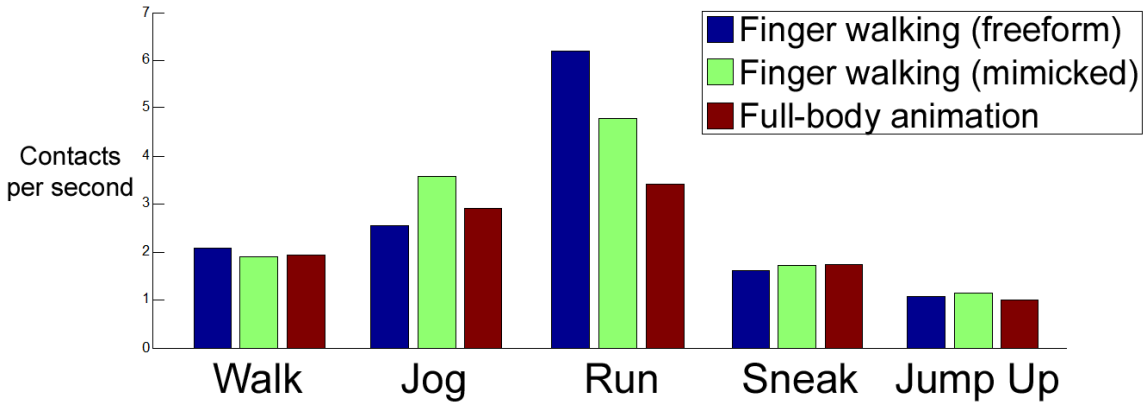

Figure 4.6: Contact frequency for a variety of motion types. Again, the relationship between freeform and mimicked finger walking and a corresponding animation differs depending on locomotion type.

4.2.4 Discussion

Based on the analysis of finger motion contact data, it appears that even when performing motions to their satisfaction, precise parameters of user performances such as contact shape, position, and state can be imprecise and unreliable. Examining more robust overall characteristics yields greater consistency - however, the correspondence between finger and full-body motions appears to be highly dependent on the motion type.

Therefore, a method of creating full-body animation by directly mapping finger motion parameters would need to handle this varying abstraction in order to generate plausible results. Instead, given the inconsistencies in user inputs, it is possible that users treat this type of performance as illustrative - demonstrating general characteristics such as motion type and overall direction, without concentrating on either the consistency or realism of specific motion and contact parameters.

4.3 Implementation

To accommodate illustrative input by the user, a *gestural* interface was developed to generate animations which match high-level characteristics of finger motions, rather than from the finger motion directly. This system consists of two components that operate on a user's performed finger motion. The locomotion type of a new user's performance can be automatically determined, based on the previous users' motions of known types. Given a particular locomotion type, an appropriate full-body animation can be selected and edited (Figure 4.7), without replicating finger motion inconsistencies which would make it look unrealistic. This approach of classification and gestural motion editing from a single input is similar to the Motion Doodles work of Thorne et al. [110], which used continuous sketched paths from a "side view" instead of finger contacts.

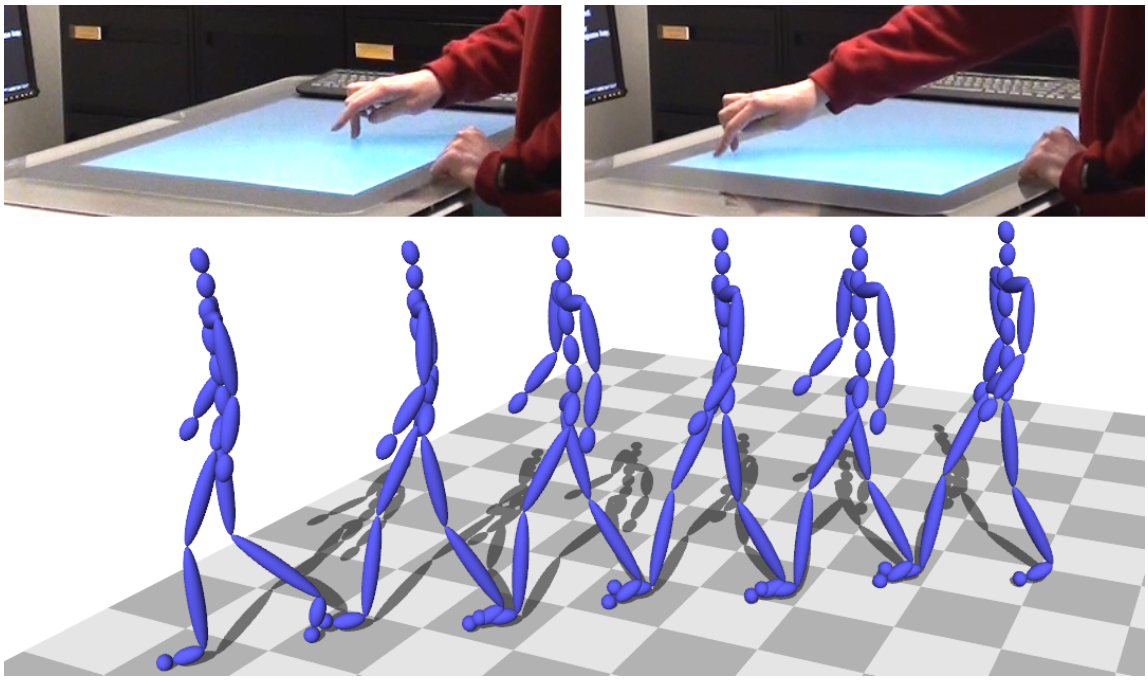

Figure 4.7: A finger motion performed on the prototype system (top) and the edited full-body animation which was automatically generated as output (bottom).

4.3.1 Feature-Based Classification

Since finger motions can vary in their similarity to full-body motion depending on the locomotion type, attempting to classify the locomotion type of a new finger motion by direct

comparison to full-body motions could prove unreliable. Therefore, an example-driven approach was chosen to classify new finger motions based solely on previously-recorded finger motions. Given a finger motion of unknown type, a feature vector is determined, which can be compared to feature vectors from existing finger performances of known types, from a variety of users.

Every feature must be valid for all locomotion types, and should produce similar results for motions of the same type which differ in the number of steps. Furthermore, these features should be robust to the types of potential irregularities examined in Section 4.2.3, such as inconsistent contact state or finger slips, or even missing steps entirely.

The chosen feature vector consists of six features of a finger motion:

- The contact time proportions for zero, one, and 2 active contacts,
- Average velocity,
- Average stride frequency,
- And average contact-to-contact distance.

Distance between contacts is measured from the first point of contact, since analysis of the contact data from the exploratory study showed the beginning of a contact to be more stable than the end point, where slipping can occur.

As an initial test, the accuracy of a number of standard classification techniques were tested using this feature vector on data from the exploratory study, with accuracy ranging between 65 and 80 percent. This indicated that accurate classification was possible; a full analysis of classifier accuracy on a larger dataset is included with the results of the second study in Section 4.4.2.

4.3.2 Data Normalization

The animation system should accommodate not only a variety of locomotion types, but a variety of users as well. While the feature vector is based on contact positions and timing, which are not explicitly user-dependent, there is one very important way in which differences among users can potentially affect the results: finger length.

Compensating for subject size is a common consideration in biomechanics applications such as gait analysis, where the consequences of different heights can be significant; for example, natural walks for two people of different heights will vary in velocity and stride length. To address this, gait velocity in particular can be rendered dimensionless - effectively, normalized - using the Froude number [115], a nonlinear quantity which relates *relative* stride length (absolute stride length divided by leg length) to velocity by a nonlinear equation. This is appropriate because morphologically similar humans of different sizes are affected by a gravity force proportional to their mass, which does not vary linearly with height.

This system takes a similar approach by normalizing distances as well, but using finger length instead of leg length. While free-standing humans require nonlinear normalization, linear normalization may be sufficient for finger walking. This has the effect of modifying the spatial units of measurement so that, for example, average finger motion velocity is measured not in centimeters per second, but finger lengths per second.

The Surface can be used to quickly and automatically measure finger lengths. The intensity images captured by the Surface (Figure 4.8, left) are thresholded at a standard value to produce a binary image (Figure 4.8, middle). Fingertips on the largest blobs in the image are detected using the technique of Malik et. al [72], and the clockwise ordering of the fingers relative to the thumb identifies the dominant hand. While the lengths of the index and middle fingers are different (Figure 4.8, right), applying the lengths appropriately to different "steps" would require identifying the finger of a particular contact, which could be complex and/or unreliable. Therefore, all distances are normalized by the average length of the middle and index fingers.

Figure 4.8: While the Surface can provide approximate depth images (left), appropriate thresholding yields a hand shape (middle) which can be automatically analyzed to identify and measure fingers (right).

Gait analysis also calculates step-to-step distance in a different way; rather than measuring the direct displacement between step positions, only the "forward" distance is considered. To approximate this, a novel path estimation technique (described in Section 4.3.4) is used to determine a central path for a finger motion based on the ordered contact positions. The contact-to-contact distance is then measured in path arc-length between the contacts' projected positions along the path.

A comparison of classification accuracy using all combinations of these variations - absolute versus relative units, and direct versus path step distances - is presented with data from additional subjects in Section 4.4.2.

4.3.3 Full-Body Motion Editing

Given a particular locomotion type, the goal of the system is to generate a corresponding full-body animation matching some aspects of the user's performance. This is accomplished by editing "canonical" animations: short animation clips representing each locomotion type, which can be automatically looped to generate a smooth animation of any number of steps. However, editing the canonical animations to closely match particular characteristics of a finger motion could result in unrealistic animation, given the inconsistencies observed in finger motions (Section 4.2.3). Therefore, user input is treated as gestural - essentially, as instructions of the form of *"do this, there"* - and the canonical animation is edited to match the broader spatial parameters of a finger motion, in the form of its motion path.

Canonical animations are edited using the path-based editing technique described in Chapter 3, which identifies a sparse set of editing handles along the path of an animation. After identifying the motion path of a finger motion using a path estimation technique (Section 4.3.4), the editing handles of a straight-ahead canonical animation can be automatically placed along the finger motion path, resulting in a new animation with a very similar path to that of the user's performance.

There are two remaining degrees of freedom in this process: the *animation scale* and the *number of cycles* for the canonical animation to be edited. One possible method for determining scale is to attempt to match the average step size of the input finger motion and the animation. Unfortunately, this could result in animated characters of significantly different sizes being

generated even for a single user. Instead, based on feedback from participants during the exploratory study, a single scale is determined per user such that the leg length of the animated character is equal to the user's finger length (shown in Figure 4.1), which results in consistent character size for a particular user, across various locomotion types and performances.

To determine the appropriate number of cycles of a scaled animation, the layout of the editing handles along the path must be considered. With a fixed scale, arranging the handles to match aspects of the performance could result in erratic motion or unrealistic gait parameters, such as unnaturally long strides. Instead, in keeping with treatment of user input as gestural, the canonical motion is cycled a sufficient number of times to lay out the editing handles along the entire path while maintaining the original distance between each pair of handles, i.e., "bending" the animation without "stretching" it. This will very nearly maintain the consistency of stride length in the output animation; however, one potential downside of this method is that the animation's timing may be significantly different from the user's performance.

4.3.4 Closest Points of Approach Path Estimation

An approximation of a central motion path can be useful for a variety of purposes, such as motion analysis and categorization, and path-based motion editing. This method approximates the motion path from a sequence of contact positions by attempting to project the contact positions onto the path, which are then interpolated by a C2 natural cubic spline.

The projected positions are determined by combining a series of local solutions of the "Closest Points of Approach" problem: given two contact positions p and q and their approximate lateral directions u and v, the projected positions are the closest points along the lines $p(t) = p + t \cdot u$ and $q(t) = q + t \cdot v$ (Figure 4.9).

To apply this to contact position projection, it is assumed that the two steps are laterally equidistant from the central path, i.e., that $||u|| = ||v||$. One lateral direction is negated if necessary, to ensure that the closest points of approach are "between" the contact positions (Figure 4.9). The full process is described in Algorithm 4.3.4.

The CPA process in Algorithm 4.3.4 is applied to each pair of subsequent contact positions, yielding one estimate for the projected positions of the first and last contacts; for all interior contacts, the projected position is calculated as the average of its two corresponding estimates.

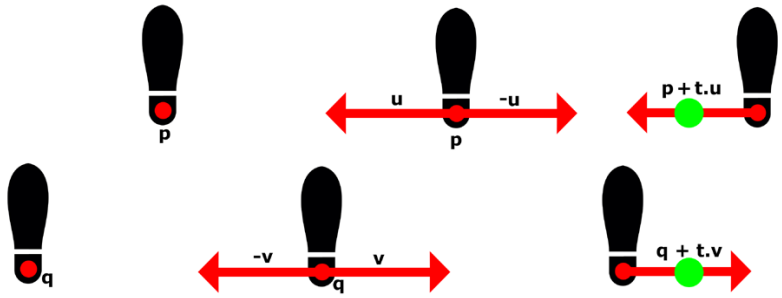

Figure 4.9: Solving for the Closest Points of Approach using contact positions (left) and their corresponding lateral directions (middle) allows the projection of the footprints onto the motion path to be estimated (right).

Algorithm 1 CPA(p, q, u, v)

Require: $||u|| = ||v||$

1: **if** $sgn(u \cdot (p - q)) = sgn(v \cdot (p - q))$ **then**
2: $\quad u \leftarrow -u$
3: **end if**
4: $t \leftarrow \frac{(p-q)\cdot(u-v)}{||u-v||^2}$
5: **return** $(p + t \cdot u, q + t \cdot v)$

Once the positions are determined, a C2 natural cubic interpolating spline can be fit to the projected positions, approximating the motion path. The lateral directions at each contact position are determined from the normal vector at the nearest point along the spline path; the initial lateral directions are determined from a temporary spline which interpolates the midpoints between pairs of subsequent contacts.

This path can be iteratively refined by repeating the process, using more accurate lateral directions from the current path to determine a new path. Though termination is not guaranteed, this procedure has been stable and converged in all uses. A sufficient termination criteria is when the maximum change in lateral direction between iterations drops below a certain threshold; a value of 1 degree seems to work well in practice. For forward paths, the process usually converges after 1 iteration; for more complex paths, 3 iterations is the usual maximum. Some results for finger motion paths are shown in Figure 4.10.

One assumption of this algorithm is that contacts occur in a generally "forward" direction. While it has not been extensively tested on more erratic motions with significant back-and-forth components such as dancing, this presents a problem even during forward motions when double stances occur (Figure 4.11, left). A simple solution to this problem is to "merge" contacts which

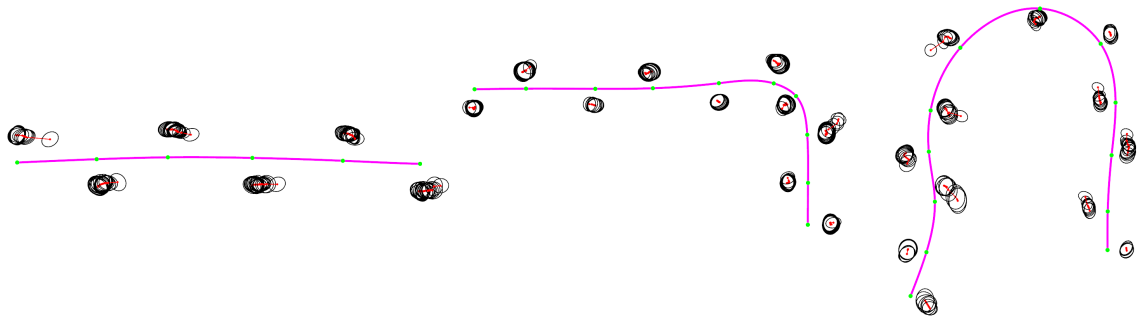

Figure 4.10: The Closest Points of Approach algorithm can quickly determine a path from footprints for a variety of motion types and path shapes, such as forward walking (top left), sharp turns (bottom left), and 180-degree turns (right).

overlap significantly in time into a single contact position at their midpoint, and to constrain the corresponding path position to this point (Figure 4.11, right). Merging contacts when the duration of the overlap is at least 50% of the duration of either contact is a suitable criteria for generating appropriate paths.

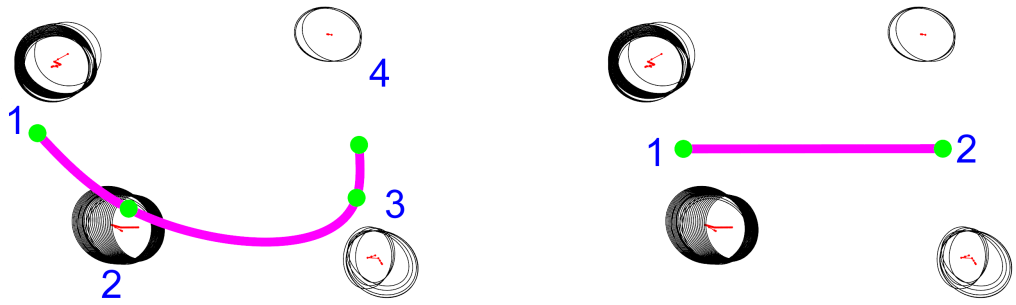

Figure 4.11: Imposing a strict ordering on contacts can generate inappropriate paths for double-stances (left), which is addressed by merging contacts which overlap significantly in time (right).

