Expressive Motion Editing Using Motion Extrema

by

Patrick Coleman

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

Abstract

Expressive Motion Editing Using Motion Extrema

Patrick Coleman

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2011

When animating characters, a key goal is the creation of a believable, expressive performance that gives a character a unique personality with a distinct style of movement. While animators are skilled at creating expressive, personalized performances, it remains challenging to change performance–related aspects of movement in existing motion data. In recent years, motion data reuse has become increasingly important as recorded motion capture data has come into widespread use. This thesis investigates the use of a sparse set of pose–centric editing controls for editing existing motion data using techniques similar to those used by keyframe animators when they create new motion. To do this, this thesis proposes the use of *motion extrema*—the poses a character passes through when there is a significant change in movement—as a means for choosing effective pose–centric editing controls.

First, I present algorithms for identifying motion extrema. Motion extrema can be associated with individual joints or the full body of the character. I introduce a set of approaches for identifying motion extrema; these include the use of *extrema of differential measures* and the explicit search for times at which the body or a joint is in a *spatially extreme configuration*.

I then present three motion editing applications that use motion extrema as a foundation for applying motion edits. The first application, *pose–centric editing*, allows users to interactively change poses in a motion, and the system modifies the motion to respect

existing ground contact. The second application—*staggered poses*, introduces a model of character pose that explicitly encodes how timing varies among motion extrema on different parts of the body. This timing variation is commonly used by animators to model overlapping action. By introducing an algorithm for finding timing variation on motion extrema in existing motion, this system enables users to make high–level changes to timing patterns to change overlap effects in existing motion.

Finally, I present a procedural motion editing application that targets a specific aspect of motion style; this technique is called *spatial exaggeration*. Spatial exaggeration changes the geometric relationships among extreme poses. Such edits cause movement to appear more or less energetic. Overall, these applications demonstrate that performance–related aspects of existing motion can be edited using a sparse set of controls in the form of motion extrema.

Contents

Chapter 1

Introduction

Keyframe animation tools have long enabled skilled animators to create compelling character movement. Key to this is direct control over every aspect of a character's motion. Animators commonly achieve this by specifying keyframe poses on a full character or keyframe constraints on parts of a character. However, it remains challenging to use such techniques to allow animators to quickly make expressive edits to existing motion, such as recorded motion capture data, to craft it into a compelling performance.

This thesis introduces techniques for identifying keyframe–like editing controls in existing motion data and presents systems that allow users to quickly edit recorded motion data using such controls. In this proposed workflow, a motion editing system assists the user with selecting keyframe poses or keyframe controls on individual joints. These keyframe controls should be qualitatively similar to the keyframes used by animators when they create new motion. Users can then edit these keyframe poses or keyframe controls on joints to specify how the motion of a character should be changed. These keyframe constraints can be specified manually, or they can be specified using new procedural techniques inspired by the classical animation principles [167] and common animator workflows [177]. The motion editing system applies these keyframe edits to the existing motion while automating lower–level editing details, such as ground contact

preservation. Overall, this approach to expressive motion editing produces new motion that has similar content to the original, but has been robustly modified to meet the keyframe editing constraints that have been specified either manually or procedurally.

As an overall goal, users should be able to quickly select and edit pose controls, either manually or procedurally. These pose controls should be qualitatively similar to the poses used in keyframe animation, and the editing system should include algorithms for automatically identifying these pose controls. This reduces the complexity of the editing task and assists unskilled users with pose selection. As the user edits the pose–based controls, the motion editing system should then update the motion in a robust, plausible way to provide feedback about the resulting motion. Also, when making these changes, users should be able to edit the motion without directly specifying low–level aspects of movement. Specifically, they should be able to focus on the high–level intention of the edit. If the intention of the edit is to make a character walk while hunched over and taking shorter steps, the user should not have to manage details of movement such as contact constraints. Finally, new algorithms and applications should provide the user with immediate, interactive feedback. Interactive editing is important to allow for quick editing workflows in which immediate feedback is presented as the user changes the poses.

As the focus of these techniques is on changing how a motion is performed without changing the basic type of movement, operations that can fundamentally change the type of motion are not investigated. Such operations include the insertion of new important poses or the removal of existing important poses. Instead, the focus will be on motion edits in which the user modifies existing poses that are similar to the keyframe poses used by animators.

Motion editing techniques allow animators to make changes to reuse existing motion, but existing approaches do not sufficiently meet the above criteria and goals. For example, kinematic editing algorithms [92, 41, 69] allow users to specify changes interactively, but these techniques require users to explicitly specify all constraints, whether related to

the intention of the edit in the form of keyframe poses or to lower–level details such as preserving motion plausibility. Techniques such as footskate correction [75, 92] can ameliorate the need to specify low–level details, but these post–processing techniques do not, in general, preserve details of an edit that relate to character pose. Physics–based optimization can be used to meet complex constraints while managing low-level motion details [178], but these approaches involve computationally intensive numerical solutions that preclude their use in interactive systems. Finally, existing techniques for motion editing do not aim to automatically identify editing controls that are similar to the poses used for keyframe animation.

However, keyframe animation techniques used for creating new motion do meet some of these design criteria. For example, keyframe animation systems are interactive and allow users to specify the design of motion at a high level using keyframe poses. However, details of movement such as contact must typically be addressed using manual constraint specification and an inverse kinematics solver. As kinematic motion editing algorithms often succeed at the goal of automating low–level details of how to change a motion once they are specified, this motivates the design of a hybrid approach: motion editing systems that incorporate keyframe workflows. Such an approach would allow users to interactively specify changes at a high level while the system automates how to manage low–level details of movement such as contact. To support such a workflow, which would combine the strengths of both keyframe animation and kinematic motion editing, contact constraints and pose–based editing controls will need to be automatically chosen. For the latter problem, I focus on the use of *extreme poses*, which are a type of pose commonly used by animators when designing new motion.

Extreme Poses

An extreme pose is a type of keyframe pose that, when taken in a temporal sequence, concisely describes the basic kinematics of body movement. In keyframe animation work-

flows, different types of poses are commonly used as controls for specifying how a character moves. Pose–to–pose editing, in particular, is a workflow that is commonly used to create new keyframe animation. In this approach, animators create a sparse initial sequence of poses, which determine how a character changes stance over time [167, 177]. At this stage, extreme poses are used to specify the major stances that a character moves through while carrying out the motion [177]; designing a sparse sequence of extreme poses is commonly called *blocking* [46]. As the sequence of extreme poses used to design new motion conveys the basic movement that takes place, it can be considered a static summary of the motion.

An example sequence of extreme poses used to design new motion is illustrated in Figure 1.1, which is adapted from an illustration by Williams [177]. In each extreme pose, the character holds a stance in which limbs are extended, which serves to illustrate the basic mechanics of the movement.

Extreme poses are also important for conveying expressive aspects of movement related to a character's performance. The specific extreme poses that a character passes through can reveal parts of their personality and indicate their mood. Goldberg emphasizes this idea in his discussion of *attitude* poses—attitude poses are poses that, in static isolation, reveal the personality of the character and their thoughts or emotions [46]. Creating a sequence of extreme poses that are also good attitude poses gives an animator control over how the personality and emotions of a character are portrayed. Williams illustrates the importance of extreme pose design in his discussion of how to animate walking motion in a variety of expressive styles [177]. In his text, a wide variety of stylized walks that reveal aspects of character personality are illustrated using repeating sequences of extreme poses. To add expressive performance to the sequence of poses illustrated in Figure 1.1, an animator would modify them such that each pose has a stance that reveals aspects of the character's personality of mood. For example, these could be modified to indicate whether the character is energetic, tired, determined, or relaxed.

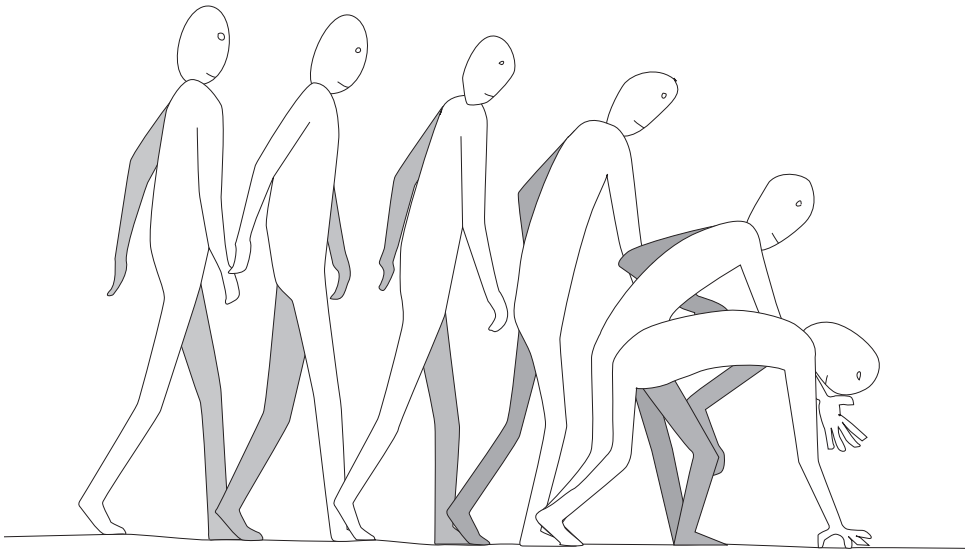


Figure 1.1: An example of extreme pose sketches created by an animator to design a new motion. Extreme poses occur when parts of a body, such as arms and legs, are at maximal swing angles. At these times, limb motion tends to change direction. For example, an arm swinging forward will change direction to swing backward at the time of an extreme pose. Adapted from Page 65 of Williams [177].

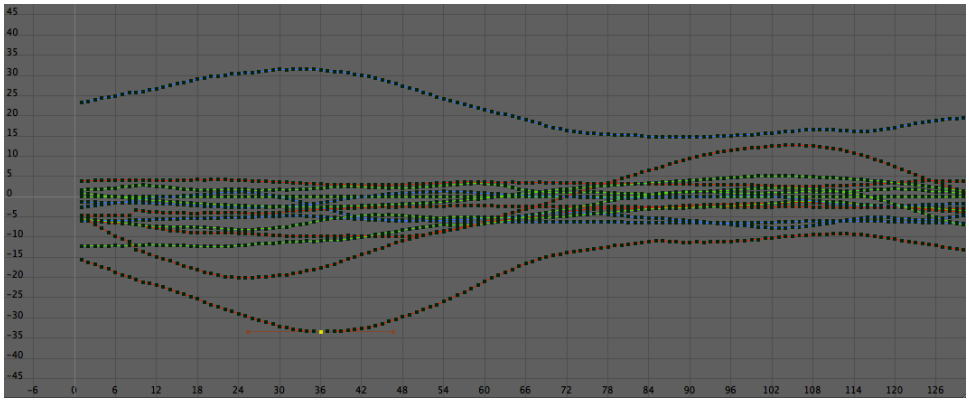
1.1 Motion Extrema

Extreme poses can be considered a specific case of what I refer to in this thesis as *motion extrema*. Motion extrema, in general, have the same qualities of extreme poses in terms of how they relate to motion, but they relate to arbitrary portions of a character's body. This generalized form of extreme–based editing control is inspired by animator workflows in which the user focuses on one portion of the character's body at a time. For example, Lasseter describes a form of layered refinement, in which he first animates a small set of parameters that affect the entire character [85]. Other aspects of movement that relate to specific parts of the body are then added by animating other parameters. This can be done be creating a sparse set of extreme configurations for each joint or part of the body, as it is animated. These extreme configurations correspond to the motion extrema described in this thesis.

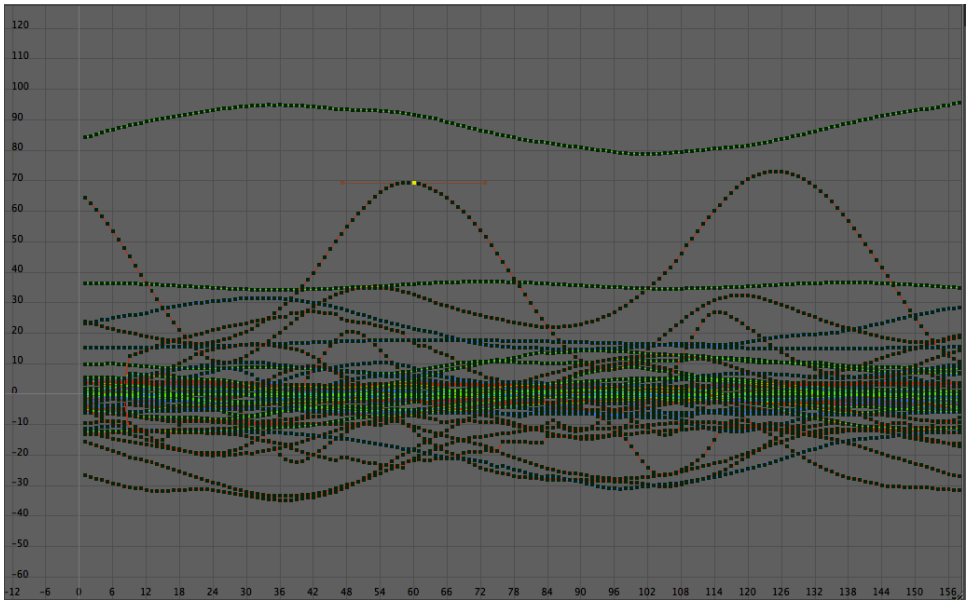
To investigate the use of motion extrema as a foundation for expressive motion editing, these computational problems will be considered:

- The identification of motion extrema in existing motion data.
- The robust application of edits specified using extreme poses or per–joint motion extrema to an entire motion.
- The modeling of timing variation among motion extrema on different parts of the body.
- The development of procedural approaches to editing motion style that modify relationships among motion extrema using simple, high–level parameterizations.

A set of motion extrema is a set of constraints over the geometric state of some portion of a character's body, potentially the entire body, at particular times. These motion extrema can relate to individual degrees of freedom, as is common in animation splines. They can also be specified for individual joints or collections of joints, as well



(a)

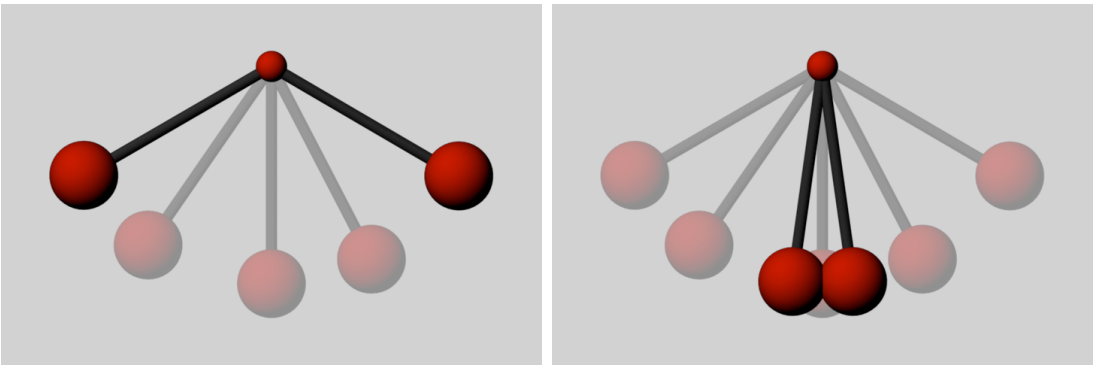


(b)

Figure 1.2: Current user interfaces for keyframe editing, such as the spline editor shown here, are difficult to use for editing recorded motion data, as they do not scale well to densely sampled data. Recorded data for one second of a moving arm is shown in a, and data for the full control skeleton is shown in b.

as for the full body as a character pose. A set of motion extrema is said to be *sparse* if there are far fewer motion extrema than animation frames. As they represent extreme configurations of the character, motion extrema should occur at times when the character or a joint passes through a state that is qualitatively similar to an extreme pose.

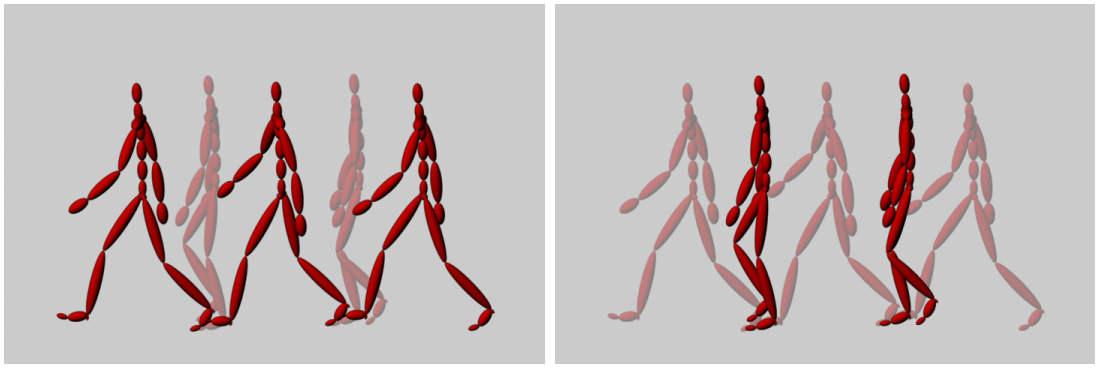
Modern systems for keyframe editing due not scale well to densely sampled motion data. As shown in Figure 1.2a, even one second of arm motion presents the user with many more degrees of freedom than is necessary to edit the motion. Working with the full body (Figure 1.2b) exacerbates this difficulty. Developing algorithms for automatically identifying a sparse set of motion extrema to use as editing controls would greatly simplify the task of using keyframe editing techniques for editing densely sampled motion. With such algorithms, only the motion extrema, such as extreme poses, would be need to be edited by the user.



(a) (b)

Figure 1.3: For this pendulum, the geometric state at extreme points of the swing can be used as pose–based editing controls to modify how much the pendulum swings (opaque, a). If controls are specified at other times, users do not have this type of direct editing control over how much the pendulum swings (opaque, b).

To demonstrate why the use of extreme poses is important, potential choices for pose–based editing controls are illustrated for a swinging pendulum in Figure 1.3. In this example, potential editing controls consist of the geometric state of the pendulum



(a) (b)

Figure 1.4: Choosing editing controls at extreme poses (opaque, in a) allows for direct editing of the primary movement in the walk—the coordinated swing of the arms and legs. Choosing non–extreme poses does does not provide direct editing control over the primary movement (b).

at the times at which it appears fully opaque. Figure 1.3a shows editing controls that are chosen at the extreme points of the swing. In this case, the user has direct editing control over the spatial range of the motion, as the extreme state can be directly edited. As a result, users can use such controls to directly edit how much the pendulum swings.

Figure 1.3b illustrates editing controls chosen at other times. In this case, the user will not, in general, have direct control over how much the pendulum swings in space, as the editable poses do not occur at the extreme points of the swing. As a result, the extreme pose–based controls shown in Figure 1.3a would be more effective for editing the overall motion of the pendulum than the non–extreme editing controls shown in Figure 1.3b.

Similarly, this idea applies to character motion, as illustrated with a walk in Figure 1.4. In Figure 1.4a, the extreme poses occur when the legs and arms are at a point of extreme swing. Users can edit the dominant motion of the character by directly editing these poses. In Figure 1.4b, the poses occur mid–stride. These non–extreme poses do not allow users to directly edit this dominant swinging motion.

A key observation for both the pendulum motion and the character walk is that

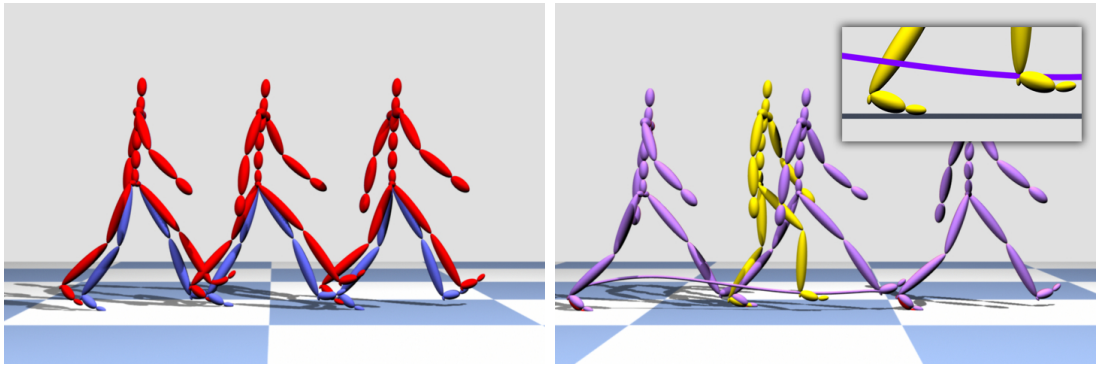
extreme–based controls over movement occur when motion changes significantly. Qualitatively, motion extrema are similar to the extreme poses described by Williams [177], which animators often use as a first pass when designing new motion. A key contribution of this thesis is a set of algorithms for finding these motion extrema to use as a sparse set of motion editing controls.

When choosing a set of editing controls, the goal is to identify times at which the body is in a pose that is similar to one that would be created by an animator when creating new keyframe animation. Considering Figure 1.1, the following observations can be made about the extreme poses:

- Extreme poses occur at times when parts of the body change direction of movement.
- Extreme poses occur when parts of the body are moving at their slowest, due to the slowing down of speed before the change of direction. Animators refer to this as *ease in and ease out*.
- Extreme poses occur when joint trajectories have high curvature.
- Extreme poses occur when joints are at locations in space that are most distant in space from corresponding locations at neighboring times. For example, at the point of maximum arm swing, the location of the wrist is at a location that is most distant to its position at both prior and subsequent times.

1.2 Overview

To set the context for the work introduced in this thesis, I will first relate the new ideas and applications to existing algorithms for motion editing (Chapter 2) and methods for aesthetically describing movement (Chapter 3). I will then describe the formal models of motion that I will use, my assumptions about character models, and the specifics of



(a) (b)

Figure 1.5: In pose–centric motion editing, users can make quick edits (red) to poses for editing control (a). This approach, presented in Chapter 6, preserves ground contacts and robustly creates ground–relative end effector motion (purple, b).

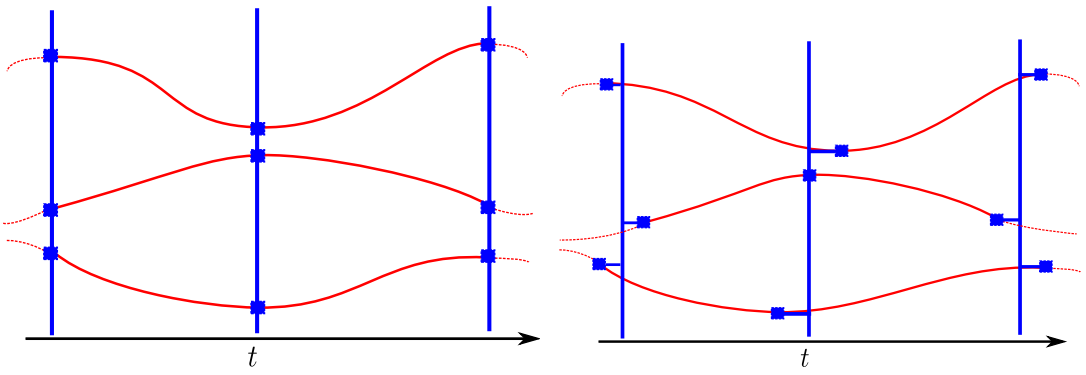
how I incorporate existing motion editing techniques into the systems presented later in this thesis (Chapter 4).

After presenting these foundations, this thesis will present models and algorithms for identifying motion extrema to be used as editing controls in Chapter 5. I will develop a set of techniques, each of which aims to select motion extrema that are similar to the extreme poses used by keyframe animators. The techniques take two approaches: finding *extrema of differential motion measures* and finding *spatially extreme body configurations*. I will consider the application of these two techniques to both the motion of individual joints and the motion of an articulated control skeleton.

This thesis then presents three specific approaches to expressive motion editing that use motion extrema as editing controls. The first application, presented in Chapter 6, considers the problem of how to robustly apply pose–centric motion edits to existing motion data that includes ground contact (Figure 1.5). In particular, I focus on the interactive application of pose–centric edits without requiring users to explicitly manage the relationship of the motion to the ground. This is important for expressive motion editing, as it enables users to quickly make meaningful performance–related changes while

focusing solely on expressive aspects of movement.

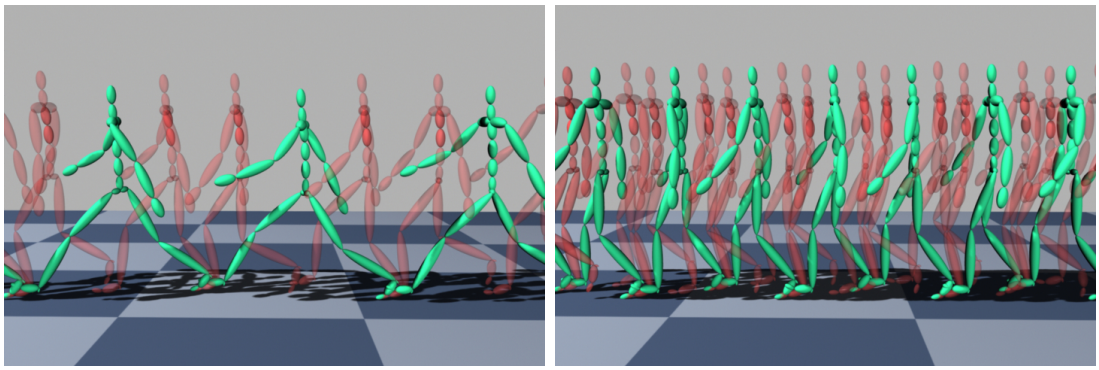
The second application considers the timing relationships among motion extrema on different parts of the body (Chapter 7, shown in Figure 1.6). I introduce a model, the *staggered pose*, that explicitly encodes these timing relationships. I will present a motion model that incorporates staggered poses to allows users to work with the timing relationships in a staggered pose as a coordinated unit. I introduce an algorithm for identifying staggered poses in existing motion data that uses the algorithms for identifying motion extrema as a foundation. I also describe how procedural timing edits can be applied to existing motion data using this representation.



(a) (b)

Figure 1.6: Traditional pose representations (blue, a) constrain knots in all degrees of freedom at the same time. Staggered poses, introduced in Chapter 7, explicitly encode timing variation, leading to a pose representation that can encode timing patterns common in natural movement (b).

The third application, described in Chapter 8, explores the use of geometric relationships among extreme–based controls as a foundation for representing high–level aspects of style (Figure 1.7). I focus on a specific aspect of motion style that is based on the spatial relationships among temporally adjacent extreme poses; modifying these spatial relationships is a form of spatial motion exaggeration. Finally, this thesis will provide concluding remarks and suggest directions for future study in Chapter 9.



(a) (b)

Figure 1.7: Spatial exaggeration, presented in Chapter 8, procedurally modifies extreme poses (green) to create exaggerated (a) or more subtle (b) motion variations using a simple, high–level editing parameterization.

In developing these algorithms, I focus on kinematic techniques for applying motion edits. This choice allows for the development of stable and efficient systems that allow for interactive editing. However, it would be advantageous to, in the future, incorporate these ideas into motion editing systems that also include efficient and stable dynamic models of movement. In addition, I focus on skeletal motion, as it incorporates many expressive qualities of full–body movement. Furthermore, control skeletons are commonly used as simplified character models in current animation systems.

Chapter 2

Related Work

This chapter describes existing work related to motion editing, expressive motion modeling, and specific algorithms described in this thesis. This includes motion editing algorithms, models of motion style used for both editing and synthesis, and algorithms for selecting poses from motion. In this thesis, motion editing is defined as making a change to a pre–existing motion. This does not include workflows and algorithms that allow for the interactive creation of new motion, such as keyframe editing.

The character animation literature includes a variety of approaches to modeling motion. These include the design of synthesis models, such as physically–based controllers, the development of motion synthesis algorithms, such as constrained optimization, and motion editing techniques. As this thesis focuses on the use of motion extrema for expressive motion editing, this chapter will cover motion editing algorithms and how they relate to expressive movement. This chapter will also discuss expressive models of movement that have been used for applications other than motion editing, as well as algorithms for identifying poses in existing motion data.

2.1 Motion Editing

Motion editing techniques reuse existing motion in new situations or for different char-

acters. This section surveys these techniques and categorizes them into four problem areas. Motion *concatenation* refers to the attachment of one motion clip to another, sequentially in time. A key aspect of concatenation is ensuring a smooth transition from the first motion clip to the second. Motion *blending* techniques take a set of motion clips as input and create a parameterized space of motion as output. In its simplest form, blending can be expressed as multi–target interpolation, but many techniques include feature alignment, through the use of time warping, as part of the blending process. Motion *transformations* take a given piece of motion and apply some edit to produce new motion, typically similar to the original. Transformation techniques include constraint–based editing, retargeting, and edits that change motion style. Motion *sequencing* is the problem of choosing a sequence of motion clips to combine into a longer piece of motion. A common approach to this problem is the choice of some constraints that express a particular goal and the subsequent search for a sequence of motion clips that meets these constraints.

2.1.1 Concatenation

Motion concatenation techniques aim to attach the end of one piece of motion to the start of another while maintaining plausibility at the time of attachment. For example, the motion should remain smooth, and constraints should remain intact. This can be done by first interpolating parameters and then applying correction techniques that enforce constraints.

The most common approach to concatenating motion clips interpolates skeletal parameters over some window of time [137, 138]. However, this requires some degree of temporal correspondence between the end of one motion clip and the beginning of the next. To address this, time warping can be used [179]. If constraints exist, they can be taken into account as part of the blending algorithm. For example, Rose et al. combine interpolation, an inverse kinematics solver, and constrained optimization to concatenate

motion clips, while maintaining contact constraints [145]. To avoid the need to manually adjust time windows, Wang and Bodenheimer investigate how long these transition durations should be for different types of motion [172]. They find that choosing window durations that minimize additional joint rotation appears most plausible for cyclic motion, while considering only joints with the highest acceleration appears sufficient for physically constrained movement.

Algorithms designed for more general problems have also been applied to the problem of motion concatenation. For example, interpolation can be done using basis function coefficients [170], velocity [144], and hierarchical displacement maps [92]. Often, large root motion changes are necessary, which can cause interpolation artifacts, especially if the orientation of the character is significantly different in the original motion clips. To address this, path alignment can be used to reduce the amount of root motion introduced by blending [44]. For enhanced realism, physical simulation [186] and damped spring models [7] can be used to track an interpolated target motion. To incorporate constraints into physics–based approaches, constrained optimization can be used with continuation and block scheduling [106]. These approaches all assume the motion is represented on articulated skeletal hierarchies. As an alternative, hand–drawn motion clips can be concatenated using layered spacetime implicit surfaces [35].

Concatenation is not typically used as a form of expressive motion editing; it more often serves as a foundation technique for algorithms that sequence motion clips to meet constraints. As this thesis focuses on editing a single motion clip, the systems presented here do not explicitly use these concatenation techniques. However, it might prove useful to develop concatenation algorithms that aim to match extrema of movement that are identified using the algorithms presented in Chapter 5.

2.1.2 Blending

Motion blending algorithms combine a set of example motion clips to create new motion. This can be done, for example, using interpolation. This creates a parameterized set of motion clips for exploratory editing, and this set can be used to find a blended motion that meets certain editing constraints. Motion clips used for blending are typically assumed to have similar content, and many approaches incorporate some form of time warping to explicitly align important events in the original motion clips.

If users have direct control over how to blend motion clips, they are typically given explicit knowledge of the motion examples and select weight values that are used to combine them. This blending can be done using skeletal parameters, frequency domain coefficients [170], multi–resolution basis coefficients [20], and measured values such as joint positions [48]. These approaches can be problematic if the motion clips are not aligned in time. To address this, the original motion clips can be time warped to a common time base [20, 8, 9]. If the paths of the character in the environment are significantly different in the source clips, blending artifacts can occur. To avoid such artifacts, the motion paths and associated contact constraints can be brought into alignment before blending [72]. None of these approaches consider physical plausibility of movement. To increase physical plausibility, flight phases and contact constraints can be explicitly preserved [147].

Motion blending algorithms can be driven by environmental constraints. This can be done, for example, to create blended motion in which an end effector follows a desired trajectory. Wiley and Hahn use a combination of nearest neighbor search and interpolation [176], and a variety of approaches use radial basis functions to find blended motion that meets parametric constraints [144, 146, 135, 134]. Assembling parameterized sets of motion clips can be tedious and time–consuming for large databases of motion. To automate this, the databases can be explicitly searched for sets of similar motion clips; clips in each set can then be blended to meet constraints [73].

Statistical models, such as Kriging, a form of Gaussian Process regression, can be used to relate parametric distance to pose distance, which allows the model to synthesize new motion that meets parametric constraints [122]. As another example of a statistical model, Lau et al. use a Dynamic Bayesian Network to model the variation among a collection of similar motion examples, and they sample the distribution learned using this model to create new variations of the given type of motion [86].

Each of the above techniques treat each source clip uniformly, regardless of body part. To allow for different source data to be applied to different parts of a character's body, blending algorithms can use data from one source clip for one portion of the body and data from another source clip for another part of the body. For example, Ikemoto and Forsyth, when editing motion to expand motion databases, randomly swap portions of motion at the waist and shoulders [61]. Heck et al. splice upper body walking movement from one source onto lower body motion from another source by aligning the two sources of motion and rotating the torso to keep the shoulders and spine in similar positions [50]. Majkowska et al. splice high resolution, local motion onto low resolution full body motion to attach detailed hand movement to full body movement that has been recorded with less accuracy [115].

The above approaches all focus on quantitative aspects of movement when determining how to blend motion. As an alternative approach, expressive style variation can be modeled by blending among a set of similar motion clips that vary in style. For example, Min et al. model variation of both style and actor using a multilinear model [120]. Users can directly edit the style and identity parameters of the model to blend between styles or actor characteristics.

To generalize blending to the motion of more than one character, crowd data can be blended. For example, Ju et al. interpolate among different crowd simulations using a model that encodes neighborhood relationships and the trajectories of individual characters [64].

Overall, blending is primarily used to combine example motion clips to create a parameterized space that can be directly edited or used to solve for motion that meets a set of constraints. In terms of expressive motion editing, these parameterized spaces can directly model style variation. However, example motion must be provided that spans the space of all potential styles that might be desired when creating the blended motion. The work presented in this thesis, in contrast, aims to apply expressive changes to a specific motion without requiring additional motion examples.

2.1.3 Motion Transformation

Motion transformations alter one motion clip to meet some goal. Many motion transformation algorithms focus on environmental constraints. In these approaches, enforcing some set of constraints that relate the character to the environment or other characters is the primary goal. Some techniques correct for errors such as footskate or physical implausibility, and others can adapt motion to a new character with different skeletal parameters or topology. Most relevant to the work in this thesis are expressive motion transformations. Expressive motion transformations change motion based on either performance–related direction given by a user or an algorithmic model of style.

Motion Layering

Motion layering is an approach to editing that allows skeletal parameter animation to be defined using a sum of signals or splines. Motion layers are commonly used in practice, as evidenced by their inclusion in commercial animation software such as *MotionBuilder*[1] and *Maya*[2]. This idea was introduced by the research community as motion *warps* [179], also known as motion *displacement maps* [20]. While general and powerful, they require the user to specify all information about how character parameters should be changed.

[1] http://www.autodesk.com/motionbuilder
[2] http://www.autodesk.com/maya

The techniques presented in this thesis use motion displacement maps algorithmically, but these techniques use extrema of movement such as full body poses as the editing foundation, rather than the layers themselves. Instead of defining layers using skeletal parameter summations, I will use a representation that concatenates transformations, which will be described in Chapter 4.

Editing for Environmental Constraints

Motion can be altered to to enforce constraints that relate the character to the environment. These constraints are commonly expressed as position constraints on end effectors, but they can also relate to other motion parameters, such as the path the character takes in the environment. Most approaches either solve a constrained optimization problem or apply a kinematic deformation to the skeletal parameters. Pose–centric editing, presented in Chapter 6, will contrast with these approaches in that the focus of the edit will be on intrinsic character stance, as opposed to the extrinsic relationship of the character to the environment.