4.4 Performance Study

To evaluate the finger walking animation system, a second study was conducted with a new set of participants, who performed finger motions and rated the resulting animations.

4.4.1 Methodology

The equipment for this study was identical to the first study (Section 4.2.1). However, one practical difficulty of this study was the need to display an animation to the user which corresponded to their finger motion, and displaying this motion solely on a secondary monitor could make this correspondence difficult for users to evaluate. While an augmented reality system to display the animated character in the space above the surface would have been ideal, it was not practical. Instead, a multi-view approach was adopted: after each performance, the resulting animation was played from a pre-set 3/4 view on the secondary monitor, as well as from an orthographic top-down view on the Surface itself, in spatial correspondence with the performance.

Participants were instructed to use only the middle and index fingers of their dominant hand to pantomime motions on the Surface. After scanning their hands to determine finger length, a first "freeform" stage of trials instructed the participants to perform different locomotion types based strictly on description. The second "directed" stage used pre-set lanes displayed on the Surface combined with locomotion type descriptions on the secondary monitor, to generate motions with particular path shapes. After each finger motion was completed, the output animation was played once automatically, and could be replayed by the participant any number of times before they rated their satisfaction with how well the animation represented their intended motion, on a scale from 1 ("very unsatisfied") to 5 ("very satisfied"). There were 5 cyclic locomotion types (walk, job, run, sneak, march) and 3 non-cyclic (jump up, short jump, long jump), with motion path shapes which were either straight ahead or with a single turn of 45, 90, or 180 degrees.

4.4.2 Results and Discussion

8 new participants (5 male, 3 female, all right-handed) volunteered for this study, and each performed a total of 21 motions (8 freeform, 13 directed). Some performances and resulting animations are shown in Figure 4.12.

Classification accuracy of locomotion type was evaluated afterward, using leave-one-subject-out cross-validation on the feature vectors of the finger walking motions from both studies. A number of standard classification techniques were used: k-Nearest Neighbor (with $k = 1, 3, 5, 7$),

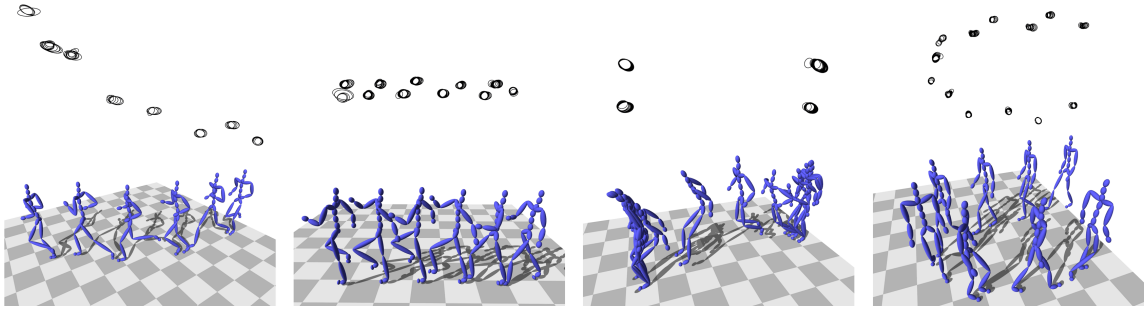

Figure 4.12: Sampled finger contacts (top) and resulting animations (bottom) from separate participants in the performance study. Left to right: running and turning, marching, jumping forward, and a walking u-turn.

Mahalanobis distance (used for motion classification by Yin et al. [125]), and Support Vector Machines using both linear and radial basis function kernels. Figure 4.13 shows the accuracy of classifiers using feature vectors calculated with either absolute or relative units, and direct or path-based contact distances (Section 4.3.2).

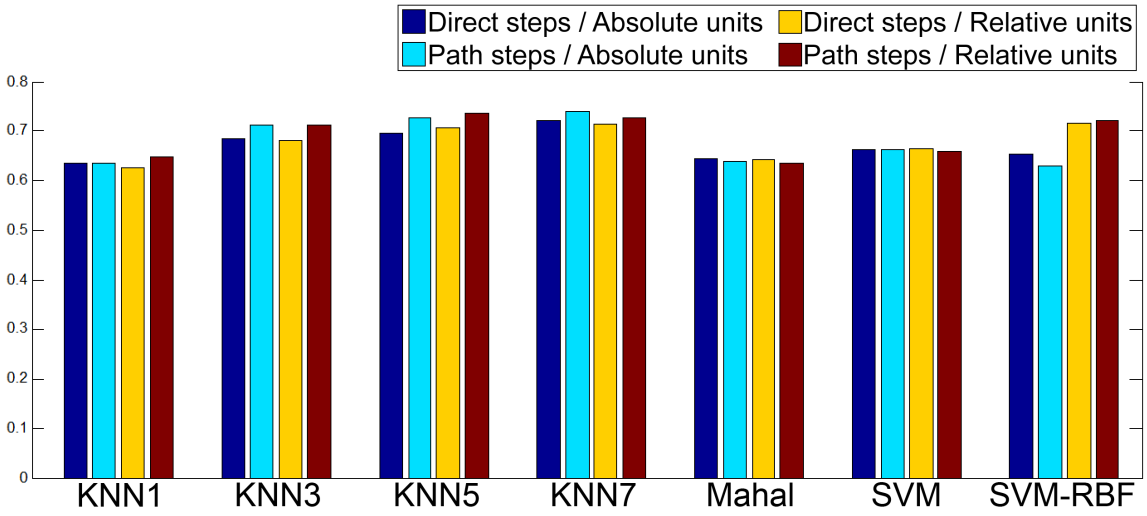

Figure 4.13: Locomotion type classification accuracy for all classifiers and varieties of feature vectors.

Classification accuracy ranged from 62 to 74 percent. Within a particular classifier, accuracy was generally improved by using path-based instead of direct contact-to-contact distances, and relative instead of absolute units, but not by a large margin (typically 1-2%). This demonstrates the usefulness of the selected motion features and data normalization methods for reliable classification, which can only be improved by the application or development of more specialized classification algorithms, which is left to future work.

Locomotion type also had an effect on classification accuracy. Figure 4.14 shows the average and standard deviation of accuracy for each locomotion type across all classifiers and feature vectors. The accuracy is significantly lower for the more "vaguely-named" locomotion types (jog, march, sneak), where greater variation in performance parameters can cause mis-classifications; for example, one user's performance of a march may be very similar to another's walk.

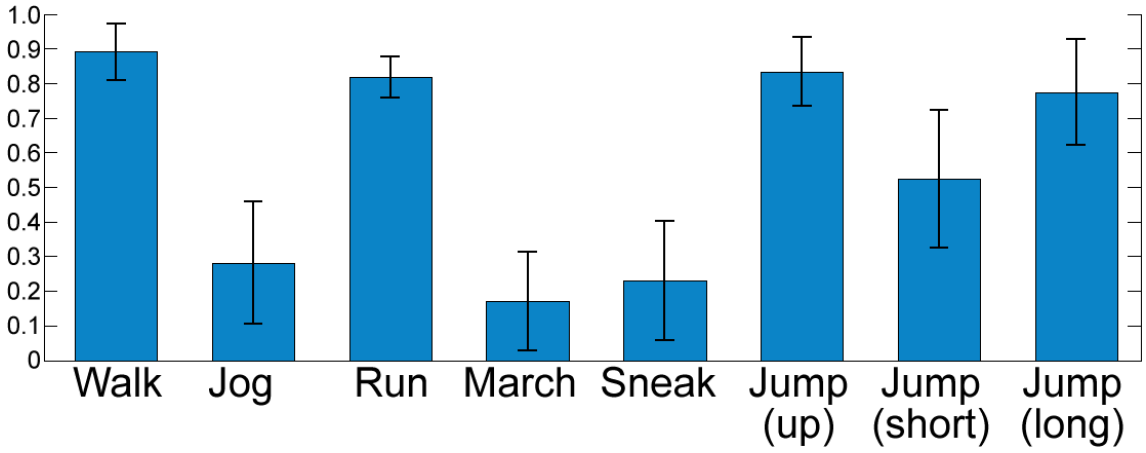

Figure 4.14: Classification accuracy for individual locomotion types, combined from all classifiers and varieties of feature vectors.

Participants rated the generated animations very high in satisfaction, with an average of 4.67 out of 5. Most participants did not comment about any discrepancies between their performances and the animations, indicating that the treatment of the input as gestural was appropriate. However, after the study was completed, one participant remarked that the animations seemed generally faster than their performances, while one other participant said the opposite: that animations seemed generally slower than the performances. To examine the potential discrepancies in timing between the performances and the animations, the ratios of the animation and performance durations and contact frequencies were examined, as shown in Figure 4.15.

Overall, animations were slightly shorter in duration than performances, which could indicate that character scale was overestimated, since a smaller character moving at the same relative speed would take longer than a larger character. The contact rate of the animations was also slightly slower. However, given the high user satisfaction ratings, it may be unwise to alter the animation timing to more closely match; this disparity is mostly accounted for by the tendency to over-perform the contact frequency of running to an unrealistic degree, as observed

in the first study (Figure 4.6).

4.5 Conclusions

This chapter has presented a user-centered approach to the development of an interactive animation system based on finger walking on a touch-sensitive tabletop. An exploratory user study was conducted which identified finger walking as a natural and comfortable method of communicating full-body motion through hand performance. Analysis of this data and its characteristics led to the development of a gestural performance interface using feature-based classification of finger motions and path-based editing of corresponding full-body motions. This system was evaluated in a second user study, which found that motions can be reliably classified using the feature vector, and satisfying animations can be generated to match user performance.

There are limitations to this system which could be addressed by future work. The focus on locomotion allows sufficient expression through surface contact, but additional information such as hand position, orientation, and pose could be very useful for expanding the types or styles of motions which could be recognized.

A system controlled by finger walking could potentially generate motion in a number of ways different from the current canonical motion editing approach. For example, motion class specification and a motion path can be used to splice new motions together using motion

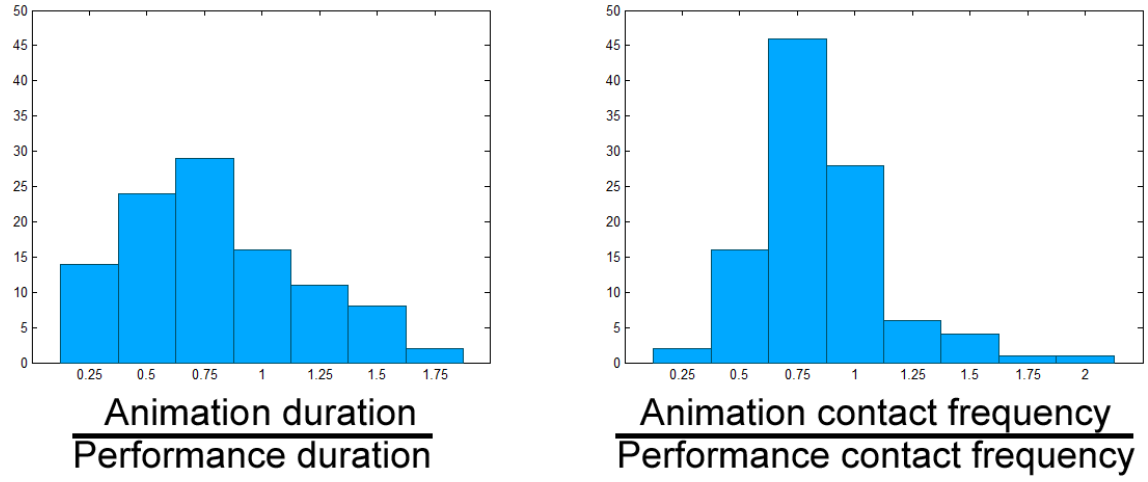

Figure 4.15: Histograms of the ratio between duration (left) and contact frequency (right) for each generated animation and its corresponding performed finger motion.

graphs [93]. Online motion generation is also possible, which would allow the user to adjust their technique during their performance while observing the results.

Finger walking interfaces could also be extended beyond this method. Different types of surface interaction could be studied, to examine, for example, whether finger walking in a "treadmill" style on a handheld screen [105] is similar to freeform finger walking on a touch-sensitive tabletop. Including additional features such as a second hand (perhaps to indicate upper body motion, or additional legs for non-bipeds) or useful props, such as the finger shoes favoured by amateur finger performers, could extend this interface even further.

Chapter 5

Handheld Gestural Editing of Motions

This chapter presents methods for editing full-body motions with by gesturing with a handheld electronic device. To accomplish this, the real-time motion sensor data commonly available from consumer smartphones and tablets was examined during the performance of gestures mimicking full-body motions. Using the identified best form of the motion data, a motion editing system was developed which edits a full-body motion using the difference between an initial "reference" gesture which mimics the original full-body motion, and a new editing gesture which demonstrates the desired change. This system is used to control discrete path-based edits of specifically parameterized locomotion motions. A continuous editing system was also developed, which utilizes continuous tilting gestures to control the steering of a walking animation, which is made cyclical by arranging the editing handles of a short walking animation to ensure continuity between the beginning and end.

5.1 Overview

The performance interface presented in the previous chapter demonstrated a method for users to control the high-level parameters of a full-body motion by moving in real time. This control is achieved through automatic manipulation of the editing handles of a path-based editing system (presented in Chapter 3) to match the user's input. The path-based editing algorithm

is fast enough to produce edited motions with very low latency, allowing multiple iterations of a motion to be produced as rapidly as they can be performed.

One disadvantage of that performance interface is that it requires specialized hardware for the user to interact with. As an alternative, mobile devices such as smartphones and tablets provide an opportunity for performance interfaces, given their increasing computational power and familiarity among many users. Motion sensors in this common hardware can be used for tracking manipulations of the device in space, with a variety of data measured in real-time, such as linear acceleration and rotation.

Two methods were developed for editing motions by performing movements, or gestures, with handheld devices. A method for applying discrete gestures, those with a defined start and end point, was developed for modifying a specially-parameterized and correspondingly discrete full-body motion. The available motion data was investigated to find the best method for recognizing the overall spatial magnitude of a gesture, which is applied to the motion by performing two gestures, one mimicking the current motion and one demonstrating the desired edit. A second method for applying continuous, ongoing gestures was also developed. By decomposing the handheld device's current rotation, steering wheel-like control can be applied to the path of a walking character. Since editing long motions may be too computationally expensive and still ultimately finite, this method was extended to manipulate a short animation clip which cycle with full continuity.

This work makes a number of contributions. An analysis of available motion data from commodity handheld devices was performed, identifying the advantages and disadvantages of each form of data. A discrete editing algorithm was developed, which allowing simple gestures to modify the spatial parameters of a motion based on an approximation of the device's motion through space. Finally, a continuous editing algorithm was also developed, which uses path-based editing to continuously control the turning of a walking character while allowing for the creation of arbitrary motion paths, which can continue indefinitely.

5.2 Motion Sensor Data from Handheld Devices

In recent years, a variety of different types of sensors have become common components in handheld consumer electronics such as smartphones and tablets. Commonly, these devices contain at least a 3-axis accelerometer, a gyroscope, and a magnetometer. The raw signal data from these sensors are usually processed and combined by the device's operating system to produce motion data in a more complete form, for a particular moment in time. This data consists of linear acceleration along each of the device's local axes, rotation from a semi-arbitrary initial world coordinate frame, and rate of rotation in a pre-set rotation order along each of the device's local axes. The local axes of two handheld devices are shown in Figure 5.1.

Figure 5.1: Local device axes for an iPad (left) and iPhone (right).

This motion sensor data can present different challenges. Bias in accelerometer data can result in inaccurate readings, such as indicating motion when the device is stationary. This bias will also accumulate over time, resulting in potentially significant drift in the calculation of device velocity or position if acceleration samples are manually integrated. Rotational data is more stable: while drift is possible, this is mitigated by the operating system by incorporating stabilizing data from other sensors, such as the directions of magnetic north and gravity. However, since the device rotation is presented in an absolute form (as Euler rotations in a pre-set order, a rotation matrix, or a quaternion), decomposing a full rotation to determine a more semantically meaningful rotation, such as solely around an arbitrary axis, requires additional manual calculation.

Utilizing this motion sensor data for performance interfaces also places requirements for the

quality of the data. The data must be smooth, so that it can be mapped to character motion without introducing noise artifacts. The data must also be sufficiently detailed, so that any desirable features within the signal can be reliably identified.

While the format of the motion data may slightly differ depending on the handheld device, it is essentially identical on devices running either the iOS [45] or Android [46] mobile operating systems. For this work, an Apple iPad tablet as well as Apple iPhones were used.

5.3 Motion Editing with Discrete Gestures

In order to determine exactly which motion data from a handheld device to use, and how it should be processed, a fuller plan is needed of exactly how this procedure of motion editing should work. A simple case to begin exploring is the editing of a predetermined motion with a defined beginning and end, which we refer to as a *discrete motion*. A corresponding *discrete gesture* for editing a discrete motion also has a defined beginning and end, and consists of a single or repeated gesture which can be processed and applied to the editing of a parameterized full-body motion. We do not incorporate the selection of a full-body motion by classifying a user's gesture, unlike with finger walking as discussed in Section 4.3.1, though such classification of handheld gestures could be pursued as future work.