A common approach to constraint–centric editing involves the creation of a physical model for the character and the solution of a constrained optimization problem to create motion that meets the constraints in a physically plausible way. This technique was introduced by Witkin and Kass, using a low–complexity character [178]. However, the constrained set of nonlinear equations can be difficult to solve, especially as character complexity increases, as the objective function often has many local minima in a high–dimensional space. A variety of techniques have investigated more efficient approaches to solving the constrained optimization problem. These include the use of partial local solutions [31], different motion representations [110, 149], simplified character models [139], and limited types of constraints and objective function terms [107, 39]. While these techniques are typically used to create new motion, they are all amenable to the problem of editing existing motion, as the optimization solvers typically require an initial mo-

tion. If the initial motion is already of high quality, then a general purpose optimization solver can be used for constraint–centric editing without a physical character model, for applications such as editing recorded motion capture data [42, 43].

A general purpose constrained optimization solver could, in principle, be used to solve the pose–centric editing problem addressed in Chapter 6. However, the optimization solution can be computationally expensive and have an unreliable solution. In addition, these approaches do not allow for the prioritization of motion editing constraints, which is a key component of the algorithm presented in Chapter 6.

Lee and Shin introduce an alternative approach to constraint–centric editing that uses a hierarchy of motion displacement maps [92]. These are progressively modified using inverse kinematics to meet single frame constraints, and local optimization problems are solved to enforce sustained, or variational, constraints. This approach can also be applied to the pose–centric editing problem described in Chapter 6. However, this approach also does not include constraint prioritization. Le Callennec and Boulic present an algorithm that applies a per–frame procedural deformation to meet prioritized constraints [87]. However, their approach only supports single–frame constraints, while robust pose–centric editing requires sustained prioritized constraints that model sustained contact for some duration of time.

Motion can also be edited using physical models without explicitly solving a constrained optimization problem, as constrained optimization can be computationally expensive. For example, Sulejmanpašić and Popović adapt jumping motion using iterative gradient descent to repeatedly make small adjustments that allow the resulting motion to closely meet desired constraints [162]. As another example, Abe and Popović modify existing motion using a tracking controller and inverse dynamics, with the goal of allowing users to change how objects are manipulated.

The above approaches primarily consider end effector locations in the environment. Higher–level editing control can be achieved by changing the path a character follows

in the environment. For example, Gleicher allows users to edit a curve on the ground that approximately follows the motion of the character in the environment [44]. As one application, Ahmed et al. use path editing in conjunction with speed and slope changes to create variations of an example motion to populate a motion database [2]. These approaches only consider the path, however. To also incorporate environmental constraints such as ground contact, Chai et al. solve an optimization problem using a linear dynamic model to find a motion that meets constraints on both the path and end effectors [26]. Kim et al. apply geometric Laplacian editing to the character's root trajectory to allow the path to change to meet end effector constraints [69]. Lockwood and Singh apply an as–rigid–as–possible geometric deformation to the root trajectory to meet positional constraints [111]. While these approaches do not primarily focus on user interface design, Min et al. allow users to sketch a desired motion path and use a Gaussian Mixture Model to generate motion that approximately follows the sketched path [119].

The above approaches primarily apply to the motion of a single character. To generalize to multiple characters, constraints can be specified among them and explicitly preserved during editing. Liu et al. address this using continuation and block scheduling with physically constrained optimization [106]. Kim et al. also incorporate constraints among multiple characters as part of their Laplacian editing framework [69]. These approaches only consider end effector constraints between characters. To handle complex contact constraints that involve many body parts, Ho et al. build a mesh representation that connects interacting joints in space and time and use Laplacian–based editing to preserve this contact information [52]. To preserve contacts between character motion and passive projectile motion while editing, Jain et al. combine physics–based rigid body control and a kinematic motion model into a solver that updates both models simultaneously [63].

Constraint–centric editing can also be used to produce motion for other editing ap-

plications. For example, Abe et al. use physically constrained optimization to create a set of motion examples that interpolate with high quality for motion blending [1].

Overall, these algorithms primarily focus on the relationship of the character to the environment or other characters, rather than characteristics of movement related to expressive performance. While some of the solution techniques could, in principle, be used to apply pose–centric edits, as done in Chapter 6, many do not scale well to complex characters in terms of computational cost, which precludes interactive editing. Furthermore, the constraints needed for robust pose–centric editing require a prioritized solver; these approaches do not incorporate prioritization of sustained constraints.

Motion Correction and Refinement

When editing motion, errors can be introduced that create physical implausibility or break constraints, such as contact with the ground. To address this, algorithms have been developed to correct for these types of errors as a post–processing step. In addition, motion can be refined to add new detail. For example, responses to physical impacts can be introduced, and detailed movement can be added to motion that is only approximately specified.

To correct for physical implausibility introduced during an edit, Shin et al. present a technique that improves static balance while a character is in contact with the ground and physical plausibility of flight when the character is not on the ground [152]. This technique was shown to introduce the body lean that would be expected when character motion is edited to introduce turns, inclines, and walk–to–run transitions.

Many editing techniques introduce ground contact error, and, as result, contact detection and correction have been well–studied. Most approaches to contact identification label individual frames and combine contiguous sequences into sustained, or variational, contact constraints. A commonly–used approach labels individual frames using a threshold on either the velocity of the potentially contacting joints or their distance to the

ground [15]. More sophisticated techniques include contact estimation based on noise patterns [88] and classification based on relative joint locations [58]. However, these frame–labeling approaches do not label all frames correctly. Mislabeled frames can be corrected by a user or algorithmically. For example, Le Callennec and Boulic [88] merge sequences of labeled frames in which small temporal gaps occur. In the editing systems presented in this thesis, velocity thresholds will be used to label individual frames and temporal gaps will be filled either manually (Chapter 7) or with gap filling [88] (Chapter 6).

Once identified, contact errors can be corrected with *footskate cleanup* methods. These are often based on inverse kinematics [75] or hierarchical displacement maps [92]. These post–processing approaches have a fundamental limitation however, in that they can undo some types of intended edits, as the post–process does not have knowledge of the desired editing goals. This becomes especially noticeable when applying pose–centric edits. The approach to pose–centric editing introduced in Chapter 6 solves for motion that incorporates pose centric edits while simultaneously preserving ground contact constraints. This accounts for how pose edits and ground contacts interact; general post–processing approaches cannot model this.

A variety of techniques have been developed to introduce responses to new physical impacts that are not present in the original motion. Zordan and Hodgins use physical simulation with a tracking controller [186]. The resulting motion appears similar to the original when no impacts occur, and the character then responds in a plausible way when new impacts are introduced, before returning to the original motion. Inverse dynamics can also be used with physical simulation to track the original motion [132, 71]; these techniques also must include explicit balance control to keep the character upright. To preserve details in pre–existing motion, controllers can also be activated only as needed. This introduces physically plausible response movement before a character returns to kinematic motion playback [188]. Physics–based simulation can be expensive and unpre-

dictable, however. To provide efficient and predictable response, motion displacement maps that respond to different impacts can be introduced by selecting the map based on the impact direction [7]. To limit the new physical response to only portions of a character's degrees of freedom, Ye and Liu modify only those degrees of freedom that are not actuated by the underlying movement [180]. Finally, Nguyen et al. apply live dynamic response as a motion is recorded by incorporated a physical model that responds to contacts with virtual objects [131].

If motion has only been approximately specified, detail can be added algorithmically. For example, the system of Liu and Popovíc allows users to specify approximate initial motion that is refined using physically constrained optimization [107]. Pullen and Bregler modify cyclic motion to create subtle variations by modeling the interdependence among different bands of a wavelet decomposition; they then synthesizing new motion using this model [141]. In later work, Pullen and Bregler explicitly use existing detail by copying it from recorded motion that correlates well, at low frequencies, to the original motion [142]. These techniques are similar in spirit to the editing approaches introduced in this thesis, in that they allow users to work with simplified movement. The approaches presented in this thesis differ in two ways. First, they reuse detail from the original motion rather than synthesizing it or creating it using another motion source. Second, they use a small set of motion extremes, such as full–body extreme poses, as the foundation for editing, rather than a full approximate motion. While Pullen and Bregler reuse detail of movement by copying it from recorded motion, they do not maintain the structure of this detail. The techniques presented in this thesis, in contrast, explicitly preserve this detail throughout the entire motion by applying low–frequency changes to the overall movement.

Retargeting

Existing motion can be adapted to a new character in a process called *retargeting*. Some techniques support changes in limb length, while others also support changes in skeletal

topology. For retargeting motion to a new character with the same skeletal topology as the original motion, inverse kinematics can be used on each frame, but it has the disadvantage that discontinuities will often occur in the resulting motion [42]. To address this, a number of techniques use inverse kinematic solutions that are smoothly introduced and removed [15]. Others use IK solutions that take into account a complete solution for a small window of prior frames [28, 153]. Kulpa et al. scale limb length and apply IK on simplified character models before mapping the change back to the full skeleton [77]. Gao et al. use heuristics to change the root trajectory before applying IK to position end effectors [92]. Hecker et al. address topology changes by copying animation from one limb to others and apply IK to set the positions of important joints to desired locations [51].

Another approach to retargeting involves the use a dynamic motion model that approximates the original. This can be useful, as secondary motion can then adapt to changes in character size. For example, Tak and Ko apply the Kalman filter to compute new motion, using the original motion as a predictive component and any constraints for correction [164]. Zordan and Hodgins use inverse kinematics to adjust motion and then use a tracking controller to smoothly follow the adjusted motion [186]. Tracking controllers have also been used to meet contact constraints before deferring to simulated contact response [76].

General–purpose constraint editing approaches can also be adapted to the retargeting problem. Gleicher accounts for changes in limb length by using constrained optimization [41] in conjunction with heuristics to adjust both root trajectories and contact constraints [42]. He also considers topology changes by adapting motion to an intermediate character with only size changes before applying it to the new character with simpler skeletal topology. Lee and Shin also use their hierarchical displacement mapping system to account for limb length changes [92]. To handle multiple interacting characters with limb length changes, Ho et al. minimize the deformation energy of their space–time mesh representation [52].

Other media, such as hand–drawn animation, can also be considered for retargeting. For example, Bregler et al. retarget hand–drawn animation from one character to another, where the target character can be either hand–drawn or a 3D model [19]. Users specify a set of corresponding key poses for each character, and a combination of affine transformations and multi–target interpolation are used to create the new motion.

While retargeting does not specifically address expressive motion editing, it does allow for motion performed in a given style to be applied to another character. The pose–centric editing system presented in Chapter 6, while not designed for retargeting, can be used for this application. To do so, a single pose could be set with the desired limb length, and the overall motion would adapt to the size change while preserving ground contact.

Expressive and Stylistic Editing

Existing motion can be transformed to alter the style—the expressive qualities of movement—while keeping the fundamental type of motion unchanged. For example, a neutral walk can be changed into a walk that appears more energetic, cartoony, or lazy. The editing techniques presented in this thesis fall into this area, as they have been designed to allow users to quickly make expressive, performance–related changes to existing motion.

Existing algorithms for editing motion style take a variety of approaches. Some apply procedural changes that directly modify the motion. Others use additional motion data to construct a mathematical style model that can be used to change the style of new motion. The approaches presented in this thesis will take two approaches. First, the systems presented in Chapters 6 and 7 allow users to directly make quick, expressive changes to existing motion using motion extrema as high–level controls. Second, the staggered poses system (Chapter 7) and the spatial exaggeration system (Chapter 8) each include tools for applying procedural style changes.

Procedural approaches that transform the style of existing motion can modify geometric content, timing, or both. Signal processing can be used to edit geometric qualities

of movement. The systems introduced by Bruderlin and Williams [20] and Unuma et al. [170] allow for a form of style modification in that users can adjust motion contributions from different basis components. Similarly, Shapiro et al. apply Independent Components Analysis and allow users to directly edit the ICA coefficients or transfer them from one motion clip to another [151]. In these systems, however, the relationship between basis coefficients and meaningful aspects of movement is indirect. Users must experiment with parameters, and it might not be possible to get a desired effect.

Filtering techniques modify the geometric properties of motion by applying a linear filter to the parameter signals. Litwinowicz [104] and Kanyuk [65] apply procedural oscillatory response filters, and Cardle et al. [24] and Wang et al. apply filters that emulate anticipation and follow–through. The spatial exaggeration techniques presented in Chapter 8 also modify geometric qualities of motion, but they focus on modifying existing characteristics of movement rather than adding new motion content.

Another approach to stylistic editing focuses on finding degrees of freedom with coordinated movement and changing them as a collective group. For example, Neff and Kim [129] change geometric content by allowing users to edit the degree to which the motion of different skeletal parameters is correlated to a body part of interest. The spatial exaggeration techniques presented in Chapter 8 differ in that they explicitly change how extreme poses at different times relate to each other, while Neff and Kim change the continuous relationship of one body part to others.

When modifying timing, the timing of the body can be modified as a whole, or the relative timing among different body parts can be changed to model effects such overlapping action. As part of their motion warping system, Witkin and Popović allow the overall motion to be retimed to meet a constraint specified at a given time [179]. Liu and Cohen apply physically constrained optimization to determine efficient timing for motion specified using keyframes [109]. However, these approaches do not take environmental constraints into account. McCann et al. address this, optimizing the timing of motion

while handling a variety of environmental constraints [117]. As part of their constrained optimization system, Liu et al. also allow for constraints to be retimed [106]. Lockwood and Singh modify locomotion timing after kinematic path edits have been applied to preserve relationships based on biomechanical observations about how stride length and speed relate [111].

These approaches all aim to improve timing in existing motion or motion that has been kinematically edited. Another approach is to use the timing in reference data as a source for modifying the timing of another example motion clip. For example, Hsu et al. present a technique to retime motion to match the timing characteristics of additional reference motion data by attempting to match the velocity of each frame to that of a similar frame in the reference [54]. In contrast to each of the above approaches to changing timing, the timing–centric procedural editing tools presented in Chapter 7 will focus on the relative timing among different body parts.

In work that is concurrent to this thesis, Kass and Anderson introduce an approach to changing the relative timing patterns in motion using oscillatory phase [66]. Changing oscillatory phase, in general, has a different effect on the resulting motion than editing timing directly. Chapter 7 will include a discussion of how changes to oscillatory phase and timing differ in terms of how they affect the motion.

Each of the above approaches to procedural style change consider geometric changes and timing changes independently. Coupled systems can also be developed that procedurally modify both geometric and timing–related qualities of motion. For example, Kwon and Lee modify both to create a rubber–like quality of movement [78] . To do this, they filter motion to emphasize arcs, resample the control skeleton to introduce extra joints, and apply a physical simulation to this resampled control skeleton. The approaches to procedural editing presented in this thesis will differ in that they change temporal and geometric aspects of motion independently. This allows users to have more specific control over the results of an edit.

To change the apparent energy present in a motion example, the physical properties intrinsic to the motion can be explicitly edited. As an example, Sok et al. allow users to edit the momentum and force profiles of the root trajectory, and they optimize the resulting trajectory to preserve the underlying dynamics [158].

Data–driven models can be constructed using example motion data to learn style transformations. Amaya et al. record neutral motion, along with emotional variations, and create timing and amplitude transforms that model the emotional variation as a form of style [4]. They then apply these transforms to new motion to change its style. Hsu et al. model the stylistic variation between two similar motion examples using a linear time–invariant model; they then use this model to alter the style of new motion data [56]. Chuang and Bregler, in their system for mapping facial motion from video to 3D geometry, use a bilinear model to separate style and content [30]. To transform the style of new motion, they transfer the style component from another motion example. A variety of other motion editing systems, while not specifically designed to transform a single motion clip, do make use of data–driven style models; these will be described further in Section 2.2. Unlike the above approaches, the systems presented in this thesis do not require any additional motion data.

To provide direct control over expressive qualities of movement, a variety of systems allow users to directly specify how motion should be stylized. Li et al. allow users to draw desired changes to 3D surfaces at a sparse set of frames and use a combination of motion displacement maps and geometric deformation to match the motion to the drawn changes [102]. Both Li et al. and Jain et al. allow users to sketch the beginning and ending poses of a motion as a drawing, and they each apply a displacement map to change a motion clip to match these beginning and ending poses [99, 62].

While the above approaches allow users to specify geometric state at selected times, other systems allow for a user's performance to change the entire motion. For example, Dontcheva et al. [38] and Terra and Metoyer [166] each present systems that allow users

to perform style of movement or timing, either with real–world prop motion or with mouse movement. In each approach, correlations are measured between the character motion and the user's performance movement, and this correlation is used to retime the character motion.

Pose edits can also be used to change an entire motion, rather than a specific frame. Ikemoto et al. allow users to edit a sparse set of characteristic poses; these pose edits are applied to the given motion such that any frame with a pose similar to those that were edited will have a similar edit applied [60]. Ma et al. use a similar approach to allow users to edit characteristic poses in facial animation data [114]. The pose–centric editing system presented in Chapter 6 and the staggered poses system (Chapter 7) each provide users with direct control over important poses in the motion. Like Ikemoto et al. and Ma et al., these systems are pose–centric. However, the systems presented here allow users to directly edit specific poses in the motion, while the system of Ikemoto et al. applies a single pose edit to any portion of the motion that has similar body stance.

Mukai and Kuriyama apply this idea of edit propagation to subsections of a motion in their Pose Timeline system [123]. In their system, users edit a small temporal region of a motion, and regions of time with similar movement will have the same edit applied.

The above approaches require users to specify changes. Data such as sound can also be used to drive the changes. For example, motion can be retimed to match auditory features in sound data. Cardle et al. extract features from MIDI and audio data; they use these to determine how to apply their anticipation and follow–through filter [24]. Lee and Lee extract feature points from both MIDI and motion data; they use dynamic programming to match and time–warp both the sound and movement to create sound–synchronized motion [89]. In the work presented here, I do not consider audio matching, although the motion extreme selection algorithms presented in Chapter 5 could be used for such an application.

2.1.4 Sequencing

Motion *sequencing* is the process of selecting a sequence of motion clips to concatenate into a longer stream of motion. This can be done to meet either environmental constraints or stylistic goals; most work in this area has focused on meeting constraints on body pose, joint positions, or annotations. Approaches have been developed to directly select a sequence of clips to meet immediate or future goals. A popular approach to practically handling large motion databases has been the construction of graph representations of motion that enable efficient search. Finally, motion clip selection algorithms can be designed for specific applications, such as speech synthesis and audio synchronization.

Sequential Motion Search

A variety of strategies can be used for the selection of a sequence of motion clips; approaches include direct selection, which can use a data–driven model, dynamic programming for refining a chosen sequence of clips, and reinforcement learning for choosing motion clips based on expected future reward. Lamouret and van de Panne directly select motion clips from a database to find a clip sequence that meets constraints on end effector positions [83]. Yin and Pai, exploring alternative user interface design, directly select clips of motion to match input from a foot pressure sensor; this approach tends to select clips that match the pose of the user [181]. These systems select motion clips based on data provided by the user. Algorithmic data can also be used to select motion clips. As part of their system for applying physics–based responses to recorded motion data, Zordan et al. present an algorithm that searches a motion collection for a clip that best matches the output of a physics–based controller [188].

Data–driven models can be used to incorporate knowledge about the content of a motion database into the selection algorithm. For example, Molina Tanco and Hilton use a Hidden Markov Model to select a sequence of motion clips that matches starting and ending pose constraints [121]. Brand and Hertzmann also use a Hidden Markov Model,

and they sample state transition probability distributions to find a motion clip sequence that is similar to a given motion, but differs in style [18]. Li et al. model state transitions between motion clips using a linear dynamic system, and they allow local constraints to by applied to each motion clip to determine how they will be selected [101]. These approaches consider only skeletal motion representations. de Juan and Bodenheimer instead focus on traditional character animation; they re–sequence existing drawn frames to create new motion that meets starting and ending frame constraints [34]. To do this, they use a low–dimensional ST–Isomap embedding of the motion and search for shortest paths in this embedding space.

Dynamic programming can also be used to find sequences of motion clips that meet various constraints. Arikan et al. use a course–to–fine dynamic programming algorithm to create frame sequences that meet constraints on pose, location in the environment, and annotation [6]. Hsu et al. also use dynamic programming to choose a clip sequence, but they instead aim to match a continuous control signal [55].

To allow for sequencing from one portion of a given motion sequence to another, Ikemoto et al. precompute blended motions that transition well from any frame to every other frame [59]. This precomputed cache can be used to allow for fast response when changing the action performed by a character.

To create acrobatic motion with long sequences of flips that would be difficult to perform, Majkowska and Faloutsos explicitly search for portions of a motion sequence that have the ballistic pattern of a flip [116]. These portions of the motion are then repeated and attached together to increase the number of flips present in the motion.

When recording motions to construct a controller that directly sequences motion clips, it can be difficult to choose an effective set of motion examples to record. To address this, Cooper et al. use active learning to suggest which motion examples a user should record to improve the controller [33].

Direct selection of a sequence of motion clips can be used to vary expressive con-

tent [18], and expressive content can also be specified through the use of annotation constraints [6]. However, most sequencing approaches have used constraints. The work presented in this thesis has not been targeted to sequencing applications, but it would be interesting to use motion extrema, such as extreme poses, as a simplified representation to aid in direct search or model construction.

Graphs for Sequencing

When continuous motion can be modeled as a set of states with a restricted set of allowable transitions, graph data structures can be used to explicitly model these states and transitions for use in constraint–based motion playback. This allows efficient algorithms for finding graph traversals to be used to reduce the computational cost of identifying a sequence of motion clips that meets a set of constraints. Graphs can be constructed manually, whether from keyframe data or recorded motion, or they can be algorithmically constructed to use them with large motion databases. In motion graph representations, nodes commonly represent poses, and edges represent motion clips, but the dual representation is also common. These representations are functionally equivalent.

To manually construct graphs, a user specifies the transition poses and the motion edges that connect them. Such graphs have been used for a variety of applications. Badler et al. introduce this idea and explicitly search the graph to play back a motion clip sequence that meets an end pose constraint [11]. This limits clips that can be played back to a specific set, however. Rose et al. extend the idea to support parameterized motion clips along the edges; these parameterized motion clips can be used to vary the expressive style of the motion [144]. Choi et el. encode the graph using foot plant patterns and use a probabilistic roadmap to assist with a search to find motion that meets start and end pose constraints [29]. Graphs can also be used as reference data to generate simulated motion. For example, Zordan and Hodgins use a manually–constructed graph to determine the motion that will be tracked by a physical controller [187].

Manually–constructed graphs can directly respond to user selection of the type of motion that should be played back. This is commonly used in game environments to choose a character's actions. Park et al. use graphs for this application; they store a collection of similar motion clips in each edge to allow for parameterization in terms of speed and turning angle [134]. Wang and van de Panne determine graph playback by allowing the user to select clips via voice command [174].

Algorithmic Graph Construction

Manually constructing graphs for large databases can be intractable. Algorithms for the automatic construction of motion graphs from large databases have been developed to address this. This is commonly done by first finding transition poses to encode in nodes; these poses are typically identified as the minima of a pose–to–pose distance metric [74, 90]. To determine a graph traversal for playback, different search strategies can be used to meet different types of constraints. For example, Kovar et al. use an iterative branch–and–bound search to find a traversal that meets constraints on the motion path and style of motion [74]. To support faster run–time playback, motion clips can be adjusted during graph construction to guarantee smoothness of transition without requiring run–time corrections to the individual clips [45]. To better meet constraints on paths and end effectors, Safanova and Hodgins solve for motion sequences in an algorithmically constructed graph that are interpolated to create the final solution [148].

Graph information can also be organized hierarchically to support faster search or flexibility during playback. Lee et al. cluster poses in transition nodes into a simplified set of states; this smaller set is used to search the graph to find a sequence that plays back a chosen action, mimics a user's motion, or follows a path [90]. Arikan and Forsyth create a hierarchy of edges for each transition, and they use an iterative sample–and–reject strategy to determine the specific edges that will be used [5]. These representations do not consider what actions take place in motion clips, however. To address this for

locomotion synthesis, Kwon and Shin use a two layer graph in which the top layer stores actions in each node and each local layer node stores a set of clips that match this action, but vary in terms of foot plant patterns [80].

To generalize algorithmically–constructed graphs to parameterized motion, the graphs can be constructed such that either a node or an edge represents a *family* of clips that transition from one pose to another. For example, Shin and Oh [154] store parameterized sets of motion clips in each edge, and Heck et al. [49] store similar sets of clips in each node. These sets of motion sequences can be blended to meet continuous constraints. Beaudoin et al. identify families of similar motion sequences, and graphs are then constructed using these families as the foundation, rather than single motion clips [14].

The quality of a graph can also be a concern during construction. To ensure good connectivity for fast response, Zhao and Safanova blend similar motion clips before constructing a graph and then prune the graph to reduce its size [185]. In subsequent work, they reduce the size of a graph by retaining an approximately minimal subgraph [184].

Graph construction and playback algorithms can also be designed for specific applications. For example, Cao et al. segment synchronized speech and facial animation using phoneme breaks and then cluster the segmented clips and speech samples [22]. This graph is then searched to play back new speech motion. Ren et al. use a graph–based encoding of two–character dance motion and add extra edges that allow frames to be skipped during playback to allow for timing flexibility [143]. A vision–based interface then allows users to lead a virtual dance partner.

Graphs for Groups

The graph representations described above encode motion for a single character. Motion graphs can also be constructed for large groups of characters, such as crowds. Lai et al. create flocking motion graphs by using the set of all character positions in a distance metric [82]. These graphs are used to create new flocking motion that follows user–

provided paths or stays within desired regions of the environment. Sung et al. create group motion using a probabilistic roadmap representation of the environment [163]. They use this representation to incrementally refine the motion of each character to match specific pose, position, and orientation constraints. Motion patches, introduced by Lee et al., represent localized motion graphs that are specific to a portion of an environment and contain motion for multiple characters; each patch is used as a node, and transitions among them are stored as edges to determine how group motion can be played back in complex environments [94]. This idea is extended to characters that interact with each other in the Interaction Patches system, in which interactions are precomputed and stored in each patch [157].

To allow for large–scale editing of large groups of moving characters, Kwon et al. represent group motion using a graph that encodes neighborhood relationships and character trajectories [79]. To edit the group motion, constraints are specified, and a minimal–distortion edit is applied to the graph.

Overall, the potential of graphs for use in expressive motion editing lies in the ability to search for annotation constraints that encode high–level style information [74] or to select a motion clip in a parameterized set that models stylistic variation. While the systems developed in this thesis have not been designed to support motion graphs, they could potentially be used to modify the motion clips in a graph to create a graph with a particular style. In addition, the sparse sets of poses that can be selected using the algorithms presented in Chapter 5 could be used to simplify graph construction algorithms by reducing the number of poses that would need to be considered.

Optimal Control

To select motion clips for immediate playback while considering future goals, reinforcement learning can be used. Lee and Lee apply this to the incremental selection of boxing clips that meet requested target states [91]. McCann and Pollard also use reinforcement

learning to build a selection table that allows users to select new motion clips using interactive controls [118]. These approaches do not scale well with control signal dimension, however.

To allow for a continuous space of high–dimensional control input when selecting motion clips, Treuille et al. approximate the control space using a low–dimensional basis function representation [169]. This allows for directional control of a walking character that can avoid obstacles while facing a desired direction. To model adversarial strategies used in multi–player games, Wampler et el. used coupled controllers based on reinforcement learning that model interactions among characters with game–theoretic models [171]. To allow for new clip selections at every frame, Lee et al. model motion as a high–dimensional vector field and use reinforcement learning to repeatedly choose among all possibilities [96].

In the context of expressive motion editing, these techniques could be combined with control terms that model style or emotion variation, while simultaneously allowing for specific constraints to be specified on the resulting sequence of motion clips. However, existing work has not yet investigated these possibilities.

Sequencing for Sound

Sequencing algorithms can use cues in audio or speech data to determine how to play back motion. For example, to match music, rhythm patterns in the audio can be used to select motion clips that have similar rhythmic patterns in the character movement. In Kim et al.'s system, described above, dominant frequencies are estimated in the motion data and used for clustering the graph; these rhythm patterns are then used as criteria for selecting graph transitions [70]. Shiratori et al. also match motion clips to rhythm patterns; they estimate rhythm using momentum patterns in the character's motion [155]. In their system, motion clips are directly selected for playback to match rhythm patterns in audio data.

Music is not the only audio source that can be considered for selecting motion clips. Speech data can also be used to create motion that follows along with existing recordings or new synthesized speech. Motion recorded in sync with audio is often segmented using phoneme labels, and new speech can be synthesized by selecting a clip sequence that matches the phoneme patterns in a new speech recording. For example, Stone et al. use hierarchical dynamic programming to select audio and motion clips that match a new spoken phrase [161]. Ma et al. explicitly select clips that model viseme transitions that match a desired speech pattern [112]; they later build upon this idea to support graph–based sequencing [113].

Expressive content, such as emotion annotations, can also be incorporated into speech–driven sequencing algorithms. Deng et al. first synthesize neutral speech motion using a graph search [37]. They then compute expressive data models as the difference between neutral motion and expressive variants of the motion. They sample the resulting models to add expressive content to new motion. Cao et al. similarly sequence speech motion and apply expressive texture; their approach uses an ICA model of style, which they use to transform neutral speech synthesized from a graph to match desired phonemes [23].

2.2 Expressive Motion Synthesis

Stylistic, or expressive, models of motion can be used for motion synthesis, as well as motion editing. Stylistic approaches to motion synthesis can be based on procedural changes to kinematic models of motion, physical simulation, data–driven models, or some combination of these. This section surveys the use of style models for the creation of new expressive character motion.

2.2.1 Procedural Models

Procedural models for motion synthesis typically use a parameterized generation algorithm that is based on principles of movement related to expressive performance. For example, in the EMOTE system, Chi et al. use the Laban Movement Analysis principles of effort and shape to parameterize the timing among and changes to a sequence of user–provided poses that are used to create interpolated movement [27]. The shape parameters, which the user can edit, modify the extent to which the arms move with respect to the body. The effort parameters modify the timing of how the character moves from pose to pose. The timing effects of the effort parameters were also applied to facial animation in Byun et al. [21].

In a series of papers, Neff and Fiume present a system for applying performance–centric control over motion that is synthesized using either keyframe interpolation or physical simulation [125, 126, 128]. Specific types of controls include the modeling of a timing variation pattern called *succession*, a parameterized change in the scale of a movement called *amplitude*, and the relationship of the limbs to the body, which they refer to as *extent* [125]. Users can model the characteristic stance of the body in this system using inverse kinematic solvers [126].

Overall, the procedural models presented by Chi et al. and Neff and Fiume have similar motivations as those I present for editing timing in Chapter 7 and extreme poses in Chapter 8. In contrast to their approaches, I apply procedural edits to motion data in which poses and existing timing relationships are algorithmically determined from the existing motion, rather than specified in advance for use in the synthesis of new motion.

As another approach to applying procedural style, Kim et al. apply anticipation effects to poses selected by users [68]. For each pose pair, they procedurally create an anticipatory pose that creates such an effect and solve an optimization problem to determine the resulting motion. This is similar in spirit to the spatial exaggeration system presented in Chapter 8. Spatial exaggeration differs, however, in that it uses algorithmi-

cally selected poses, and it exaggerates existing movement, rather than introducing new motion.

2.2.2 Physics–Based Models

Physics–based models of style incorporate some notion of expressive motion qualities into simulation parameters that are used during motion synthesis. As one example, Neff and Fiume parameterize motion synthesis using physical joint controllers that incorporate tension [124]. By varying the tension parameters, users can cause a character's motion to appear either stiff and rigid or loose and relaxed. Allen et al. also incorporate tension control when creating motion that interpolates a sequence of poses [3]. In their system, they analytically solve for PD control parameters that allow the motion to match the desired poses.

While the above approaches are powerful, they rely on the user to specify how to change the style of the motion. An alternative is to estimate the parameters of a style model using existing motion data performed in a specific style. Liu et al. model physics–based style with musculoskeletal parameters that are used during physically–constrained optimization [105]. Given an example motion performed in a desired style, they use inverse optimization to determine the muscoloskeletal parameters that fit that style. These parameters are then used during the synthesis of new motion that has the same style of the original performance.

While the approaches to expressive motion editing presented in this thesis do not explicitly include physics, they can produce motion edits that are similar in spirit to physics–based edits. For example, timing variation can be increased or reduced using the techniques presented in Chapter 7 to make motion appear more loose or more rigid. Similarly, spatial exaggeration (Chapter 8) can be used to make motion appear more or less energetic.

2.2.3 Data–Driven Models

Data–driven models can be used to model probabilistic distributions of motion to allow for the generation of new motion that meets given constraints. To synthesize new motion in a desired style, Brand and Hertzmann use a Hidden Markov Model to determine a set of states common in a collection of example motion clips, along with a set of style parameters. They then synthesize new motion similar to an example motion, but with different style, by progressing through the same state sequence of the example motion using the trained model. New motion can also be synthesized by taking a probabilistic walk through the states of the trained model.

Another approach uses one or more motion examples and a regression model to create a distribution of likely poses or motion. For example, Grochow et al. introduce a Gausian Process regression model that creates a distribution of likely poses, given a recorded motion clip [47]. This model, in conjunction with local optimization, is used as an inverse kinematics solver that creates new poses in the style of the original motion. These poses can be used as keyframe poses to create new motion in the same style. This approach considers only poses in the style model, not entire motion clips. To generalize this idea to full motion clips, Min et al. use a Gaussian Mixture Model to create a distribution of motion clips from a set of similar example clips [119]. They use this model to synthesize new variations of the motion that meet user–provided constraints on the motion path or end effector trajectories. In later work, they use a multilinear model to create identity and style parameterizations [120]. These identity and style parameters can be directly edited by the user.

Gesture data can also be used to create new expressive motion that responds to a given piece of text or speech. Neff et al. synthesize new gesture motion that responds to a given piece of text, but in the style of a speaker whose gestures have been recorded, by training a Hidden Markov Model [130]. Levine et al. also use such a model, but they record prosody features from recorded speech to generate new corresponding gesture

motion [98]. They then extend this idea to create an optimal policy controller by using reinforcement learning to select the gesture clips based on the hidden states [97].

While these approaches are powerful, they rely on a set of example motion clips to train the underlying data model. This limits the motion that can be created, as the training set must often contain motion that is similar in content to the desired result. In contrast, the approaches developed in this thesis assume the given motion has some degree of desirable detail that should be preserved. These new approaches focus on changing this given motion to meet performance–related goals.

2.3 Pose Selection

A key contribution of this thesis is the algorithmic selection of a sparse set of motion extremes to be used as a foundation for expressive motion editing (Chapter 5). In the case of a full character, extreme poses capture important changes in the movement. Full–body pose selection has been investigated before. In particular, it has been applied to the problems of compression and visualization.

Curve simplification algorithms have been applied to pose selection by a variety of authors. Lim and Thalmann introduce the idea, in which motion is considered a high–dimensional curve that will be approximated by iteratively refining a piecewise–linear approximation [103]. This refinement is done by repeatedly adding the pose—a high dimensional curve point—that has the greatest error in the piecewise linear approximation. The order of pose selection is used as a prioritized ordering for all frames in the motion, and the highest priority frames are used as interpolation samples for compression or for static illustration.

Assa et al., as part of their visualization system, extend this idea by applying dimensionality reduction before curve simplification [10]. In their approach, they first use non–metric reduced multiple dimensional scaling. This creates a low–dimensional embedding

that aims to correlate the relative low–dimensional spatial relationships among motion samples to the corresponding high–dimensional spatial relationships. Bouvier–Zappa et al. also use a low–dimensional embedding for determining poses for visualization; they apply Principal Components Analysis to create the low–dimensional space. Each of these curve simplification approaches creates a full prioritized ordering of all pose samples in the motion; users must select how many of the poses they wish to use for any given application.

Other approaches can be used to select important poses. For example, both Li et al. [100] and Togawa et al. [168] repeatedly remove frames to create a prioritized ordering of poses. Liu et al. [108] and Park et al. [134] cluster all poses in the motion and use the cluster centers as important poses. Matrix factorization can also be used to determine a set of poses that can used for approximating a motion by multi–target interpolation. Huang et al. apply this to full body motion [57] and Lee et al. apply it to mesh–based facial animation [95] . These approaches also require the user to select the number of poses. This can correspond to the number of poses to cull from a prioritized ordering, the number of clusters to use, or the number of interpolation targets to use. Furthermore, none of these approaches consider motion editing as a goal application.

Motion segmentation can be considered the dual problem of pose selection. For example, motion graph construction algorithms choose segmentation boundaries using distance metrics [74, 90]. Barbič et al. investigate three approaches for action segmentation: change in PCA dimensionality, change in pose distribution, and change in the parameters of a Gaussian mixture model [13]. However, these approaches do not consider the use of poses at segmentation boundaries as editing controls.

In work completed subsequently to that in this thesis, Lockwood and Singh use the techniques that will presented in Chapters 5 and 7 to identify times at which a motion path can be meaningfully edited [111]. Specifically, times of minimal height during ground contact are selected as control points for editing a motion path. In contrast to their path–

based approach, the work presented in this thesis uses extrema of motion for choosing times at which to edit character poses and joint configurations.

In Chapter 5, I will introduce techniques for algorithmically selecting a *sparse set* of motion extremes, such as meaningful full–body poses. These approaches are distinct in the following ways. First, they are the first approaches that explicitly look at properties of the pose that indicate that it is unique or that the motion is changing. Early experiments with a curve simplification algorithm [10] indicate that it too can select poses at times when the motion changes. However, this approach tends to intermix extreme–like poses with poses that have no corresponding significant change in body movement, as such poses still reduce approximation error. However, as was illustrated in Figure 1.4, these poses are not typically meaningful controls for editing. Second, I introduce the first approaches to pose selection that chooses a sparse set; all existing approaches require users to set how many should be selected. This significantly reduces the amount of data users must consider. While for manual editing (Chapters 6 and 7), users might still need to refine the sparse set we provide, fully algorithmic editing approaches such as spatial exaggeration (Chapter 8) require the algorithm to choose a meaningful sparse set without user intervention. Finally, the applications presented in Chapters 6, 7, and 8 are the first to use algorithmically–selected poses as controls for motion editing; existing systems that allow for poses to be used for editing offer no means for algorithmic selection.

Chapter 3

Animation Techniques

A variety of approaches to describing aesthetic qualities of movement have been developed. This chapter will describe the classical animation principles [167, 84], additional concepts used by animators when describing motion [175, 177, 46], and concepts from the Laban Movement Analysis (LMA) community [81, 16]. In particular, the discussion in this chapter focuses on how these ideas relate to the goals of this thesis. Chapter 9 will further reflect on how these ideas relate to specific work that will be presented later in this thesis.

3.1 The Classical Animation Principles

The classical animation principles were developed by Disney animators to describe aesthetic qualities of movement, workflows for animating, and useful background skills for character animation. I refer the reader to the text by Thomas and Johnston [167] and the paper by Lasseter [84] for a full description of these principles, and I describe here how they relate to the goals of this thesis.

Straight Ahead and Pose to Pose: As described in Chapter 1, *straight ahead* and *pose to pose* refer to two workflow strategies that animators can take when creating new motion. The pose–to–pose approach influences the design of the motion editing systems

presented in this thesis. In particular, the common use of a small set of poses to design new keyframe motion motivates the use of a sparse set of motion extrema for motion editing.

Overlapping Action: *Overlapping action* refers to the tendency of one body part to experience a change of movement at slightly different times than other body parts. For example, during a wave, the shoulder will often change direction slightly before the elbow, and the elbow will change direction slightly before the wrist. Keyframe animators model this by moving the keyframe constraints on different body parts to slightly different times. The constraints on the different body parts are related in that they all represent a direction change in arm movement. To create compelling keyframe animation, it is important to create a believable timing relationship among these related keyframe constraints. This idea will be applied to motion extrema in the staggered poses model presented in Chapter 7.

Timing: *Timing* refers to both the overall timing of a character performance and the timing characteristics of movement such as secondary action [175]. Timing variation among different body parts, as described above, will be used in the staggered poses model (Chapter 7). This will allow for the development of procedural editing tools that change timing effects related to overlapping action.

Exaggeration: When *exaggerating* a character performance, animators take a particular aspect of movement or design and to make it stand out more. A key goal is to make that aspect of animation more readable to the audience. Different aspects of animation can be exaggerated, ranging from color to geometric design to movement. In terms of movement exaggeration, different specific aspects of movement can be exaggerated, such as overall timing, the relative timing among characters, and timing differences between different parts of the body. The procedural editing tools presented in Chapters 7 and 8 will take this approach to applying edits. Users will be able to procedurally edit *specific* aspects of motion to exaggerate them or make them more subtle.

3.2 Additional Workflows and Principles

The classical workflow ideas of pose–to-pose and straight–ahead each have their practical strengths and drawbacks, as discussed by Williams [177]. In practice, animators develop individual workflows that often incorporate ideas from each, as well as other ideas. For example, Williams advocates using a pose–to–pose approach to plan how motion will be created and then animating transitions from each pose to the next using a straight ahead approach. In computer animation, the equivalent workflow is the creation of all extreme poses before refining the interpolation between each pose and the next. Goldberg [46] refers to this process of planning important poses before adding details as *blocking*. This blocking–level control represents the same type of high level control that I aim to achieve in this thesis through the use of motion extrema.

In addition to the distinction between pose–to–pose and straight ahead workflows, animators also commonly work across different parts of a character. For example, this can be on different subsets of a character's body. Lasseter describes this in terms of animating the character parameters that control large scale movement and then animating other parameters to add detail [85]. This effectively partitions the character parameters into different sets. This approach will be adapted in Chapter 6 through the development of a multi–step algorithm in which each step modifies a subset of the character's parameters.

In addition, the animation for any given parameter can be specified through the use of layers to allow for multiple passes of refinement; this idea is supported in a variety of commercial animation packages such as *Motion Builder*[1] and *Maya*[2]. In the motion editing literature, layers are commonly known as as motion *warps* [179] and motion *displacement maps* [20]. Motion displacement maps will used through this thesis to allow for a series of changes to be applied by any given algorithm.

When designing motion, animators refer to different types of poses; both Williams [177]

[1]http://www.autodesk.com/motionbuilder
[2]http://www.autodesk.com/maya

and Goldberg [46] describe these types of poses. *Key* poses are the poses that are necessary to tell the story of what is happening. They do not typically convey details about how the character moves, and they are intended to communicate *what* is happening in a scene. *Extreme* poses are used to lay out approximate motion, and they are necessary to communicate the basic motion that a character goes though. They typically occur when parts of the body change direction, and animators often create these as a first pass when blocking out motion. *Breakdowns* specify details about how the character moves from one extreme pose to the next, and they are often used to specify curved paths in space.

Goldberg further describes poses in terms of how they relate to acting performance. In particular, he advocates the use of *attitude* poses for the extreme poses; these are poses that capture the emotional state of the character. As a result, creating effective extreme poses is necessary for creating motion with expressive performance control. This motivates the use of poses similar to extreme poses for editing controls when changing existing motion. To find these poses, I will use motion extremes that correlate to times when the body's movement changes (Chapter 5). In practice, these extreme "poses" need not be specified for full body. For example, an extreme "pose" can also apply to an individual part of the body that is animated in isolation.

Motion timing can refer to many different aspects of movement [175]. One key aspect of the timing is the general layout, in time, of how a character passes through important poses, such as extreme poses. These poses can thus give high–level control over the timing of a performance, and this timing is often developed early in the creation of new motion. One can also consider the *relative* timing among different parts of the body. This relative timing refers to different body parts that are moving as a coordinated unit having slightly different times at which they change direction. This can be used to model effects such as a force propagating motion from one body part to another [175]. It can also be used to avoid *twinning*—unnatural symmetrical movement on different limbs. Force propagation effects are often modeled and explicitly exaggerated through the use of sequential timing,

also called *succession* [125]. Twinning can be avoided by introducing *asymmetric action*. These specific types of relative timing control motivate the development of similar tools for procedurally modifying the timing of existing motion (Chapter 7).

3.3 Laban Movement Analysis

Laban Movement Analysis (LMA) is a model for describing and interpreting human motion that is commonly used for dance and other forms of expressive performance [81]. In recent years, it has also been used to describe [16] and synthesize [27] animated movement. LMA includes five key components for describing movement. Four components—*body*, *effort*, *shape*, and *space*—describe aspects of movement related to body stance and timing of movement. The fifth component—*phrasing*—describes how the other components change over time and can combine into a longer cohesive movement.

The *body* component of LMA describes which parts of the body are moving, which parts are not, and how movement propagates from one body part to others. This movement propagation is similar to how follow–through can be used to describe force propagation in articulated characters [16], which further motivates the need to explicitly model overlap in character motion (Chapter 7). *Shape* describes how the body changes in shape over time. This is intrinsically related to poses, which describe stance at specific times. *Space* describes the relationship of the character to the environmental space around it. For example, the amount of physical space occupied by the character while moving is related to the shape component. A key conceptual idea related to space is the *kinesphere*, which describes the extent of reach around the body that is used during a movement. The concept of a kinesphere is one aspect of motion that can be procedurally edited. Chapter 8 investigates approaches for doing this.

Chapter 4

Motion Representations and Editing Foundations

This chapter presents foundational material and notation that will be used in the remainder of this thesis. This includes assumptions about the character structure, transformation representations, and motion models. This chapter also describes how displacement maps are used to modify motion and how to convert among motion representations. The notation introduced in this chapter will be used in subsequent chapters to describe the motion editing systems and associated algorithms.

The work in this thesis will focus on articulated body movement represented using skeletal hierarchies. Three motion representations will be used. These three representations are called the Articulated Skeletal (AS) representation, the Global Cartesian (GC) representation, and the Local Cartesian (LC) representation. These different representations use different parameterizations; certain algorithms can be expressed more concisely or implemented more efficiently by using a motion representation with a convenient parameterization. Poses can also be represented using these three representations.

Motion displacement maps encapsulate the idea of layering an edit on top of a base motion. Since rotations are non–commutative, displacement maps will be formally de-

fined in a restricted way, such that the concept of applying a motion displacement map produces a unique result. This idea is extended to single frame poses; these single frame displacement maps are called *difference poses*. Finally, this chapter will present methods for converting among the three motion representations using either forward or inverse kinematics.

4.1 Character Model

The work in this thesis focuses on the development of expressive motion editing techniques that modify skeletal character motion. The choice to use skeletal character motion is made for a variety of reasons. First, abstract control skeletons approximate the large–scale movement of the human body well. Second, many expressive qualities of movement, such as gesture and body stance, can be represented using skeletal motion. Third, most current motion recording systems have a workflow that transforms motion to control skeletons. Finally, skeletal motion representations are commonly used as a high–level control representation for significantly more complex character models [25].

The control skeleton abstraction used in this thesis uses a hierarchy of joints. Each joint has associated parameters for controlling the movement of any joints and geometry below it in the hierarchy, such as translation and rotation. Depending on the needs of any given algorithm, these joints might be parameterized in different ways. For example, two of the motion representations used in this thesis do not include joint rotation parameters. Representations that do not include rotation parameters can be more effective when searching for motion extrema for two reasons. First, common rotation parameterizations are not uniquely determined, as multiple joint rotation parameter values can represent the same overall rotation. Second, rotation parameters have a nonlinear effect on character stance; this can bias many types of measurement toward specific joints.

By convention, subscripts represent the joint that a parameter is associated with. The

subscript j denotes a general joint, while other subscripts refer to specific joints. For the latter case, p denotes a parent joint in the hierarchy. The subscript r denotes the root joint of the character hierarchy, and a denotes an ankle joint. The Cartesian coordinates of any given joint are denoted as $\mathbf{x}_j$; for example, $\mathbf{x}_a$ denotes the world space position of the ankle.

4.2 Transformation Representations

Joint transformations can include translation and rotation parameters. Translations, denoted as $\mathbf{t}_j \in \Re^3$, define the directional offset of a joint, relative to its parent joint. Although it is common to hold these fixed in length, the motion representations used here allow them to change in length; this will be important for the pose–centric editing system presented in Chapter 6. This also allows users to intentionally change limb length when editing motion. Rotations are represented using unit–length quaternions, denoted as $\mathbf{q}_j \in SU(2)$.

In this thesis, I will often use operator notation to designate the transformation of a point or vector. The translation T of a point $\mathbf{p}$ by a vector $\mathbf{t}$ is denoted $T_{\mathbf{t}}(\mathbf{p})$; this is defined to be $T_{\mathbf{t}}(\mathbf{p}) = \mathbf{p} + \mathbf{t}$. The rotation R of a point $\mathbf{p}$ by the unit quaternion representation $\mathbf{q}$ is denoted $R_{\mathbf{q}}(\mathbf{p})$; this is defined to be the vector component of $\mathbf{q}(0, \mathbf{p})\mathbf{q}^*$. The notation $\mathbf{q}^*$ denotes the quaternion conjugate of $\mathbf{q}$. If a joint transformation includes both a translation $\mathbf{t}_j$ and a rotation $\mathbf{q}_j$, we follow the common convention of applying the translation after the rotation to result in the transformed point $T_{\mathbf{t}_j}\left(R_{\mathbf{q}_j}(\mathbf{p})\right)$.

At times, it will be convenient to abstract out which transformation operator is being modified by a given algorithm. To do this, I define the general transformation concatenation operator $\oplus$. For translation concatenation, the follow expressions are equivalent:

$$T_{\mathbf{t}} = T_{\mathbf{t}_a} \oplus T_{\mathbf{t}_b}$$

$$\mathbf{t} = \mathbf{t}_a + \mathbf{t}_b.$$

For rotation concatenation, $\oplus$ is, in general, non–commutative. The definition for rotation concatenation is such that these expressions are equivalent:

$$R_{\mathbf{q}} = R_{\mathbf{q}_b} \oplus R_{\mathbf{q}_a}$$

$$\mathbf{q} = \mathbf{q}_b \mathbf{q}_a.$$

Therefore, $R_{\mathbf{q}_b} \oplus R_{\mathbf{q}_a}$ represents a rotation by $R_{\mathbf{q}_a}$, followed by a rotation by $R_{\mathbf{q}_b}$.

I also define the generalized difference operator $\ominus$ for transformations. For translations, the following expressions express this difference. This is represented as a difference in Euclidean values, which is explicitly shown in the second expression.

$$T_{\mathbf{t}} = T_{\mathbf{t}_a} \ominus T_{\mathbf{t}_b}$$

$$\mathbf{t} = \mathbf{t}_a - \mathbf{t}_b.$$

For rotations, I define $\ominus$ such that it results in the rotation that would be applied *after the right hand operand* to *result in the left hand operand.* The order of operands is important for rotations, as rotations are non–commutative. To define the generalized difference uniquely, this is represented as:

$$R_{\mathbf{q}} = R_{\mathbf{q}_b} \ominus R_{\mathbf{q}_a}$$

$$\mathbf{q} = \mathbf{q}_b \mathbf{q}_a^*.$$

The latter expression follows from the requirement that $\mathbf{q}_b = \mathbf{q}\mathbf{q}_a$. Overall, $R_{\mathbf{q}}$ is then the rotation that needs to be applied after $R_{\mathbf{q}_a}$ to achieve the same overall effect as a rotation by $R_{\mathbf{q}_b}$.

4.3 Pose and Motion Representations

Depending on needs of any given algorithm, it can be convenient to use one of three basic representations of pose or motion, which are defined here.

- The *Articulated Skeletal* (AS) representation, which is similar to the skeletal representation most commonly used in the literature, models intrinsic state, such as joint angles.

- The *Global Cartesian* (GC) representation explicitly models the world space position of each joint. This representation can be useful for analyzing joint trajectories in space or expressing constraints on end effectors.

- The *Local Cartesian* (LC) representation models the Cartesian coordinates of each joint, *relative to the location of the parent joint.* This representation is most useful for analyzing spatial characteristics of movement for individual joints in isolation.

Each representation is formally described below.

4.3.1 The Articulated Skeletal Representation

The articulated skeletal representation is similar to most representations used in the literature. The root is represented using its world space position $\mathbf{x}_r$ and global orientation $\mathbf{q}_r$. Each joint has a local translation $\mathbf{t}_j$ and rotation $\mathbf{q}_j$. For the motion representation, limb length *is allowed to change in time*, which differs from many representations used in the literature. Formally, the motion representation is

$$\mathbf{M}(t) = (\mathbf{x}_r(t), \mathbf{q}_r(t), \{(\mathbf{t}_j(t), \mathbf{q}_j(t))\}),$$

where the set contains all joints j except the root. Using operator notation for the joint transformations, this is

$$\mathbf{M}(t) = \left(\mathbf{x}_r(t), R_{\mathbf{q}_r}(t), \left\{\left(T_{\mathbf{t}_j}(t), R_{\mathbf{q}_j}(t)\right)\right\}\right).$$

Similarly, articulated skeletal poses are formally represented as

$$\mathbf{P} = (\mathbf{x}_r, \mathbf{q}_r, \{(\mathbf{t}_j, \mathbf{q}_j)\})$$

$$\mathbf{P} = \left(\mathbf{x}_r, R_{\mathbf{q}_r}, \left\{\left(T_{\mathbf{t}_j}, R_{\mathbf{q}_j}\right)\right\}\right).$$

Pose representations, in general, are similar to motion representations, but do not include time dependence. For brevity, formal pose definitions will be omitted from the remainder of this section. The only distinction is that in a motion representation, each parameter is a function of time. Such parameters are denoted, for example, as $\mathbf{x}(t)$. For a pose, the parameters are not dependent on time. Pose parameters are thus denoted, for example, as $\mathbf{x}$.

4.3.2 The Global Cartesian Representation

The Global Cartesian representation stores joint locations in world space as Cartesian coordinates. This is useful for making changes to or measurements from joint positions in a single Cartesian coordinate frame. One example application is the computation of distance metrics. The formal notation is

$$\mathbf{M}(t) = (\mathbf{x}_r(t), \{\mathbf{x}_j(t)\}).$$

There is no equivalent operator notation for the global Cartesian representation, as world space positions are not treated as transformations.

4.3.3 The Local Cartesian Representation

The local Cartesian representation includes Cartesian coordinates for each joint, specified relative to the corresponding parent joint. This representation is useful for analyzing the motion of individual joints, as it does not incorporate rotation parameters, which have a nonlinear effect on spatial distance. Formally, this is equivalent to the articulated skeletal representation, but excludes joint angles:

$$\mathbf{M}(t) = (\mathbf{x}_r(t), \{\mathbf{t}_j(t)\})$$

$$\mathbf{M}(t) = \left(\mathbf{x}_r(t), \left\{T_{\mathbf{t}_j}(t)\right\}\right).$$

Note that in the first expression, $\mathbf{t}_j$ represents a vector offset, as this is necessary to represent spatial relationships among joints when no rotation is included in the parameterization.

When modeling motion with this representation, the root position $\mathbf{x}_r(t)$ is simply the world space location of the character root. For each other joint j,

$$\mathbf{t}_j(t) = \mathbf{x}_j(t) - \mathbf{x}_p(t),$$

where p is the parent joint of joint j.

4.4 Displacement Maps

Motion displacement maps, also known as motion warps, capture the notion of encoding motion as a layering of transformation data. For example, applying a displacement map to one motion clip creates another motion clip, and the displacement map can be considered to be the "difference" between the two motion clips. These ideas can also be applied to poses; this captures the intuitive notions of layering pose data and computing pose differences.

Given a pair of motion clips or poses, I indicate the application of a displacement map or difference pose using the transformation concatenation operator $\oplus$. Similarly, I indicate the computation of a displacement map or difference pose using the generalized difference operator for transformations, $\ominus$. While these operators are defined for translation and rotation transformations, the addition of Cartesian points in space is not geometrically defined, and the difference between a pair of points will be represented as a direction vector; i.e. a translation [160]. As such, the formal displacement map model will be slightly different than the corresponding motion models. These formal definitions will be briefly presented for completeness, and the concatenation and factorization operations will be defined for applying and computing displacement maps.

4.4.1 Displacement Map Representations

The three displacement map representations are similar to the corresponding motion representations, but geometric spatial locations are replaced with translations, i.e. direction vectors. For the articulated skeletal representation, the displacement map is represented as

$$\mathbf{D}(t) = \left(T_{\mathbf{t}_r}(t), R_{\mathbf{q}_r}(t), \left\{\left(T_{\mathbf{t}_j}(t), R_{\mathbf{q}_j}(t)\right)\right\}\right).$$

For the global Cartesian representation, the definition is

$$\mathbf{D}(t) = \left(T_{\mathbf{t}_r}(t), \left\{T_{\mathbf{t}_j}(t)\right\}\right),$$

and for the local Cartesian representation, it appears similar:

$$\mathbf{D}(t) = \left(T_{\mathbf{t}_r}(t), \left\{T_{\mathbf{t}_j}(t)\right\}\right).$$

As with the motion representations, these can also be stated using notation that explicitly includes the transformation representations.

Difference poses are also similar in notation, but they do not include the dependence on time; I use the notation $\mathbf{D}$ to indicate a difference pose.

4.4.2 Pose and Motion Operations

As concatenation and factorization operations are defined for translation and rotation transformations, these definitions can be used component–wise to explicitly define the operations for applying and computing displacement maps. Concatenation captures the intuitive notion of layering one motion clip or pose on top of another. Generalized difference models the intuitive notion of computing the difference between a pair of motion clips or poses.

The concatenation operator $\oplus$ applies a motion displacement map or difference pose to a given motion clip or pose, respectively. Assume a displacement map $\mathbf{D}(t)$ exists.

The application of $\mathbf{D}(t)$ to a motion clip $\mathbf{M}^o(t)$ to create is new motion clip $\mathbf{M}(t)$ is then denoted as

$$\mathbf{M}(t) = \mathbf{D}(t) \oplus \mathbf{M}^o(t).$$

Similarly, the application of a difference pose $\mathbf{D}$ to a pose $\mathbf{P}^o$ to create a new pose $\mathbf{P}$ is denoted as

$$\mathbf{P} = \mathbf{D} \oplus \mathbf{P}^o$$

Note that the operations in the reverse order, $\mathbf{M}^o(t) \oplus \mathbf{D}(t)$ and $\mathbf{P}^o \oplus \mathbf{D}$, are not defined to account for the non–commutativity of rotations. This keeps the application of a displacement map unique. Also note that the application of multiple displacement maps can be order–dependent, as they might incorporate non–commutative rotation concatenations.

The generalized difference operator $\ominus$ represents the computation of a motion displacement map or difference pose, given two motion clips or poses, respectively. These computations are formally stated as

$$\mathbf{D}(t) = \mathbf{M}(t) \ominus \mathbf{M}^o(t)$$

and

$$\mathbf{D} = \mathbf{P} \ominus \mathbf{P}^o$$

The generalized difference operation is not commutative; i.e. $\mathbf{M}(t) \ominus \mathbf{M}^o(t) \neq \mathbf{M}^o(t) \ominus \mathbf{M}(t)$ and $\mathbf{P} \ominus \mathbf{P}^o \neq \mathbf{P}^o \ominus \mathbf{P}$.

4.5 Converting Among Motion Representations

Converting from the articulated skeletal representation to the the Cartesian representations is straightforward; standard techniques for evaluating transformed geometry can be used. Converting from the Local Cartesian representation to the Global Cartesian

representation is also straightforward; each joint position $\mathbf{x}_j$ in the Global Cartesian representation can be evaluated as

$$\mathbf{x}_j = \mathbf{x}_r + \sum_i \mathbf{t}_i,$$

where the summation is over the sequence of ancestor joints i of joint j in the hierarchy. Converting from either Cartesian representation to the articulated skeletal representation is nontrivial, however, and requires an inverse kinematic solution. The remainder of this section will discuss the conversion from the Global Cartesian representation to the articulated skeletal representation, as the above equation can be used in conjunction with this to transform a Local Cartesian representation to an articulated skeletal representation.

There are a number of issues to consider when converting from the Global Cartesian representation to the articulated skeletal representation. First, rotation parameters are not uniquely determined, as multiple joint rotations can achieve the same change in orientation. Second, limb lengths can change. Third, as it is common to store skeletal motion in rotation parameters instead of translation parameters when possible, the solver should primarily set rotation values. To resolve ambiguity, it will use a default character state, which is referred to as the rest pose. Finally, if a joint rotation affects multiple child joints, it is not possible, in general, to set the rotation such that all child joints are rotated to desired positions, as a rotation is a rigid transform.

Taking these issues into consideration, the inverse kinematic solver sets each joint's parameter values, using a recursive, depth–first ordering of the skeletal hierarchy. As the system is under–constrained, the solver takes a greedy approach that minimizes changes to local joint translations. For each iteration of the solver, two steps take place. For the first step, the goal is to first find a joint rotation that aligns child joints to goal positions in a least squares sense. The second step applies local joint translations *to the child joints*, if needed, to account for remaining error in matching child joints to the goal positions. These two steps are iterated for the each joint in the character hierarchy, using a depth–first traversal. This approach is motivated by the common practice of using joint rotation

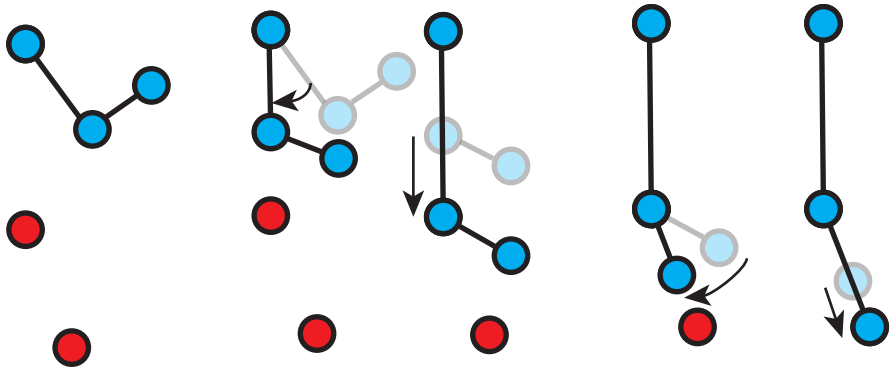


Figure 4.1: Two iterations of the IK algorithm for fitting a skeleton with rotation parameters (blue) to desired joint positions (red). In the first image, the root joint is in the desired position. Recursively, for each descendant joint, a rotation is applied to align it to the desired position (second and fourth images). A translation is then applied to account for changes in limb length (third and fifth images).

parameters as the primary means of animating characters whose motion is represented using skeletal hierarchies.

In practice, when taking a joint–centric view, the translation step takes place first. This is necessary to account for the error that remains from processing the parent joint. The solver first sets the translation of the joint to achieve the desired position. It then rotates the joint to set all child joints as best possible. This can be done in closed–form; the solver uses an approach similar to that taken by Hecker et al. [51]. For each joint, it solves for the smallest rotation change, relative to the rest pose, that moves the child joints to the desired joint positions. If there is only one child joint, and limb lengths are preserved, this can be done exactly. If there are multiple child joints, an approximate solution must be used. This process is illustrated in Figure 4.1. In this example, limb lengths are not preserved, and the solver handles this length change by increasing the length of the translation component. If this limb length is not changed, the translation

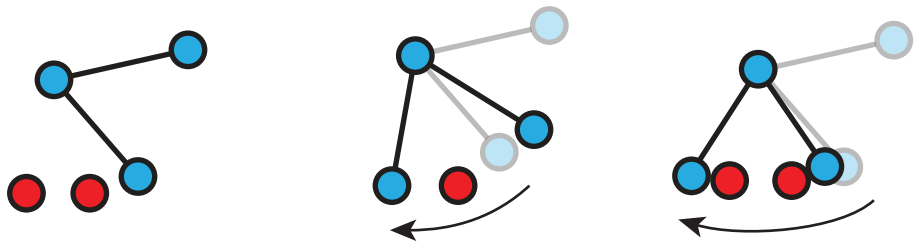


Figure 4.2: When multiple child joints exist, the rotation step cannot always align the child joints exactly. The approach of Hecker et al. [51] resolves this by using the first child joint to compute the rotation, which leaves all the rotation error in one child joint (center). The solver used for the systems presented in this thesis uses a least squares solution that balances the error among the child joints (right). In both cases, this error would be corrected in the subsequent translation step for each child joint.

applied by the solver will be zero for this example.

The solver uses the closed–form least squares solution of Horn [53] to determine such joint rotations. Any residual error in positioning the child joints will be accounted for when setting the child joint's translation parameters. This differs from the approach of Hecker et al.; their approach instead uses a subset of the child joints to compute the approximate rotation. The difference between these approaches is illustrated in Figure 4.2.

If editing tools incorporate limb length constraints and rigidity constraints on each set of sibling joints, relative to their parent, this solution will not change any translation parameters, relative to the rest pose, with the exception of the root. However, when these assumptions are not met, the solver still sets the skeletal state such that joints are positioned as desired, but it biases how they are set such that rotations are favored

over translations, and parameter changes higher in the skeletal hierarchy are favored over those lower in the hierarchy.

Chapter 5

Methods for Identifying Motion Extrema

This chapter introduces models and algorithms for identifying motion extrema to be used as a sparse set of motion editing controls. The central goal in choosing these motion extrema is finding times with poses or joint configurations that are qualitatively the most similar to those that would be created by keyframe animators. Chapter 1 lists a set of observations about the extreme poses animators create when designing new animated motion. These observations will guide the development of a set of quantitative models for finding motion extrema. Using these models, robust algorithms are presented for finding the motion extrema. One particular model—*spatial extremeness*—will result in poses and joint configurations that most often have the qualities observed about extreme poses created by animators.

These models can be applied to either individual joints or a larger collection of degrees of freedom, such as the entire body. I will distinguish between these two approaches by referring to motion extrema as either *joint–centric extrema* or *extreme poses*. The motion editing systems presented in Chapters 6, 7, and 8 will use either joint–centric extrema or extreme poses as a foundation for applying motion edits.

This chapter will formally describe the problem of identifying meaningful motion extrema (Section 5.1) and present the principles for selecting these motion extrema (Section 5.2). Two general approaches to finding motion extrema will be presented—the use of extrema of differential motion measures (Section 5.3) and the use of spatially extreme configurations (Section 5.4). To illustrate these techniques for joint–centric extrema, a rotating hip will be used, and for extreme poses, a walk will be used. Algorithmic options for robustly selecting motion extrema in real–world data will be discussed in Section 5.5, and the models will be applied to a wider variety of motion examples in Section 5.6.

5.1 Identifying Motion Extrema

To formally state the concept of what constitutes a motion extreme, let the motion extreme occur at some time t_i and apply to a set of parameters $\{p\}$. A motion extreme is then the tuple $(t_i, \{p\}, \mathbf{p}(t_i))$, where $\mathbf{p}$ is a vector containing parameter values for each element of $\{p\}$. The entire set of motion extrema is then $\{(t_i, \{p\}, \mathbf{p}(t_i))\}$.

A motion extreme can apply to a single degree of freedom, a joint or larger portion of the body, or the full body of the character. For a single degree of freedom, $\{p\}$ contains a single element. The extreme time t_i and state $p(t_i)$ is then analogous to a knot in a traditional animation spline. For a full character, $\mathbf{p}(t_i) = \mathbf{M}(t_i)$ represents the values of all character parameters; this effectively represents a character pose at time t_i. To explicitly relate this to the motion representations presented in Chapter 4, the parameter set $\{p\}$ must be related to specific quantities modeled by the corresponding motion representation.

The problem of finding motion extrema is then the problem of finding the times t_i and the associated parameter sets $\{p\}$, where something is distinct about the state vector $\mathbf{p}(t_i)$. This chapter presents algorithms for finding times at which $\mathbf{p}(t_i)$ is distinct, and different algorithmic approaches will be considered. Recall that $\{p\}$ can include

parameters for only a portion of the body, such as an individual joint, or it can include all character parameters, in which case $\mathbf{p}(t_i)$ is equivalent to the full character pose at time t_i. The algorithms presented here consider parameter sets that include parameters for a single joint, as well as sets that include all parameters in a skeletal hierarchy, such as the full body.

For a given motion representation, the problem of finding motion extrema involves analyzing the information explicitly stored in that representation to find times at which movement changes. For example, if finding motion extrema using the Articulated Skeletal representation, then $\{p\}$ can contain both translation and rotation parameters, or a subset of them. If using a Global Cartesian representation, $\{p\}$ can include only positions in space, and if using a Local Cartesian representation, it can include the root position and local joint translations.

Different algorithms for finding motion extrema will use different parameter sets and motion representations. The choice of the parameter set will depend on whether the algorithm aims to find joint–centric extrema or full–body extrema, i.e. extreme poses. The choice of motion representation will depend on the algorithm used. In particular, algorithms based on extrema of differential motion measures can consider any set of parameters. In contrast, algorithms based on spatially extreme configurations will only consider the Cartesian representations, as the Cartesian parameterizations are linearly related to spatial changes in joint position.

In Chapters 6–8, motion extrema will be used as a foundation for motion editing. When doing so, it is not necessary that the same motion representation be used both for identifying motion extrema and applying a motion edit. For example, motion extrema associated with a particular joint can be found using one representation and then used to edit the same motion, but using a different motion representation. To do so, the parameter set $\{p\}$ and state vector $\mathbf{p}$ must be converted to use the parameterization of the joint under the motion representation used for editing.

5.2 Approaches to Identifying Motion Extrema

When identifying motion extrema, the goal is to find times in which a character's body or part of a character's body is in a state that is similar to the extreme poses created by keyframe animators. To find such times, recall the observations made of extreme poses in Chapter 1. These observations will be used both as motivating ideas for designing algorithms and as qualities to determine whether selected motion extrema meet the goal of identifying times that are similar to the extreme poses used by keyframe animators. These qualities are:

- Motion of a joint or many joints changes direction.
- Motion of a joint or many joints has low speed, due to the character easing into and out of a keyframe–like pose.
- The trajectory of a joint or joints has high curvature.
- Joints are at spatial locations that are maximally distant to the locations of the same joint at neighboring times. This measure of distance to positions at neighboring times is spatial extremeness.

The first three qualities can be stated in terms of quantities that can be directly measured from motion. Curvature and speed are both differential measures of motion, and direction change is correlated to curvature. The concept of spatial extremeness is not a local measure, as it considers the state of the character at other times. It is less straightforward to measure than speed or curvature, and a method for measuring spatial extremeness will be presented in Section 5.4. This, along with the differential motion measures of velocity, acceleration, and curvature of joint trajectories, will be used to identify motion extrema.

For each approach, a set of motion examples will be used to find a set of motion extrema. First, two motion examples will be used from a human walk. Walking motion

allows for a visual comparison of the motion extrema to extreme poses that animators would create, as walking is commonly used as a textbook example and classroom exercise for teaching keyframe animation techniques. Other motion examples will then be used to further test how well the approaches select times that have the above listed qualities of extreme poses.

Each algorithm will use a scalar function $p(t)$; local maxima or minima of $p(t)$ are then chosen as the times at which motion extrema occur. The choice of whether to use maxima or minima will depend on the algorithm. For some editing applications, in particular those in which users directly edit joint–centric motion extrema or extreme poses, the selected set of motion extrema might need to be refined. In other applications, such as procedural editing algorithms that encapsulate the use of motion extrema, the set of motion extrema might need to be chosen without the possibility of user refinement. Different algorithms might select different sets of motion extrema, although, in practice, many chosen motion extrema will have similar states.

Ongoing work in the event detection literature supports the use of these ideas. The field of event detection develops theories that explain how people discretely partition time into a series of events. Boundaries of such events correspond to times at which subjects label a change in task, action, or movement, depending on the nature of the study. Two theories for explaining low–level event segmentation correlate to the principles used here for identifying motion extrema. For example, Zacks found that subjects, when considering only 2D shape motion, segment motion at *times that correlate to differential measures of motion* [182]. As another example, in more recent work, Zacks et al. found that this is also true when considering human motion, and that *spatial proximity* of joints has a high correlation to the labeling of event boundaries [183]. These findings correllate to the ideas of using differential measures of motion and spatially extreme configurations to find motion extrema.

5.3 Extrema of Differential Measures

The use of the extrema of differential measures is motivated by the idea that when motion changes significantly, associated differential properties of the character's movement will often have an extreme value. To investigate this idea, three differential measures will be considered here—velocity, acceleration, and curvature. These values are measured for the motion of each joint using trajectories that represent either local or global movement, depending on whether joint–centric extrema or full–body poses are being identified.