While motions can be parameterized in a variety of ways, we seek a parameterization that is as simple as possible, in order to accommodate a correspondingly simple, and thus reliable, gesture. The path-based editing technique described in Chapter 3 is a method which provides a compact set of parameters for editing motion: only the 3D positions of a small set of handles. However, these handles are distributed along the path of a motion and can be used to affect any part of it. To affect only a particular part of a motion, a subset of handles can all be moved together, potentially along a preset vector of displacement. Identifying such a subset of handles to move, and the vector to move them along, would result in an edit with a single degree of freedom, which meets the goal of a simple parameterization.

To control a paramaterized edit of a discrete motion, a more precise method needs to be determined for how the user will provide a discrete gesture. The results of our exploratory study described in Section 4.2 suggest that gestural mimicry of full-body motions is a natural

way for users to express motion. However, the results of the study also suggest that while users may be consistent in their own gestures, the gestures may not accurately represent the timing or spatial parameters of full-body motion, and should be treated as illustrative. In the next section, we explore the motion data available from a handheld device, with the goal of establishing a method to process gestures for motion editing.

5.3.1 Discrete Gesture Data

As discussed in Section 5.2, there is a variety of motion data from handheld devices which could be applied to the processing of discrete gestures. Given the goal of enabling gestures which mimic full-body locomotion, utilizing the data from the accelerometer should provide at least partial information about how a user is moving the device through space, moreso than directly using the rotation information from the gyroscope. The device's current rotation, however, can be used to transform its acceleration into the world coordinate system that the device established upon initialization of the motion sensors. This coordinate system uses a pre-established axis for the vertical direction (i.e., in the opposite direction of gravity), while the horizontal axes are initially arbitrary but tracked by the device with as much stability as possible. Acceleration in a world coordinate system can also be manually integrated once to obtain approximate world velocity, and integrated again to obtain approximate world position. Figure 5.2 shows these values along only the world vertical axis during a "jump up" gesture executed with the device.

There is substantial noise easily observable in the raw acceleration data. While there are some obvious features in the acceleration due to the up-and-down motion, the noise could make identifying those features more difficult, especially during subtler gestures. Conversely, the twice-integrated position data, which should reflect the vertical motion of the device during the gesture, is extremely smooth. However, the position data is dominated by drift introduced from the accelerometers, resulting in a clear quadratic effect which makes any features introduced by actual device motion difficult to identify. This drift is a result of bias in the accelerometers, which can also be observed as non-zero acceleration when the device was stationary at the beginning and end of the gesture. While this bias could potentially be measured and accounted for, it still presents a significant impediment to using integrated position data.

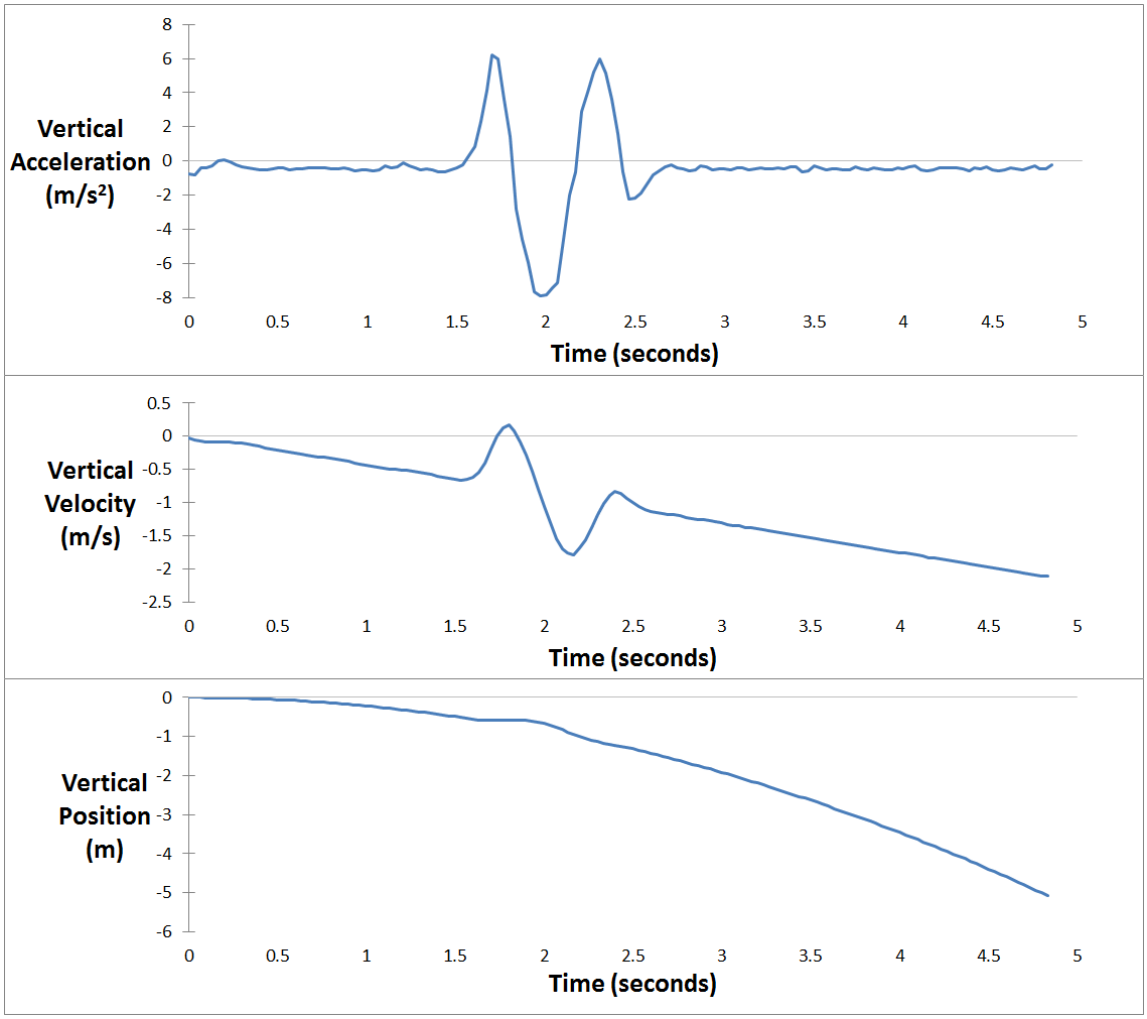

Figure 5.2: Vertical acceleration, integrated velocity, and twice-integrated position during a "jump up" gesture.

Integrating the acceleration once to provide velocity data seems to provide the best option for processing gestures. Unlike acceleration, velocity appears somewhat smooth and well-behaved, but also with identifiable features during the gesture. The aforementioned accelerometer drift results in a linear offset over time due to the single integration, which results in an approximate skew of the velocity data, affecting its shape far less than that of the position data. These qualities make single-integrated velocity the best data for processing discrete gestures.

5.3.2 Processing Handheld Velocity Data

In order to determine a method for processing velocity data during gestures, further investigation into the variation of this data was needed. The vertical velocity during a series

of "jump up" gestures of various heights was recorded, and is shown in Figure 5.3. We refer to this velocity data over time from a gesture as its *velocity profile*.

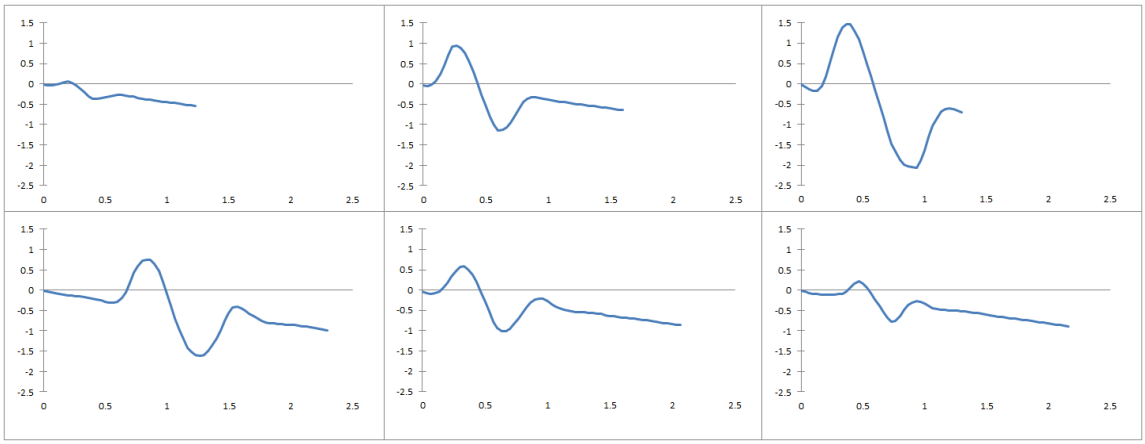

Figure 5.3: Vertical velocity profiles from "jump up" gestures of various heights.

Despite the differences in the heights of the gestures as well as the gesture durations, the shapes of the velocity profiles are very similar, with clear and consistent features in addition to a smooth shape. Unfortunately, these features don't necessarily coincide with intuitive parts of the gesture - for example, the velocity maximum naturally occurs midway through the "upswing", before slowing and becoming negative prior to the apex of the device's trajectory. This makes it difficult to identify exactly which parts of the velocity profile correspond to particular parts of the full-body motion. In addition, the results from our previous exploratory study (Section 4.2) indicate that while users may be consistent in their own methods of expressing motion, these expressions may be more illustrative of full-body motion than an accurate miniature depiction.

Therefore, we propose an approach which utilizes gestures to mimic full-body motion, but which is also example-based, using an initial *reference gesture* which mimics the current motion, and a subsequent *editing gesture* which mimics the desired edit to the motion. With this approach, only the difference between the two gestures needs to be used to edit the full-body motion. This approach is advantageous since it avoids relying on comparing an editing gesture to the corresponding full-body motion, and instead relies only on the user performing gestures in a consistent manner.

Our approach determines a *scaling factor* between the spatial "extent" of the reference and

editing gestures; for example, a scale factor of 2.0 for a "jump up" gesture would indicate that the editing gesture was twice as high as the reference gesture. This scaling factor is calculated from the scaling in both dimensions of an approximate similarity transform between the velocity profiles of the reference and editing gestures. Since the spatial extent of a motion - its position - corresponds to its integrated velocity, the scaling of this extent can be approximated by the product of the scaling in each of the velocity and time dimensions.

Due to the relative simplicity of the velocity profiles, our approach determines the approximate similarity transform from a small number of corresponding feature points identified in the profiles, rather the between the entire curves. The most reliably-identified features of the velocity profiles are the local extrema, and a small set of motion-dependent rules are used to identify corresponding extrema in the profiles of the reference and editing gestures. If any gesture doesn't match these identification rules, it is discarded as invalid and no edit occurs.

In the simple case of two identified features in each gesture, the scaling factor can be determined as follows. The features of the reference gesture, $\mathbf{r}_0$ and $\mathbf{r}_1$ correspond respectively to the features of the editing gesture, $\mathbf{e}_0$ and $\mathbf{e}_1$, with points in a two-dimensional space where the axes are time and velocity:

$$\begin{aligned} \mathbf{r}_i &= (r_{it}, r_{iv}) \\ \mathbf{e}_i &= (e_{it}, e_{iv}) \end{aligned} \tag{5.1}$$

The deltas between each gesture's pair of features, $\Delta\mathbf{r}$ and $\Delta\mathbf{e}$, are the vectors from the first to second features:

$$\begin{aligned} \Delta\mathbf{r} &= \mathbf{r}_1 - \mathbf{r}_0 = (\Delta r_t, \Delta r_v) \\ \Delta\mathbf{e} &= \mathbf{e}_1 - \mathbf{e}_0 = (\Delta e_t, \Delta e_v) \end{aligned} \tag{5.2}$$

The final scaling factor, s, is the product of the ratios between each dimension of the delta vectors:

$$s = \frac{\Delta e_t}{\Delta r_t} \cdot \frac{\Delta e_v}{\Delta r_v} \tag{5.3}$$

This scaling factor represents the change in *spatial extent* between the reference and editing gestures. That same scaling factor can be applied to the current motion to obtain a new motion

with a correspondingly edited extent, using a one-dimensional parameterization of the motion as described in the next section.

5.3.3 Motion Parameterization

The scaling factor determined from the reference and editing gestures provides a compact measure of how to modify a full-body motion to match the gestures: the motion's *extent* should be scaled correspondingly. In order to facilitate this, a motion is prepared for editing by manually specifying a context-specific parameterization, which represents how the entire motion changes relative to an appropriate spatial extent. This parameterization works in addition to the path-based editing method described in Chapter 3, by specifying the positions for all of the editing handles of the motion, depending on the input parameter.

While the exact parameterization is manually determined depending on the motion, the general method of parameterization and how it is modified by the gestural scaling factor is consistent. After the path-based editing method is initialized on the motion, n editing handles are detected, with initial positions $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n$. The parameterization is a set of functions $\hat{\mathbf{h}}_i(t)$, each of which determines the new position of a handle based on the scalar parameter t:

$$\hat{\mathbf{h}}(t) = \left\{ \hat{\mathbf{h}}_1(t), \hat{\mathbf{h}}_2(t), \ldots, \hat{\mathbf{h}}_n(t) \right\} \tag{5.4}$$

When the user performs a reference gesture, we refer to the state of the motion that they are mimicking as the *current motion*, which is the result of the current parameter value $\bar{t}$. Once the editing gesture is performed, a scale factor s has been determined as described in Section 5.3.2. To scale the extent of the current motion, the new edited motion is determined from a new parameter $\hat{t}$ which is the product of the scaling factor and the current parameter, i.e.:

$$\hat{t} = s \cdot \bar{t} \tag{5.5}$$

For all but the first edit to a motion, the current parameter $\bar{t}$ is already known since it was the result of a previous edit. However, for the first edit, no parameter has been specified, as the motion is simply in its initial unedited state. This requires the parameterization to be inverted, to determine the parameter from the initial motion state. An iterative or

approximating approach is not necessary; instead, the initial parameter is directly calculated when the parameterization is determined.

The exact method of motion parameterization, and the velocity profile feature templates required to calculate the scaling factor, necessarily vary depending on the specific motion and the gesture used for editing. The following sections describe the details for three gestural editing scenarios: editing the height of a jumping motion, editing the distance of a jumping motion, and editing the step length of a walking motion. All of these use manually-specified motion parameterizations and velocity profile features, though methods to determine those automatically could be investigated as future work.

5.3.4 Editing Jump Height

Since the example of gestural editing for a jumping motion was used during the investigation described in Sections 5.3.1 and 5.3.2, the final method for editing the height of a jumping motion is very similar. During the reference and editing gestures, the device's world vertical velocity is used for the velocity profiles. To identify the features in each profile, the maximum and minimum of all local extrema are determined. The jumping gesture is recognized if the maximum occurs first, and is immediately followed by the minimum with no intervening other local extrema. In our testing, this is sufficient to recognize a single smooth up-and-down jumping gesture with good accuracy. Once identified, the corresponding pairs of features are used to calculate the scaling factor as described in Section 5.3.2.

To edit the height of a jumping motion, only the height of the vertical editing handle (Section 3.3.2) in the jump needs to be modified; leaving the other handles unmodified will maintain the placement of the character during take-off and landing. A straightforward parameterization to enable the editing of the height of jump is to use the height of the jump itself as the editing parameter. However, the height of a jump should not be measured as the absolute height of the handle from the ground plane. This is because any edit that produces a jump of near-zero height should result in a character which moves almost entirely horizontally, rather than towards the ground plane, where the absolute height would be zero.

Therefore, the jump is parameterized such that a jump height of zero results in the vertical editing handle being positioned to approximately interpolate the neighbouring handles. A plane

is fitted to the positions of these neighbouring handles, with a normal as parallel as possible to the vertical axis. A strictly vertical normal (and thus, a level plane) will not be possible if the heights of the surrounding handles are different, though they are often close since the poses which generate the handles are similar. The parameterized height of the vertical handle is calculated as its vertical displacement from this plane, resulting in a zero-height position which interpolates the heights of the surrounding handles.

The parameterization is thus calculated as follows: we assume that the vertical editing handle is the j-th handle, and that its vertical projection onto the aforementioned plane occurs at point $\mathbf{b}$. Therefore, the functions which determine new editing handle positions parameterized by jump height are:

$$\hat{\mathbf{h}}_i(t) = \begin{cases} \mathbf{h}_i & \text{if } i \neq j \\ \mathbf{b} + t \cdot (0, 1, 0) & \text{if } i = j \end{cases} \tag{5.6}$$

This results in only the position of the vertical editing handle being modified as a result of the parameterization; all other handles remain at their initial positions. As per Equation 5.4, this fully defines all editing handle positions and can be automatically modified by the scaling factor s after the reference and editing gestures are complete.

Figure 5.4 shows an example of this technique for editing jump height. The features within the velocity profiles of the reference and editing gestures are determined, which results in the calculation of a scale factor of 2.98. This scale factor is applied to the initial jump height of 23cm, resulting in a new jump height of 68cm.