When finding extrema, a key goal is to find those times that the joint or body has a configuration similar to that of an extreme pose. Depending on the nature of the motion, it might not be clear what times should be chosen such that extrema occur at animator–like extreme poses. For coordinated movement such as walking, extreme poses are strongly apparent as occurring at the times at which the legs have the widest stance and the arms are at furthest points of swing [177]. For a single joint, the analogous configuration to an extreme pose is a time at which the limb alone is at the extreme point of a swing, as was illustrated in the motivating example in Chapter 1. However, if a single joint moves in a circular motion, there will be no apparent times at which extreme poses should occur. *For most human motion, important poses can be found, as evidenced by the long history of animators using such poses to design and create character movement.*

To find joint–centric extrema, in which each extreme is identified with the motion of a single joint in isolation, I use the Local Cartesian representation of motion. Using a Cartesian measure avoids any need to take into account the nonlinear effects of rotation parameters on joint motion. Using the *Local* Cartesian representation of motion isolates the motion of each joint by considering how it moves with respect to its parent joint in a joint hierarchy. Also note that as the intrinsic motion of a character is typically driven by joint rotation; it is the rotation movement of the parent joint that causes the child joints' Cartesian motion. The extrema found using any joint's Cartesian trajectory are

thus associated with the rotation motion of the parent joint.

To find extreme poses, I use the Global Cartesian representation of motion. This representation effectively represents the collected trajectories of all joints in the world. The differential measures of these trajectories are combined as a single overall measure; extrema of this measure are selected as extreme poses. Note that the motion of a parent joint can hide motion extrema in a child joint using this representation. As such, it is more useful for finding full–body motion extrema than joint–centric motion extrema.

Real motion data contains noise, which can make the identification of extrema difficult. A variety of strategies for dealing with noise will be discussed in Section 5.5; in all results shown in this chapter, the semi–local extrema approach will be used, which selects local extrema that are extreme with respect to all samples within a given time window. For consistency, all examples use a time window of 20 frames for semi–local extrema identification. The recorded and simulated motion data have a frame rate of 120 frames per second., and the keyframe example has a frame rate of 24 frames per second.

Throughout this section, I will use two examples of real human motion data to demonstrate each algorithm. In Section 5.6, the measures will be compared for a larger set of motion examples. To illustrate the selection of joint–centric extrema, I use the motion of a hip from a walking motion, as it is approximately periodic, has visually identifiable extrema, and has the irregularities of movement present in real motion data. For extreme poses, the full walking motion will be used.

For both examples, the goal is to select times at which the limbs are maximally spread. These are the times at which the character is in a pose that is similar to the extreme poses illustrated in Chapter 1. Also, at these times, the limbs are changing direction of movement from rotating forward to rotating backward, and vice versa. At these times, speed is low in comparison to the middle of a swing from once stance phase to the next. Joint trajectories have high curvature at these times, due to the change in direction of

movement. Finally, joints are at spatial locations that are maximally distant to positions at other times. As such the times at which limbs are maximally spread have the four qualities of extreme poses listed in Section 5.2.

5.3.1 Joint–Centric Differential Extrema

Joint-centric differential extrema are the extrema of some differential measure of the motion of a single joint. As the local Cartesian representation of motion is used to identify joint–centric extrema, the general form of these extrema is

$$\{t_i : f(\mathbf{t}_j(t_i)) \text{ is locally extreme}\}.$$

Recalling that $\mathbf{t}_j = \mathbf{x}_j - \mathbf{x}_p$ is the local joint translation, t_i denotes a specific time, and f denotes a differential measure. For any given measure, extreme configurations might occur at the local maxima or at the local minima. For some types of motion, the measure might be locally extreme at both extreme configurations and at times when the joint is in a less extreme configuration. This section will explicitly show when these extreme configurations occur, and whether they occur at minima or maxima, for different differential measures when considering human motion. Section 5.6 aims to identify how these minima and maxima vary for different types of motion.

Velocity: Velocity extrema have the form

$$\{t_i : ||\dot{\mathbf{t}}_j(t_i)|| \text{ is locally extreme}\}.$$

To evaluate velocity for the examples shown in this section, central differences is used. Figure 5.1 illustrates the resulting measure for the rotating hip during a walk and the configuration of the limb at the times chosen as extrema. Figures 5.1a and 5.1b plot the velocity, and the chosen maxima (a) and minima (b) are labeled in red. The corresponding configurations of the hip are shown from a side view in Figure 5.1c (maxima) and

Figure 5.1d (minima). The videos that accompany this thesis include illustrations of the resulting extrema within the context of the motion of the limb.

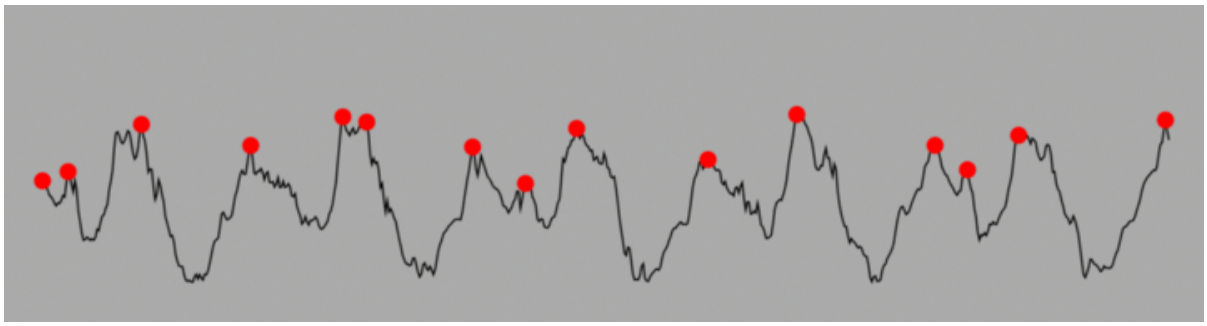
For this example, note that velocity maxima occur at many times, but not at the extreme points of the hip swing (Figure 5.1a). However, most velocity minima do occur at extreme points of the leg swing, while a small number occur mid–swing. This indicates that, *for recorded human motion, velocity minima typically correspond to extreme configurations, but that user–guided refinement might be needed if the selection should contain only extreme configurations of the joint.*

Acceleration: Acceleration extrema have the form

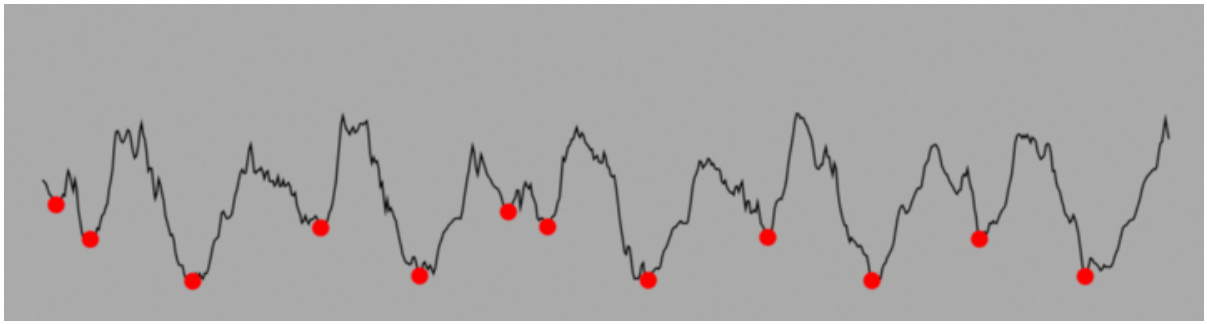
$$\{t_i : ||\ddot{\mathbf{t}}_j(t_i)|| \text{ is locally extreme}\}.$$

Acceleration is also evaluated using central differences. The acceleration plot and joint configurations at the times of acceleration maxima are shown in Figure 5.2. Note, first, that acceleration measures are significantly more sensitive to noise (Figure 5.2a), as discretized differential measures use linear combinations of the joint position samples, which increases the range of the resulting noise in the acceleration measure. This causes significantly more extrema to be selected, as the underlying signal has less contribution to the resulting measure (Figure 5.2b). However, acceleration maxima do include the extreme points of the hip swing. Overall, this indicates that *for recorded human motion, acceleration maxima include the times of extreme joint configurations, but the increased noise sensitivity can cause a larger number of non–extreme configurations to be chosen.* This increased noise sensitivity can be addressed by filtering the motion data more aggressively and having users refine the selection, but significantly more user–refinement will be necessary than when using velocity minima.

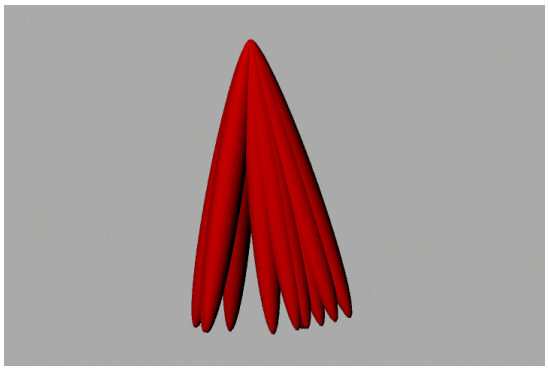
Curvature: For motion of joints with more than one degree of freedom, the curvature of joint trajectories can also be used to indicate that a joint is in an extreme configuration,



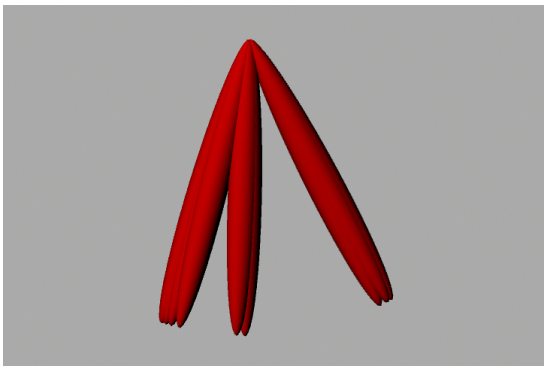
(a)



(b)

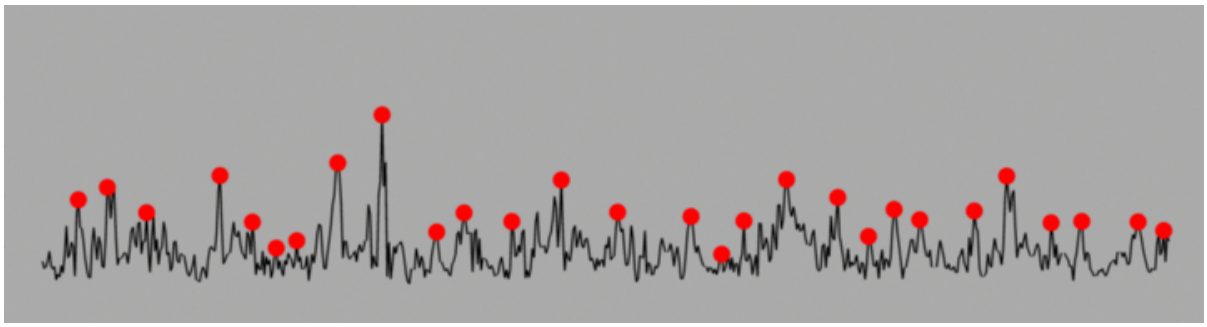


(c)

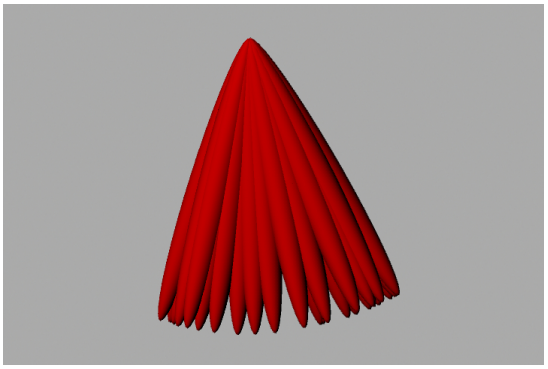


(d)

Figure 5.1: Velocity plots and corresponding extrema of the joint are shown for a rotating hip during a walk. Velocity plots include local maxima (a) and minima (b). Joint configurations at the corresponding times are shown for maxima and minima, respectively, in c and d.



(a)



(b)

Figure 5.2: The acceleration plot and resulting maxima are shown in a for a rotating hip during a walk. The joint configurations at the resulting motion extreme times are shown in b.

as the trajectory will tend to curve most at times when the joint changes direction of movement. The curvature will be discontinuous at these times for a joint with one degree of freedom. The approach described here is based on the rotational curvature measure described by Taubin [165]. He gives the discrete curvature measure at a sample i as

$$\kappa_i = 2 * \mathbf{n}_i \cdot \mathbf{e}_i / ||\mathbf{e}_i||^2,$$

where $\mathbf{n}_i$ is the normal vector of a Frenet frame and $\mathbf{e}_i$ is an edge vector between two samples i and $i+1$. This is the curvature approximation relative to one edge; to account for both edges adjacent to a sample, the average value as computed using each adjacent edge is used. This discrete measure has a range of $[0, \infty)$, which can be problematic when extending this approach to the full body, as a fast, small change of direction in a

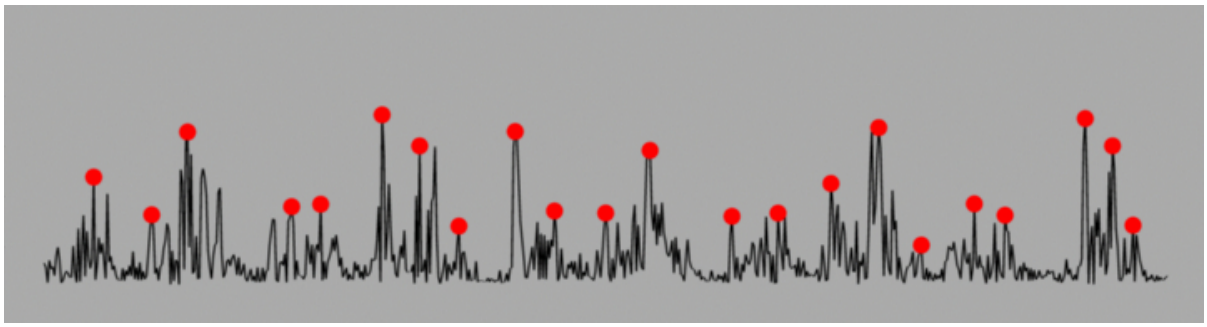
small joint can easily dominate any other motion in the body. To account for this, I use a modified *pseudo–curvature* measure $\hat{\kappa}$, which has a range of $[0, 1]$. This measure is

$$\hat{\kappa}_i = \mathbf{n}_i \cdot \mathbf{e}_i / ||\mathbf{e}_i||,$$

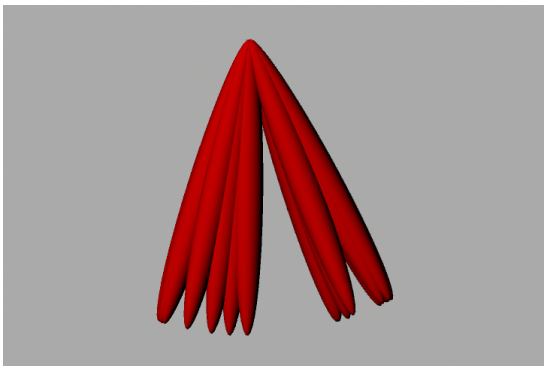
which effectively clamps the range of the measure to avoid arbitrarily large values.

Applying this pseudo–curvature measure to the joint motion, the motion extrema are then given as

$$\{t_i : \hat{\kappa}(\mathbf{t}_j(t_i)) \text{ is locally extreme}\}.$$



(a)



(b)

Figure 5.3: The pseudo–curvature plot and selected maxima are shown for the rotating hip (a). Joint configurations at the corresponding times are shown in b.

Figure 5.3a shows the plot of this pseudo–curvature measure for the rotating hip, along with the selected maxima. Again, noise has a significant affect on the measure, as curvature is also a second order differential measure. The joint configurations at the

times of the resulting motion extrema are shown in Figure 5.3b. Similar to acceleration, extreme configurations are included, but a number of non–extreme configurations are also included. This indicates that, similarly to acceleration, *when finding extrema in recorded human motion, curvature measures will select extreme configurations at local maxima, but also a number of non–extreme configurations.* Again, the increased effect of noise can be accounted for by additional filtering of the motion data or by user refinement of the selected extrema.

Associating Joint Motion with Parent Joint Rotation: The above approaches account for the motion of a single joint, and, as they measure joint motion in space, the resulting extrema relate to the *rotation* motion of the *parent* joint. Depending on the application, these extrema might need to be explicitly associated with the skeletal motion of the parent joint. This can be necessary when simplifying a skeletal motion representation into keyframe splines. If multiple child joints are present beneath one parent joint, all child joints must be taken into account, as it can be the case that a parent rotation moves one child joint significantly more than others.

Given multiple child joints j, the resulting measure for finding rotation extrema on the parent joint is evaluated as the maximum of the Cartesian measures of each child joint. This is given as

$$f(t) = \max_{j}(f_j(t)),$$

where $f_j(t)$ can be any of the above differential measures, or any other joint–centric measure, such as the spatial extremeness measure that will be presented in Section 5.4.

These approaches also do not account for limb twist in the parent joint, i.e., rotation about the limb itself, as this does not affect the Cartesian motion of the child joint. To account for this, an additional virtual limb $\hat{j}$ can be created that is perpendicular to the real limb, but undergoes the same skeletal motion. As $\mathbf{t}_{\hat{j}}$ is perpendicular to $\mathbf{t}_j$, it will be affected by limb twist, but might or might not be affected by limb swing, depending

on the direction of the swing. A second measure $f_{\hat{j}}(t)$ is evaluated on this virtual limb as $\max\left(f_j(t), f_{\hat{j}}(t)\right)$.

Combining this with the approach described above to handle multiple child joints, *the overall measure used to assign joint motion extrema to the rotation motion of a parent joint is*:

$$f(t) = \max\left(\max_j(f_j(t)), \max_{\hat{j}}(f_{\hat{j}}(t))\right).$$

5.3.2 Differential Extrema for Extreme Poses

The use of extrema of differential measures of motion to find extreme poses extends the ideas of Section 5.3.1 to handle full body motion. Full body differential measures use the Cartesian motion of joints in the world, i.e. the Global Cartesian motion representation. As such, the general form of the extrema of full–body differential measures is

$$\{t_i : \sum_j f(\mathbf{x}_j(t_i)) \text{ is locally extreme}\}.$$

The remainder of this section will apply the same differential measures used in Section 5.3.1 to find extreme poses, using walking an an example. Section 5.6 will later compare these approaches for different types of motion.

Velocity: Velocity extrema for full body motion have the form

$$\{t_i : \sum_j ||\dot{\mathbf{x}}_j(t_i)|| \text{ is locally extreme}\}.$$

Again, central differences is applied to the motion of each joint to evaluate the total velocity as a sum of individual joint velocities. The resulting velocity plots and the first portion of the set of chosen poses are shown in Figure 5.4 for a walk. Figures 5.4a and 5.4b again plot the velocity, and the chosen maxima (a) and minima (b) are again labeled in red. Of the selected poses, the first several are illustrated for maxima (Figure 5.4c) and minima (Figure 5.4d), and the full sets of poses can be seen in the videos associated with

this thesis[1].

In this example, note that velocity maxima consistently correspond to times between extreme poses, when the legs are close to passing each other (Figure 5.4c). In contrast, velocity minima correspond to those times at which the character has widest leg stance (Figure 5.4d); these are similar to the extreme poses illustrated by Williams [177]. This suggests that *velocity minima correspond well to extreme poses in recorded motion of the full body.*

Acceleration: Acceleration extrema for full body motion have the form

$$\{t_i : \sum_j ||\ddot{\mathbf{x}}_j(t_i)|| \text{ is locally extreme}\}.$$

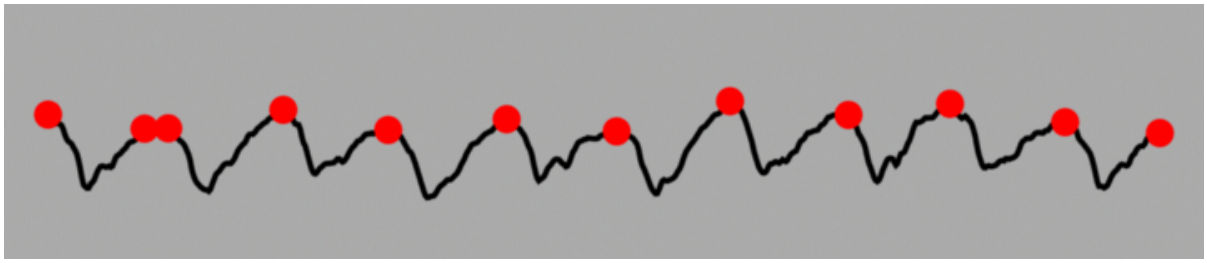
Again, acceleration is measured using central differences. The total acceleration is plotted in Figure 5.5a, and the selected maxima are labeled. While noise is still present in an increased amount, relative to total velocity, the underlying signal is still well represented. The poses corresponding to maxima of total acceleration are shown in Figure 5.5b. Note that while the majority of the selected poses appear to be extreme poses, some do not. The suggests that *acceleration maxima often correspond to extreme poses, but some user refinement of the selection might be needed.*

Curvature: Extending the pseudo–curvature measure for joint motion to the full body movement, motion extrema are given as

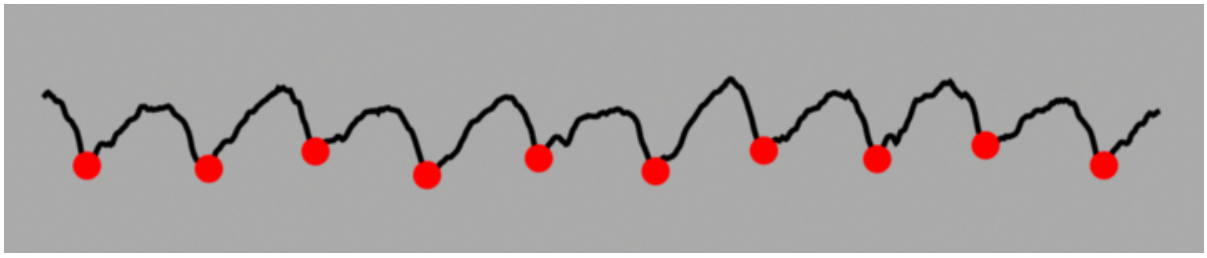
$$\{t_i : \sum_j \hat{\kappa}(\mathbf{x}_j(t_i)) \text{ is locally extreme}\}.$$

This measure is plotted in Figure 5.6a, and the selected maxima are shown. Figure 5.6b shows the corresponding poses. Again, noise makes a significant contribution to the

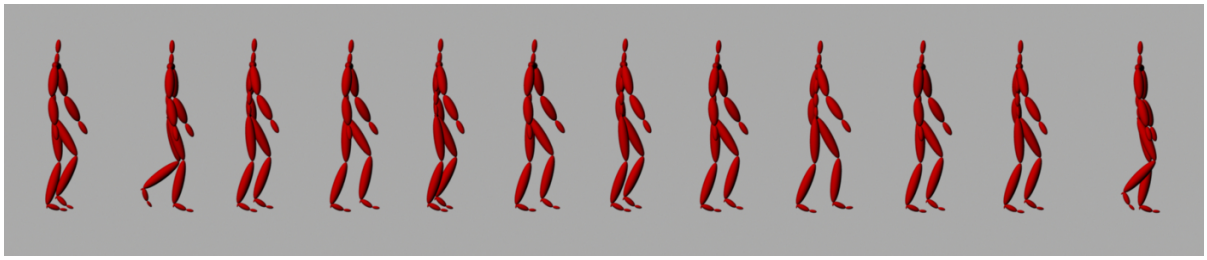
[1]Some algorithms select a very large number of poses for certain motion examples. For conciseness, only a sufficient number are included in figures to demonstrate the *types* of poses that are selected. For consistency, these will always be the first poses selected by the method.



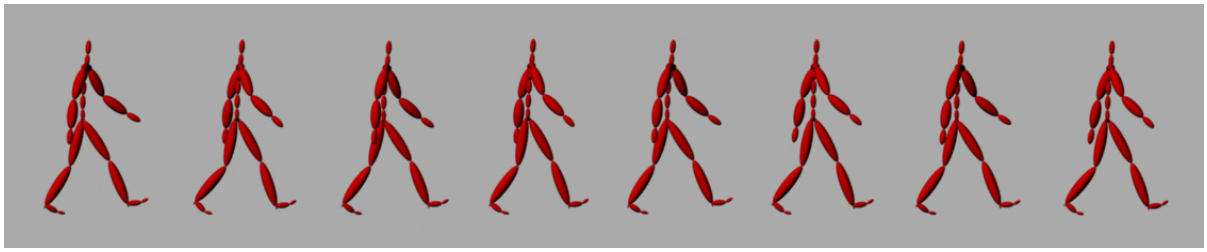
(a)



(b)

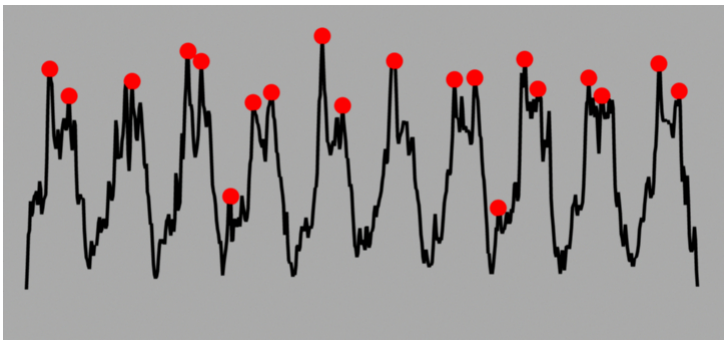


(c)

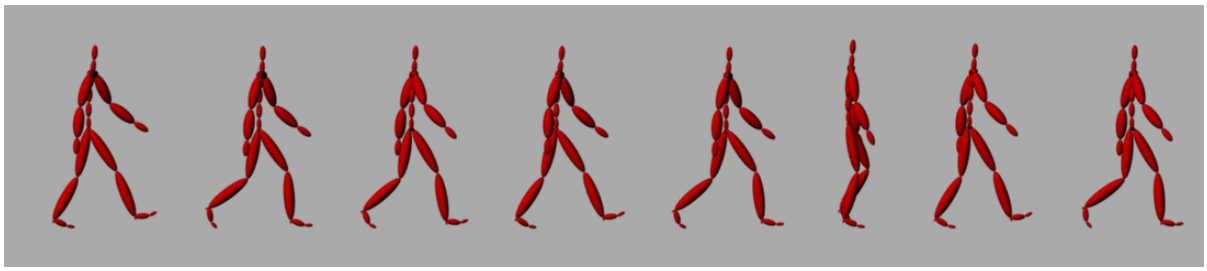


(d)

Figure 5.4: The velocity plots and resulting poses selected from a human walk. Velocity minima are shown in a, and the first handful of corresponding poses are shown in c. Velocity maxima are shown in b, and the first handful of corresponding poses are again shown in d.



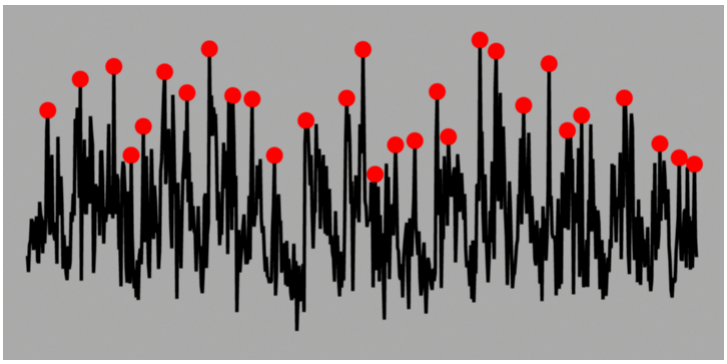
(a)



(b)

Figure 5.5: Totol acceleration of all joints is plotted in a, and selected maxima are labeled. Corresponding poses are shown in b.

resulting measure. However, many of the selected poses are extreme poses, although a fair number of non–extreme poses are also chosen. This suggests that *maxima of curvature measures often correspond to extreme poses for recorded full–body motion, but some user refinement of the selection will be needed.*



(a)

(b)

Figure 5.6: Pseudo–curvature and selected maxima (a) and the corresponding poses (b).

5.4 Spatially Extreme Configurations

The second approach to finding motion extrema aims to explicitly identify times at which either a joint or the body is in a spatially extreme configuration. Extreme poses occur at times when the geometric state of the body or joint is most distant to positions in temporally neighboring poses or joint configurations. The distance metrics used here are based on Cartesian distance, as it linearly relates to how much the body or a joint moves in space. As with approaches based on differential measures of motion, joint–centric motion extrema will be identified using the Local Cartesian motion representation, as it isolates the motion of each joint relative to the position of its parent joint. For the full body, the Global Cartesian motion representation will be used.

To find motion extrema at spatially extreme configurations, I define a continuous *spatial extremeness* measure $e(t)$ that measures how unique a pose or joint configuration is at any given time. Joint–centric extrema or extreme poses are then selected at those

times at which $e(t)$ has a local maximum. Given a temporal neighborhood $[t_a, t_b]$ that contains t, the spatial extremeness at t is defined as the average distance, under an appropriate metric, of a pose or joint to its neighbors in that temporal neighborhood. Generically denoting the pose or joint configuration at a given time as $\mathbf{P}(t)$, spatial extremeness is then

$$e(t) = \frac{1}{t_b - t_a} \int_{t_a}^{t_b} ||\mathbf{P}(t) \ominus \mathbf{P}(\tau)||\ d\tau, \tag{5.1}$$

where $||\mathbf{P}(t) \ominus \mathbf{P}(\tau)||$ denotes the appropriate distance metric, measured in terms of the difference between two poses or joint configurations. For joint–centric extrema in the local Cartesian representation, this metric is

$$||\mathbf{P}(t) \ominus \mathbf{P}(\tau)|| = ||\mathbf{t}_j(t) - \mathbf{t}_j(\tau)||, \tag{5.2}$$

and for full body poses in the global Cartesian representation, this metric is

$$||\mathbf{P}(t) \ominus \mathbf{P}(\tau)|| = \sum_{j \neq r} ||\ (\mathbf{x}_j(t) - \mathbf{x}_r(t)) - (\mathbf{x}_j(\tau) - \mathbf{x}_r(\tau))\ ||, \tag{5.3}$$

where r denotes the root joint.

The latter metric measures joint positions *relative to the root position*; this causes the metric to measure the stance of the body independently of its location in space. This can be important, as many motion examples, including walking, will have joint positions that are bounded with respect to the root, but follow long, smooth trajectories in the world. As this measure has been designed to *explicitly measure uniqueness of body stance*, the effect of overall location in space is subtracted.

It is common in the literature to factor out root orientation when designing distance metrics for determining times at which to align motion clips [74]. This is not done here, as some motion examples, particularly those in which the character turns while standing still, can have most of the motion represented as a change in the root orientation. Unlike when choosing pose alignment times, the information contained in the root orientation is important for the selection of extreme poses that are meaningful for editing, as it can

contain the movement information that represents turning, which can be the dominant effect of stance change for such motion examples.

To determine the temporal neighborhood used to evaluate Equation 5.1, an adaptive search strategy will be used that searches for time windows that represent locally smooth movement away from the pose or joint configuration at time t. This will be discussed in more detail in Section 5.4.3.

5.4.1 Joint–Centric Spatial Extremeness

To find joint–centric extrema, Equation 5.1 is evaluated using the metric given in Equation 5.2. This results in the joint–cenric spatial extremeness measure

$$e(t) = \frac{1}{t_b - t_a} \int_{t_a}^{t_b} ||\mathbf{t}_j(t) - \mathbf{t}_j(\tau)|| d\tau.$$

This measure can be discretely evaluated as

$$e(t) = \frac{1}{N} \sum_{\tau_i} ||\mathbf{t}_j(t) - \mathbf{t}_j(\tau_i)||,$$

where the summation is over all times $\tau_i \in [t_a, t_b]$, and N is the number of time samples in $[t_a, t_b]$.

Consider the swinging pendulum shown in Figure 5.7, which is similar to the motivating example given in Chapter 1. $\mathbf{P}_0$, $\mathbf{P}_2$, and $\mathbf{P}_3$ are spatially extreme configurations, as they occur at extreme points of the swing. $\mathbf{P}_1$ is not spatially extreme, and tells us little about how much the pendulum is swinging. When choosing neighborhoods for the spatial extremeness metric, the resulting neighborhoods should include neighboring times that are bounded by temporally adjacent changes in the direction of movement. For $\mathbf{P}_1$, this is $[t_0, t_2]$, and for $\mathbf{P}_2$, this is $[t_0, t_3]$. The neighborhood for $\mathbf{P}_2$ includes configurations that are spatially more distant than any in the neighborhood of $\mathbf{P}_1$, and this will result in a higher extremeness value for $\mathbf{P}_2$ than is measured for $\mathbf{P}_1$. Formally, this means $e(t_2) > e(t_1)$. Section 5.4.3 provides the details of how to choose neighborhoods to meet these goals.

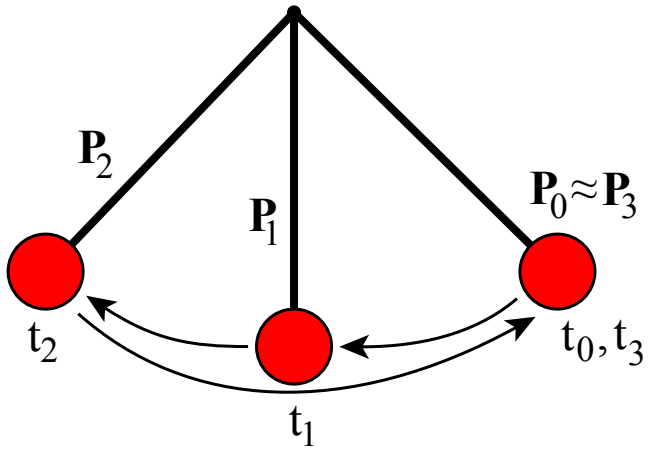
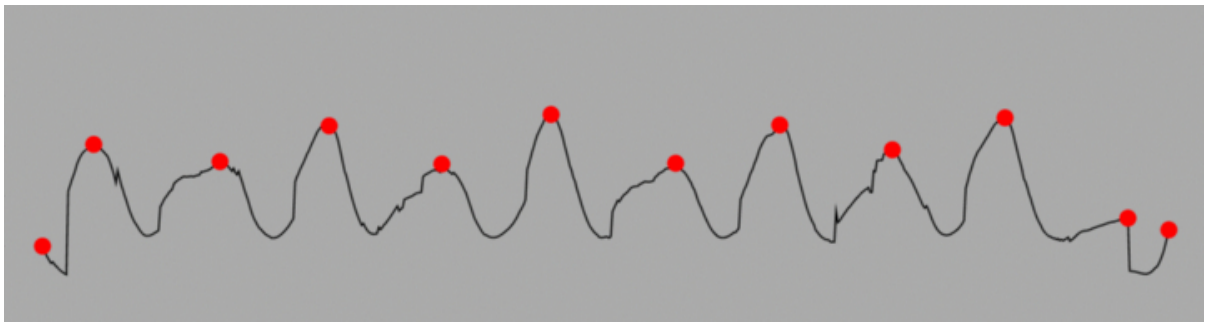


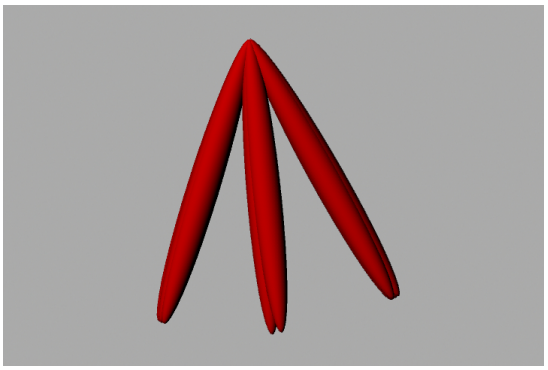
Figure 5.7: The spatial extremeness measure explicitly chooses poses when a joint or body, such as this pendulum, is in a spatially extreme configuration. This is true of poses $\mathbf{P}_0$, $\mathbf{P}_2$, and $\mathbf{P}_3$, but not $\mathbf{P}_1$.

Figure 5.8a shows the plot of the spatial extremeness measure for the rotating hip joint. Note that the measure is significantly less sensitive to noise than the differential measures, and the maxima, in general, are much more distinct than for differential measures. However, as the temporal bounds can be discontinuous, the measure can have sudden changes, such as occur between the last two maxima. The small spikes in the the plot, such as that occurring between the second and third maxima, are also due to jumps in the temporal neighborhood. In practice, these tend to occur when the measure is changing significantly, and the semi–local extrema selection algorithm will reject them, as other samples in the small neighborhood used to find semi–local extrema will have a greater/lesser value. These could also be removed by modifying the neighborhood selection algorithm to choose neighborhoods whose bounds monotonically increase in time; this remains as future work.

Also note that the spatial extremeness measure has extrema at the first and last frames. This is expected, as the endpoint of a trajectory must be more distinct than its temporal neighbors, unless the joint begins or ends at rest.



(a)



(b)

Figure 5.8: The spatial extremeness measure and selected maxima for the rotating hip are plotted in a. The joint configurations corresponding to the maxima are shown in b.

The joint configurations that correspond to the selected maxima are shown in Figure 5.8b. All joint configurations occur at times when the hip is at the extreme point of the swing, with the exception of two. These occur at the first and last frame. In practice, these two local extrema can be explicitly rejected, depending on the needs of the application that the joint–centric extrema are used for, as they are easily identified.

5.4.2 Full–Body Spatial Extremeness

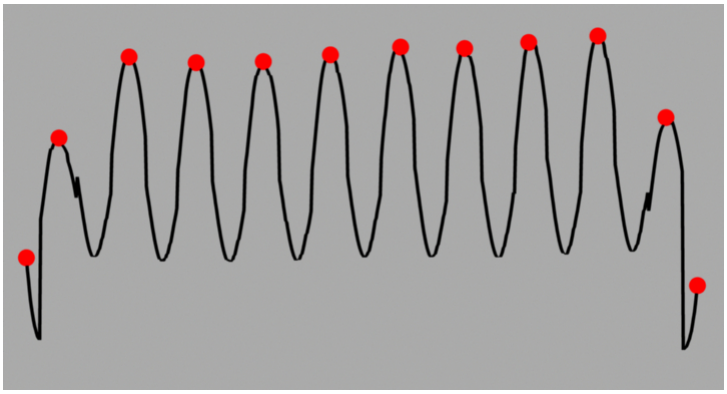
For full body motion, Equation 5.1 is evaluated using the metric in Equation 5.3. The resulting full–body spatial extremeness measure is

$$e(t) = \frac{1}{t_b - t_a} \int_{t_a}^{t_b} \sum_{j \neq r} || \left(\mathbf{x}_j(t) - \mathbf{x}_r(t)\right) - \left(\mathbf{x}_j(\tau) - \mathbf{x}_r(\tau)\right) || d\tau.$$

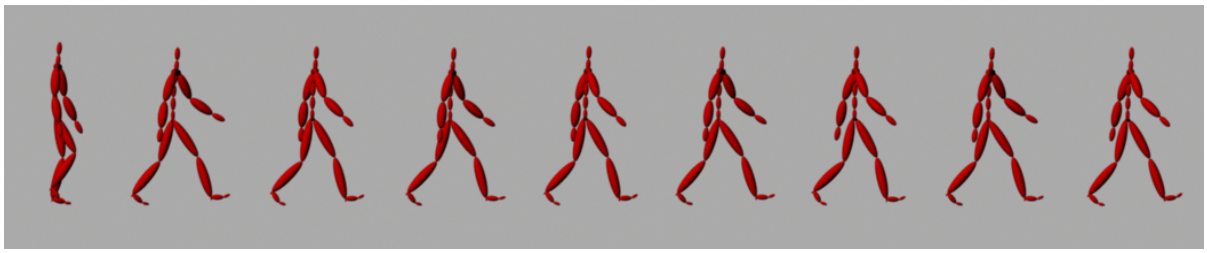
which has the discretization

$$e(t) = \frac{1}{N} \sum_{\tau_i} \sum_{j \neq r} || \left(\mathbf{x}_j(t) - \mathbf{x}_r(t)\right) - \left(\mathbf{x}_j(\tau_i) - \mathbf{x}_r(\tau_i)\right) ||.$$

Figure 5.9a shows the plot of the full–body spatial extremeness measure for the human walk. Again, the amount of noise present in the measure is significantly less than that present in the differential measures, although the discontinuities due to jumps in the temporal neighborhoods also remain. Again, the first and last frames are selected. The poses corresponding to the spatial extremeness maxima are shown in Figure 5.9b. All poses, except the first and last, occur at times when the legs have approximately the widest stance and the arms are near the extreme swing points, meeting the goal of the spatial extremeness measure. As with joint–centric extrema, the first and last frames can be easily culled if needed.



(a)



(b)

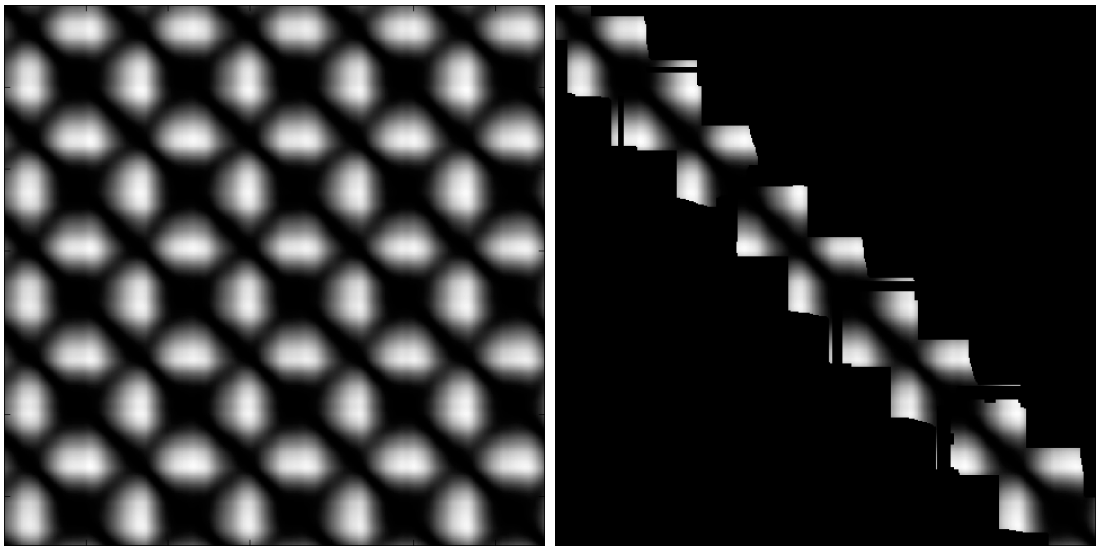
Figure 5.9: The spatial extremeness measure and selected maxima for the human walk are shown in a. The poses corresponding to the maxima are shown in b.

5.4.3 Neighborhood Selection

The temporal neighborhood used to evaluate the spatial extremeness measures is chosen such that it includes neighboring regions of time that are bounded by changes in the direction of movement. To do this, the time window is iteratively expanded to find temporally adjacent regions in which the distance metric increases approximately monotonically. When distance begins to decrease, the neighborhood bounds are set to the times with the nearby states that are most distant under the chosen metric.

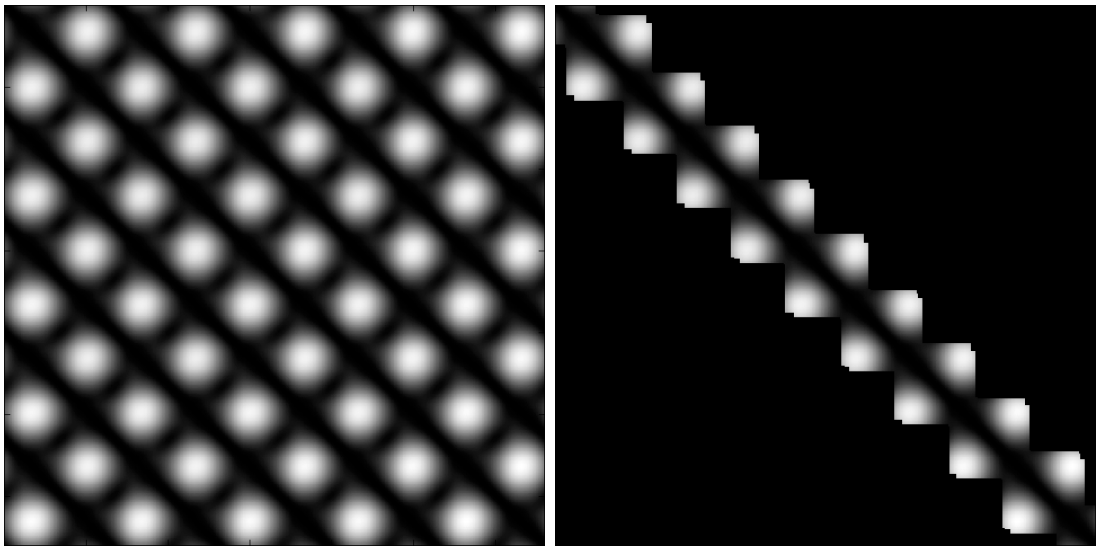
To search forward, the algorithm initializes t_b to $t+1$. It then iteratively measures $||\mathbf{P}(t_b+1) \ominus \mathbf{P}(t)||$ and increases t_b by one frame if this value is larger than $||\mathbf{P}(t_b) \ominus \mathbf{P}(t)||$. The algorithm halts the search when $||\mathbf{P}(t_b+1) \ominus \mathbf{P}_t||$ is less than $||\mathbf{P}(t_b) \ominus \mathbf{P}_t||$ by some tolerance measure. This tolerance measure is set to account for noise in the data that might prevent the distance from increasing in a truly monotonic manner. For the examples in this thesis, this tolerance is adaptively set such that a 5% decrease in the distance metric relative to the metric's value at t_b causes the search to halt. A similar strategy is used in reverse to find t_a. The neighborhood is also initialized to a minimum width, again to account for noise; a width of 10 frames is used for the examples shown in this thesis.

When evaluating the distance metrics, note that for any two time samples t_1 and t_2, $||\mathbf{P}(t_1) \ominus \mathbf{P}_{t_2}|| = ||\mathbf{P}(t_2) \ominus \mathbf{P}_{t_1}||$. Typically, this value will be needed twice, when searching forward from t_1 and when searching backward from t_2. As evaluation of the distance metric is the costliest part of the algorithm, re–computation is avoided by caching values in a symmetric distance matrix. The full distance matrix for the hip motion is shown in Figure 5.10a, and the full distance matrix for the walk is shown in Figure 5.11a. The distance metric values that are actually used for evaluating the full spatial extremeness measure are shown in Figures 5.10b and 5.11b. The values appear in an approximately banded structure and represent the evaluations used when searching for neighborhoods. In Figure 5.10b, some entries are missing along partial rows or columns that cause sig-



(a) (b)

Figure 5.10: The full matrix of the distance metric values for the rotating hip motion (a) and those entries that are used to evaluate joint–centric spatial extremeness (b).



(a) (b)

Figure 5.11: The full matrix of the distance metric values for the human walk (a) and those entries that are used to evaluate full–body spatial extremeness (b).

nificant instantaneous discontinuities to appear in the neighborhood bounds. These are the neighborhood discontinuities that cause the spatial extremeness measure to have discontinuous jumps. The underlying cause for these omissions is that the noise in the data at these times causes the search to stop sooner than at adjacent times. While the use of semi–local extrema to identify spatial extremeness maxima prevents any problems associated with this, alternative strategies include increasing the tolerance used for halting the neighborhood search and increasing the minimum neighborhood size. In these figures, the full distance matrices are shown for illustration purposes; only the partial matrices are used when evaluating spatial extremeness.

5.5 Robust Identification of Extrema

As recorded human motion data contains some amount of noise, the skeletal reconstruction process often involves the application of a low–pass filter to remove noise from the motion data. However, any remaining noise, while not often visually significant, can have a significant effect on how well joint–centric motion extrema and extreme poses can be found, as noise leads to the presence of many local extrema that do not result from the underlying signal. A variety of strategies can be used to robustly detect extrema in the presence of noise; those that have been used in the various editing systems are described here.

Laplacian Filtering: Laplacian filtering can be applied to either the joint trajectories or the resulting measures, whether the measure is a differential measure or spatial extremeness. It has the benefit of simplicity, but many iterations will be needed if a significant amount of noise is present, which can significantly impact computational performance. The staggered poses editing system applies Laplacian filtering to joint trajectories before selecting local extrema, and this is the computational bottleneck of that

system.

Linear Low–Pass Filters: Linear low–pass filters, such as box, tent, or Gaussian filters, can also be applied to reduce noise. However, aggressive filtering might be needed if a significant amount of noise is present, which reduces the information retained from the underlying signal. This approach was initially used in the pose–centric editing and spatial exaggeration systems, but some measures required aggressive filtering to reduce noise to the point that local extrema were easily identified. This aggressive filtering resulted in significant information loss, and a number of expected extrema were not identified after the necessary amount of filtering was applied.

Semi–Local Extrema: To achieve fast performance, a stricter condition than local extremeness can be required for the of the selection of extrema from a noisy signal. This condition requires that a selected extrema, rather than being local extrema with respect to neighboring samples, be extreme with respect to all samples within some time window. Formally this condition requires that

$$f_i > f_j \quad \forall j \in [i - W/2, i + W/2]$$

for sample f_i to be considered a local extremum. In this condition, W is the width of the time window, and the interval represents the sample indices in the time window surrounding index i. As this condition extends the notion of a local extremum to a condition about a wider window of time, values that meet this condition are referred to as *semi–local extrema*.

This approach to selecting extrema in the presence of noise has been used in all examples presented in this chapter, and the approach is also used in the pose–centric editing and spatial exaggeration systems.

5.6 Comparing Extreme Selection Models

The above sections use a single motion example to demonstrate how the different approaches for selecting motion extrema vary with respect to each other. For joint–centric extrema, a rotating hip from a human walk has been used. For extreme poses, the full walking motion has been used. This section aims to compare how well each approach identifies extrema for a variety of motion examples, in terms of their ability to select extrema that have qualities similar to the extreme poses used by keyframe animators.

For joint–centric extrema, this section focuses on comparing how the measure associated with each approach identifies extrema for *different types of motion data*, as an isolated joint captures variation in qualities of movement independently of complexities that arise with full–body movement, such as coordination. For the full body, however, coordination plays a much stronger role in how extreme poses are chosen. Therefore, the full–body comparisons focus on *varying the type of movement*, which affects how much coordination is present. Recorded human motion data is used for the full–body comparisons, primarily due to the ready availability of high–quality recorded motion data.

5.6.1 Joint–Centric Extrema

For joint–centric extrema, three types of motion data have been used to compare the different approaches for extreme selection. The recorded hip motion, used in Section 5.3 as an illustrative example, is repeated here for comparative discussion. A keyframe–animated pendulum motion is used to determine how well the various models identify extreme configurations in interpolated motion data. Finally, physically–simulated motion is used for the third example, in which a pendulum is simulated as a constrained rigid body. As in the prior sections, semi–local extrema are used to identify local minima and maxima in the presence of noise, using a time window of twenty frames.

For each example, the goal is to select extrema at times when the individual joint is

at the the extreme point of a swing. These times match the observations made of the extreme poses that animators create: changes in direction of movement, slow speed, large motion trajectory curvature, and high spatial extremeness.

Recorded Hip Motion:

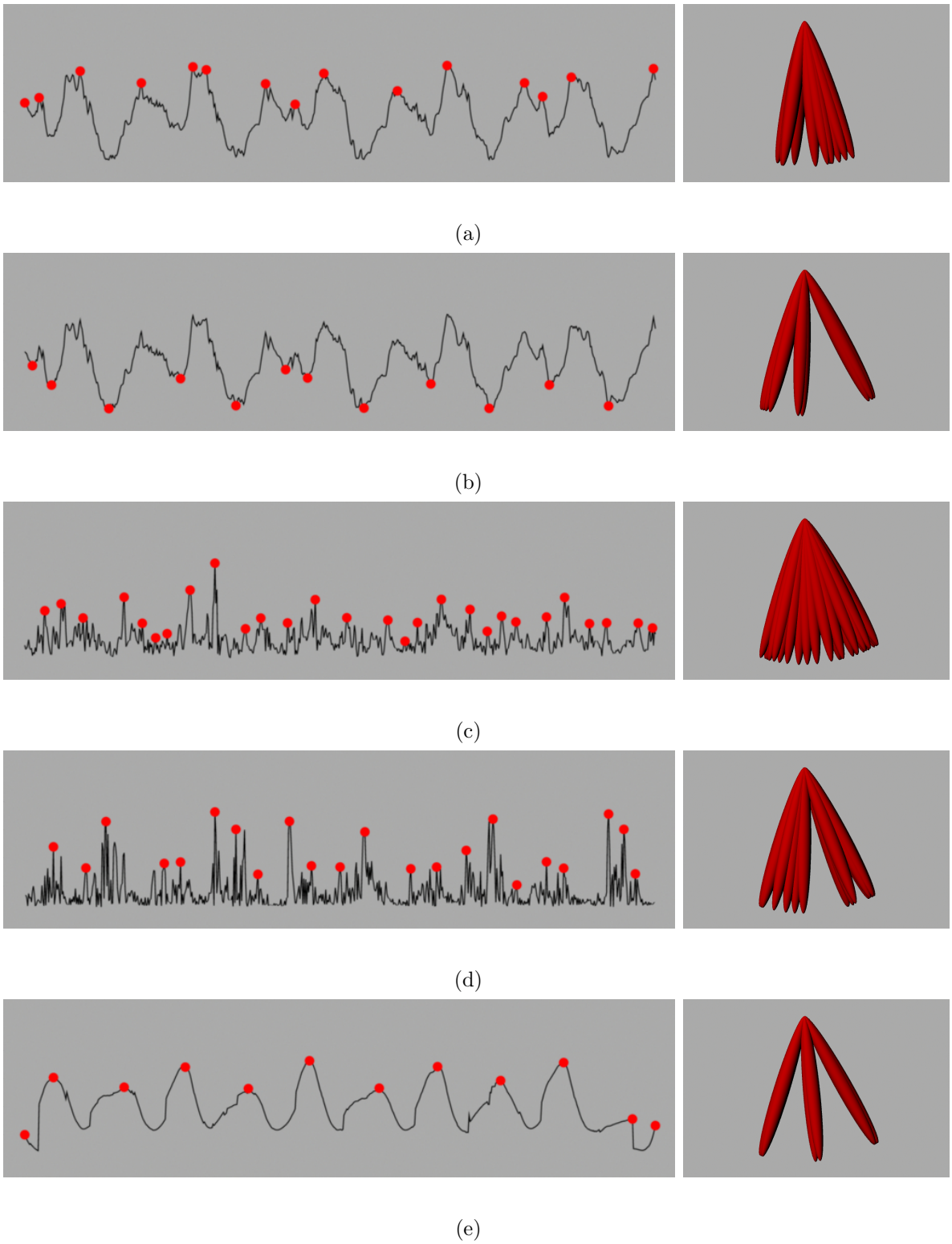
The recorded hip motion results are collectively illustrated in Figure 5.12. For this and all subsequent examples, plots of the various measures are shown on the left, along with the selected extrema highlighted as red circles, and the joint configurations at the times of the selected extrema are shown on the right. For this and all subsequent examples that include recorded motion, the motion has a frame rate of 120 frames per second.

As noted previously, the pseudo–curvature and acceleration measures are the most sensitive to noise, while the spatial extremeness measure is the least sensitive. All measures pick at least some times that do not match the observations made of extreme poses, but the spatial extremeness measure selects these at predictable times—the first and last frames. Among the other measures, velocity minima most consistently occur at times that match the observations made of extreme poses. However, some minima of velocity occur at less predictable times that do not meet these goals. Overall, the spatial extremeness measure most consistently selects extreme joint configurations in this motion example, as the selected non–extreme configurations occur at predictable times.

Keyframe–Animated Pendulum:

The keyframe–animated pendulum was created using the commercial animation software *Maya*[2]. Keyframes were specified at two times—the two extreme configurations—and these were repeated to cause the motion to repeat twice. Maya's default keyframe settings were used, which constrain tangents on cubic splines to remain constant in length. These tangent constraints were set to zero slope to create motion that eases into and out

[2]http://www.autodesk.com/maya



(a)

(b)

(c)

(d)

(e)

Figure 5.12: Plots, selected extrema (left), and joint configurations at the resulting extreme times (right) for the rotating hip. Velocity maxima (a), velocity minima (b), acceleration maxima (c), pseudo–curvature maxima (d), and maxima of spatial extremeness (e).

of the extreme joint configurations, similar to how a real pendulum would swing. For extreme identification, the motion was sampled at 24 frames per second.

The results for the keyframe–animated pendulum are shown in Figure 5.13. The joint–centric extrema for velocity maxima (a), velocity minima (b), and acceleration (c) appear to be consistent with those selected for the rotating hip motion. However, note the subtle local maxima in the acceleration measure that occur between the selected, significantly larger, local maxima. This are due to the form of interpolation used in Maya's spline representation. In this example, the subtle local maxima are narrow enough that they are not selected as semi–local maxima, but this is not always the case, as the acceleration magnitude depends on the state of the extreme poses. In interactive experiments with the keyframe–animated pendulum, it was found that the interpolation–induced acceleration maxima that occur between extreme configurations grow significantly larger when the pendulum swings less, causing them to be chosen as extreme times, even though the pendulum is not in a state that matches the observations made of extreme poses.

Pseudo–curvature maxima are shown in Figure 5.13d, and the green joint configurations have been manually selected to provide illustrative context. Note that, in this case, the maxima occur mid–swing, unlike in the rotating hip. Figure 5.13e shows the pseudo–curvature minima, which occur at extreme pose times. These selected extreme times are the opposite of those that would be expected, given the rotating hip example. However, this can be explained by the constant underlying curvature of the joint trajectory, as the pendulum only rotates about one degree of freedom. All variation in pseudo–curvature is due to change in edge length, which affects the value of the discrete approximation. In this case, the pseudo–curvature measure does not work well, as there is not sufficient curvature variation in the underlying motion of the pendulum.

In Figure 5.13e, the last frame would be expected to be selected as a local minimum. Upon investigation, it was found that two subsequent samples had the same floating point value, causing them to be rejected by the semi–local extremum test, which effectively

considers this a small flat region.

As shown in Figure 5.13f, the spatial extremeness approach selects the times that match the observations made of extreme poses, and no times that do not. Overall, maxima of this measure correspond to extreme joint configurations, as do velocity minima. However, the effectiveness of velocity minima depends on the animator having created approximately natural timing in the motion. Acceleration maxima correspond to times that match the observations of extreme poses for this example, but, again, different timing patterns or geometric keyframe configurations can cause times to be selected that do not match these observations. Pseudo–curvature is not effective for this example, as the underlying curvature is constant, but the measure might work well for other types of keyframe–animation.

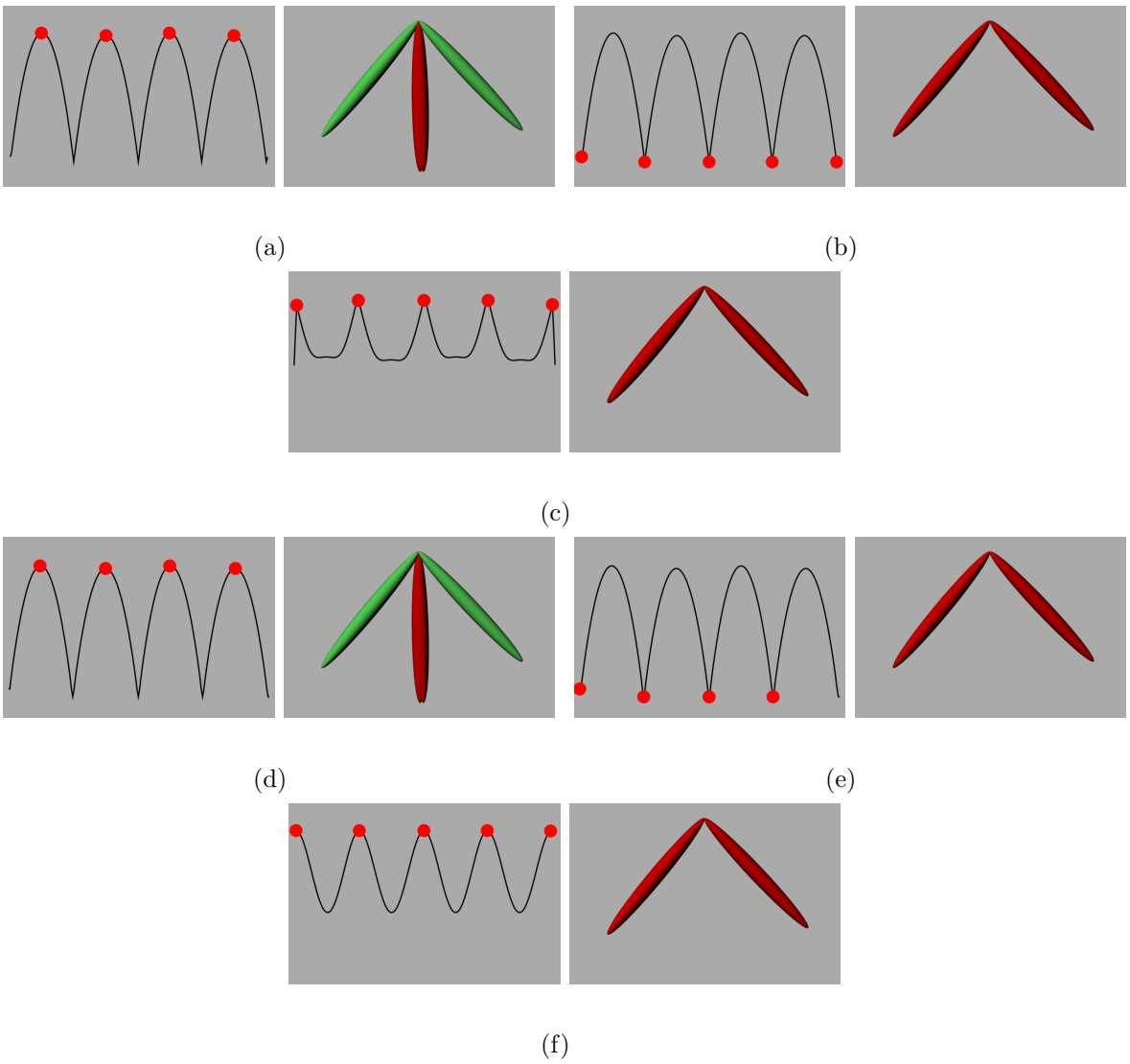


Figure 5.13: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d) and minima (e), and maxima of spatial extremeness (f) for the keyframe–animated pendulum.

Simulated Pendulum:

The joint–centric extrema chosen for the simulated pendulum motion are shown in Figure 5.14. The simulated pendulum motion was created using Maya's rigid body simulator, using a single rigid body with a hinge constraint on one end. Gravity and drag fields were applied to created damped oscillatory movement, and the pendulum motion was directly mapped to a rotating skeletal joint for analysis.

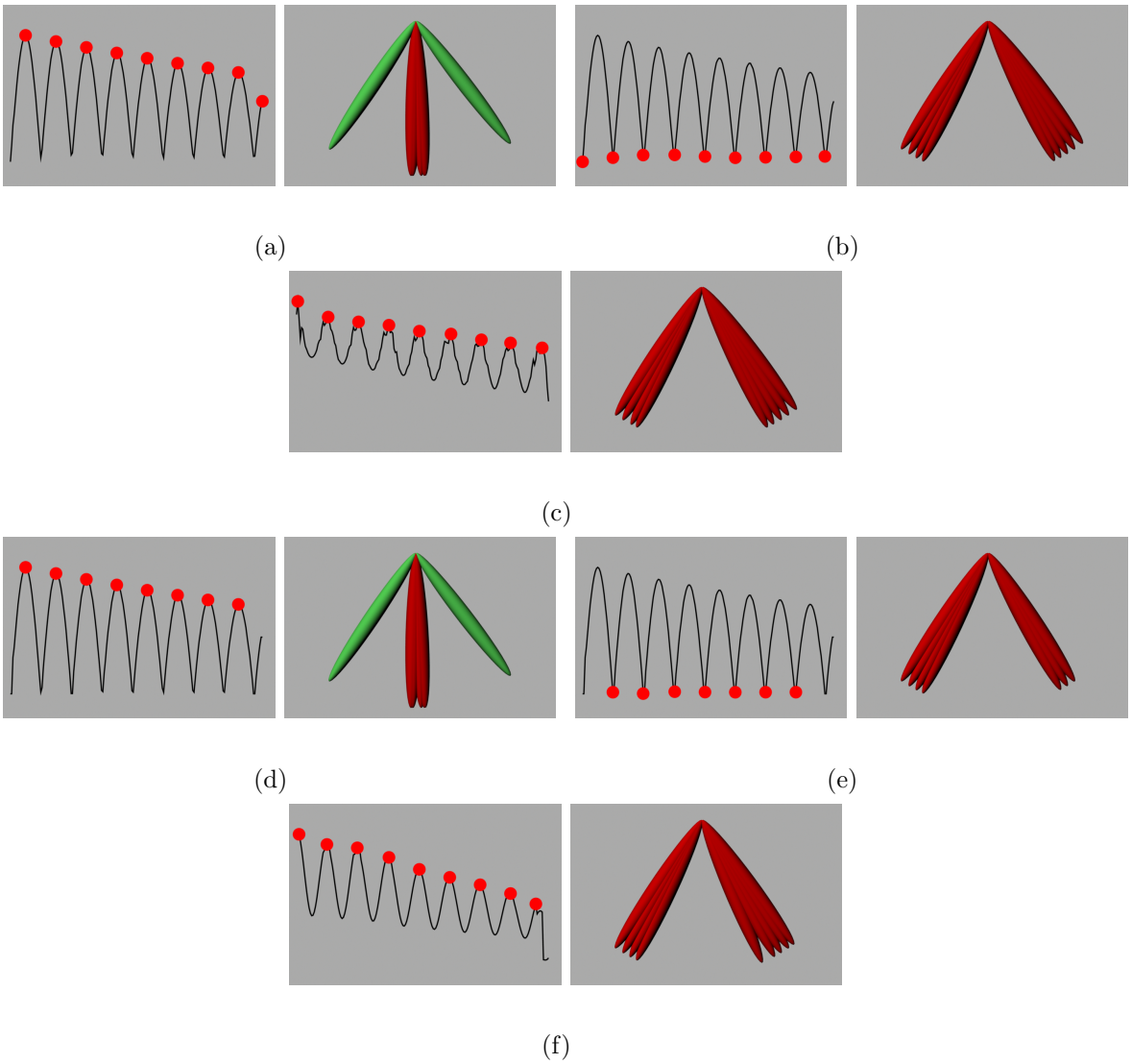


Figure 5.14: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d) and minima (e), and maxima of spatial extremeness (f) for the simulated pendulum.

As with prior examples, velocity maxima occur *between* times that match the observations made of extreme poses (Figure 5.14a), and velocity minima occur *at* these times (Figure 5.14b). In this case, acceleration maxima also occur at times that match the observations made of extreme poses (Figure 5.14c). Similarly to the keyframe–animated pendulum, pseudo–curvature maxima occur between the goals times (Figure 5.14d), and minima occur at times that meet the goals of matching the observations made of extreme poses (Figure 5.14e). This is again due to lack of curvature in the trajectory of the bottom of the pendulum. This lack of significant curvature indicates that pseudo–curvature is not a reliable measure for finding joint–centric extrema on constrained simulation systems. Spatial extremeness (Figure 5.14f) again leads to the choice of joint extrema at times that match the observations made of extreme poses.

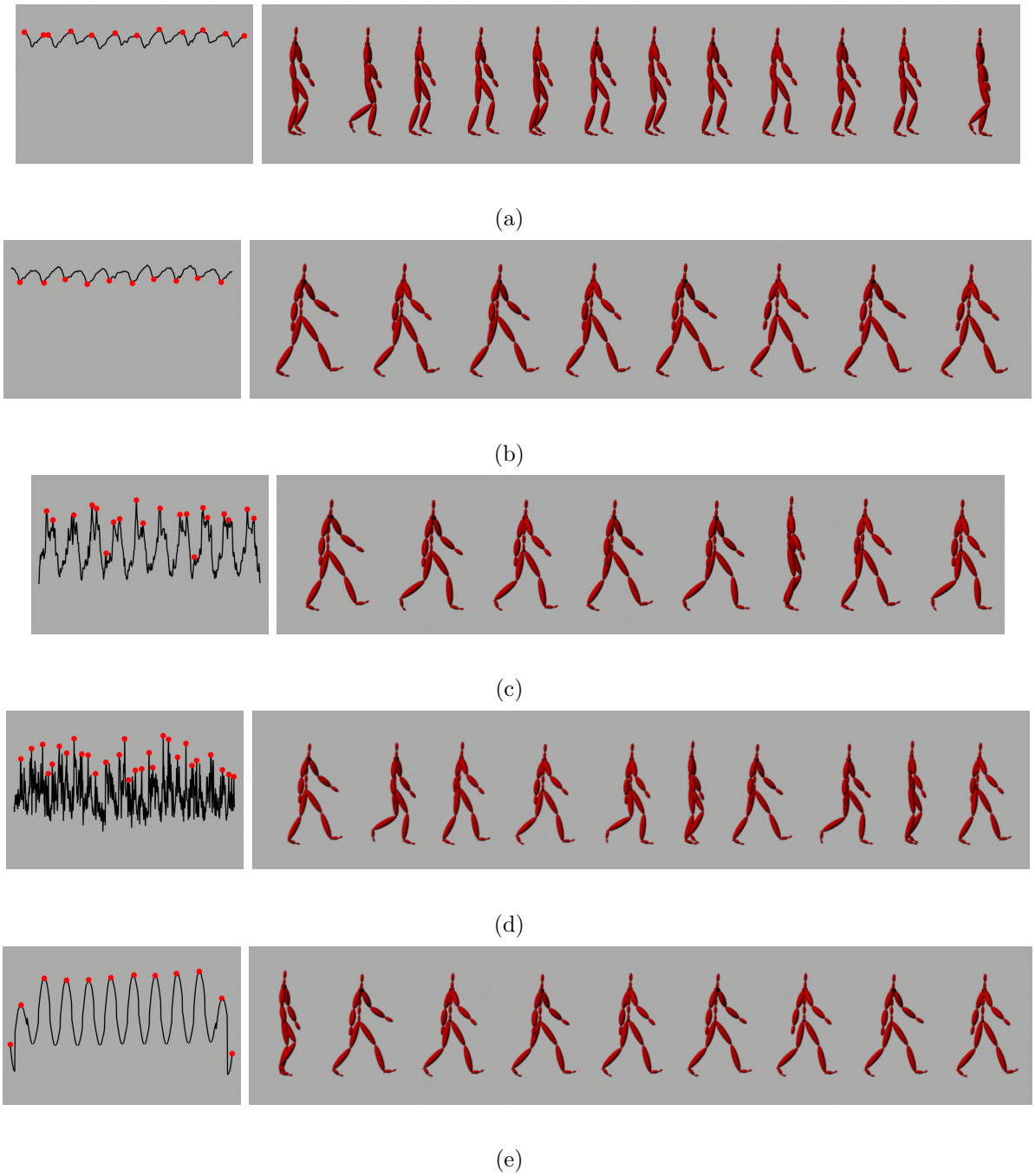
Overall, velocity minima, acceleration maxima, and spatial extremeness maxima all predict the times that match the extreme joint configurations for this example. Pseudo-curvature does not, due to the constraints in the simulation.

5.6.2 Extreme Poses

To compare the various approaches to selecting extreme poses, a variety of motion examples have been chosen that vary in terms of the activity the character is carrying out. These examples include coordinated motion, less coordinated motion, standing, and cartwheel motion. For each example, the plot of the measure used to identify extreme poses is shown, and the extrema that are selected are again highlighted in red. Again, the first selected poses are shown, and the videos associated with this thesis illustrate the selection of all extreme poses chosen by the various methods. The walk motion is repeated here for comparative illustration.

Walk:

The poses chosen for the walk motion, an example of coordinated motion, are shown



(a)

(b)

(c)

(d)

(e)

Figure 5.15: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d), and maxima of spatial extremeness (e) for the walk motion.

in Figure 5.15. As each example was discussed in more detail earlier in this chapter, a quick summary follows: Both velocity minima and spatial extremeness maxima correspond well to desired extreme poses—these occur at times when the arms and legs are maximally spread. Again, spatial extremeness maxima also occur at the first and last frames, which are not extreme poses. Acceleration maxima and pseudo–curvature maxima correspond to many poses that are close to extreme, but they also correspond to a number of non–extreme poses, with pseudo–curvature maxima corresponding to the most non–extreme poses.