5.3.5 Editing Jump Distance

The procedure for editing a jump's distance is similar to editing jump height. The reference and editing gestures are two "forward jump" gestures demonstrating the current and desired distance of the jump. While the vertical velocity of the handheld device is relevant in these gestures, the horizontal velocity is also important, as it demonstrates the horizontal distance of the gesture. However, since the device's world horizontal axes can be oriented arbitrarily, the gesture's motion may occur along a combination of those axes. Therefore, the absolute

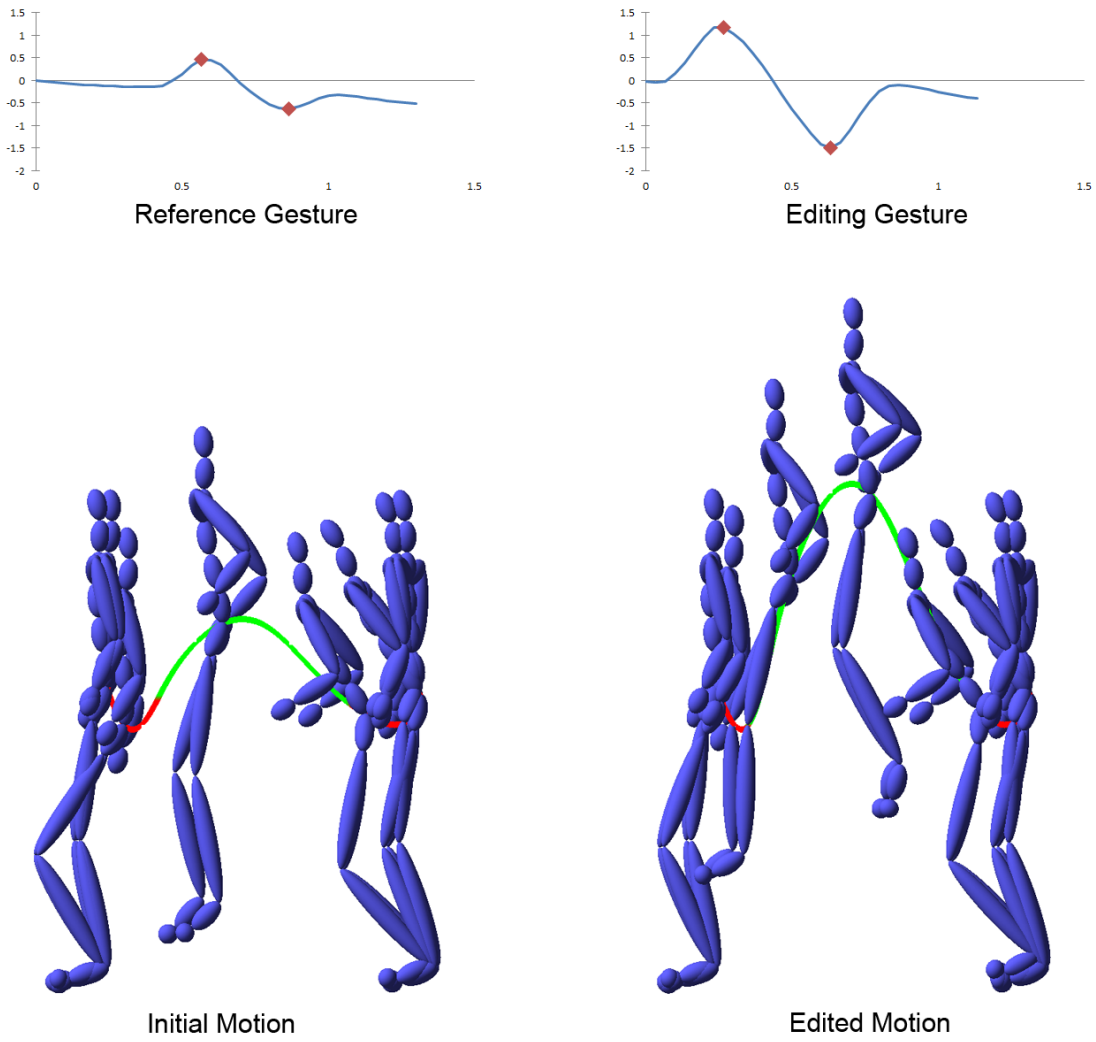

Figure 5.4: Velocity profiles with identified features and the corresponding motions for the reference gesture (left) and the editing gesture (right).

magnitude of the horizontal velocity is used:

$$\|\mathbf{v}_{horiz}\| = \sqrt{\mathbf{v}_x \cdot \mathbf{v}_x + \mathbf{v}_z \cdot \mathbf{v}_z} \tag{5.7}$$

Reducing the horizontal velocity to a non-negative scalar in this manner eliminates the problem of considering the device's direction relative to the horizontal axes, and maintains the velocity profile as a one-dimensional function over time. This does remove any information about the direction of the movement altogether, which would make distinguishing between "backward" and "forward" periods of a gesture impossible. We assume that the gesture consists of primarily forward motion along some direction in world space, which seems reasonable for

gestures which demonstrate a forward jump, and thus can safely utilize the absolute horizontal velocity.

However, the horizontal velocity profile during forward jump gestures may not contain enough data to enable comparisons between different gestures. Figure 5.5 shows the horizontal velocity profiles from four different "jump forward" gestures of increasing distance, along with the corresponding vertical velocities. In these examples, the differing magnitude of the peak horizontal velocity offers a clear indication of the magnitude of the gesture, but the smoothness of the velocity over time makes determining exact correspondences between gestures difficult. Conversely, the vertical velocity has useful features, as utilized in the previous section to edit jump height, but the magnitude of the gesture's horizontal distance would be unsurprisingly difficult to determine from this strictly vertical data.

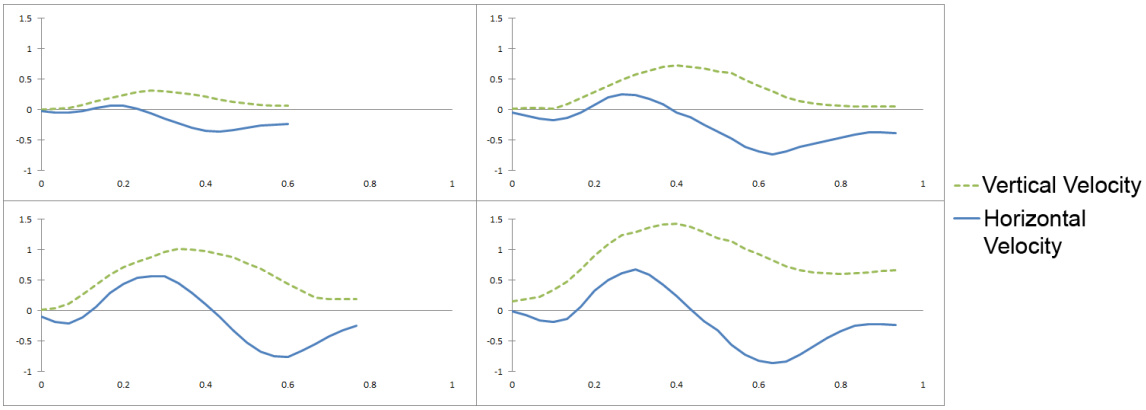

Figure 5.5: Horizontal and vertical velocity profiles of four different "jump forward" gestures of varying distances.

Therefore, to determine the gestural scaling factor for jumping distance edits, aspects of both velocity profiles are used: the magnitude of the horizontal velocity, and the timing of the vertical velocity. Exact times of features within the vertical velocity are determined with the same method as for jump height gestures (Section 5.3.4). As per Equation 5.1, we denote these pairs of feature times from the vertical velocity as r_{0t} and r_{1t} for the reference gesture, and e_{0t} and e_{1t} for the editing gesture.

The vertical velocity values at these features are unused; only their times are relevant for the calculation of the scaling Similar to Equation 5.3, the scaling factor is determined by the product of the ratio of the duration between each pair of features with the ratio of the maximum

horizontal velocity in those durations from each gesture:

$$s = \frac{\Delta e_t}{\Delta r_t} \cdot \frac{\mathbf{max}\{e_{v\_horiz} \in (e_t, e_{v\_horiz}) : e_{0t} \leq e_t \leq e_{1t}\}}{\mathbf{max}\{r_{v\_horiz} \in (r_t, r_{v\_horiz}) : r_{0t} \leq r_t \leq r_{1t}\}} \tag{5.8}$$

The parameterization for this type of editing defines the parameter as the horizontal distance between the editing handles prior and subsequent to the vertical editing handle. The parameterization is defined as follows. We again assume that the vertical editing handle is the i-th handle. The initial jump displacement $\mathbf{d}$ is the horizontal vector between the initial positions of the neighbouring handles, i.e.,

$$\mathbf{d} = (\mathbf{h}_{(j+1)_x} - \mathbf{h}_{(j-1)_x}, 0, \mathbf{h}_{(j+1)_z} - \mathbf{h}_{(j-1)_z}) \tag{5.9}$$

Let $\mathbf{b}$ be the point closest to the initial position of the vertical editing handle, $\mathbf{h}_j$, along the line parallel to $\mathbf{d}$ which passes through $\mathbf{h}_{j-1}$. The scalar l is the proportion of the vertical editing handle's projected distance along the jump displacement:

$$l = \frac{\|\mathbf{b} - \mathbf{h}_{j-1}\|}{\|\mathbf{d}\|} \tag{5.10}$$

The final parameterization is then:

$$\hat{\mathbf{h}}_i(t) = \begin{cases} \mathbf{h}_i & \text{if } i < j \\ \mathbf{h}_{j-1} + t \cdot l \cdot \frac{1}{\|\mathbf{d}\|} \cdot \mathbf{d} + (\mathbf{h}_j - \mathbf{b}) & \text{if } i = j \\ \mathbf{h}_i + (t - \|\mathbf{d}\|) \cdot \|\mathbf{d}\| & \text{if } i > j \end{cases} \tag{5.11}$$

This results in all editing handles prior to the vertical editing handle, i.e., the "lift-off", remaining unchanged. The vertical editing handle is moved to retain its proportional distance along the new jump displacement vector. Its vertical and lateral offset from that vector is also unchanged. Finally, all handles after the vertical editing handle are rigidly translated either "forward" or "backward" depending on whether the new jump distance t is larger or smaller than the original jump distance $\|\mathbf{d}\|$.

Figure 5.6 shows an example of this technique for editing jump distance. The features within

the velocity profiles of the reference and editing gestures are determined, which results in the calculation of a scale factor of 1.77. This scale factor is applied to the current jump distance of 88cm, resulting in a new jump distance of 156cm.

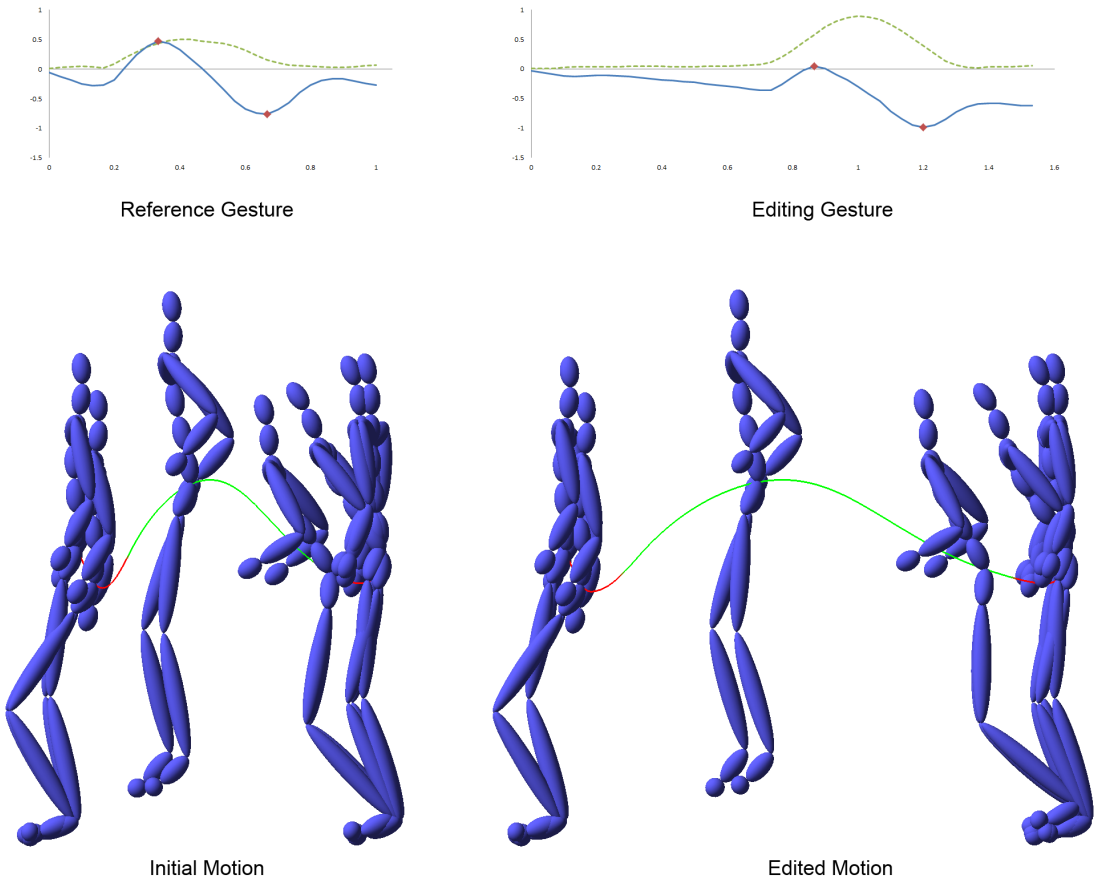

Figure 5.6: Velocity profiles with identified features and the corresponding motions for the reference gesture (left) and the editing gesture (right).

5.3.6 Editing Walking Stride Distance

With some modifications, these methods can be applied to the editing of non-jumping motions as well. To edit the stride distance of a walking motion, a "forward walk" gesture is used, similar to the forward jump gesture used to edit jump distance as described in the previous section. However, a gesture which demonstrates forward walking should allow multiple "steps" instead of a single jump. Furthermore, we seek a method which doesn't require direct correspondence between each step in the reference and editing gestures. Avoiding this relieves the user of

needing to perform the same number of steps and allows concentration on the performance instead.

For this type of editing, both the horizontal and vertical velocity profiles are used, as with editing jumping distance as described in the previous section. However, instead of identifying a single pair of features within each profile, the longest continuous sequence of such pairs is identified. Each pair signifies a "hop" in the gesture which corresponds to a performed stride.

The scaling factor is calculated similar to Equation 5.8, using the average of the calculations on all "hops" in each gesture. However, Equation 5.8 cannot be applied to corresponding pairs of hops between the two gestures and then simply averaged, since the number of hops in each gesture can differ. Therefore, instead of an average of ratios, the calculation is structured as a ratio of averages:

$$s = \frac{\frac{1}{m} \cdot \sum_{i=1}^{m} \left((e_{(2i)_t} - e_{(2i-1)_t}) \cdot \mathbf{max}\{e_{v\_horiz} \in (e_t, e_{v\_horiz}) : e_{(2i-1)_t} \leq e_t \leq e_{(2i)_t}\} \right)}{\frac{1}{n} \cdot \sum_{j=1}^{n} \left((r_{(2j)_t} - r_{(2j-1)_t}) \cdot \mathbf{max}\{r_{v\_horiz} \in (r_t, r_{v\_horiz}) : r_{(2j-1)_t} \leq r_t \leq r_{(2j)_t}\} \right)} \quad (5.12)$$

Similar to the method for editing jumping distance, stride distance in a walk can be edited by parameterizing the walk by its total walking distance, which we measure as the horizontal distance between the first and last editing handles. This parameterization based on total distance, rather than attempting to identify potentially-varying stride distance within the motion itself, is advantageous since it ignores partial strides which may be truncated by the start or end of the motion clip.

Let $\mathbf{d}$ be the horizontal vector between the initial positions of the first and last editing handles, $\mathbf{h}_1$ and $\mathbf{h}_n$, and $b(\mathbf{h}_i)$ the point closest to the initial position of the i-th handle along the line parallel to $\mathbf{d}$ which passes through $\mathbf{h}_1$. Then the parameterization is as follows:

$$\hat{\mathbf{h}}_i(t) = \begin{cases} \mathbf{h}_i & \text{if } i = 1 \\ \mathbf{h}_1 + t \cdot \frac{\|b(\mathbf{h}_i) - \mathbf{h}_i\|}{\|\mathbf{d}\|} \cdot \frac{1}{\|\mathbf{d}\|} \cdot \mathbf{d} + (\mathbf{h}_i - b(\mathbf{h}_i)) & \text{if } i \neq 1 \end{cases} \quad (5.13)$$

This results in the first editing handle remaining stationary, and all other handles being "stretched" along the direction of the walking vector. An additional post-processing step adjusts the height of all editing handles depending on the amount of stretching, to ensure that the

character's feet can still remain stationary during footplants; i.e., a longer stride necessitates a lower stance. The height adjustment is a piecewise linear function of stretching amount that was manually determined in advance, though an automated method could be developed as future work.

Figure 5.7 shows an example of this technique for editing stride distance. The features within the velocity profiles of the reference and editing gestures are determined, which results in the calculation of a scale factor of 1.75. This scale factor is applied to the current walking motion with a total distance of 3.36m, resulting in a new walk with a total distance of 5.89m.