In general, for the remaining examples, the goal is to select poses that have qualities similar to the extreme poses of the walk:

- Limbs change direction of movement.
- Limbs are at their slowest speed.
- Joints have high trajectory curvature.
- The positions of the joints are the most distant to corresponding positions at other times.

As these tend to correlate to the extreme points of swing of major limbs, the desired poses will occur at times when limbs are maximally spread.

Run:

Running is an another example of coordinated motion—typically more so than walking. The extreme poses selected by the different extreme selection methods are shown in Figure 5.16. Velocity minima occur at two typical points in time over the course of the run cycle—at mid–flight and again at landing (Figure 5.16a). At landing, the legs are spread, but arms are typically close to the body and are not at extreme swing. In mid–flight, the arms are wider in the swing, but the legs are closer together.

Velocity maxima, shown in Figure 5.16b, include poses where both legs are spread and the arms are at points of wide swing. However, they also include a number of poses in which most of the body is not in an extreme point of movement, but the knees are bent the most. The latter type of pose would be useful if users wish to edit the extent of knee swing, but they are less useful for editing how much the rest of the body moves. This is an example where the type of poses chosen can vary based on the intention of the edit. Acceleration maxima are similar to velocity minima, but acceleration maxima include both types of poses in a more robust way (Figure 5.16c), as the local extrema of the velocity measure are sometimes subtle (Figure 5.16b, left).

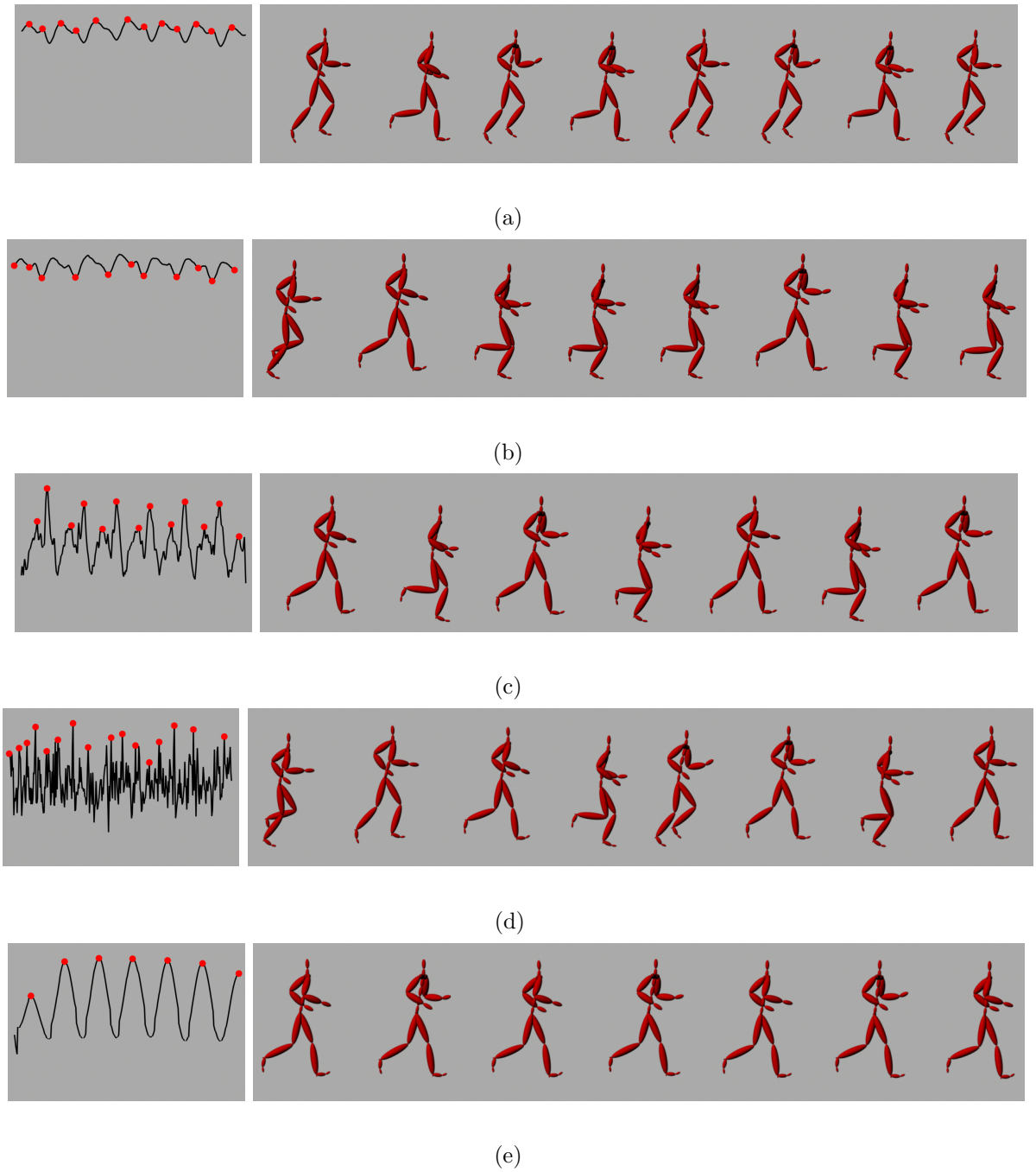
Pseudo–curvature is again very sensitive to noise in the data, and, while some extreme poses are chosen, as well as some poses where the knees are fully bent, the amount of noise sensitivity causes them to be inconsistently chosen over the duration of the motion (Figure 5.16d).

Spatial extremeness, for this example, is very robust at choosing times when the legs are maximally spread, as the feet have the largest affect on the underlying distance metric (Figure 5.16e). These poses would be less useful, however, if users wanted to specifically edit knee motion, as extreme knee configurations occur out of phase with extreme leg and arm configurations.

Overall, spatial extremeness most robustly selects the poses in which most of the body is in an extreme configuration. Acceleration maxima also include poses that capture the extreme times of knee bending, as do velocity minima, but less robustly. Other measures are less consistent.

Jump:

The jump motion, shown in Figure 5.17, starts with a step, transitions to a leap, and then continues with walking. For this example, velocity minima occur twice during the leap—once on the way up and once on the way down (Figure 5.17a). More occur during



(a)

(b)

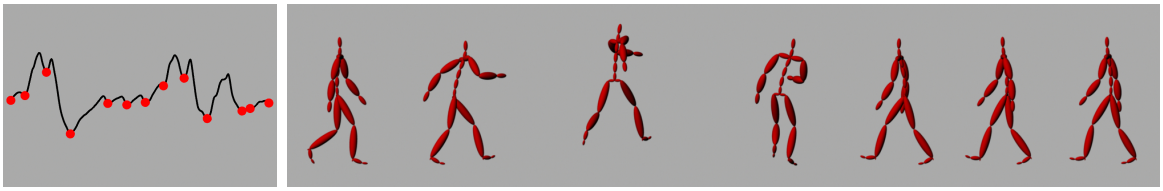
(c)

(d)

(e)

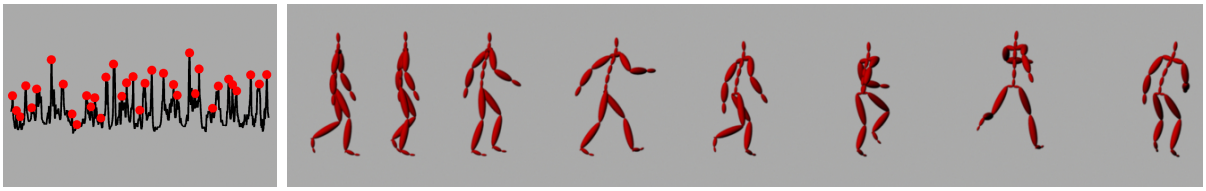
Figure 5.16: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d), and maxima of spatial extremeness (e) for the run motion.

(a)



(b)

(c)



(d)

(e)

Figure 5.17: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d), and maxima of spatial extremeness (e) for the jump motion.

the subsequent walking portion of the motion, at times between extreme stance times.

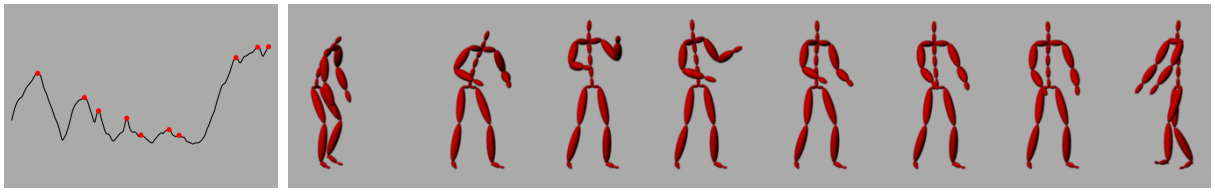
Velocity minima occur at times of wide leg stance, and also at the apex of the jump (Figure 5.17b). Acceleration is significantly sensitive to noise in this example, and a large number of extra pose are chosen, often close together in time (Figure 5.17c). Pseudo–curvature is even more sensitive to noise, and a large number of poses are chosen, many similar to each other (Figure 5.17d).

Spatial extremeness maxima correspond to poses that are very similar to those that correspond to velocity minima (Figure 5.17e). One notable difference is that the spatial extremeness approach selects an anticipatory pose (the third) that occurs just before the jump, which is not chosen as a velocity minimum. This could be used to edit anticipatory motion, for example.

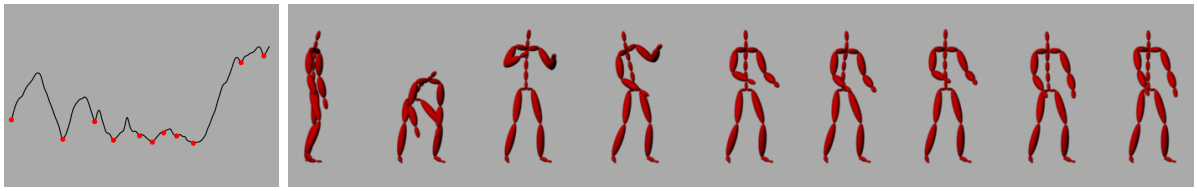
Overall, spatial extremeness maxima and velocity minima both correspond to a number of times that match the observations made of extreme poses, including the apex of the jump, but spatial extremeness also has a maximum at the pose that anticipates the jump. Acceleration maxima often correspond to similar poses, but a number of additional poses also occur at acceleration maxima, due in part to the noise sensitivity of the acceleration measure. The pseudo–curvature approach chooses even more additional poses, as it is very sensitive to noise in this example.

Lift–and–Carry:

A lift–and–carry motion is shown in Figure 5.18, in which a character steps to an object (not shown), lifts it, and walks while carrying it. In this example, no velocity maxima occur while the character is crouched to lift the object (Figure 5.18a). The poses that correspond to velocity minima do include a pose that occurs during the crouch, as well as a number of poses that occur while the character holds the box before beginning to walk (Figure 5.18b). Both acceleration and pseudo–curvature maxima (Figures 5.18c and 5.18d) correspond to a significantly larger number of poses, many occurring at times



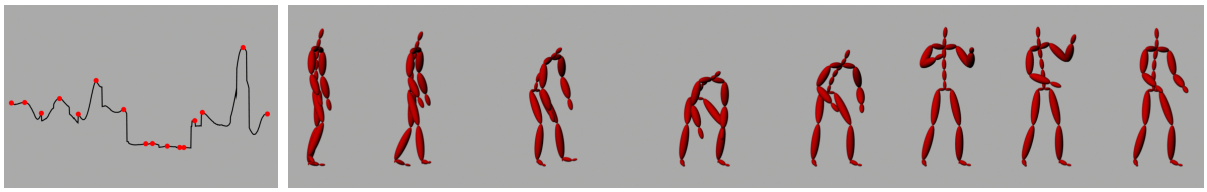
(a)



(b)

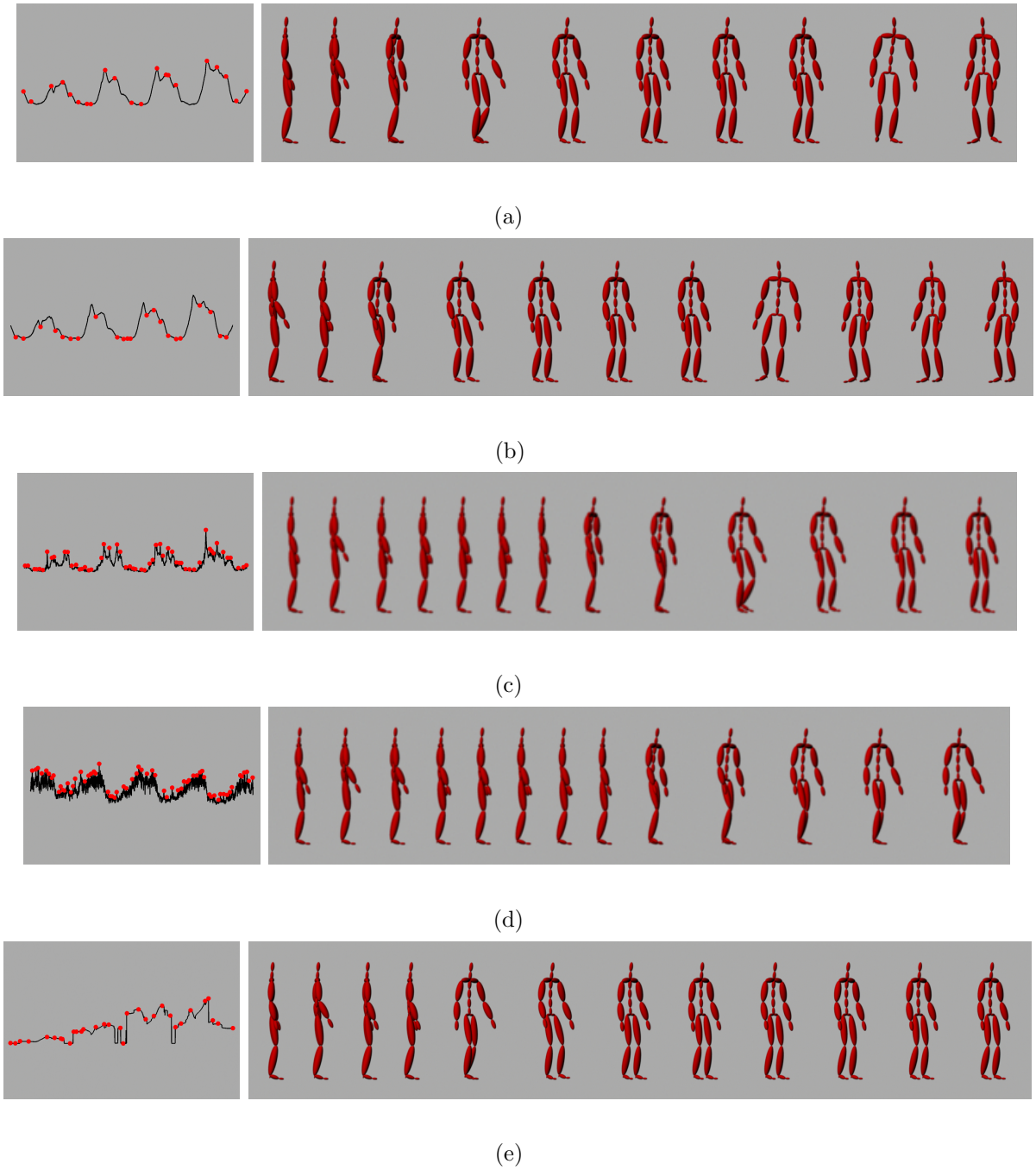
(c)

(d)



(e)

Figure 5.18: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d), and maxima of spatial extremeness (e) for the lift–and–carry motion.



(a)

(b)

(c)

(d)

(e)

Figure 5.19: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d), and maxima of spatial extremeness (e) for the stand–and–turn motion.

of non–extreme body stance. Poses that correspond to spatial extremeness maxima, which are again similar to those that correspond to velocity minima, include the crouching pose, as well as additional poses that occur as the character bends down and stands up (Figure 5.18e).

Stand–and–Turn:

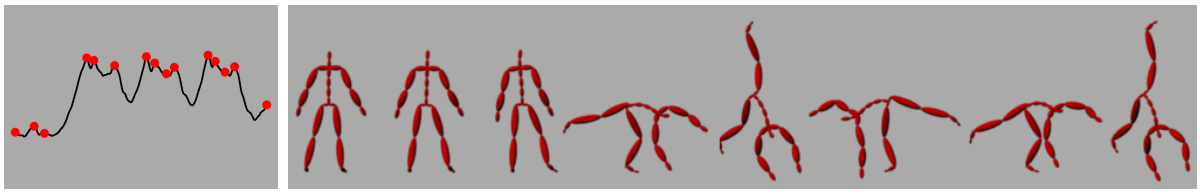
The stand–and–turn motion, shown in Figure 5.19, has a character standing approximately still, making a short change of direction, and repeating. This motion example is problematic for all the approaches presented in this chapter, as there are long regions of time in which the character has little movement. This causes subtle motion to create local maxima in the various measures, leading to the selection of poses that do not appear to be distinctive. In addition, each approach chooses sequences of poses that are approximately similar to each other; for example, each method chooses a number of similar standing poses at the start of the motion.

The spatial extremeness measure (Figure 5.19e) has a number of unusual discontinuities. This is due to the neighborhood bounds changing significantly over time, as subtle motion and noise dominate the stopping criterion in the neighborhood search. Enforcing monotonicity in the neighborhood bounds could lead to a smoother measure.

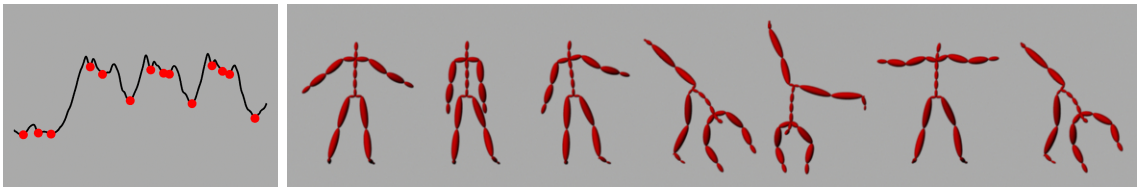
The fundamental challenge with this motion example is that all the approaches aim to identify distinctive local extrema of some measure. To effectively handle motion such as this, in which the character remains still for regions of time, the overall strategy would need to be extended to identify regions of nearly constant stance. A single representative pose could then be used to allow users to edit this temporal region of near–constant stance.

Cartwheels:

The final motion example, shown in Figure 5.20 is a cartwheel motion, in which a

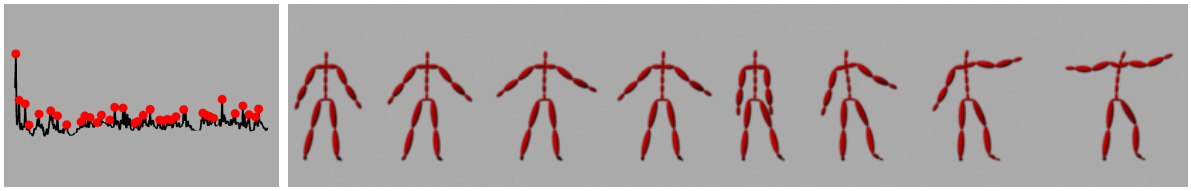


(a)

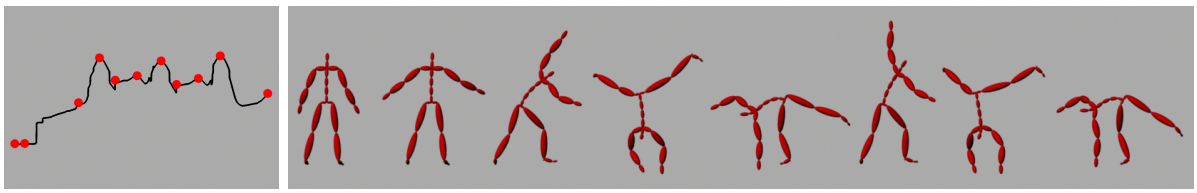


(b)

(c)



(d)



(e)

Figure 5.20: Velocity maxima (a) and minima (b), acceleration maxima (c), pseudo–curvature maxima (d), and maxima of spatial extremeness (e) for the cartwheel motion.

character stands for a moment and then executes a series of cartwheels.

Velocity maxima, at the start of the motion, occur at times when the character is in a neutral stance (Figure 5.20a). As the cartwheels begin, they primarily occur at times when the character is transitioning from one double stance state to the next; i.e., at times when one limb is on the ground. During the standing portion of the motion, velocity minima occur at times of maximal arm swing, and during the cartwheel portion of the motion, they occur at times of two–limb support (Figure 5.20b).

Both acceleration maxima (Figure 5.20c) and pseudo–curvature maxima (Figure 5.20d) lead to a larger number of poses being chosen, especially near the beginning of the motion. Spatial extremeness maxima correspond to the most concise set of poses, and, as with velocity minima, the poses occur at times of double support (Figure 5.20e).

In this example, it is less clear which poses would work "well" for direct editing during the cartwheel portion of the motion. One notable observation is that the cartwheel poses chosen at times of velocity minima and spatial extremeness maxima tend to capture greater variation in the distance between end effectors, while these also are sparse sets.

5.7 Discussion

This chapter has presented a number of techniques for choosing a sparse set of motion extrema from existing motion data. These approaches have been based on two principles for finding extrema of movement that correspond to extreme poses—the use of extrema of differential motion measures and the explicit search for times at which a character or specific joint is in a spatially extreme configuration. These approaches have been applied to the specific problems of finding joint–centric extrema and extreme poses. Approaches to robustly identifying a sparse set of extrema in noisy data have also been presented.

For joint–centric extrema, the techniques were applied to motion from different data sources, as isolating a single limb removes the influence of varying body coordination from

the comparison of the techniques. Three data sources have been considered: recorded human motion data, keyframe animation data, and physically simulated motion data. For full–body motion, only recorded motion has been considered, due to the availability of high–quality data, and a set of motion examples was chosen that varies in terms of task and coordination. For each motion example, each technique for finding a sparse set of motion extrema has been applied, and the resulting joint–centric extrema or extreme poses have been compared.

The remainder of this section summarizes the results of these experiments, provides recommendations for how to use the techniques, and makes suggestions for future work in extending these ideas.

5.7.1 Observations

The follow points summarize the observations made in the prior section, in which various techniques for identifying motion extrema were applied to a variety of motion examples.

- Velocity minima and spatial extremeness maxima both correspond, most often, to joint configurations and poses at times that have qualities similar to the extreme poses created by keyframe animators: direction change, low velocity, high trajectory curvature, and joint configurations that are the most distant to positions at other times. Velocity minima, occasionally, can correspond to times at which a joint or the body is not in such a configuration, and, as a result, using velocity minima might require explicit user refinement of the selection. Spatial extremeness maxima might also correspond to joint configurations or poses that do not match the observations made of extreme poses, but these nearly always occur at predictable times—the first and last frames. Finally, the use of velocity minima requires close–to–natural timing to exist in the motion, while spatial extremeness only considers geometric aspects of movement.

- Acceleration and curvature measures can be significantly sensitive to noise in the data. As a result, extrema of these measures often correspond to joint configurations or poses that do not have the qualities of extreme poses.

- Motion examples that have periods of time during which the character is close to still are problematic for the methods presented here, as these methods are based on finding distinct local extrema. The measures themselves could still be used as part of a more robust technique, but close–to–constant regions would need to be explicitly identified to avoid choosing subtle local extrema that occur when the character is nearly still.

- Curvature measures do not work well for motion in which joint movement has constraints on the rotation degrees of freedom that cause the joint motion to lie in a plane, such as can occur in elbow and knee motion.

5.7.2 Recommendations

Based on the observations made here, the following recommendations are given for finding joint configurations and poses that are similar to extreme poses created by animators:

- In most cases, use spatial extremeness maxima to find extreme joint configurations or extreme poses, as it does not require natural timing in the motion. In addition, this measure has been explicitly designed to find extreme poses.

- Use velocity minima if computational cost is constrained, and natural timing is present in the motion. As it is a strictly local measurement, the computational cost is $O(n)$, where n is the length of the motion. Spatial extremeness has a higher computational cost, due to the need to evaluate distance metrics over local regions of time. The computation cost of the spatial extremeness measure is $O(nm)$, where m is the neighborhood width. In the worst case, this can be $O(n^2)$.

- Acceleration maxima can be used for low–noise data, but the measurement is significantly more sensitive to noise than velocity. This can result in the selection of many extra joint extrema or poses.

- Pseudo–curvature can be used for low–noise data, provided there are no constraints on the rotation degrees of freedom that cause the motion to lie in a plane.

5.7.3 Future Work

The focus of the work presented in this chapter has been on the selection of motion extrema associated with individual joints or the full body. Another approach that would be useful would be the identification of extrema of movement associated with individual degrees of freedom. For example, this could be used to identify meaningful knots for spline approximations of the parameter signals of individual degrees of freedom.

As the approaches presented here focus on identifying local extrema of various measures, they do not handle motion that has regions of time where the character remains still. This was illustrated in the turning motion, in which the character repeatedly stands still for some time before turning to face a new direction. One possibility would be to develop a technique that identifies these near–constant regions of time.

The spatial extremeness measure relies on the motion being bounded in space with respect to some coordinate frame. For joint–centric extrema, this coordinate frame is the position of the parent joint. For the full body, it is the root position. If the motion is not bounded in space, the neighborhood bound searches might not terminate. To extend the spatial extremeness approach to motion with unbounded trajectories, different criteria would be needed to determine meaningful neighborhoods.

In addition, the neighborhoods currently used in the spatial extremeness measure can suddenly change if the search tolerance parameter is set too low or if the motion is very subtle. It would be useful to enforce monotonicity in the bounds to avoid outliers, which

can cause short discontinuities in the measure.

The comparison presented in this chapter has focused on comparing the techniques, given the motion data and a consistent approach for finding local extrema of some measure. The time window used to find semi–local extrema has been held constant, with the goal of understanding how the different measures act on different data sources as well as showing how sensitive they are to noise. This results in a comparison of the different measures that is independent of how the data is processed or how the resulting measure is processed. In real applications, data filtering and extrema identification parameters could be adjusted separately for each measure, as well as for different types of data to enable any chosen measure to perform as best possible for the chosen type of data.

Other measures could be used as well. One possibility is rotation–based measures, but nonlinearities and parameterization ambiguities would need to be handled effectively. Task–specific measures could also be developed, such as the distance between two end effectors or the distance of an end effector to important locations in the environment.

While the techniques presented in this chapter were in development, additional techniques were partially investigated, but not taken to completion. One such technique measured the incremental rotations that a limb undergoes for each time step from one frame to the next. The angles of these incremental rotations were then considered as a measure. This measure is similar to curvature, as it is correlated to angular acceleration, and it tends to choose motion extrema that are similar to those chosen using the pseudo–curvature measure. This similarity to curvature, along with its conceptual complexity, motivated the investigation of simpler differential measures, which have been presented in this chapter.

Finally, it would be interesting to combine the measures into a predictive model. The staggered poses system (Chapter 7) takes this approach to determine the central time associated with each staggered pose when finding them in recorded motion data. This idea could also be applied specifically to finding extrema. To determine how to combine

these measures, it would be useful to create a model of what the "correct" poses would be. This can be difficult, however, as it can be application–specific or example–specific. For example, editing the knee movement in the run example would require different pose times than editing the overall stride.

Chapter 6

Ground–Aware Pose–Centric Motion Editing

This chapter presents *ground–aware pose–centric motion editing*, the first of three motion editing applications that use motion extremes as a foundation for applying expressive motion edits. The system presented in this chapter allows users to apply expressive edits to a sparse set of extreme poses, and the system applies these edits to the remainder of the motion in such a way that ground contacts are preserved and ground–relative motion remains plausible. Unlike existing approachs to applying pose–centric edits in the presence of contact constraints, this system ensures that pose edits that remain plausible with respect to the ground will be exactly met in the resulting motion. If the pose edits provided by the user are not plausible with respect to ground contact, the system accounts for this error by matching poses that are similar to the specified edits, but retain ground contact.

To accomplish this, I model the pose–centric editing problem as a set of prioritized constraints—a set of constraints in which any given constraint should be met as best possible, given that all higher–priority constraints are also met as best possible. These constraints include ground contact preservation, the matching of the user's pose edits, and

additional constraints that assist in retaining the plausibility of the character's motion with respect to the ground.

Rather than solving the resulting prioritized optimization problem as a constrained set of nonlinear equations, I introduce a greedy, closed–form solution that applies a sequence of changes to the motion, each of which makes certain guarantees about a subset of the constraints. This closed–form solution enables the robust *interactive* editing of extreme poses without requiring users to explicitly account for the relationship of the character's motion to the ground.

6.1 Motivation and Overview

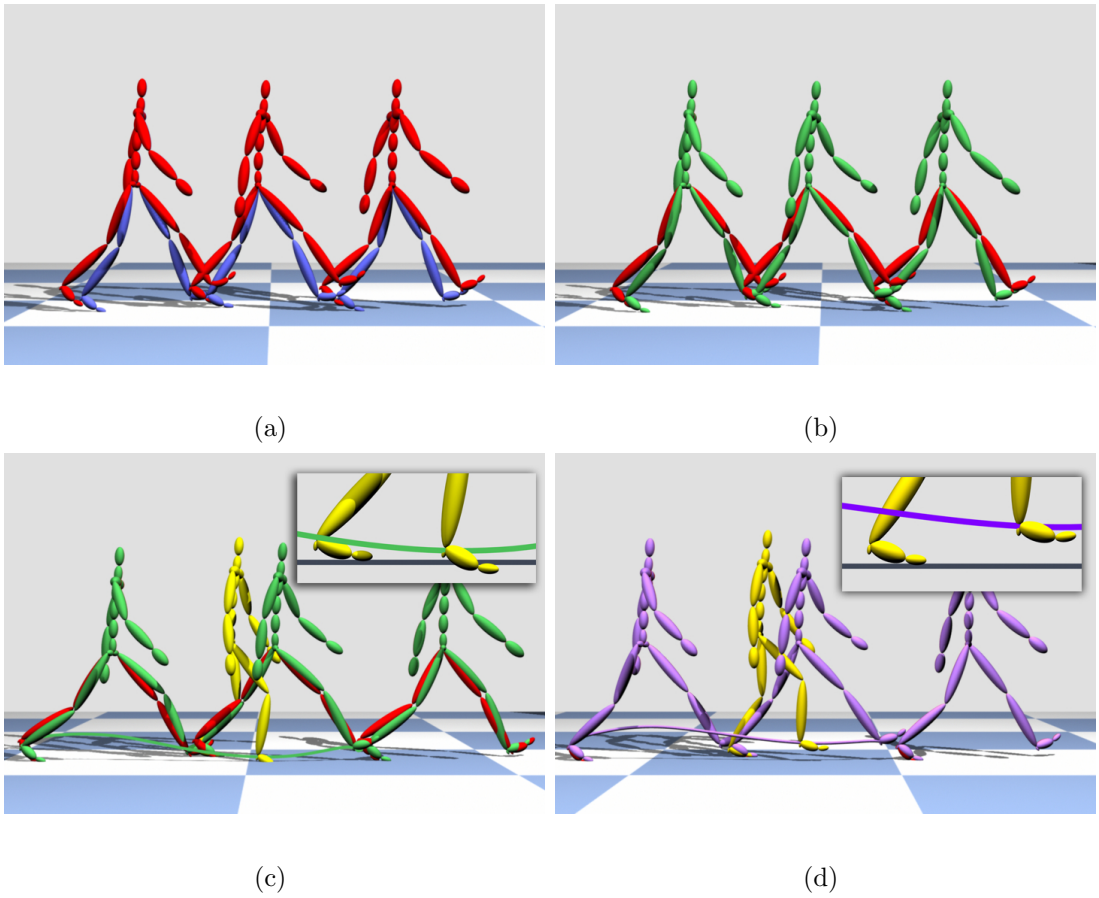
Animators often think in terms of specific poses, especially when creating movement. Detailed control over pose is an important part of both character design and movement creation. For example, poses are used to specify characteristic stances that show personality when creating model sheets [167], and poses are often used as a first pass when animators create new motion using keyframe workflows. Pose specification is a critical tool in creating *expressive* characters, as animators use it to convey many of the subtle, subjective aspects of movement such as personality, intent, and emotional state. Animators have considerable tools and expertise for working with pose: skilled animators understand how to use pose and master an arsenal of tools for manipulating poses, most often for keyframe animation. However, while pose control is the central part of how animation is created manually, effective pose control has not been developed for editing existing motion. The integration of pose–control into motion editing would allow the considerable talent and tools available for pose manipulation to be used for the oft–needed task of editing existing motion.

This chapter presents a new approach to motion editing that combines the expressive power and animator–friendliness of pose editing into a convenient system for adjusting

existing motion, particularly motion created using motion capture. This *pose–centric* editing system allows animators to specify desired poses, which I refer to as *target poses*, for particular frames of the animation, and the system automatically adapts the motion to meet these pose goals. The key challenge is to adapt the motion in a way that preserves the important interactions of the character with the environment, particularly in terms of ground contacts. Unlike existing approaches, this system can precisely meet target poses and preserve ground interactions, as it is given the flexibility to alter the global motion of the character if necessary. The techniques support a workflow that allows animators to use their existing tools and talent to specify poses for a sparse set of frames, and to have the motion altered accordingly. Such a workflow has been previously infeasible: existing approaches either do not provide detailed pose control or require tedious, per–frame adjustments to maintain a character's interaction with the world.

When editing existing motion, in contrast to creating new motion from scratch, a sparse set of poses must be chosen for editing from the much larger set of poses in the full motion. While this, in principle, could be done manually, this can be tedious, especially for long motion clips or motion with repetitive patterns, such as walking. In addition, it can be error–prone; if the existing motion extrema are not chosen well, users might inadvertently introduce a high frequency jerk to the edited motion. To assist users with selecting poses that are meaningful for expressive motion editing, the pose–centric editing system includes a tool for extreme pose selection. Any of the algorithms presented in Chapter 5 can be used for pose selection; the pose–centric editing system currently supports the algorithmic selection of extreme poses using either velocity minima or spatially extreme configurations.

Another challenge in creating an effective pose–centric editing approach stems from the need to balance between the often conflicting goals of matching the specified pose targets, preserving ground interactions, and remaining similar to the original motion. This can be seen in the simple example of Figure 6.1. In this example, a user edits poses



(a) (b)

(c) (d)

Figure 6.1: A user edits poses from the original motion, shown in blue, to widen the stance, shown in red (a). Applying changes with displacement maps and footskate correction, shown in green, can fail to increase the stride length (b). Target poses can be moved to reduce the effect, but this can cause ground penetration on swinging limbs, as shown in yellow (c). Our approach, shown in purple, matches target poses while preserving ground contacts and avoiding ground penetration between contacts (d).

of an existing walk (blue, Figure 6.1a) to create a more energetic variation by widening the stance (red, Figure 6.1a). Previous approaches are ineffective, as they correct contact error *after* applying the edit. For example, while pose targets can easily be applied using motion displacement maps to interpolate the specified changes [20, 179] and footskate correction to repair resulting ground contact error [75], these algorithms can interact to result in motion (green, Figure 6.1b) that does not match the desired edit. This results from the edit introducing a large amount of contact error that post–process correction approaches, such as footskate correction, remove by returning feet to average contact locations. Manual positioning of the poses can help avoid this effect (Figure 6.1c), but it is tedious and can lead to unnatural warps of end effector trajectories, which can result in foot motion that passes through the ground between contacts (yellow, Figure 6.1c). In contrast, pose–centric editing (Figure 6.1d) meets specified pose edits precisely and preserves the relationships between the end–effectors and the ground by adjusting the overall motion such that the character's movement fits with the larger steps.

The type of edit illustrated in this example is not well supported with current tools. In conversations with skilled animators, including four who were specifically asked to recreate the edit in Figure 6.1 using existing tools, it was found that most would prefer to create a new motion by keyframing, merely using the source motion a reference, rather than using available motion editing solutions. This suggests a real need for an effective pose–centric editing workflow: animators are familiar with these kinds of tasks, find existing tools overly cumbersome—even more so than the tedium of recreating a reference motion—and strongly prefer to work with their familiar keyframe editing tools.

To enable a pose–centric editing workflow that allows users to specify expressive edits to a sparse set of extreme poses, the approach presented here addresses the key technical challenge: adjusting existing motion in a manner that balances meeting target poses, preserving ground relationships, and minimizing overall adjustments. This problem is modeled as a set of prioritized constraints, in which the system modifies the motion to

match desired pose edits and preserves vertical motion of end effectors between ground contacts, subject to the constraint that contacts are always preserved, although they can be moved. Contact preservation is explicitly prioritized over matching target poses; this avoids requiring users to create poses that exactly conform to ground contact requirements, as this can be tedious using some pose–editing workflows. As a result, the system matches poses that are adjusted as necessary to allow for contact preservation. When the target poses already have plausible contact relationships with the ground, the pose–centric editing system exactly matches the limb configurations of the target poses (Figure 6.1d).

The pose–centric editing system solves the resulting constrained optimization problem using a new greedy algorithm that determines the resulting motion by making a sequence of changes, each of which makes certain guarantees about the given constraints. By taking this approach instead of solving a collection of nonlinear equations, two key benefits are gained. First, as each step applies a change to the motion using a closed–form solution, computational cost scales well with respect to the complexity of the character and the length of the motion. Second, as each step addresses specific constraints, the system can guarantee that lower priority goals are met whenever possible, which can be difficult to do using a single system of equations.

Overview: The key contribution presented in this chapter is the realization of an effective approach to pose–centric motion editing that allows for the leveraging of animators' talent and skills with existing tools for manipulating poses. The main technical contribution is the first algorithm for applying sparse pose–based edits to existing motion data while preserving ground contact and important ground relative motion. The notion of contact preservation for arbitrary foot structures is formally modeled as a rigid transformation, for each sustained contact, of the original foot motion during contact (Section 6.4). I characterize the problem of applying desired pose changes while preserving the existence of ground contact and important ground–relative motion as a

prioritized constrained optimization problem (Section 6.5). To solve this optimization problem, I provide an efficient, robust, and scalable algorithm that applies a sequence of changes using closed–form solutions (Section 6.6). I demonstrate the editing flexibility the approach enables in Section 6.7 by showing that significant changes to extreme poses can be made to expressively alter motion while the system automatically manages the interdependency between pose changes and ground–relative motion.

6.2 Related Techniques

While Chapter 2 surveys alternative approaches to motion editing, this section focuses on how these existing approaches relate to the pose–centric editing problem addressed in this chapter. In addition, I describe existing approaches to detecting and preserving ground contact constraints and describe how existing constraint detection techniques have been integrated into the pose–centric editing system. While the pose–centric editing system uses a greedy prioritized optimization approach to apply the edits, prioritized optimization has been applied to other animation problems; these applications will be discussed as well.

Existing methods for motion editing make pose–centric editing possible, but fail to achieve the requirements for an effective approach. Motion displacement maps allow users to directly specify desired geometric changes at particular times, and they adapt motion to interpolate these pose targets [20, 179]. However, these methods do not provide mechanisms for preserving ground contact or important ground–relative motion. These contact issues can be addressed by using a variety of footskate cleanup techniques [75, 92]. However, this post–processing correction can undo the desired changes made to the target poses to correct the contact constraint error. The pose–centric editing system integrates these steps to balance between meeting target poses and preserving ground contacts.

Existing commercial animation systems integrate these displacement–map style tools

with sophisticated pose editing. Systems specifically designed for motion editing, such as Motion Builder, provide inverse kinematic solvers that can be applied per–frame to address foot–sliding issues. However, none, to the author's knowledge, integrate pose–centric editing with simultaneous contact preservation. Therefore, these existing systems are unable to accurately meet target poses and do not adjust root trajectories to meet target poses while preserving contacts in a natural fashion. Conversations with skilled animators conversant with the best current systems suggest that they prefer to re–animate new motion using keyframe animation tools, rather than edit existing motion, to make these kinds of pose–centric changes.

Constraint–based approaches to motion editing, surveyed in Chapter 2, explicitly consider contact relationships when adjusting motion. Such problems can be solved kinematically, using techniques such as optimization [41] or displacement maps [92], and these approaches can even consider physical constraints [139]. These approaches can, in principle, be applied to pose–centric editing, as it is possible to consider target poses as constraints. The use of an existing, general–purpose, solver to reposition contact locations as been considered in constrained optimization approaches [42]. However, using such an approach to adjust motion to meet the sparse constraints of pose–centric editing would require an expensive and unreliable global optimization, rather than a more practical local approach [44]. Even then, these general solvers do not handle priorities well, although recent work in physical controller optimization has developed algorithms for handling prioritized quadratic constraints [36]. Instead of using a general–purpose motion editing solver, this chapter introduces a novel solver that solves the prioritized constraints in an efficient, robust, and scalable way, which is required for interactive pose–centric editing with automatic ground contact preservation.

Generalized contact preservation has also been addressed using a variety of approaches. These include inverse kinematics [15], displacement maps [92], and spacetime optimization [178, 41]. For interacting characters in particular, Liu et al. [106] apply continuation

strategies to solve difficult optimization problems. As constraint–centric approaches, those that allow pose constraints do not attempt to match pose–centric edits whenever it is plausible to do so.

Contact identification is well–studied. Most approaches label individual frames and combine contiguous sequences into sustained, or variational, contacts. A commonly–used approach applies a threshold to either the velocity of the potentially contacting joints or their distance to the ground [15]. More sophisticated techniques include estimation based on noise patterns [88] and classification based on relative joint locations [58]. These frame–labeling approaches do not, in general, label all frames correctly; mislabeled frames can be corrected by a user or algorithmically. For example, Le Callennec and Boulic [88] merge sequences of labeled frames in which small gaps occur. In the pose–centric editing system, sustained contacts are identified by applying Le Callennec and Boulic's gap–filling technique to frames labeled using velocity thresholds.

Prioritized optimization has been used for other animation–related problems such as pose specification and controller design. Baerlocher and Boulic [12] and Pienado et al. [136] develop prioritized inverse kinematic solvers for setting character pose. Khatib and Sentis use task prioritization to control the pose of robots while keeping them physically stable [67, 150]. de Lasa et al. use prioritized optimization to design physically–based character controllers that include constraints on motion features such as the center of mass and end effectors [36]. Unlike these approaches, the pose–centric editing system uses a prioritized solver to modify an entire motion. Le Callennec and Boulic also consider prioritized constraints when editing motion [87]. However, their solver meets constraints on a per-frame level as part of an inverse kinematics solver and uses filtering to ensure motion smoothness. As such, it cannot incorporate constraints that affect the motion at different times, such as the interdependency between target poses and sustained ground contact constraints.

6.3 Approach

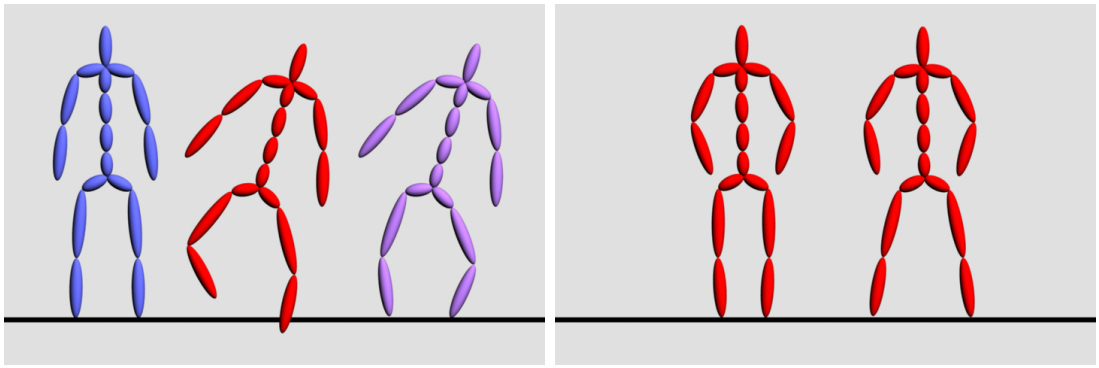
The goal of the pose–centric editing system is to enable efficient expressive motion editing using a sparse set of meaningful poses, such as the extreme poses chosen using the techniques described in Chapter 5. To ensure that contact is plausibly preserved, the pose–centric editing system applies an edit that results in the character passing through the target poses while preserving ground contact and plausible end effector motion. The simultaneous goals of matching desired poses and preserving contact can be contradictory, however. To address this problem, the pose–centric editing system models the problem as a set of prioritized constraints, in which contact must be preserved, and target poses are matched subject to this contact preservation constraint.

To match target poses, the pose–centric editing system modifies the entire motion, paying particular attention to the root trajectory, the locations and orientations of ground contacts, and the motion of the limbs that make contact. Users can adjust the root orientation and any joint parameters; the root trajectory is determined by the editing system to match target poses while preserving contact constraints. As foot motion is explicitly preserved during contact, the system prevents users from breaking existing foot contact motion such as foot roll. During flight phases, such as jumps, users can edit the vertical root translation, as it is not constrained by contact. The system, as a design choice, does not allow users to change ground–parallel root translation during flight phases, as it can be difficult to specify such edits in a way that preserves plausible trajectories.

To allow motion to be reused in a new environment or in a different location of a given environment, the pose–centric editing system modifies contact height and end effector trajectories to avoid any interpenetration with ground geometry. For example, this allows users to add obstacles to the environment that the character can step over.

As the pose–centric system is focused on applying geometric changes specified using poses, it does not alter the overall timing of the motion. However, the approach could

be used in conjunction with a variety of retiming algorithms.



(a) (b)

Figure 6.2: Target poses and contact constraints can be contradictory. A single target pose can contain height inconsistency (red, a); the pose–centric editing system changes the motion to match a similar pose with no contact error (purple, a). A pair of target poses can create sliding motion that breaks contact (b); the pose–centric editing system will again edit the motion to avoid the application of this sliding error.

6.3.1 Goals for Applying Pose Changes

The pose–centric editing system prioritizes ground contact over exactly matching target poses for two reasons. First, motion appears fundamentally incorrect if an edit does not preserve ground contact. Second, it can be difficult for users to specify poses that are consistent with contacts. For example, Figure 6.2a shows a pose (left) that does not have consistent foot height after editing (middle). Inverse kinematic (IK) tools can be used to correct this type of error, but IK rigs can make certain types of pose edits inconvenient for users. In addition, IK corrections would not be reliable if the environment is not uniform in height, as the solver is free to change contact locations, which, in turn, would could make the IK solution incorrect. If this inconsistency exists in a target pose, the pose–centric editing system produces motion that will be close to the target pose, but with correct contact (Figure 6.2a, right).

A second type of inconsistency can exist among two or more target poses, specified at different times, that share contact constraints. For example, if two target poses specified for a standing motion have different geometric spatial relationships between the feet (Figure 6.2b), matching both target poses when applying the edit will create footskate. Again, prioritizing contact preservation over pose edits avoids this.

The pose–centric editing system incorporates additional goals as lower priority constraints. To avoid ground interpenetration between contacts, the third priority of the solver is the preservation of the vertical motion of end effectors. The system does allow pose edits to override this goal, as they are higher priority. In addition, the solver applies edits with smooth, approximately minimal changes, a common goal when applying kinematic motion edits [41].

Finally, it can be useful to preserve the orientation of the root velocity, parallel to the ground, with respect to the orientation of the character. That is, the system aims to keep velocity, in the local coordinate frame of the root, similar to that in the original motion. Velocity patterns on the root can otherwise appear unnaturally jerky if the user rotates the root significantly when specifying target poses, as root motion and direction of overall movement are often correlated.

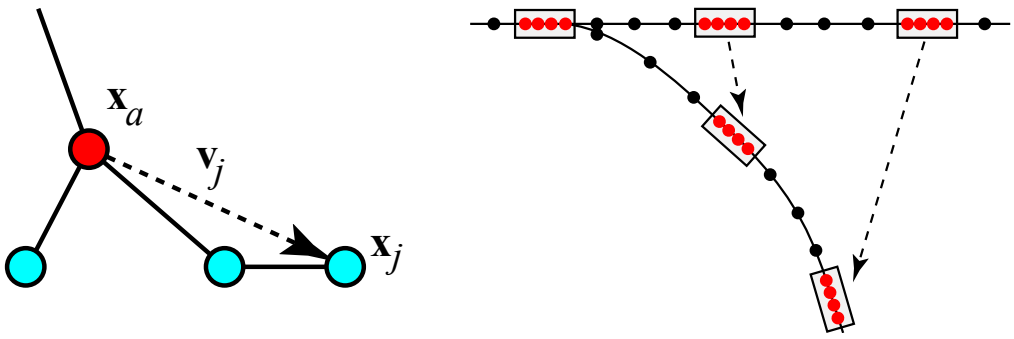
6.3.2 Assumptions and Representations

The pose–centric editing system applies edits to the root and limbs that make contact with the ground. Each limb that makes contact is assumed to have a *hip* joint, an intermediate *knee* joint, and an *ankle* joint as an end effector. Beneath each ankle joint, there can be an arbitrary hierarchy of foot joints, any number of which can make ground contact (Figure 6.3). While I use leg terminology in the remainder of this chapter, the system supports any number of limbs that have this topological structure. For example, wrists can act as end effectors.

The pose–centric editing system applies changes to motion represented using the

Articulated Skeletal representation (Chapter 4). The local joint translations $\mathbf{t}_j(t)$ can vary subtly in magnitude for the leg limbs, as rigidity and smoothness of motion can be conflicting at times [75]. Recall that the Cartesian coordinates of a joint are denoted as $\mathbf{x}_j(t)$. The world up axis is assumed to lie along the positive z axis of the world coordinate frame.

When applying displacement maps, gaps between knots are filled using cubic Bézier splines with uniform knot spacing and repeated endpoint values. For rotations, the system uses recursive spherical linear interpolation [156]. This simple approach approximates smooth motion sufficiently well for purposes of this system, although a rotation spline form with stronger continuity guarantees could also be used if needed.



(a) (b)

Figure 6.3: Contact constraints on joint position can exist for any foot joint, shown in blue (a). All joint position constraints that overlap in time are merged into a single sustained contact constraint associated with the ankle, shown in red (a). When applying a motion edit, each sustained contact constraint, illustrated as a box (b, top–down view), is considered to be preserved if the foot motion during the contact undergoes a rigid transform parallel to the ground plane, along with a change in height.

6.4 A Model of Contact Preservation

The pose–centric editing system preserves sustained ground contacts associated with the foot as a whole. Each contact is associated with a particular ankle a (red, in Figure 6.3a) and has an associated time duration $[t_s, t_e]$. To identify contacts, the system uses velocity thresholds [15] to label frames for foot joints that make contact (blue, in Figure 6.3a). Sustained contacts on each of these foot joints are then determined using gap filling [88], and the system merges each set of temporally overlapping joint contacts into a single sustained contact associated with the ankle.

Contact preservation is modeled as a constraint, for each sustained contact, that the change to the motion of the foot consist of a 2D rigid transformation parallel to the ground plane and a vertical translation to account for changes in ground height. This is illustrated using a top-down view of a single ankle trajectory in Figure 6.3b; red samples indicate contact frames. Each contiguous sequence of contact frames undergoes the same rotation and translation, and this change smoothly varies for samples between contacts, shown in black. For individual foot joints, the rotation is about the ankle. This constraint can be formally stated as

$$\mathbf{x}_a(t) - \bar{\mathbf{x}}_a - R_{\theta_a}\left(\mathbf{x}_a^o(t) - \bar{\mathbf{x}}_a^o\right) = 0 \tag{6.1}$$

$$z_a(t) - z_a^o(t) - \bar{g}(t) + \bar{g}^o(t) = 0 \tag{6.2}$$

$$\mathbf{v}_j(t) - R_{\theta_a}\left(\mathbf{v}_j^o(t)\right) = 0 \tag{6.3}$$

for each $t \in [t_s, t_e]$ and each foot joint j. R_{θ_a} is a 2D rotation by θ_a about the z axis, positions $\mathbf{x}$ are specified as 2D cartesian coordinates in Equation 6.1 only, z_a is the ankle height, and $\mathbf{v}_j = \mathbf{x}_j - \mathbf{x}_a$ (see Figure 6.3a). $\bar{\mathbf{x}}_a^o$ and $\bar{\mathbf{x}}_a$ are the average ankle positions over the duration of the contact in the original motion and edited motion, respectively. Similarly, $\bar{g}_a^o$ and $\bar{g}_a$ represent the average height of the ground underneath the ankle, over the duration of the contact, in the original motion and edited motion.

To preserve a contact using this model, $\bar{\mathbf{x}}_a$ and θ_a are available as free parameters. This is a stronger condition than is strictly necessary to preserve contact for individual joints, but it allows the system to handle arbitrary foot structures with contact motion such as foot roll or toe flex without requiring users to maintain this motion when editing. This also assumes the foot motion during contact is correct in the original motion.

6.5 Problem Formulation

The goals of the pose–centric editing system are prioritized and potentially contradictory, and they can not in general all be achieved for a desired edit. The overall goal of the system is to match target poses as best possible, subject to the constraints that ground contacts be preserved and no ground penetration of end effectors occurs. Table 6.1 lists the specific goals and their associated priorities; these will correspond to steps of the solver that will be presented in the next section. The remainder of this section formally describes the remaining goals as either hard or soft constraints; Section 6.6 describes the greedy algorithm used to meet these prioritized goals.

Priority	Goal	Equations
1	Preserve Contacts	6.1–6.3
2	Avoid Ground Penetration	6.4
3	Match Target Poses	6.5
4	Preserve End Effector Motion	6.6
5	Preserve Root Velocity	6.7

Table 6.1: The goals, priorities, and associated constraint equations of the pose–centric editing system.

As contact preservation is highest priority, it is modeled as a hard constraint using Equations 6.1– 6.3. The edited motion should strictly avoid having any ground interpen-

etration. This is modeled using the inequality constraint

$$(z_a(t) - g(t)) - (z_a^o(t) + g^o(t)) \geq 0 \tag{6.4}$$

for the entire motion. This requires the vertical distance of the ankle above the ground to be *at least* as high as it was in the original motion. By allowing it to increase, the system can retain smooth motion when introducing obstacles that the character must step over.

The remaining goals are formulated as soft constraints, that is, as penalty terms of an objective function. If target poses are matched by an edit, then $\mathbf{M}(t_i) = \mathbf{P}_i$ for each pair $(t_i, \mathbf{P}_i)$, where $\mathbf{P}_i$ is the set of skeletal parameter values for a target pose at time t_i. The resulting objective function term is

$$f_P(\mathbf{M}) = \sum_i ||\mathbf{M}(t_i) \ominus \mathbf{P}_i||^2, \tag{6.5}$$

where $||\mathbf{M}(t_i) \ominus \mathbf{P}_i||$ is a pose distance metric does that not include root translation, with the exception of vertical root height during flight phases.

To keep the vertical component of the end effector trajectory in the edited motion similar to that in the original motion, this soft constraint is used:

$$f_A(\mathbf{M}) = \sum_a \int_t |z_a(t) - z_a^o(t)|^2 dt. \tag{6.6}$$

Finally, for root velocity parallel to the ground plane to remain oriented with the character, it is desired that $\dot{\mathbf{x}}_r(t) = R_{\alpha(t)}\left(\dot{\mathbf{x}}_r^o(t)\right)$, where $\dot{\mathbf{x}}_r(t)$ is the 2D root velocity parallel to the ground plane and $R_{\alpha(t)}$ is the 2D rotation about the z axis closest to the 3D rotation applied by the edit. With the root as the pivot, the corresponding term is

$$f_R(\mathbf{M}) = \int_t |\dot{\mathbf{x}}_r(t) - R_{\alpha(t)}\left(\dot{\mathbf{x}}_r^o(t)\right)|^2 dt. \tag{6.7}$$

One approach to applying a pose–centric edit would be to use a general–purpose constrained optimization solver. In this case, the goal would be to solve for motion $\mathbf{M}$ and each contact constraint transformation $(\theta_a, \bar{\mathbf{x}}_a)$ such that $f(\mathbf{M}) = w_P f_P(\mathbf{M}) +$

$w_A f_A(\mathbf{M}) + w_R f_R(\mathbf{M})$ is minimized and the constraints given in Equations 6.1–6.4 are met. This would be done by setting up a system of equations and adding terms that penalize large changes and bias the solution toward smoothness. This can be difficult to solve efficiently, however, as the system is nonlinear and has nonlinear constraints. In addition, it can be difficult to prioritize Equation 6.5 over Equations 6.6 and 6.7 by adjusting objective function coefficients, which is necessary to allow pose edits to override both root velocity reorientation and the preservation of vertical end effector motion between contacts.

Another approach would be to use a general–purpose solver to solve a series of problems, each of which addresses a specific term of the objective function. For example, one could first minimize $f(\mathbf{M}) = f_P(\mathbf{M})$, subject to Equations 6.1–6.4 to achieve a solution where $f_P(\mathbf{M}) = c_P$. The constant c_P represents the minimal value of $f_P(\mathbf{M})$. Second, one would need to minimize $f(\mathbf{M}) = f_A(\mathbf{M})$, subject to Equations 6.1–6.4, as well as an additional constraint that $f_P(\mathbf{M}) = c_P$, achieving a solution where $f_A(\mathbf{M}) = c_A$. Finally, one would minimize $f(\mathbf{M}) = f_R(\mathbf{M})$, subject to Equations 6.1–6.4 as well as $f_P(\mathbf{M}) = c_P$ and $f_A(\mathbf{M}) = c_A$. See de Lasa et al. for an example of such an approach, which has been applied to the optimization of physical controllers [36]. This approach remains difficult to solve for the pose–centric editing problem, however, as the the constraints and objective function terms remain nonlinear. Efficiency also remains a key concern, as the goal is to create an interactive system, and, for pose–centric editing, each step must solve for the entire motion.

The pose–centric editing system instead uses a greedy algorithm that solves a sequence problems to apply a pose–centric edit. Each step of the algorithm addresses a subset of Equations 6.1–6.7. For efficiency, each step only modifies a subset of the motion parameters, and each step is applied using a closed–form solution.

6.6 A Closed–Form Solver for Pose–Centric Editing

To efficiently apply pose–centric edits while considering constraint priorities, this section presents a greedy algorithm that solves the problem as a sequence of steps. Each step computes a partial solution[1], as illustrated in Figure 6.4. The solver first matches the target poses specified by the user, with the exception of the feet, resulting in the partial solution $\mathbf{M}^p$. It then solves for new motion for the root of the character that allows contact constraints to be *approximately* met, while keeping the root velocity oriented with the character and the root motion smooth; the result is $\mathbf{M}^r$. Third, it solves for motion of the feet alone ($\mathbf{M}^f$) that combines the foot motion in $\mathbf{M}^o$ and $\mathbf{M}^r$ to *exactly* meet contact constraints while resulting in vertical end effector motion that is similar to the original. Finally, the solver resolves the difference between $\mathbf{M}^r$ and $\mathbf{M}^f$ using an analytic IK solver that allows subtle leg stretching [75].

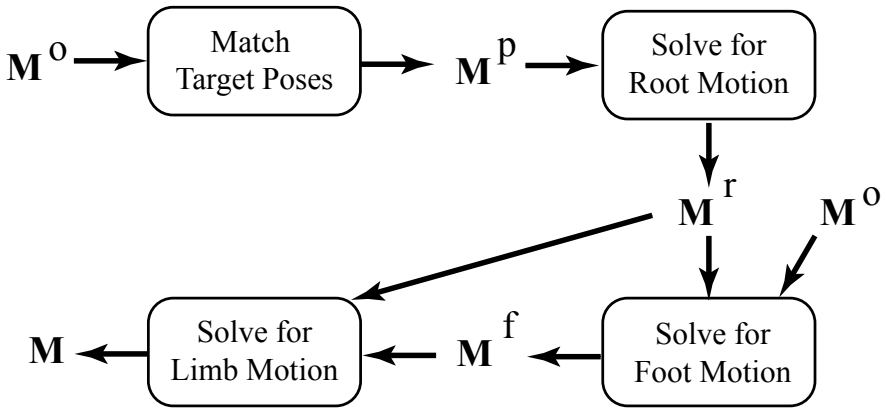


Figure 6.4: To efficiently apply pose changes while considering constraint priorities, the pose–centric editing system uses a multi–step greedy algorithm that creates a sequence of partial solutions.

Table 6.2 summarizes how each step of the algorithm enforces the constraints. A key step is the solution for $\mathbf{M}^r$. This step assumes that the foot motion remains vertically oriented with the character, but flat with respect to the ground. This will be

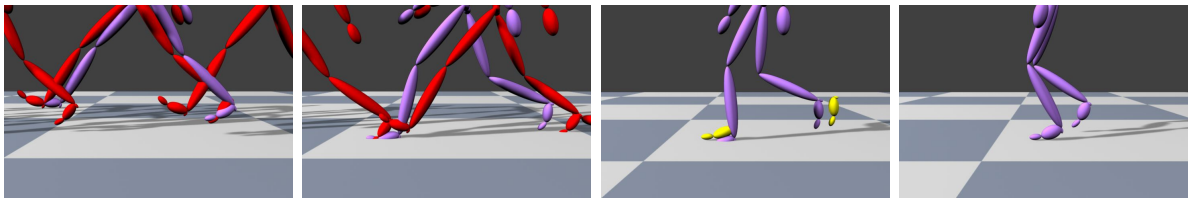
[1]Superscripts on the motion or on parameters indicate the corresponding partial solution.

Goal:	1: Preserve Contacts (Eq. 6.1–6.3)	2: Avoid Ground Penetration (Eq. 6.4)	3: Match Target Poses (Eq. 6.5)	4: Preserve End Effector Motion (Eq. 6.6)	5: Preserve Root Velocity (Eq. 6.7)
$\mathbf{M}^p$	N/A	N/A	Applies displacement map to match poses (exact), except joints below ankle.	N/A	N/A
$\mathbf{M}^r$	Computes a smooth root trajectory that preserves contacts (approximate).	Root trajectory approximately avoids ground penetration (approximate).	No change.	N/A	Initialized to exactly preserve velocity; then modified to approximately preserve contacts.
$\mathbf{M}^f$	Compute rigid transforms for each sustained contact (exact).	Explicitly avoid ground penetration (exact).	No change.	Displacement map, best possible smooth approximation that preserves contacts and matches poses.	No change.
$\mathbf{M}$	No change.	No change.	Pose matching error introduced as needed to preserve contacts.	No change.	No change.

Table 6.2: Each step of the solver creates an intermediate solution ($\mathbf{M}^p$, $\mathbf{M}^r$, $\mathbf{M}^f$, and $\mathbf{M}$, the final solution). Each step of the solver applies changes to approximately or exactly meet the given constraints; each row of the table briefly describes how each step achieves these goals.

enforced when solving for $\mathbf{M}^p$. It also only approximately meets the constraints given in Equations 6.1–6.4, but key to achieving the overall goals of the solver is that this step approximately meets these constraints in a way that allows the solution for $\mathbf{M}^f$ to exactly meet them without changing how well the target poses are met. Also, the solution for $\mathbf{M}^r$ uses an initial solution that exactly meets Equation 6.7, but then modifies it to allow for Equations 6.1–6.4 to be approximately met, as they have the highest priority. Finally, note that the final step might reduce how well Equations 6.5 and 6.6 are met to account for the fact that $\mathbf{M}^f$ exactly meets Equations 6.1–6.4.

Figure 6.5 illustrates the effect of each step of the solver by showing the intermediate solutions. After matching target poses, $\mathbf{M}^t$ introduces a large amount of contact error (Figure 6.5a). Solving for the root motion in $\mathbf{M}^r$ reduces this contact error significantly, but not exactly, to keep the overall motion smooth (Figure 6.5b). Figure 6.5c shows the detached foot motion that does meet contact constraints, $\mathbf{M}^f$, in sync with $\mathbf{M}^r$. Finally the limb motion is modified to resolve the difference between the two (Figure 6.5d).



(a) (b) (c) (d)

Figure 6.5: The main steps of the solver used by the pose–centric editing system correspond to a sequence of solutions. $\mathbf{M}^p$, shown in purple, matches the target poses, shown in red (a). $\mathbf{M}^r$ has a new root trajectory and contact locations that cause the motion to *approximately* meet contact constraints (b). $\mathbf{M}^f$, shown in yellow, includes detached foot motion that exactly meets contact constraints (c). The final motion $\mathbf{M}$ combines the body motion of $\mathbf{M}^r$ with the foot motion of $\mathbf{M}^f$ (d).

The remainder of this section presents the details for each step of the algorithm. For each step, the algorithm is first given. The algorithm for solving each step is then

related to the constraint equations, to describe how it meets the goals, as summarized in Table 6.2.

6.6.1 Matching Target Poses

The first step of the solver alters the motion to match target poses, with the exception of root translation and foot parameters below the ankle. These parameters will be addressed in later steps. To smoothly apply changes, the solver uses motion displacement maps. The solver does not apply changes to skeletal parameters of joints below the ankles during this step, and the overall change in ankle orientation is constrained to be a rotation about the world up (z) axis. This constrains the change in foot motion such that contact frames stay "flat" with respect to the ground, but oriented with the character about the vertical axis.

To constrain the the change in overall ankle orientation to be about the z axis, the solver projects it to a rotation in the ground plane. Let $\mathbf{q}^d = \mathbf{q}^d_r ... \mathbf{q}^d_a$ be the overall orientation of the ankle after applying the displacement map, determined by concatenating the rotation parameters of ancestor joints (Figure 6.6, center). Similarly, let $\mathbf{q}^o = \mathbf{q}^o_r ... \mathbf{q}^o_a$ be the overall ankle orientation in the original motion (Figure 6.6, left). The world space change in 3D orientation of the ankle is then $\mathbf{q} = \mathbf{q}^d \mathbf{q}^{o*}$, where $*$ is the quaternion conjugate operator.

The solver projects $\mathbf{q}$ to $\hat{\mathbf{q}}$, a similar rotation about z (Figure 6.6, bottom), which results in the change of world space ankle rotation remaining parallel to the ground plane. The approach for constraining $\mathbf{q}$ to determine $\hat{\mathbf{q}}$ is described in Appendix A.

Given $\hat{\mathbf{q}}$, the solver must set a new local orientation for the ankle, $\hat{\mathbf{q}}^d_a$. First, note that

$$\hat{\mathbf{q}}^d = \hat{\mathbf{q}} \mathbf{q}^o, \tag{6.8}$$

since $\hat{\mathbf{q}}$ acts on $\mathbf{q}^o$ to result in $\hat{\mathbf{q}}^d$.

Second, note that $\mathbf{q}^d \mathbf{q}^{d*}_a = \mathbf{q}_r ... \mathbf{q}_k$, the concatenation of rotations up to, but not

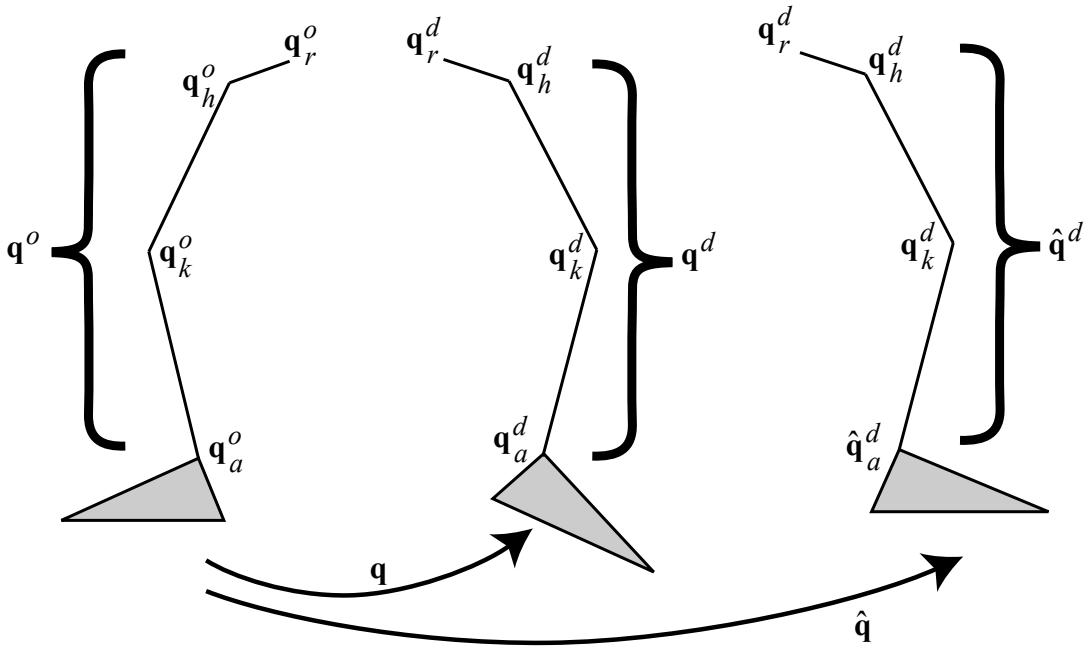


Figure 6.6: When applying a displacement map in the solution for $\mathbf{M}^p$, the original stance of the character (left) can have foot orientations that are no longer parallel to the ground (center). The overall ankle orientation is then constrained to remain parallel to the ground by solving for a new local ankle orientation $\hat{\mathbf{q}}^d_a$ (right).

including, the ankle, as $\mathbf{q}_a^{d*}$ acts to factor out the local ankle rotation from $\mathbf{q}^d$. Similarly, $\hat{\mathbf{q}}^d\hat{\mathbf{q}}_a^{d*} = \mathbf{q}_r...\mathbf{q}_k$, since the rotations above the ankle remain unchanged. Therefore, these terms are equivalent:

$$\mathbf{q}^d\mathbf{q}_a^{d*} = \hat{\mathbf{q}}^d\hat{\mathbf{q}}_a^{d*} \tag{6.9}$$

To find $\hat{\mathbf{q}}^d$, the local orientation to be set on the ankle, solve Equation 6.9 for $\hat{\mathbf{q}}^d$, set the result equal to Equation 6.8, and solve for $\hat{\mathbf{q}}_a^d$:

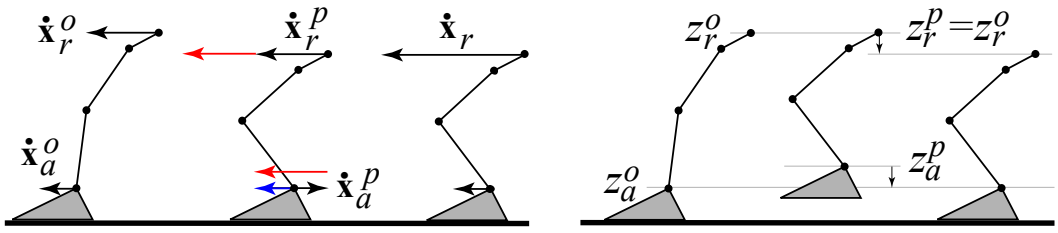
$$\hat{\mathbf{q}}_a^d = \mathbf{q}_a^d\mathbf{q}^{d*}\hat{\mathbf{q}}\mathbf{q}^o.$$

On a conceptual level, this expression takes the original world space ankle orientation $\mathbf{q}^o$, applies the constrained rotation $\hat{\mathbf{q}}$, and factors out the displacement mapped local joint rotations above the ankle ($\mathbf{q}_a^d\mathbf{q}^{d*}$).

In $\mathbf{M}^p$, the contributions to Equation 6.5 are due to root translation and foot parameters remaining unchanged. Equations 6.1 and 6.3, which model the constraint that foot motion during a sustained contact undergoes a rigid rotation about z, do not yet hold. The rotation change applied to the ankles has been constrained to be about z, but neither the contact rotation nor the contact translation is constant for the duration of each constraint. Equation 6.4 also does not yet hold, as the solver has not yet accounted for change in ground height below the ankle.

6.6.2 Root Motion and Contact Changes

When the system solves for the root motion and changes in contacts, the result should match the target poses and preserve contacts as best as possible. The root velocity should also remain oriented with the character, as determined by the changes in root orientation specified in the target poses. During flight phases, users can specify changes to the height of the root. The system first solves for the ground–parallel root motion and contact changes, as the ground height at the new contact locations will be needed to determine root height. The system then solves for the vertical motion of the root.



(a) (b)

Figure 6.7: During contact, root motion is horizontally adjusted (red) to preserve ankle velocity parallel to the ground (blue, a) and vertically adjusted to preserve ankle height above the ground (b).