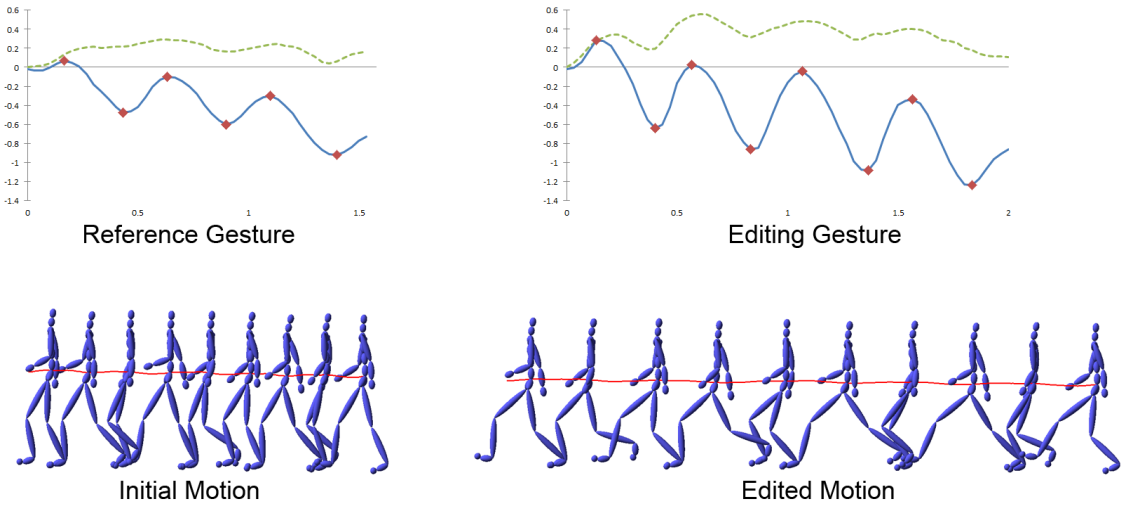

Figure 5.7: Velocity profiles with identified features and the corresponding motions for the reference gesture (left) and the editing gesture (right).

5.4 Motion Editing with Continuous Gestures

While the method described in the previous section for motion editing with discrete gestures can be applied to a variety of motions, there are scenarios where such a method wouldn't be as effective. More complex motions with subtle performance aspects may require more precision to edit than the velocity-based "extent" editing can provide. Longer motions could require complex and correspondingly lengthy gestures for editing, or potentially a sequence of gestures performed separately, which could still prove complicated to perform.

Any gestural editing method which has the requirement that gestures are completed prior to displaying the results has the drawback of delaying feedback to the user, and this delay

necessarily increases for longer motions. While this style of interaction may be more appropriate for editing discrete and static aspects of a motion, ongoing motions can instead be edited using *continuous gestures* which affect the motion during playback, in real-time. This method of gestural editing can affect an aspect of the motion which can vary continuously over time, giving the user instant feedback. The method of motion parameterization must change as well: while discrete gestural editing can utilize a motion parameterization which strictly depends on aspects of the gestures, editing with continuous gestures requires a parameterization which also depends on the preceding state of the motion. Without taking at least part of the history of the continuous edit into account, varying any gestural parameter could produce a motion that might not appear consistent with the motion up to that point, which was generated with a different gestural parameter.

We address these issues by developing a continuous gestural editing technique for controlling the direction of a walking motion, demonstrated in Figure 5.8. As with the scenarios of discrete gestural editing, this is a case of a relatively simple motion with a single spatial degree of freedom. The following sections discuss how the user's continuous gestural control is determined, how this control is applied in a history-dependent manner to produce consistent motion during editing, and how this technique is extended to a cyclic motion of arbitrary duration.

5.4.1 Continuous Steering Control

For continuous gestural control of the direction of a walking motion, there is a simple and familiar model of how the user can specify the direction: a steering wheel. When the device is held level, the character will proceed straight ahead, and when the device is tilted to either side, the character will turn to the appropriate side with the angle of the turn corresponding to how far the device is tilted. This "steering angle" of the device can be measured solely instantaneously, without considering any preceding measurement or its rate of change, if the parameterized motion will ensure continuity of the character's pose as the motion progresses.

The instantaneous steering angle can be determined from the device's current rotation, without the measuring the device's linear acceleration or rotation. This avoids potential drift introduced by integrating non-instantaneous motion sensor data over time, as was demonstrated in Section 5.3.1. We measure steering angle as the device's rotation around an axis normal

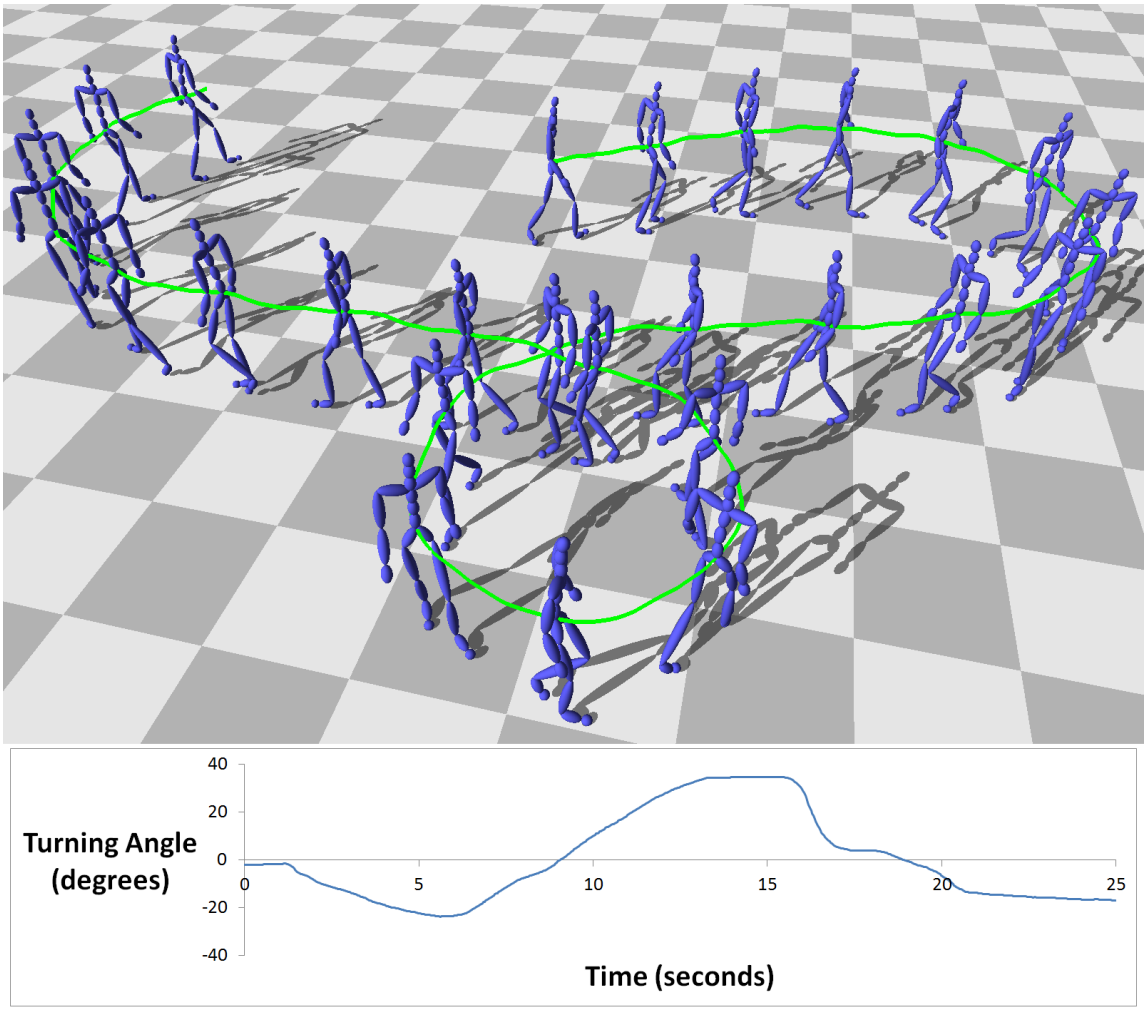

Figure 5.8: A novel motion generated with continuous gestural control of the motion's turning angle in real-time.

to its display, which allows the "steering column" to be oriented however the user is most comfortable. This means that the measuring the steering angle should compensate for both the facing direction of the user, as well as the device's forward-backward tilting. Steering angle should also be robust to the device's screen orientation, which determines which direction on the device is "up" in screen-space. These requirements are illustrated in Figure 5.9.

As discussed in Section 5.2, the instantaneous rotation data available from the device consists of only a final rotation matrix or quaternion, or the three component rotations of that final rotation, which are around each world axis and applied in a pre-specified order. However, this means that the device cannot provide the steering angle, which is a rotation of the device around a local axis normal to the display surface, which must be measured after the other

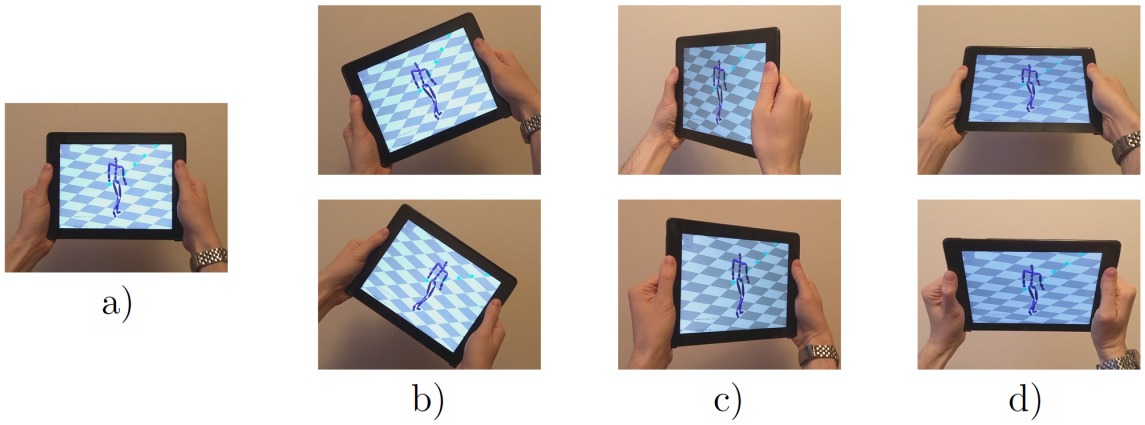

Figure 5.9: Starting with a neutral "steering wheel" grip of the device (a), steering angle is measured as the rotation left or right around an axis normal to the display (b). Rotating the device around a vertical axis (c) or tilting up or down (d) should have no effect on the measured steering angle.

two rotational degrees of freedom have been applied. Therefore, the steering angle must be manually calculated.

Our method for determining the steering angle from the current device rotation is as follows. We define the steering angle of a handheld device as the difference between the device's current rotation and a "level" version of that rotation, which is constructed to have a steering angle of zero. This level rotation is obtained by applying the shortest rotation around an axis normal to the device's display which would make the device's screenspace horizontal axis parallel with the "world" ground plane. Since there are potentially two such rotations differing by 180 degrees, the rotation is chosen which would also make the device's screenspace vertical axis as close as possible to the world up direction.

Using this method, the steering angle can be calculated for any instantaneous rotation of the device, with the exception that it is undefined when the device's display is parallel to the ground plane. To avoid this, when the device's rotation is sufficiently close to this state, the steering angle is forced to be zero. While this could cause potential discontinuities in the steering angle over time as the device is manipulated, in practice we have not noticed any artifacts in the final motion resulting from this safeguard.

5.4.2 Parameterizing Turning in a Walking Motion

A parameterized motion, similar to those used for discrete gestural editing (Section 5.3.3), forms the basis of the motion that will be generated with continuous control. The parameterization takes the scalar steering angle as input, and modifies a straight-ahead walking motion to have a constant turning rate by arranging the handles along an approximate circular arc. The entire parameterized motion is modified in real-time as the steering angle changes based on how the user manipulates the device.

A simple geometric approach is used to map the steering angle to the arrangement of the editing handles of the parameterized motion. Equal successive rotations around a vertical axis are applied at each editing handle, so that the direction of each segment (formed between pairs of editing handles) differs from that of the previous segment by an angle equal to the steering angle multiplied by a constant factor. The distance between successive editing handles remains unchanged, and thus the handles "bend" as if a rigid chain.

We have found through informal experimentation that bending the path between successive handles at the same angle as the steering angle provides good results for the chosen walking motion, which results in a significant range of possible turning values by manipulating the device, while still maintaining precision. Figure 5.10 shows the results of this path bending for a variety of steering angles.

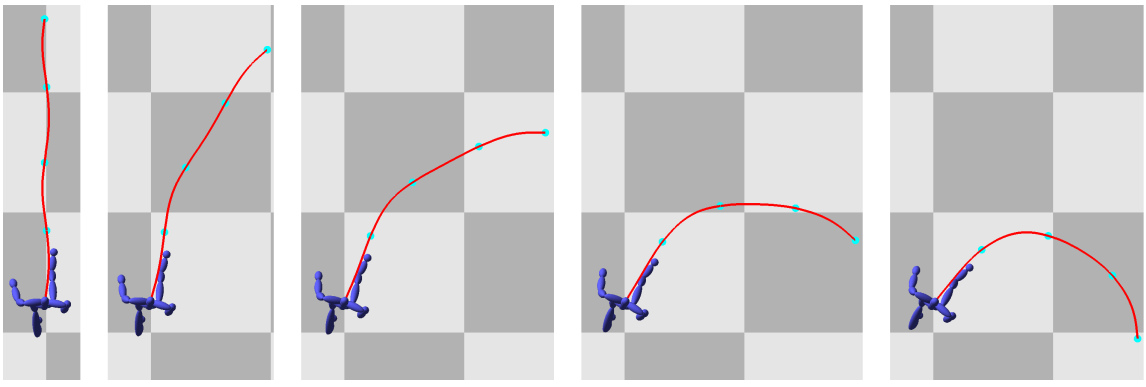

Figure 5.10: A short walking motion with successive handles bent by interactive steering angles of 0, 10, 20, 30, and 40 degrees.

However, updating all of the editing handles during playback has a significant drawback. Since differing handle arrangements produce different motions, the pose displayed from the

current motion may not be consistent with the pose displayed on the previous frame, from an even slightly different motion. This lack of consistency during editing is addressed by the technique in the following section.

5.4.3 Online Continuous Editing with Steering Control

Our goal in generating user-driven motion in real-time is to develop a method which is flexible enough to respond quickly and accurately to user input, but robust enough to generate smooth and consistent motion without apparent artifacts or errors. However, a straightforward application of the handle-based path editing technique will not meet both of these criteria simultaneously. By moving the editing handles of a motion during playback as described in the previous section, the path direction could be precisely updated as the user's steering angle changes, but the character could slide side-to-side or otherwise move impossibly. Conversely, attempting to represent the path defined by the user's steering over time with a static arrangement of handles could generate a single consistent motion, but this would require arranging the handles to match a completed gesture, and thus would sacrifice responsiveness.

To overcome these drawbacks, our approach uses the handle-driven path-based editing method to generate a sequence of poses in response to user input, but arranges the poses during playback with more flexibility than can be provided by high-level path manipulation. "Streaming" the poses in this manner allows arbitrary path shapes to be generated which precisely match the user's steering, which controls the derivative of the the *overall* path shape through the editing handles. This approach also retains higher-frequency motion, such as hip sway, from the unedited motion, which would be eliminated if the character's root were constrained to precisely follow the steered path.

This is accomplished by treating the parameterized motion from the previous section as a secondary "sidecar" motion which controls the pose as well as the relative, rather than absolute, translation and rotation of the primary displayed skeleton. At each timestep, the editing handles of the sidecar motion are re-arranged to form a turning motion according to the steering angle, as previously described, and the pose of the skeleton from the sidecar motion at the new time is applied to the primary skeleton, except for the skeleton's root.

The primary skeleton's new root position is obtained by applying the same relative

translation between the sidecar skeleton's transformations at the previous and current times. The primary skeleton's new root rotation is equal to the rotation of the sidecar skeleton at the current time, but rotated around a vertical axis so that the relative change in the skeleton's forward facing direction — its heading — from the previous time is the same as the relative change in heading for the sidecar skeleton. We have found that using the sidecar skeleton's rotation in this fashion, rather than simply applying the relative rotation between the sidecar skeleton's previous and current rotations, prevents numerical errors from accumulating by repeated application of relative rotations.

This process for a single timestep is shown in Figure 5.11.

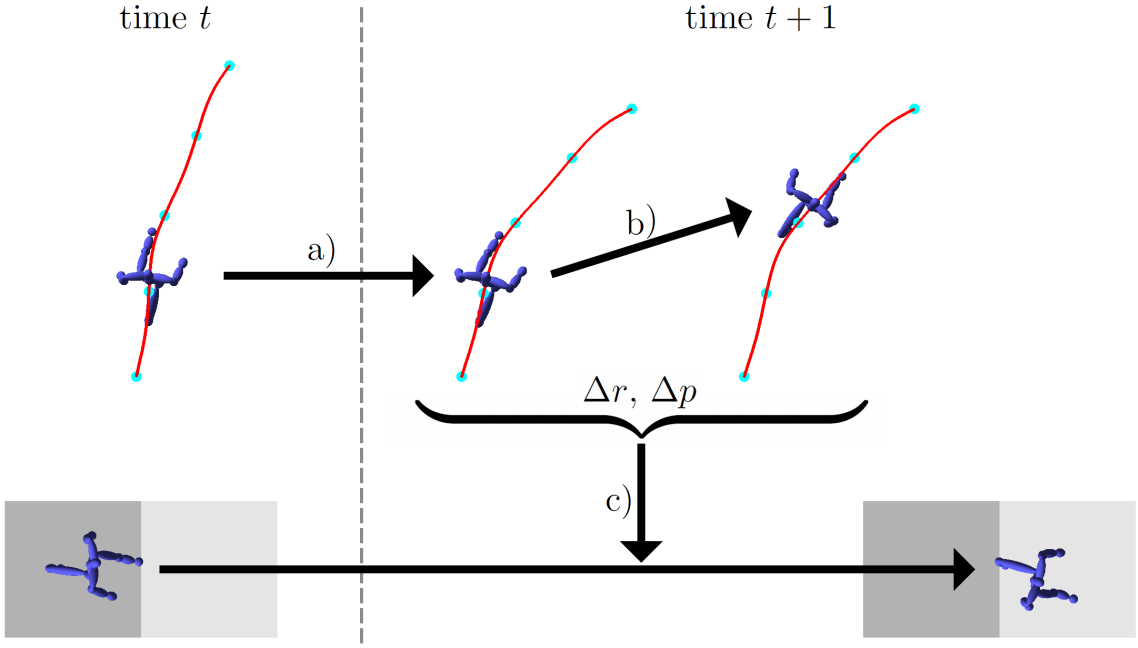

Figure 5.11: To determine the new primary pose during a continuous edit, the editing handles of the sidecar motion are first updated (a). Then the time within the sidecar motion is advanced (b), and the relative difference in position and heading direction are applied to the previous primary pose (c).

5.4.4 Online Continuous Motion Editing with Cycling

The method described in the previous section generates consistent motion directed by the continuous steering control of the user, but is ultimately limited by the length of the original motion. Unfortunately, since the cost of the numerical optimization in the path-based editing algorithm increases as the size of the edited motion increases, extending the sidecar motion

by a significantly large amount would prevent the path-based editing from operating at an interactive rate.

However, the cyclic repetition in a motion such as forward walking means that edits to a small representative motion can be applied repeatedly, to produce a streamed motion very similar to the results of editing a large motion composed of many cycles. In the case of forward walking, the cyclic portion of the motion is two forward steps, one left and one right.

Thus, to enable the generation of motions of arbitrary length, the sidecar motion is a "cyclified" clip from a forward walking motion, with the same skeleton pose at the first and last frames. The method described in the previous section then only needs to be modified to loop from the end of the sidecar motion back to its beginning. To accomplish this, whenever a timestep would require a pose from the sidecar motion beyond the last frame, the method conceptually splits this timestep in two parts. First, the sidecar motion is stepped forward to exactly the last frame, and then "rewound" to the first frame. Then, the remaining timestep is taken and the procedure continues as before.

For efficiency, it is desirable to have a sidecar motion which is as short as possible. However, a sidecar motion consisting of the smallest repeatable part of the motion, two forward steps, will not generate consistent motion. This is due to the path-based editing technique's behaviour at the endpoints of an edited motion; because of the as-rigid-as-possible deformation applied to the motion path, even during a motion which has been uniformly turned to produce a circular walk, the facing direction of the character at the endpoints relative to the overall motion path will not be the same as on interior points (see, for example, Figure 3.6).

Therefore, the final sidecar motion is a cyclified walk with four steps, and the actual cycled portion of the motion is in the interior two steps, with the first and last steps providing continuity in the character's facing direction. The difference between using the entirety of a two-step sidecar motion and the interior two steps of a four-step motion is shown in Figure 5.12.

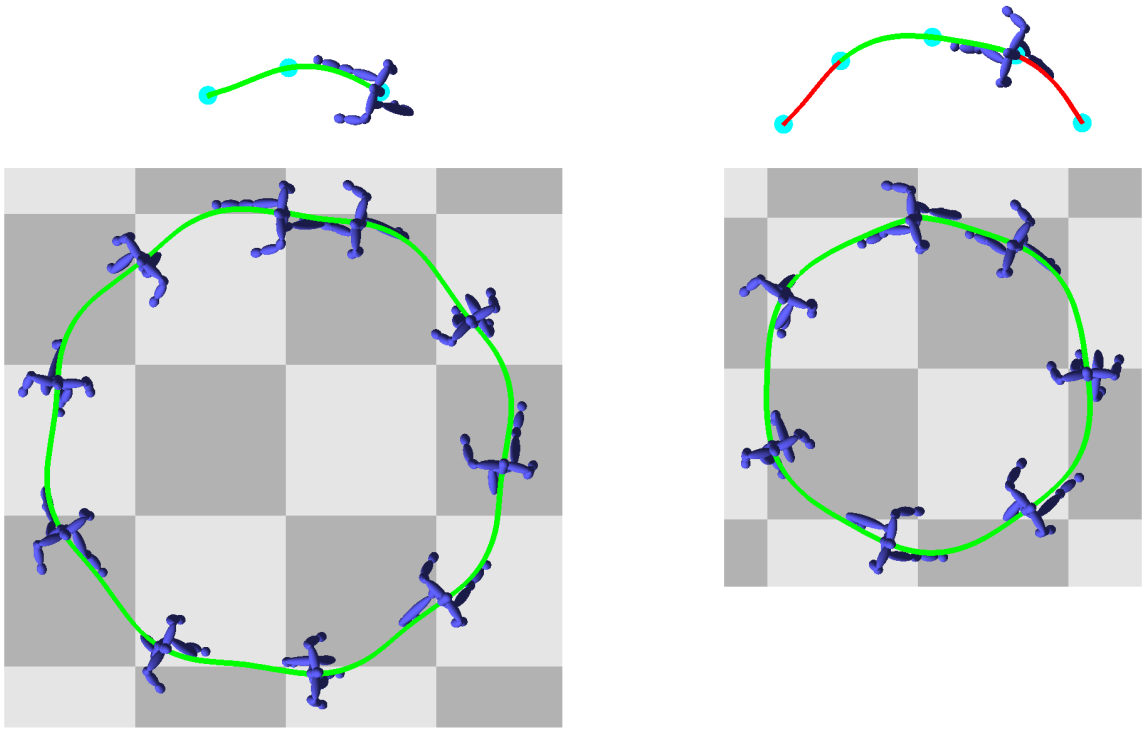

Figure 5.12: A two-step sidecar motion (top left) results in a path shape that does not appear smooth when repeated (bottom left). Using the interior two steps of a four-step motion (top right) results in a tighter, smoother path when repeated (bottom right).

5.5 Conclusions

This chapter has presented two methods for editing of motions by gesturing with handheld electronic devices. A method for discrete editing of motions identifies features in the velocity profile of the device during both an initial reference gesture and subsequent editing gesture. The difference between these gestures is applied to a variety of motions which have been specifically parameterized using a path-based editing algorithm. A second method for continuous editing of motions reformulates the rotation of the handheld device to allow orientation-independent steering wheel-type control by the user. This allows continuous steering control of a cyclical walking animation, which has continuity during cycling enforced by careful automatic arrangement of the path editing handles.

There are limitations to these methods. There can be noticeable noise as well as a lack of precision in handheld gestures, which can be caused by either the device's sensors, instability in the user's gestures, or a combination thereof. Different methods for tracking

the device's movement through space, such as computer vision-based approaches using the onboard camera(s), might provide more precise information, or at least another source which can augment the acceleration data used in the discrete gestures. Additional data about device position could be useful for continuous gestures as well; though there are potential advantages to directly manipulating the device which displays an online motion edit, our steering control is still one-dimensional and similar to other control methods such as a joystick. Incorporating accurate device position could allow many different types of continuous gestures; an exploratory study could reveal how to best use an oriented device moving through space to represent full-body motion.

Another limitation of the discrete gestural control is that the corresponding full-body motions must be specially parameterized to match the particular type of gesture being recognized. While this preparation is not particularly labour-intensive, an automatic approach for parameterizing motions would increase this method's flexibility. For example, a method similar to that of Dontcheva et al. [23] could allow the performance of the reference gesture to be used to identify which aspects of the full-body motion should be modified by the editing gesture.

Finally, a system which unifies the use of discrete and continuous gestures by seamlessly switching between those methods could be useful for generating full-body motions with more variety. It is possible to allow continuous control of a variety of cyclic motions, and then to transition that motion to a parameterized discrete motion when a correspondingly discrete gesture is detected. This could also allow the use of discrete gestures for selection of various motion parameters, for example, a single discrete gesture could be used to signify the change from a continuous walk to a continuous run.

Chapter 6

Conclusion

In this chapter, we first review the capabilities of the techniques presented in this thesis, and discuss some potential contemporary applications. We then discuss some limitations of the techniques and potential future work to address those limitations. We conclude with a summary of the contributions of this work and a discussion of some emergent themes.

6.1 Capabilities and Applications

In this thesis, we have presented a collection of techniques for interactive high-level editing of the spatial and timing parameters of animated human locomotion. The foundation of this work is a path-based editing technique which works quickly, identifies few but meaningful spatial controls for the user to edit, and based on simple biomechanical and physical rules automatically edits a variety of locomotive motions to satisfy user input. This technique was extended by two additional techniques which utilize alternative methods of input. Based on how users express full-body motion with their hands, an interface for using finger walking on a touch-sensitive table was developed, which can identify motion type as well as control path edits from a single input user performance. Using commodity mobile devices familiar to many users, techniques using the gestural motion of the devices in both discrete and continuous fashion were developed to control path edits and generate single edited or ongoing motions.

There are a variety of potential applications for these techniques. In Section 1.1, we discussed four usage scenarios which could benefit from fast methods of controlling the motion of animated

characters, and our techniques present solutions for each of those.

In the first scenario, we discussed how current methods of visualizing coordinated sports team plays can be insufficient. Even though, as sketches of simple symbols and arrows depicting paths of movement, they can be quickly produced and revised, the relative timing and interactions of separate motions can be difficult to depict statically. However, the same method of input - sketched paths - could be used as input to our path-based editing technique. As shown in Section 4.3.3, the editing handles of a character's motion path can be straightforwardly arranged to follow a new path input by a user, and a motion with the appropriate number of strides can be automatically determined. Timing relationships between separate motion paths could be specified with additional sketched connections, and the timing of related motions can be adjusted similar to the complementary work of Kim et. al [50]. Their work uses a (manually-controlled) one-dimensional as-rigid-as-possible deformation of keyframe timings; as our timewarping approach has already been augmented with such an approach for contact-to-ballistic transitions (Section 3.3.2), it would prove simple to add inter-motion dependencies in a similar way.

The second usage scenario involved the creation of motion for pre-visualization of live action or animated productions. While pre-visualization is a useful planning tool which reduces the amount of reshooting or revision during production, it is often accomplished using conventional animation techniques, which may be inefficient for creating rough visualization and rapid iteration. Our path-based editing algorithm, however, allows for fast exploration and refinement of the movement of characters to accomplish the staging of a scene. While our performance techniques for controlling a character's path might be useful in this scenario, an extension to sketch-based specification of paths (as discussed for the previous scenario) could be even more appropriate, given the usage of directional lines, arrows, and other indicators of motion commonly used in the two-dimensional storyboards which guide the creation of animated pre-visualization. Simple rules could allow motion paths to be modified by sketched paths or arrows in perspective, even through the point of view of the final camera.

In the third scenario, we discussed the usage of animated characters which can be interacted with in real-time, which are especially important in the increasingly prominent medium of virtual reality experiences. An impactful method of providing an immersive experience is to

maximize the user's agency, allowing free movement in a virtual environment and providing surrounding and characters which react believably. As shown in Sections 5.4.3 and 5.4.4, our path-based editing algorithm can be used in real-time to edit an existing single or cyclic motion, which would allow the automatic modification of animated characters' motion to react to a moving user. As well, since virtual reality goggles are fully enclosed, representing the users' motion on their virtual body, or avatar, also impacts the experience's immersiveness. Similar to interactive steering control (Section 5.4.2), the motion of the user as detected by the virtual hardware could control the avatar's motion path. Alternately, similar to the technique of Sugiura et. al [105], to enable stationary users to explore a virtual space, our finger walking technique (Chapter 4) could be adapted to a simple handheld touchscreen device, which has the added benefit of being straightforwardly usable solely by touch while wearing enclosed virtual reality goggles.

The fourth usage scenario involved the application of high-level control of animated characters in a feature animation setting where the standards for final animation are extremely exacting. While low-level control is ultimately necessary for precise nuance in high-quality animated performance, the process of developing an animated performance generally occurs in a coarse-to-fine fashion, with rough poses proceeding to smoother and more polished motion after continual iteration. Sometimes, the necessity of changes in the animation only become clear after significant effort on the part of the animator. For "blocking"-type decisions which require the changing of animation due to the composition of the final animated frames, our path-based editing technique could allow the "stage direction" of an animated performance to be broadly changed while preserving the painstaking fine details crafted by the animator.

6.1.1 Animator Feedback on Potential Usage

To explore the possible applications of our techniques in a high-quality feature animation setting, we discussed our techniques with five professional visual effects animators with experience varying from four to twenty-two years in the industry, whose work involves creating lifelike performances for both realistic digital doubles as well as impossible digital creatures. After demonstrating our basic path-based editing technique as well as the performance techniques, we allowed the animators to hands-on test the basic and gestural editing techniques

on an iPad, and discussed any aspects of the techniques which they found interesting or potentially useful in any scenario they encounter when animating.

All of the animators confirmed that our identified scenario, involving the challenge of modifying existing animation based on directorial feedback, presents a serious challenge when it occurs, which can be often. Especially in visual effects, a director may have their expectations shaped by their expertise in live-action production, making it difficult to communicate what they are looking for in an animation. This can mean that the animator's job is partially one of educated guesswork - as one animator put it, of "trying to visualize what's in someone else's head".

It can also be difficult for a director to evaluate and give meaningful feedback on an animation which is still in progress, as an animator's work process often begins with rough poses arranged in time without interpolating motion, the results of which can appear choppy and difficult to interpret to a viewer unaccustomed to animated production. However, once an animator has taken the time to clean up an animation to appear closer to a final version and be more easily evaluated, the director may request changes that require large changes or even starting the animation work over again entirely.

Because of these difficulties in the labourious back-and-forth of animation production over time - essentially, of visual communication - all of the animators agreed that our techniques could have a significant impact by allowing them to more easily and quickly produce smooth animation for evaluation by the director, and also to edit the animation according to feedback in order to hone in on the action of a scene; as this work often involves many iterations, our techniques could result in a large amount of effort saved by reducing iteration time. As one experienced animator and animation supervisor said, our techniques have "amazing potential to streamline the editing of performances based on director's comments".

The animators were also enthusiastic about additional scenarios and applications of our techniques, given that editing motion through their usual keyframing tools can be extremely time-consuming, and there are no widely used or robust tools for such tasks. One aspect of our techniques that was particularly exciting to the animators was the preservation of foot contacts using path-based deformation for end effector paths (Section 3.2.3), as foot contacts seem to be invariably disrupted whenever an animation is modified, and correcting or maintaining them

manually is painstaking work.

An additional application discussed by animators brought to light the different considerations and techniques which are useful as an animated performance is crafted over time. As one animator described, their work progresses "from blocking to polish", as a character's performance is initially very rough and often consisting of sparse poses, which are gradually filled in and adjusted, as additional nuance is added. Our techniques could be especially useful during the initial blocking work; this is similar to pre-visualization (as discussed in the previous section) but occurs afterward, as an animator explores the many large and small variations which are still possible even when the high-level action has been pre-visualized. Our techniques could enable this exploration to occur much more quickly.

The animators also discussed how often their work incorporates pre-existing animation, in the form of either motion capture data, or example animation cycles crafted by a lead animator to help with consistent creature performance. However, this pre-existing data will often not work directly in a scene. It was noted that motion capture performances involving environmental interaction rarely match up with the virtual environment of the corresponding character. In addition, the animation of prominent creatures which must be believable in a scene often cannot be expressed by motion cycles.

One animator noted that their animations have not only spatial goals but *performance goals*; while our techniques aim to enable high-level control of animation through solely spatial control (Section 1.3), they can still be useful to meet the more complex goals of feature animation. For example, one animator described a scenario where they had to choose between a variety of takes of motion capture data for a particular performance; only one take had the correct attitude for the character, but the motion didn't align with the environment when applied to the character. Our techniques could be used to modify the spatial aspects of selected pre-existing animations or motion data; the animator noted that even small changes to the path of a character can require a lot of work, but that it appeared that our techniques could produce changes of high enough fidelity as to require no additional touch-up work at all.

The discussions with animators also made clear that our techniques might require modification to accommodate one important part of how animators work: the placement and tuning of keyframes. There is a large variation in how animators select and craft keyframes,

which often expresses their own personal approach to creating and refining an animated performance. In general, as discussed by Coleman et al. [15], professional animators often create keyframes for joint chains that staggered in time, for motion to appear to propagate realistically. As well, keyframes from a professional animator may vary significantly in their density, as compared to the uniformly timesampled motion data we used as input for our techniques; for any given part of a character, its keyframes may be densely authored for some of a performance, and sparse at others.

While our underlying path-based editing technique does not require its keyframes to be unformly or densely sampled in time, nor does it require all bones to be keyframed at the same times, the results of using such input data have not been significantly tested. Converting between such representations is possible: a handcrafted animation could be uniformly timesampled in order for our techniques to be applied, and the results could be reduced back to the number and relative arrangement of keyframes of the input in a best-fit manner. Working directly with an animator's keyframes and allowing nondestructive editing - another feature of our techniques the animators were enthusiastic about - would require more testing and perhaps additional constraints, but would present a very powerful technique that could be applied to any motion than an animator would work with.

One of the themes that emerged from our discussions was the variation in the prominence of characters on-screen; animators craft performances for characters along a spectrum, varying from far background characters in a crowd, to midground bystander characters, to the most important foreground (or "hero") characters. While all character performances must be believable, the reduced prominence of background or midground characters often doesn't correspond to a significant reduction in the effort and time to animate using conventional techniques. Our path-based editing technique could be incorporated into the specialized tools for creating crowd animation (usually through agent-based simulation of a swarm or herd model). Furthermore, there could be an even greater impact by using our techniques to generate the motion of midground characters, whose performances must demonstrate more variation than a crowd and in a sparser group.

Even the more prominent performances which require significant manual animation polish could benefit from our techniques in a supporting role. Our techniques can apply directly to

the animation parameters controlled by the animators, accelerating the initial blocking stage of an animation or modifying a pre-existing animation to use as the basis for manual performance refinements. Our techniques could also apply to the creation of in-scene reference animation, which some animators prefer having alongside their character in three dimensions, rather than as video.

Based on our discussions with animators, there are applications of our techniques throughout this spectrum of character prominence. Table 6.1 shows a breakdown of some categories of character prominence, the typical method for animating them, and how our path-based motion editing techniques could be applied.

Character Prominence	Typical Method	Application of Path-Based Motion Editing
Far Background (Crowd)	Agent-based crowd simulation	Simulation system could generate animations by editing pre-exisiting motion to fit simulated agent paths; alternately, post-simulation, editing can be used to refine agent animations generated by other algorithms (e.g., motion graphs)
Background	Cycled animation or motion capture, with manual cleanup	Quickly edit pre-existing motion to fit desired path, with minimal cleanup
Midground	Manually keyframed; or using pre-existing animation/motion capture as a base or reference	Modify motion with desired performance to meet new spatial goals, to serve as a base for animation or as reference
Foreground (Hero)	Manually keyframed; ocassionally using pre-existing animation/motion capture as a base or reference	The most prominent characters require significant manual polish; motion editing can accelerate the blocking stage to prevent polishing unnecessary iterations of a performance

Table 6.1: An overview of the works we will discuss, classified with respect to their animation operation and user–specified parameters.

Overall, our techniques were very well received by the animators, who saw a variety of applications in their work: enhancing visual communication with a director; minimizing the

painstaking work of maintaining animated footplants; accelerating the process of initial blocking of a performance; and streamlining the modification of pre-existing motion to fit particular goals. As one animator said after seeing our techniques demonstrated, "this is how I want to work".

6.2 Limitations, and Future Work

Our techniques are not without their limitations. The following are some potential future work which would expand the techniques' capabilities:

Improved Effectiveness and Flexibility

As described in Sections 3.5, 4.5, and 5.5, there are a variety of ways in which the effectiveness of these techniques could be improved for their particular tasks. The path-based editing algorithm could incorporate more rules which would account for additional observed behaviour during motion, such as simple dynamics calculations to maintain character balance, or further biomechanical rules for leaning during turns. The recognition of locomotion type in the finger walking technique could be improved and expanded to other locomotion types, using more advanced machine learning techniques or perhaps a more expressive feature vector describing each performance. And the handheld gestural editing technique could potentially have its accuracy improved if the device's position were tracked accurately through space, which could be accomplished using the built-in camera(s) without requiring new sensors.

Beyond improving the efficiency of the techniques in their pre-existing usage scenarios, the techniques could be made more flexible and apply to more types of motions beyond locomotion. While the basic path-based editing algorithm has been applied experimentally to the motion of both non-human animated characters and physical objects (Section 3.4), further investigation into other types of motion are worthwhile. For example, stationary but expressive motion such as conversational gestures can be defined primarily in terms of the motion paths of their end effectors, which our technique might be applicable to. This is complementary to the approach for locomotion editing, where the end effector paths are secondary in importance and automatically edited based on changes to the root path.

Our user performance techniques could also be expanded beyond specifying locomotion. The

finger walking technique could be expanded to recognize user mimicry of the aforementioned examples of non-human motion and bouncing rigid objects, or any other motions which involve environmental contact. The handheld gestural editing technique, as well, has a straightforward application to the motion of rigid objects, since that is what the user is producing in the first place. But other, more complex manipulations of the device through space, perhaps involving translation and rotation simultaneously, could be used to express additional types of character motion as well. Aspects of these two performance techniques could also be combined, by allowing a user to manipulate the motion of a device while simultaneously interacting with its touchscreen, which would potentially allow greater control over a character's motion path and type of motion. User performances could also be applied to different parts of a character skeleton in addition to representing the root or overall motion, allowing motions to be built in a layered approach over a series of performances, similar to the approach of Dontcheva et al. [23].

Advanced Techniques and Hardware

While it would be possible to improve the effectiveness and scope of our techniques without changing their core approach, the capabilities of the techniques could potentially be vastly expanded by incorporating additional advanced techniques and hardware. Our core path-based editing algorithm could incorporate other animation operations, which could still enable interactive operation and quick iteration with fast kinematic operations such as blending (Section 2.1.3) - both splicing/interpolation and sequencing - as well as kinematic synthesis (Section 2.1.4), if enough suitable pre-existing motion data was available as input. These techniques could complement our algorithm by producing variety in the generated motions, since our motion transformation approach is deterministic and its results, while appearing plausible, do appear similar upon repeated edits since the nuances are all drawn from the same single input motion. Variation could be introduced while still meeting user goals, such as by automatically adjusting any stylistic parameters of an interpolated motion model, or by splicing whole parts of related motion, such as novel upper-body motion during walking.

Other animation operations could expand our techniques to apply to heterogeneous motions, such as a character walking and then running, since our algorithm can edit these motions (Section 3.4) but not generate them. Our path-based approach would still be a useful approach

to guiding the generation of such mixed motions either through motion sequencing or synthesis, or the operations could be performed consecutively, with a "straight ahead" mixed motion generated first, and then its path edited to match user input.

Advanced hardware with input and display capabilities beyond today's commodity devices could also expand our techniques. For user input, our performance techniques depend on direct contact with a multitouch surface and simple gestures tracked imprecisely. Given the results of our study exploring how users can communicate locomotion with their hands (Section 4.2), we would still expect hardware which allows perfect tracking of user's performance to require treating that input as not only imprecise but illustrative and gestural. Despite that, the types of performances which could be captured and recognized could still be expanded by passively detecting the complete pose and position of the user's hands, which would provide complete information about a user's hand performance above a contact surface, or remove the need for handheld devices entirely in favour of freehand gestures.

Advanced display hardware could also have a significant impact on the usage of these techniques, not only by allowing a user to view the generated animations immersively, but in providing feedback on their performed motion input as well. In our finger walking performance study (Section 4.4), the cause-and-effect of the users' performances was not directly clear, as the generated animations were displayed on the tabletop display from an orthographic top-down view and in another view on a separate monitor. Augmented reality displays could allow a user to view animated characters in the spatial context of their performance, potentially simultaneously. Virtual reality displays could also allow the user to more easily understand how the generated animations move through space, along with representations (such as a traced path of a manipulated device) of their performance, to allow for quicker iteration and a better understanding of exactly how their input motions appear.

Dynamic Motion Path Parameterization

One of the most significant limitations of the path-based editing algorithm is that it maintains a static parameterization of the character's poses along the path; that is, it has a fixed one-to-one correspondence between the vertices of the root path and the poses at each keyframe. While the timing of any motion produced by a user edit is adjusted by the final timewarping step of

the algorithm, this fixed relationship means that user manipulations of the editing handles can have unintended consequences. For example, we have observed during a number of informal sessions that novice users can easily stretch the path while positioning the editing handles of a walking motion in a seemingly benign way, resulting in a character which takes unintentionally or even impossibly long strides. This effect can be useful to intentionally modify the stride length of a motion if carefully managed, as in Section 5.3.6. However, in a freeform editing situation, the user's intention (path modification without significantly changing the motion's poses) can be "over-interpreted" into a too-complicated edit (changing the stride length).

One potential way to address this is to modify the parameterization of the keyframes along the path's length dynamically after each edit, which would allow keyframes to "slide" along the path and even past an editing handle when the path is stretched or compressed. Since the algorithm shouldn't introduce motion which doubles back along the path, this correspondence between path vertices and keyframes should always be monotonic, which means that this mapping may be computable through a simple method analogous to how the timewarping is calculated (Section 3.2.4).

One way to accomplish this would be to present a simplified path - perhaps initialized with the overall path (Section 3.3.3) - for user editing instead of the full root path. Then, following a user manipulation of this simple path, the algorithm could automatically determine the parameterization of the editing handles of the root path along this simple path. This maintains the advantages of using the path deformation algorithm on root paths, while introducing an automatic additional step with a small problem size, as only the editing handles' parameterization would have to be recalculated, rather than that of every keyframe. The techniques presented in Chapters 4 and 5 could also be adapted accordingly to modify this simplified path based on the user performance.

Transforming Motion Type and Style

The path-based editing algorithm is effective at modifying the broad spatial parameters of locomotion which can be expressed in the root path, and maintains the starting style and type of the motion; for example, a straightahead walk can be modified to follow a new path, but it will still be recognizable as a walk with the same behaviour or style. Rose et al. [91] used

a grammatical analogy for motion generation, likening motion type to verbs and motion style to adverbs. In that vein, the path-based editing algorithm is at present capable of modifying only the *object* of a motion, such as changing a motion from "the character walks *here*" to "the character walks *there*", or from "the character jumps *this* high" to "the character jumps *that* high".

However, the basic concepts of the algorithm - modifying motions through path editing and timewarping - could conceivably be used for transforming motion type and style by being applied non-uniformly to different parts of a character. For example, changing a walk to a run requires not only changes to the root path and to the poses, but to the basic pattern of the character's contact pattern with the environment, from the constant contact and periodic double-stance of walking to the alternation of single contact and ballistic periods in running. Those changes to the character's leg motion could be effected by localized path edits and timewarps to the motion of the feet. While more extreme changes to motion type in general would be extremely difficult, if not nonsensical (for example, changing a walk to a jump) without additional data, changing gait type or even generating transitions between gaits should be possible.

Modifications to the style of a motion (for example, making a walk more "happy") could also be expressed as a combination of root path edits and localized limb path edits and timewarps. While additional example motion data might be necessary for an automated approach, a large amount of data might not be required, as previous research has shown that style differences can be extracted and applied by considering the differences between just two motions [40, 95]. In addition, to enable interactive modification of a motion's style, any path presented for user editing might be better visualized in a relative space rather than the limb's absolute motion through space. For example, modifying the swing of an arm might be better accomplished by viewing the more spatially compact back-and-forth path of the arm relative to the body, rather than the long path traced out through space as the character also moves.

Automatic Analysis of Input Motions

One of the advantages of our path-based editing algorithm is that it can operate on a large variety of locomotive motions, which is enabled by its unique procedure for preprocessing the input motion. The automatic identification of environmental contacts not only drives the

identification of the editing handles for user interaction, but also the overall path and particular types of contact periods (such as ballistic motion), all of which affects both the path deformation and timewarping.

While the performance-based editing techniques which we built upon the path-based editing algorithm do utilize specialized higher-level representations of their input motion data, they are limited in that the representation is manually specified in advance rather than automatically determined. For example, the finger walking technique uses pre-selected and specially edited "canonical motions" (Section 4.3.3), while the handheld discrete gestural editing technique uses pre-determined parameterizations of the input motions' editing handles (Sections 5.3.4-5.3.6).

Instead, an expanded and more automatic approach to this analysis of the input motions could greatly improve the range and quality of the edits which these techniques can accomplish. More robust analysis could automatically identify editable segments within an input motion, allowing more complicated or mixed motions to be edited, rather than requiring specially-prepared clips. For handheld gestural editing, the user's initial reference gesture could be used to automatically identify the relevant editing handles of the input and automatically determine an appropriate parameterization, similar to how Dontcheva et al. [23] use an initial motion to select particular parts of a character's skeleton for editing.

Statistical and Machine Learning Techniques

Broadly speaking, in our techniques there are two aspects which are manually pre-determined, that can be potentially limiting: the biomechanical/physical rules used to control path deformation and timewarping (Sections 3.2.4 and 3.3.1–3.3.3), and the correspondence of a user's performance to how the editing handles of a motion are modified (Sections 4.3.1–4.3.4, 5.3.1–5.3.6, and 5.4.2). Both of these could potentially be improved by applying statistical or machine learning techniques if a large enough amount of example data were collected.

The rules governing path deformation and timewarping are generally simple constraints or equations which nonetheless enforce realistic results, since they have been drawn from empirical biomechanics research. This explicit representation of particular effects is limited because each must be manually specified, and desirable behaviours in how very particular parts of a character should be modified may be difficult to specify at all. An alternative approach is to allow these

effects to emerge implicitly through statistical techniques applied to a large collection of example motion data; for example, enough example turning motions of various curvature could allow a technique to build a model of how velocity is affected by path curvature, without representing such explicitly.

It might not be possible for a statistical technique to implicitly capture our explicit rules in a way that allows extreme motions outside of any training data to be modified in a plausible way (i.e., extrapolation), such as an impossibly high jump. However, the potential for identifying additional effects or correspondences is very powerful; for example, since humans generally bend down further in anticipation of higher jumps, that correspondence could be modelled automatically from examples, to allow the bending prior to a jump to be modified by the system whenever the jump height is edited by the user.

Further statistical or learning approaches could be applied to how user performances edit their corresponding motions, since our techniques compute this using explicit rules. While this works for some users, the rigidity in how the performance data is analyzed and used for editing may be disadvantageous if there are users whose performance style vastly differs from what the system expects. Both of our performance techniques would be able to incorporate example performances from a variety of users; as shown by the results of our example-based classification of finger walking performances (Section 4.4.2), simple learning techniques can be successful when applied to performance data of even a small sample size. In theory, with the proper parameterization, a large number of example performances and their corresponding motion edits could allow that mapping to be determined automatically with high accuracy.

User Studies and Participatory Design

In Section 1.1, we discussed that the conventional approach of producing animation can require a large amount of time and skill because it is both high-dimensional and low-level. The descriptions of these qualities and the difficulties they present were based on years of informal observations in the form of personal experience as well as first-hand observations of, and discussions with, both novice and professional animators. While we strongly feel that these descriptions are valid, and the resulting research motivated by them has shown a significant contribution, they are still anecdotal data. To address this, a principled user study could be

performed to more closely and quantitatively examine how animating is difficult. Such a study could test the effect of control dimensionality by presenting characters of varying complexity, as well as exploring low-level versus high-level controls, though a more formal definition of such would be required. The study could include a variety of methods for creating animation with specific goals, including keyframing, and gather data on user satisfaction with the advantages and disadvantages of each technique, as well as the resulting animations.

Forming general conclusions about how animation is difficult is not the only way that users could be more involved. Our finger walking technique was developed after an exploratory user study (Section 4.2), the conclusion of which indicated that the technique could be useful and intuitive, and the gathered data was essential for developing the technique. While the handheld gestural editing techniques were not developed following a user study, pursuing such a study in the future could inspire new methods of manipulating a handheld device for editing motion. A performance study is also useful for validation, as was also conducted for the finger walking technique (Section 4.4). However, beyond simply exploratory and performance studies which occur at the beginning and end of the process of developing a new technique, involving users throughout the process, in a form of participatory design, could also provide useful feedback and ideas.

6.3 Summary and Discussion

In this thesis, we have presented a collection of related techniques for interactively editing the spatial and timing parameters of animated human locomotion. In Chapter 2, we surveyed related work for generating motion, categorized both by the operation used to generate the new motion and the type of user-specified parameters which serve as input. In Chapter 3, we presented an algorithm for editing a motion through user manipulation of the motion's path. This algorithm uses simple rules based on biomechanical principles to automatically transform a wide variety of motions to match the user's path edits, and includes a timewarping step which preserves the timing of the character as they move along the new path. In Chapter 4, based on the data from an exploratory study, we presented a finger walking technique to allow users to specify motion type and path edits by mimicking the motion on a touch-sensitive tabletop, and

evaluated the technique in a performance study. Finally, in Chapter 5, we presented techniques for editing motions and motion paths based on a user's gestural performance with a handheld mobile device.

As discussed in Section 1.3, one factor which contributes to the difficulty of creating animation is that the space and time of motion are tightly coupled, as they are in the real world by physics; this means that the posing and timing of an animated character's motion must be coordinated, and with high precision. As per our thesis statement (Section 1.4), the techniques we have presented leverage the coupling of the spatial and timing aspects of motion, by utilizing simple rules based on biomechanics and physics as well as the spatiotemporal information in a user's physical performance.

These techniques are all related in that the path-based editing algorithm is, of course, a fundamental part of how the two performance interface techniques operate. The path editing handles form an abstraction of the motion to be edited, which are more easily manipulated automatically to accommodate the user's performance input. However, all of these techniques are also related in three fundamental ways which are essential to their effectiveness, and which could also prove to be useful themes for future work on motion generation of any form.

First, it should be noted that in recent years, there has been an increasing tendency in motion generation algorithms to rely on larger amounts of input data, larger amounts of offline computation, or both. As shown by the representative works in Table 2.1, much of the published research in the past ten years has been in techniques for motion blending and synthesis, which require multiple input motions and more computation, respectively, unlike many motion transformation techniques. Some of these works process a large corpus of motion data in a "big data"-type approach to build motion models, and others perform extensive and expensive offline optimization using physics simulations to train controllers for use in real-time. These techniques are novel and produce good results, but they have a large reliance on data and computation, and are often very limited in user interaction.

In contrast to those approaches, the first theme among the techniques in this thesis is that they demonstrate that not only can motion generation algorithms enable a broad range of user goals interactively, but that this can be accomplished with low data and computational requirements. While there is other contemporary research along these lines, our path-based

editing algorithm is unique as it complements its single input motion with biomechanical and physical rules which, while simple, are robust enough to apply to a broad range of motions and remove the need for extra example data or expensive simulation. The performance interface techniques in this thesis also use relatively simple data as user input, in the form of touch surface contacts and basic motion sensor data. Based on observations of how this simple data appears in practice, the techniques still demonstrate useful and expressive capabilities in motion editing without requiring complex models or significant computation.

Second among these themes is that higher-level semantic representations of input motion are highly useful for motion editing and can also serve the same function as more data or computation. As discussed in the previous Section, the preprocessing performed by the path-based editing algorithm allows different parts of a motion to be edited differently, such as the acceleration-based timewarping of ballistic periods in a motion, compared to the velocity-based timewarping during contact periods. This higher-level representation of what the character is doing is essential for the algorithm's efficiency and flexibility, as contrasted with other approaches which solely consider the "signals" of every transformation in the character's skeleton over time, or which represent a character strictly as a collection of articulated rigid bodies.

Our performance interface techniques also use a higher-level understanding of their input motions, though they are manually specified. The various canonical motions used in the finger walking work were identified and prepared for looping in advance, while the parameterizations of the editing handles modified by the handheld gestural editing were specially determined for each editable motion. While there is a significant amount of research into analyzing and classifying motion data, such approaches are often completely separate from motion generation. The techniques in this thesis demonstrate that even a simple "understanding" of the content of a motion can greatly increase the fidelity with which it can be transformed.

Third, and final, among these themes is that the spatial and temporal aspects of motion data are not independent but coupled, and can be treated as such. This means that editing one can and should affect the other. However, in the common approach of treating motion data as a signal over time, the sampling rate of the motion is almost always considered fixed and constant, and the output motion is generated to match. Even when a variable temporal

parameterization is utilized for motion generation, it is often for the purposes of calculating a correspondence between motions through a process called dynamic timewarping, rather than being applied to a new motion. However, the basic laws of motion as well as other rules such as the Froude number (Section 3.2.4) and one-third power law (Section 3.3.3) do not treat spatial and temporal variables as independent, but each include them in single equations.

To this end, our path-based editing algorithm includes a flexible timewarping formulation which is an essential and mandatory part of every edit. That this timewarping occurs as the last step in an edit means that spatial changes can drive the timewarping changes necessary to keep the edited motion consistent in timing with the original, where "consistent" does not necessarily mean equivalent. Our performance interface techniques also utilize both the spatial and temporal parameters of the user's input in order to build a better representation of the user's performance. For example, the finger walking technique uses the spacing between the contacts on the touch surface in addition to their frequency, and the handheld gestural editing technique uses the relative timing between the reference and editing gestures in its calculation of its scale factor. All of our techniques have been greatly enhanced by considering the interdependency between a motion's spatial and temporal parameters, and perhaps this could also prove useful for future motion generation techniques.

Bibliography

[1] Yeuhi Abe, C. Karen Liu, and Zoran Popović. Momentum-based parameterization of dynamic character motion. *Graph. Models*, 68(2):194–211, 2006.

[2] Yeuhi Abe and Jovan Popović. Interactive animation of dynamic manipulation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[3] R. Mc N. Alexander. Estimates of speeds of dinosaurs. *Nature*, 261:129–130, 1976.

[4] Brian Allen, Derek Chu, Ari Shapiro, and Petros Faloutsos. On the beat!: timing and tension for dynamic characters. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2007.

[5] Gustavo Arechavaleta, Jean-Paul Laumond, Halim Hicheur, and Alain Berthoz. An optimality principle governing human walking. *IEEE Transactions on Robotics*, 24(1):5–14, 2008.

[6] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, 2003.

[7] Okan Arikan, David A. Forsyth, and James F. O'Brien. Pushing people around. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005.

[8] Ron Baecker. *Interactive Computer-Mediated Animation.* PhD thesis, Massachusetts Institute of Technology, 1969.

[9] Connelly Barnes, David E. Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. Video puppetry: a performative interface for cutout animation. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, 2008.

[10] Daniel Bennequin, Ronit Fuchs, Alain Berthoz, and Tamar Flash. Movement timing and invariance arise from several geometries. *PLoS Computational Biology*, 5(7), July 2009.

[11] Matthew Brand and Aaron Hertzmann. Style machines. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[12] Armin Bruderlin and Lance Williams. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[13] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 2005.

[14] Patrick Coleman. *Expressive Motion Editing Using Motion Extrema*. PhD thesis, University of Toronto, 2012.

[15] Patrick Coleman, Jacobo Bibliowicz, Karan Singh, and Michael Gleicher. Staggered poses: A character motion representation for detail-preserving editing of pose and coordinated timing. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2008.

[16] Seth Cooper, Aaron Hertzmann, and Zoran Popović. Active learning for real-time motion controllers. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, 2007.

[17] Stelian Coros, Philippe Beaudoin, Kang Kang Yin, and Michiel van de Pann. Synthesis of constrained walking skills. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, 2008.

[18] Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. Locomotion skills for simulated quadrupeds. In *ACM SIGGRAPH 2011 papers*, 2011.

[19] James Davis, Maneesh Agrawala, Erika Chuang, Zoran Popović, and David Salesin. A sketching interface for articulated figure animation. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003.

[20] Timothy Davis. Algorithm 832: Umfpack v4.3 - an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30:196–199, June 2004.

[21] Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. Feature-based locomotion controllers. In *ACM SIGGRAPH 2010 papers*, 2010.

[22] Paul C. DiLorenzo, Victor B. Zordan, and Benjamin L. Sanders. Laughing out loud: control for modeling anatomically inspired laughter using audio. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, 2008.

[23] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, 2003.

[24] George ElKoura and Karan Singh. Handrix: animating the human hand. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003.

[25] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001.

[26] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, 2003.

[27] Andrew W. Feng, Yuyu Xu, and Ari Shapiro. An example-based motion synthesis technique for locomotion and object manipulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2012.

[28] William Froude. On useful displacement as limited by weight of structure and of propulsive power. *Transactions of the Royal Institution of Naval Architects*, 15:148–155, 1874.

[29] Michael Gleicher. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, 1997.

[30] Michael Gleicher. Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998.

[31] Michael Gleicher. Motion path editing. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, 2001.

[32] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: assembling run-time animations. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, 2003.

[33] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, 2004.

[34] Sehoon Ha and C. Karen Liu. Iterative training of dynamic skills inspired by human coaching techniques. *ACM Trans. Graph.*, 34(1), 2014.

[35] Rachel Heck and Michael Gleicher. Parametric motion graphs. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 2007.

[36] Rachel Heck, Lucas Kovar, and Michael Gleicher. Splicing upper-body actions with locomotion. *Comput. Graph. Forum*, 25(3):459–466, 2006.

[37] Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. Spatial relationship preserving character motion adaptation. In *ACM SIGGRAPH 2010 papers*, 2010.

[38] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[39] Eugene Hsu, Marco da Silva, and Jovan Popović. Guided time warping for motion editing. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2007.

[40] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 2005.

[41] T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial keyframing for performance-driven animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005.

[42] Leslie Ikemoto, Okan Arikan, and David Forsyth. Knowing when to put your foot down. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, 2006.

[43] Leslie Ikemoto, Okan Arikan, and David Forsyth. Quick transitions with cached multi-way blends. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 2007.

[44] Leslie Ikemoto and David A. Forsyth. Enriching a motion collection by transplanting limbs. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004.

[45] Apple Inc. Event Handling Guide for iOS: Motion Events. `https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/motion_event_basics/motion_event_basics.html`. Accessed: 2014-12-16.

[46] Google Inc. Android API Guide: Motion Sensors. `http://developer.android.com/guide/topics/sensors/sensors_motion.html`. Accessed: 2014-12-16.

[47] Satoru Ishigaki, Timothy White, Victor B. Zordan, and C. Karen Liu. Performance-based control interface for character animation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, 2009.

[48] Eakta Jain, Yaser Sheikh, and Jessica Hodgins. Leveraging the talent of hand animators to create three-dimensional animation. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009.

[49] Sumit Jain and C. Karen Liu. Interactive synthesis of human-object interaction. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009.

[50] Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. Synchronized multi-character motion editing. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, 2009.

[51] Evangelos Kokkevis, Dimitri Metaxas, and Norman I. Badler. User-controlled physics-based animation for articulated figures. *Computer Animation*, 0:16–26, 1996.

[52] Lucas Kovar and Michael Gleicher. Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003.

[53] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, 2004.

[54] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002.

[55] Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002.

[56] Taesoo Kwon and Sung Yong Shin. Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005.

[57] Yu-Chi Lai, Stephen Chenney, and ShaoHua Fan. Group motion graphs. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005.

[58] John Lasseter. Principles of traditional animation applied to 3d computer animation. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, 1987.

[59] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[60] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Interactive control for physically-based animation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.

[61] Manfred Lau, Ziv Bar-Joseph, and James Kuffner. Modeling spatial and temporal variation in motion data. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, 2009.

[62] Benoît Le Callennec and Ronan Boulic. Interactive motion deformation with prioritized constraints. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004.

[63] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002.

[64] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999.

[65] Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. Motion patches: building blocks for virtual environments annotated with motion data. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, 2006.

[66] Sergey Levine, Christian Theobalt, and Vladlen Koltun. Real-time prosody-driven synthesis of body language. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, 2009.

[67] Q. L. Li, W. D. Geng, T. Yu, X. J. Shen, N. Lau, and G. Yu. Motionmaster: authoring and choreographing kung-fu motions by sketch drawings. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[68] Yi Lin. 3d character animation synthesis from 2d sketches. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, 2006.

[69] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002.

[70] A. Majkowska, V. B. Zordan, and P. Faloutsos. Automatic splicing for hand and body animations. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[71] Anna Majkowska and Petros Faloutsos. Flipping with physics: motion editing for acrobatics. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2007.

[72] Shahzad Malik, Abhishek Ranjan, and Ravin Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST '05: Proceedings of the ACM symposium on User interface software and technology*, 2005.

[73] J. Martinez Esturo, C. Rössl, and H. Theisel. Generalized metric energies for continuous shape deformation. *Springer LNCS (Proc. Curves and Surfaces 2012)*, 8177(1):135–157, 2013.

[74] James McCann and Nancy Pollard. Responsive characters from motion fragments. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, 2007.

[75] S. Ménardais, R. Kulpa, F. Multon, and B. Arnaldi. Synchronization for dynamic blending of motions. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004.

[76] Jianyuan Min, Yen-Lin Chen, and Jinxiang Chai. Interactive generation of human animation with deformable motion models. *ACM Trans. Graph.*, 29(1), 2009.

[77] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 2005.

[78] Tomohiko Mukai and Shigeru Kuriyama. Pose-timeline for propagating motion edits. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009.

[79] Michael Neff and Eugene Fiume. Aesthetic edits for character animation. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003.

[80] Michael Neff and Eugene Fiume. Aer: aesthetic exploration and refinement for expressive character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005.

[81] Michael Neff, Michael Kipp, Irene Albrecht, and Hans-Peter Seidel. Gesture modeling and animation based on a probabilistic re-creation of speaker style. *ACM Trans. Graph.*, 27(1):1–24, 2008.

[82] Rubens F. Nunes, Joaquim B. Cavalcante-Neto, Creto A. Vidal, Paul G. Kry, and Victor B. Zordan. Using natural vibrations to guide control for locomotion. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2012.

[83] A. Cengiz Öztireli, Ilya Baran, Tiberiu Popa, Boris Dalstein, Robert W. Sumner, and Markus Gross. Differential blending for expressive sketch-based posing. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2013.

[84] Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. On-line locomotion generation based on motion blending. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002.

[85] Aftab E. Patla, Stephen D. Prentice, C. Robinson, and J. Neufeld. Visual control of locomotion: Strategies for changing direction and for going over obstacles. *Journal of Experimental Psychology: Human Perception and Performance*, 17(3), 1991.

[86] Ken Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, 1995.

[87] Ken Perlin and Athomas Goldberg. Improv: a system for scripting interactive actors in virtual worlds. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[88] Zoran Popović and Andrew Witkin. Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999.

[89] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: texturing and synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002.

[90] Cheng Ren, Liming Zhao, and Alla Safonova. Human motion synthesis with optimization-based graphs. *Computer Graphics Forum*, 29(2):545–554, 2010.

[91] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*, 18(5):32–40, 1998.

[92] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[93] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, 2007.

[94] Yeongho Seol, Carol O'Sullivan, and Jehee Lee. Creature features: online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2013.

[95] Ari Shapiro, Yong Cao, and Petros Faloutsos. Style components. In *GI '06: Proceedings of Graphics Interface 2006*, 2006.

[96] Ari Shapiro, Marcelo Kallmann, and Petros Faloutsos. Interactive motion correction and object manipulation. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 2007.

[97] Hyun Joon Shin, Lucas Kovar, and Michael Gleicher. Physical touch-up of human motions. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, 2003.

[98] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94, 2001.

[99] Hyun Joon Shin and Hyun Seok Oh. Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[100] Ben Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*, 1997.

[101] Hubert P. H. Shum, Taku Komura, Masashi Shiraishi, and Shuntaro Yamazaki. Interaction patches for multi-character animation. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, 2008.

[102] Karl Sims. Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994.

[103] Kwang Won Sok, Katsu Yamane, Jehee Lee, and Jessica Hodgins. Editing dynamic human motions via momentum and force. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010.

[104] Justin Solomon, Mirela Ben-Chen, Adrian Butscher, and Leonidas Guibas. As-Killing-As-Possible Vector Fields for Planar Deformation. *Computer Graphics Forum*, 2011.

[105] Yuta Sugiura, Gota Kakehi, Anusha Withana, Charith Fernando, Daisuke Sakamoto, Masahiko Inami, and Takeo Igarashi. Walky: An operating method for a bipedal walking robot for entertainment. In *ACM SIGGRAPH Asia 2009 Emerging Technologies*, 2009.

[106] Mankyu Sung, Lucas Kovar, and Michael Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005.

[107] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24(1):98–117, 2005.

[108] S. C. L. Terra and R. A. Metoyer. Performance timing for keyframe animation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004.

[109] Frank Thomas and Ollie. Johnston. *The Illusion of Life: Disney Animation.* Disney Editions, 1981.

[110] Matthew Thorne, David Burke, and Michiel van de Panne. Motion doodles: an interface for sketching character motion. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, 2004.

[111] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models*, 62(5), 2000.

[112] M. Tournier and L. Reveret. Principal geodesic dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012.

[113] Adrien Treuille, Yongjoon Lee, and Zoran Popović. Near-optimal character animation with continuous control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, 2007.

[114] Winnie Tsang, Karan Singh, and Eugene Fiume. Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005.

[115] C. Vaughan and M. O'Malley. Froude and the contribution of naval architecture to our understanding of bipedal locomotion. *Gait & Posture*, 21(3):350–362, 2005.

[116] Kevin Wampler and Zoran Popović. Optimal gait and form for animal locomotion. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, 2009.

[117] Kevin Wampler, Zoran Popović, and Jovan Popović. Generalizing locomotion style to new animals with inverse optimal regression. In *SIGGRAPH '14: ACM SIGGRAPH 2014 papers*, 2014.

[118] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Optimizing walking controllers. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, 2009.

[119] Jue Wang, Steven M. Drucker, Maneesh Agrawala, and Michael F. Cohen. The cartoon animation filter. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, 2006.

[120] Z. Wang and M. van de Panne. Walk to here: a voice driven animation system. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006.

[121] Andrew Witkin and Zoran Popovic. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[122] Katsu Yamane, James J. Kuffner, and Jessica K. Hodgins. Synthesizing animations of human manipulation tasks. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, 2004.

[123] Po-Feng Yang, Joe Laszlo, and Karan Singh. Layered dynamic control for interactive character swimming. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004.

[124] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: simple biped locomotion control. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, 2007.

[125] KangKang Yin and Dinesh K. Pai. Footsee: an interactive animation system. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003.

[126] Peng Zhao and Michiel van de Panne. User interfaces for interactive control of physics-based 3d characters. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 2005.

[127] Victor Brian Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002.

[128] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 2005.