Ground–Parallel Root Motion and Contact Changes

Ground–parallel root motion and the locations and orientations of contacts are mutually dependent. The system should find each such that contacts are preserved and target poses are matched when possible. Character orientation has been set as desired in $\mathbf{M}^p$; this is used to determine the change in orientation θ_a of each contact. The system then solves for root motion that reorients the original root velocity and approximately matches the desired contact motion of the ankles. To match target poses when possible, the system then refines the root motion, based on implicit constraints that exist among the positions of the root in temporally adjacent target poses that share contact constraints.

Contact Rotations: The system should find the rotation for each contact constraint that aligns the foot motion in $\mathbf{M}^o$ during contact to the foot motion in $\mathbf{M}^p$. If target poses occur during a contact, the system computes the contact rotation that aligns the foot, in an average sense, to those target poses. This is an exact alignment if one target pose occurs or the target poses that occur during the contact have consistent overall ankle orientations. If no target poses occur during the contact, the system computes a rotation that aligns the foot, in an average sense, to all frames over the duration of the contact.

The 2D rotation that aligns foot joint positions with respect to the ankle is $\theta_a =$

$\arg(\mathbf{z}_{\theta_a})$, where $\mathbf{z}_{\theta_a}$ is complex and has the closed–form expression

$$\mathbf{z}_{\theta_a} = \frac{1}{\sum_{t,j} |\mathbf{v}_j^o(t)|^2} \left(\sum_{t,j} \mathbf{v}_j^p(t) \cdot \mathbf{v}_j^o(t), \sum_{t,j} \mathbf{v}_j^o(t) \times \mathbf{v}_j^p(t) \right).$$

Each $\mathbf{v}_j$ is 2D, consisting of the x and y coordinates of its 3D equivalent. The summation index t is over the times $\{t_i\}$ of the target poses; if none occur, the summation is over $t \in [t_s, t_e]$. This is a simplification of the transformation described by Horn [53]; the system does not need to account for scaling, since $|\mathbf{v}_j^o| = |\mathbf{v}_j^p|$.

Approximate Root Velocity: The system solves for root motion that *reorients the original root velocity* and *approximately matches the desired contact motion of the ankles*. This is solved in the velocity domain for two reasons. First, this approach allows for explicit inclusion of the rotated root velocity of the original motion. Second, it allows for the formulation of the ground–parallel portion of contact preservation as a constraint on ankle velocity, with respect to the ground.

Ground–parallel root motion affects Equations 6.1 and 6.7. To determine the velocity domain contact constraints, differentiate Equation 6.1 and apply it to $\mathbf{M}^r$; this results in a constraint for each contact that

$$\dot{\mathbf{x}}_a^r(t) = R_{\theta_a}(\dot{\mathbf{x}}_a^o(t));$$

i.e. the system should preserve ankle velocities, but rotate them by θ_a. Equation 6.7 is met exactly if $\dot{\mathbf{x}}_r^r(t) = R_{\alpha(t)}(\dot{\mathbf{x}}_r^o(t))$. To meet the latter goal as best possible while approximately enforcing the contact constraints, the system finds a root motion similar to $R_{\alpha(t)}(\dot{\mathbf{x}}_r^o(t))$ that approximately matches the rotated contact velocities of the ankles. This approximation error will be accounted for when determining $\mathbf{M}^f$.

The system first finds an initial root velocity that takes Equation 6.7 into account by explicitly setting $\dot{\mathbf{x}}_r^\alpha(t) = R_{\alpha(t)}(\dot{\mathbf{x}}_r^o(t))$. To rotate the ground–parallel velocity of the original motion, it finds the the closest rotation about the z axis to the change in root rotation from $\mathbf{M}^o$ to $\mathbf{M}^p$; the latter is $\mathbf{q}(t) = \mathbf{q}_r^p(t)\mathbf{q}_r^{o*}(t)$. The system uses $\hat{\mathbf{q}}(t)$, a rotation

similar to $\mathbf{q}(t)$, but constrained to be a rotation about the z axis (See Appendix A). The angle encoded in $\hat{\mathbf{q}}(t)$ is $\alpha(t)$.

The system then finds the approximate root velocity $\dot{\mathbf{x}}_r$ by applying a smooth change to $\dot{\mathbf{x}}_r^\alpha$ to approximately match contact velocities. Figure 6.7a illustrates the matching of contact velocity for a single limb, without considering rotation. $\mathbf{M}^o$, at left, has subtle ankle movement and larger root movement. In $\mathbf{M}^p$, at center, the ankle is sliding significantly backwards, although the root is unchanged. The system applies the difference (red) to restore the original ankle velocity (blue). The resulting root velocity, at right, is $\dot{\mathbf{x}}_r = \dot{\mathbf{x}}_r^p + (\dot{\mathbf{x}}_a^o - \dot{\mathbf{x}}_a^p) = \dot{\mathbf{x}}_r^o + (\dot{\mathbf{x}}_a^o - \dot{\mathbf{x}}_a^p)$. Incorporating rotation and generalizing to multiple limbs, the system take an average over the N simultaneous contacts, resulting in the expression

$$\dot{\mathbf{x}}_r = \dot{\mathbf{x}}_r^\alpha + \frac{1}{N}\sum_a \left(R_{\theta_a}\left(\dot{\mathbf{x}}_a^o\right) - \dot{\mathbf{x}}_a^\alpha\right),$$

where $\dot{\mathbf{x}}_a^\alpha$ takes into account the effect of R_α on the ankle velocity. This is applied at each contact frame and the system fills gaps to attain $\mathbf{x}_r(t)$. In general, $\mathbf{x}_r(t)$ is not smooth, again due to the discrete boundaries of contacts; a low–pass filter is used to smooth the result.

The system uses backwards differences to compute velocity and explicit Euler integration to determine the root motion $\mathbf{x}_r(t)$, using the root position at the first frame as the integration constant. If no target poses have been specified, $\mathbf{x}_r(t) = \mathbf{x}_r^o(t)$, as these methods are inverse functions. The low–pass filter is a tent filter with a filter width of half a second.

Overall, this step computes a smooth displacement, relative to the rotated original velocity of the root, that approximately matches ground–parallel ankle velocity to the desired ankle velocity.

Root Motion Refinement: The ground–parallel root motion in $\mathbf{x}_r(t)$ approximately meets contact constraints. The system must now refine it such that *new contact locations can be determined that allow target poses to be matched whenever possible.* A

key observation a this point is that *any temporally adjacent pair of target poses that share a contact constrain the geometric relationship between the root positions at the times of the target poses.*

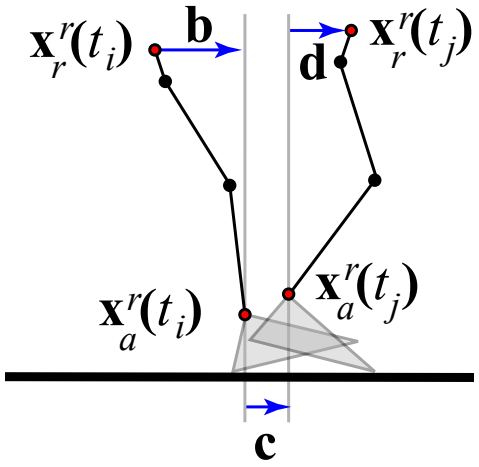


Figure 6.8: The root positions of a pair of target poses that share a contact constraint are spatially constrained, parallel to the ground.

For a pair of poses at times t_i and t_j, the root position must be set such that the shared contact constraint can be positioned to meet the ankles at both times t_i and t_j. The geometric constraint, illustrated in Figure 6.8 using a single limb swinging over a subtly moving ankle, is $\mathbf{x}_r^r(t_j) = \mathbf{x}_r^r(t_i) + \mathbf{b} + \mathbf{c} + \mathbf{d}$, where $\mathbf{b} = \mathbf{x}_a^r(t_i) - \mathbf{x}_r^r(t_i)$, $\mathbf{c} = \mathbf{x}_a^r(t_j) - \mathbf{x}_a^r(t_i)$, and $\mathbf{d} = \mathbf{x}_r^r(t_j) - \mathbf{x}_a^r(t_j)$. Taking contact rotation into account, $\mathbf{c} = R_{\theta_a}\left(\mathbf{x}_a^o(t_j) - \mathbf{x}_a^o(t_i)\right)$.

To match the target poses, which are correct in $\mathbf{M}^p$, it is required that $\mathbf{b} = \mathbf{x}_a^p(t_i) - \mathbf{x}_p^r(t_i)$ and $\mathbf{d} = \mathbf{x}_r^p(t_j) - \mathbf{x}_a^p(t_j)$. Neither $\mathbf{b}$ nor $\mathbf{c}$ are rotated, as they have the correct orientation in $\mathbf{M}^p$. When considering multiple contacts that are shared by the two poses, use the average values of $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$; this leads to the expression

$$\mathbf{x}_r^r(t_j) = \mathbf{x}_r^r(t_i) - \mathbf{x}_r^p(t_i) + \mathbf{x}_r^p(t_j) +$$

$$\frac{1}{N}\sum_a \left[\mathbf{x}_a^p(t_i) + R_{\theta_a}\left(\mathbf{x}_a^o(t_j) - \mathbf{x}_a^o(t_i)\right) - \mathbf{x}_a^p(t_j)\right].$$

As both root positions in $\mathbf{M}^r$ are unknown, the system fixes the root position at the time of the first target pose frame as $\mathbf{x}_r^r(t_0) = \mathbf{x}_r(t_0)$ and sets the corresponding refinement

as $\mathbf{r}_0 = (0, 0)$. It then iterates through the target poses, forward in time, to determine a refinement for each t_i. If $\mathbf{P}_i$ and $\mathbf{P}_{i-1}$ share contacts, the desired value for $\mathbf{x}_r(t_i)$ is found using the above expression; the corresponding refinement is $\mathbf{r}_i = \mathbf{r}_{i-1} + (\mathbf{x}_r^r(t_i) - \mathbf{x}_r(t_i))$. Otherwise, $\mathbf{r}_i = \mathbf{r}_{i-1}$. The system applies a smooth displacement map with knots $\{(t_i, \mathbf{r}_i)\}$ to $\mathbf{x}_r(t)$ to get $\mathbf{x}_r^r(t)$, the final root motion in $\mathbf{M}^r$.

The resulting ankle positions in $\mathbf{M}_r$, for a pair of adjacent target poses, are now set such that the system can determine contact locations that exactly align whenever the target pose pair has been specified with consistent contacts. *This refinement only accounts for the ability to exactly match target poses and contacts at target pose times; the system has not yet corrected for the approximation error that results from keeping root motion smooth at other frames.* Similar constraints could be applied to the root orientation to match leg twist to temporally adjacent target poses; by design, the system allows users to retain control over root orientation.

Contact Locations: The contact location parameters $\bar{\mathbf{x}}_a$ are set such that ankle motion in $\mathbf{M}^o$ during contact, when transformed, aligns to the target pose frames in $\mathbf{M}^r$ whenever possible. For a single target pose, the system solves Equation 6.1 for $\bar{\mathbf{x}}_a$, plugging in $\mathbf{x}_a^r(t_i)$ for $\mathbf{x}_a(t_i)$. If multiple target poses are specified, the system finds a value that aligns the transformed ankle motion in an average sense:

$$\bar{\mathbf{x}}_a = \frac{1}{N} \sum_t \left(\mathbf{x}_a^r(t) - R_{\theta_a} \left(\mathbf{x}_a^o(t) - \bar{\mathbf{x}}_a^o \right) \right).$$

In this expression, the summation is over the N target pose frames t_i that take place during the contact. If the target poses have been specified consistently, root motion refinement guarantees that the contribution from each frame will be equal, and the alignment will be exact. If not, this will balance the position error among the target poses. If no target poses take place during the contact, the system sets $\bar{\mathbf{x}}_a$ to align the original ankle motion during contact to that in $\mathbf{M}^r$ in an average sense by taking the summation over all frames $t \in [t_s, t_e]$.

Vertical Root Motion

The system sets the vertical motion of the root to approximately match the ankle height during contact frames to the original ankle height, while accounting for changes in ground height due to the change in contact locations and any user edits made to the environment. At target pose frames, this is set in an average sense, considering all limbs in contact and potential inconsistencies in the pose. At other frames, this will be approximated to keep the motion smooth; approximation error will be accounted for when determining the final foot motion, $\mathbf{M}^f$. To do this, the system sets the root height individually on each frame and then modifies the values for smoothness.

To compute a new vertical position for the root on a given frame, as illustrated for a single limb in Figure 6.7b, the system computes

$$z_r = z_r^p + \frac{1}{N}\sum_a [(z_a^o - z_a^p) + (g_a - g_a^o)],$$

where the summation is over the N ankles in contact at that frame. The result, $z_r(t)$, is not smooth, as contact constraints have discrete temporal bounds. A smooth final result $z_r^r(t)$ is created by applying a low–pass filter, and contact errors introduced by the filter are corrected with an interpolating spline. Again, a tent filter with a filter width of half a second is used for the low–pass filter.

To allow users to adjust height during flight phases, the system applies a local, smooth displacement map that affects the vertical root motion for the duration of each flight phase. To bound the change in time, zero–valued knots are set at the start and end of flight.

Relating $\mathbf{M}^r$ to the Constraints

As the goal of this step of the solver is to find smooth root motion and changes to contact locations that approximately preserve contacts while matching target poses, this section discusses how the solution for $\mathbf{M}^r$ relates to Equations 6.1–6.7. The vertical

root motion in $\mathbf{M}^r$ reduces the error in meeting Equations 6.2 and 6.4, considering all contact frames, when the approximate solution is applied. Refinement to match ankle height to contacts at target pose times further reduces the error in meeting Equation 6.2 at these times; the error at these frames becomes zero if the target pose has consistent contacts. Values at other times can increase slightly; this is acceptable, as it is important to keep the root motion smooth. The system accounts for the remaining error in meeting Equations 6.2 and 6.4 when solving for $\mathbf{M}^f$. Consequently, the contribution of contact frames to Equation 6.6 will be reduced.

When solving for ground–parallel root motion, the system first finds contact orientation changes. Since it finds rotations that exactly align the original ankle motion to target poses whenever possible, and in an average sense when not possible, the error in meeting Equation 6.3 is reduced at those times, and the contribution becomes zero when target poses are consistent with contacts. The approximate root motion solution effectively reduces Equation 6.7 to zero, but then increases it, as needed, to decrease the error in meeting Equation 6.1. This is consistent with Equation 6.1 having higher priority than Equation 6.7. The root motion refinement and determination of contact location changes then act to reduce error in meeting Equation 1 to zero at contact frames whenever possible.

6.6.3 Solving for Foot Motion

The foot motion in $\mathbf{M}^r$ exactly meets the contact constraints on the frames of target poses, if possible, and approximately meets the contact constraints on other contact frames. In this step, the solver computes the final foot motion $\mathbf{M}^f$ that exactly meets the contact constraints in Equations 6.1–6.4. $\mathbf{M}^f$ contains only foot motion; each ankle acts as a root. The last step of the solver will modify $\mathbf{M}^r$ to match the foot motion in $\mathbf{M}^f$ when determining the final edited motion $\mathbf{M}$.

The foot motion $\mathbf{M}^f$ is initialized to the foot motion in $\mathbf{M}^r$ by detaching the feet at

the ankles. This foot motion is then modified to smoothly move each foot from contact to contact, while adapting the foot swing to avoid any ground penetration. Finally, a displacement map is applied to match foot parameters to target poses that occur during a leg swing.

First, the solver sets ankle parameters to explicitly preserve contacts to meet the constraints given in Equations 6.1–6.3. These are set as follows:

$$\mathbf{x}_a^f(t) = \bar{\mathbf{x}}_a + R_{\theta_a}\left(\mathbf{x}_a^o(t) - \bar{\mathbf{x}}_a^o\right)$$

$$z_a^f(t) = z_a^o(t) + \bar{g}_a - \bar{g}_a^o.$$

In the first expression, $\mathbf{x}_a$ is 2D . This affect is optionally attenuated as ankle height increases if the end effecter moves far from the ground, such as occurs in the cartwheel motion. To meet Equation 6.3, the solver sets the ankle rotation to match the concatenation of the overall rotation of the original ankle with the contact rotation θ_a. Letting $\mathbf{q}_{\theta_a}$ be the unit quaternion representation of R_{θ_a}, this is

$$\mathbf{q}_a^f(t) = \mathbf{q}_{\theta_a}\mathbf{q}_r^o(t)...\mathbf{q}_a^o(t).$$

The solver has not changed any skeletal parameters below the ankle; as a result, the ankle rotation now meets Equation 6.3.

Second, the solver applies a displacement map to create smooth ankle motion that moves from contact to contact, remaining similar to the swing in the original motion, but avoiding any ground interpenetration. A layered displacement map is applied; each layer has bounding knots at t_e and t_s, where t_e is the end of one contact and t_s is the start of the next. Vertical height in the first layer is set to match the original motion's height as $z_a^f(t) = z_a^o(t)$. Additional layers are recursively applied to smoothly avoid ground interpenetration. For each layer, the solver measures the penetration error of each frame during the leg swing and chooses the frame with the largest error. A displacement map layer is applied to account for the error on this frame. This process is repeated until no

frames have ground penetration. Overall, this approach keeps the motion smooth and similar to the original, but allows leg swings to move among contacts that change height while smoothly passing over any obstacles.

Finally, the solver applies a displacement map to the parameters of foot joints below the ankle, to account for target poses that occur between contacts. This further reduces the value of Equation 5. This displacement map has zero–valued knots at t_e and t_s, bounding the change to the gap between contacts.

6.6.4 Solving for Limb Motion

In the final step, the solver combines the body and limb motion of $\mathbf{M}^r$ with the foot motion of $\mathbf{M}^f$ to determine the final edited motion, $\mathbf{M}$. A per–frame IK solver is used to align the ankle positions and orientations in $\mathbf{M}^r$ to $\mathbf{M}^f$. This is applied to every frame of the motion and then a displacement map is applied to the limb configurations of target poses during swing phases. The skeletal parameters of joints below the ankles are set equal to those in $\mathbf{M}^r$.

The system uses a closed form IK solver that introduces stretching to avoid fast accelerations when the knee is close to straight [75]. This is important, as it keeps the overall change to the motion smooth. Stretching is introduced when the knee angle exceeds 170 degrees, and the knee extension is bound to 175 degrees; only stretching is applied beyond this bound. If a given frame has a knee extended beyond these bounds, in either the original motion or $\mathbf{M}^r$, the system overrides the bounds to be no less than the existing knee angles. This avoids modifying close–to–straight leg configurations to introduce extra bending and stretching, as this could cause the solver to fail to match target poses in which users have intentionally set legs to be nearly straight.

If target poses are consistent with respect to contacts, the limbs do not change at target pose frames, as the ankle positions and orientations match in $\mathbf{M}^r$ and $\mathbf{M}^f$. If they are not consistent, the contact transformation parameters have been set to match them

in an average sense, and the resulting leg configurations at these target pose frames will be aligned to these contacts. This is consistent with contact preservation having higher priority than matching target poses.

Finally, the solver applies a displacement map to the skeletal parameters of each limb during the gaps between contacts to match target poses that occur when a limb is not in contact. This is done using the same technique that was used for foot motion, but with the solver applying the change to the parameters affected by the IK solver (hip rotation, and both translation and rotation for the knee and ankle). This will likely increase Equation 6.6. This is acceptable, as it reduces Equation 6.5, which has higher priority.

6.7 Examples and Discussion

The pose–centric editing system has been been integrated into the Maya[2] commercial animation system. To demonstrate pose–centric editing, a variety of motion examples have been edited to meet various goals. This section describes these editing examples, discusses the capabilities of the pose–centric editing system, and relates the approach to existing techniques and workflows.

Given the original motion, specified on a Maya control skeleton as time samples or animation splines, the pose–centric editing system presents users with a set of *editing poses* for specifying the target poses used for applying pose–centric edits. Users directly manipulate a Maya control skeleton for each pose, and Maya's IK tools can be used to assist users with respecting ground contact, although any edits that users make to the feet at times of contact will be modified as needed to preserve ground contact. This will be a benefit to users wishing to quickly make large, pose–centric edits, as manually applying edits to foot joints in a way that retains contact plausibility can be extremely

[2]http://www.autodesk.com/maya

time–consuming and tedious.

As users update these poses, the system interactively updates the character's motion. A 93 degree–of–freedom character with 700 frames of motion updates at about 5 frames per second on a 2.16 GHz MacBook Pro, and there is significant room for optimization in the current implementation. Since the pose–centric editing system modifies the root trajectory, the system includes a tool that repositions the editing poses to the locations at which they occur in the edited motion. These root positions are not updated interactively; this would result in Maya's direct manipulation widgets moving in space while the user edits them, which would make pose editing difficult.

The system allows users to algorithmically select extreme poses at times of velocity minima or at times of maximal spatial extremeness (Chapter 5), and other approaches to algorithmic pose selection ([10, 17]) can be added to the system as plugins. Depending on the algorithmic pose selection algorithm used, users might need to refine the set of selected poses; the pose–centric editing system includes tools for refining the set of selected extreme poses.

Ground height is modeled using an implicit height field representation. One height field is specified as the *original ground height*, and another height field is specified as the *edited ground height*. These are typically the same, but if users add obstacles to or remove objects from the environment, the objects are added to or removed from the edited ground height. Typically, the original ground is constant, as most recorded motion is recorded in flat environments. Objects can be specified using arbitrary polygon meshes, and a signed distance function is evaluated as as needed to determine ground height values.

6.7.1 Examples

The first example demonstrates the editing of a cartwheel motion to give the character a wider stance whenever both arms or both legs are in ground contact (Figure 6.9). Note that the pose–centric editing system successfully widens the stance, but moves contact

locations in the environment. This example demonstrates how pose edits can be applied while preserving ground contact on limbs other than the legs.

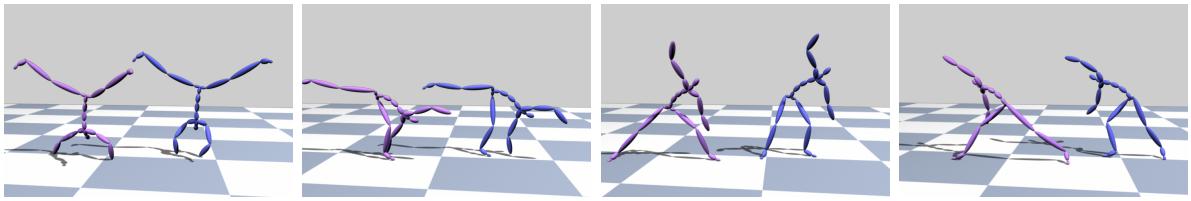


Figure 6.9: Cartwheel motion (blue) is edited (purple) to increase the stance at times when both feet are on the ground and at times when both hands are on the ground.

The next example demonstrates pose–centric edits for running motion. In this example (Figure 6.10), the poses are edited to increase the rotational change of the root and back and to loosen up the motion of the arms. Speed stays approximately the same, as stride length has not been significantly changed, but the nature of the stride has. This creates a stylized, looser feel to the motion that is less constrained than typical running motion.

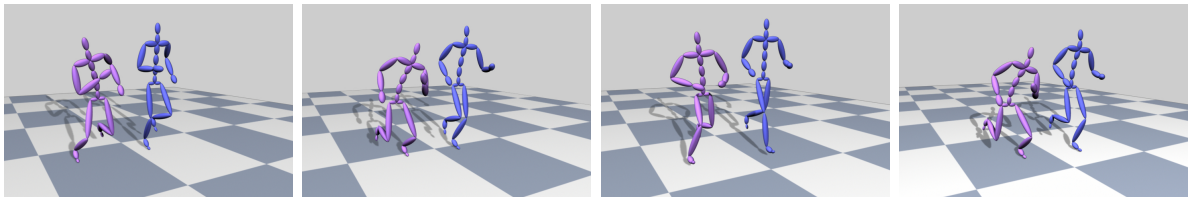


Figure 6.10: Running motion (blue) is edited (purple) to create a looser sense of style.

To demonstrate the system's ability to handle contacts at different heights, Figure 6.11 demonstrates an edit to motion that includes a step up and a step down. Stride length and arm swing are decreased and the back is tilted to create a slower, more relaxed version of the motion. Since the editing approach is pose–centric, a smaller box is needed for the character to walk across to account for the shorter stride length. Approaches that increase the number of steps in gait motion [69] could be used in conjunction with the pose–centric editing system if object size needed to be held constant.

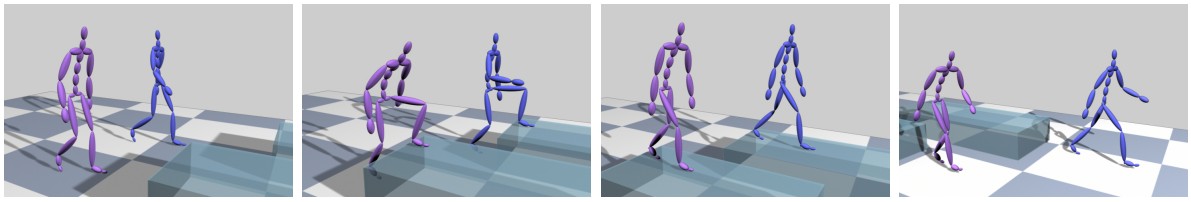


Figure 6.11: The motion of a character walking and stepping up (blue) is edited (purple) to create a less energetic walk with a shorter stride.

In the next example (Figure 6.12), a jump is edited to increase the apparent effort used by the character. This is done by increasing the bend of the legs to lower the character's root just before and just after the leap.

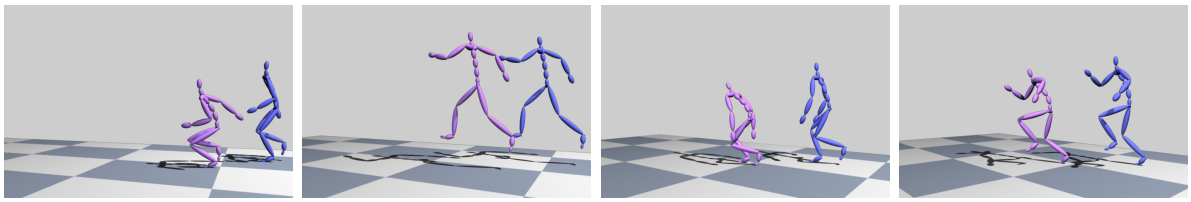


Figure 6.12: A jump (blue) is edited to exaggerate how much the character crouches before jumping and after landing (purple).

The walk motion is then edited to create a stylized variation (Figure 6.13). In this example, the arm swing and back rotation are increased and the character makes small changes in direction; this demonstrates how the system can be used to create stylistic motion variations that include a sense of personality.

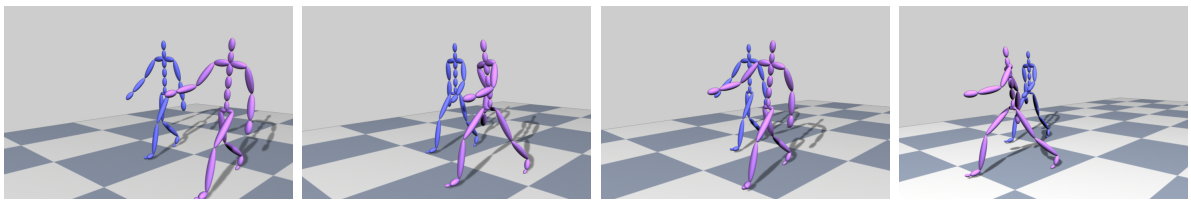


Figure 6.13: A neutral walking motion (blue) is edited to add a sense of personality (purple).

In 3D character animation, *moving holds* are used to avoid having characters stand

or sit completely still, which would cause the character to appear lifeless [85]. The subtle motion that humans make when attempting to remain still can be described as *ambient motion* or *motion texture*. In Figure 6.14, the pose–centric editing system is used to apply the subtle ambient motion of a standing human to a character holding an object. This edit was made by changing a single pose; the object is constrained to the character using Maya's existing tools.

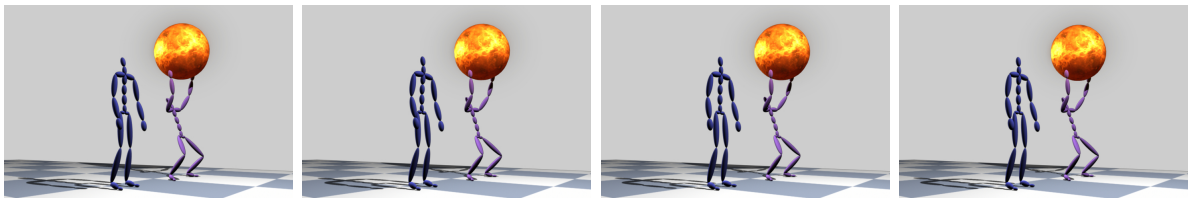


Figure 6.14: The motion of a character standing nearly still (blue) is modified with a single pose edit to create subtle body movement while the character holds an object (purple).

The pose–centric editing system can also be used to retarget motion to characters with different limb lengths. In Figure 6.15, the walking motion is applied to a character whose legs and arms are half as long as the original character, but whose torso, feet and hands remain the same in size. This edit is specified by choosing to edit a single pose and then setting the desired skeletal dimensions; the pose centric editing system automatically adapts the entire motion to account for the shorter limb length.

To demonstrate the pose–centric editing system's ability to adapt to changes in ground height, objects are introduced for a character to step onto during a walk (Figure 6.16). When combined with pose edits, the pose–centric editing system allows motion to be tailored specifically to environments with complex changes without requiring users to manually manage contact (Figure 6.17).

To show the effect of keeping root velocity oriented with the character, Figure 6.18 shows an edit in which the poses are all rotated by ninety degrees about the world up axis. When velocity is reoriented (middle), the motion of the character matches the original

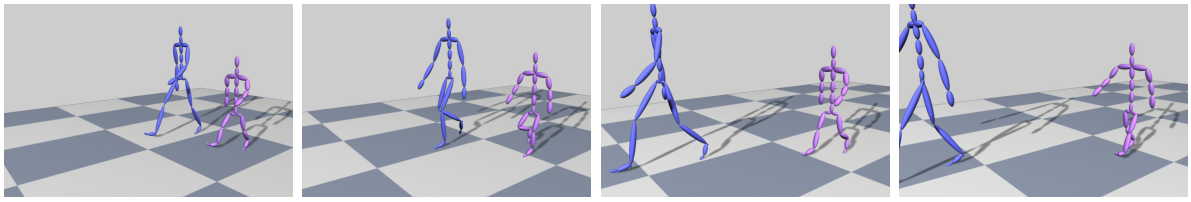


Figure 6.15: Retargeting can be done by editing the limb lengths of a single pose to match the limb lengths of a desired character.

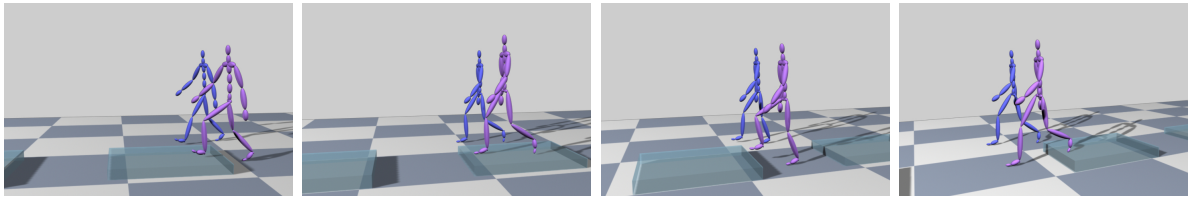


Figure 6.16: The walking motion (blue) is automatically adapated by the pose–centric editing system (purple) to account for the introduction of new obstacles over which the character must walk (transparent blue).

motion, but rotated in space. If root velocity is not reoriented, a subtle, side–to–side jerkiness results (right). This jerkiness is due to the natural variation in forward speed not staying aligned to the forward direction of the character. While appearing subtle in Figure 6.18, this effect can be objectionable when viewing the character in motion.

Figure 6.19 illustrates the effect of ground–parallel root motion refinement. When disabled, leg stance does not match that in the target pose (Figure 6.19a). When enabled, the leg stance does match (Figure 6.19b). Note that, in this example, the feet do not

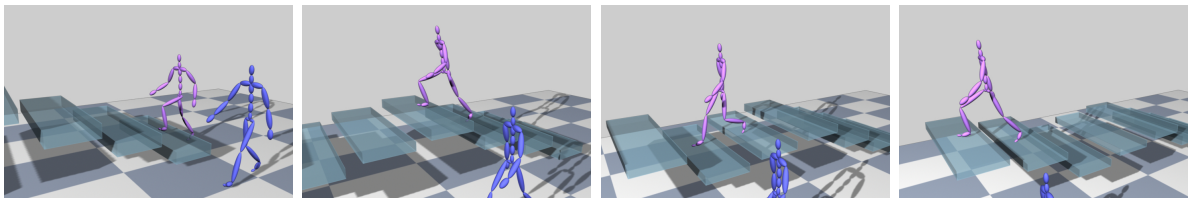


Figure 6.17: The pose–centric editing system adapts the walking motion to a complex environment with many changes in step height.

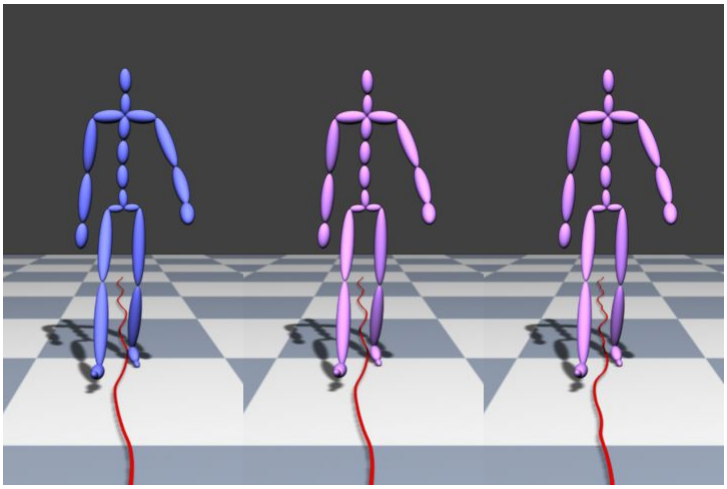
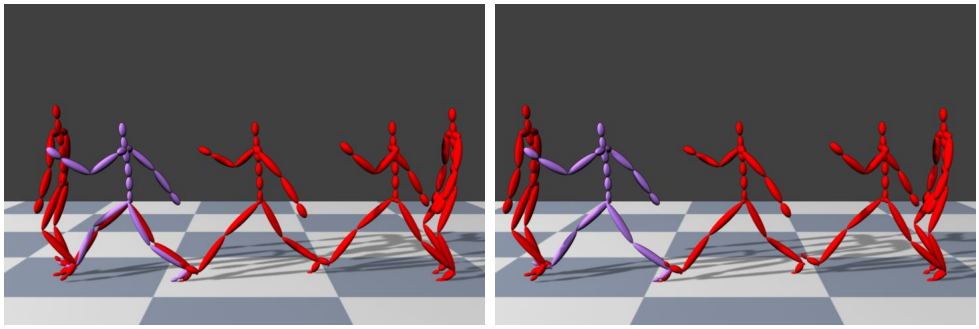


Figure 6.18: In this example, the effect of the root velocity constraint is illustrated. The original motion, shown at left, has the root trajectory projected to the ground and illustrated in red. If the target poses are rotated by ninety degrees about the world up axis, but otherwise left unchanged, the pose–centric editing system effectively rotates the root trajectory without changing its shape (middle, view is also rotated by ninety degrees). If root velocity is not considered, the shape of the root trajectory is altered, and subtle jerkiness can be introduced to the resulting motion (right, view is again rotated by ninety degrees).

match the target poses, as they do not preserve contacts in the target poses, and the system corrects for this.



(a) (b)

Figure 6.19: In this example, the effect of the root trajectory refinement is illustrated. If refinement is not included, the state of the legs in the target poses is typically not matched (a). If refinement is included, legs are successfully matched to their state in the the target poses (b).

6.7.2 Discussion

Relationship to Other Approaches: While the comparison in this chapter has focused on have how pose–centric editing differs from the use of displacement maps with subsequent footskate correction, the reasons for the errors that occur when doing so are not specific to that choice. On a fundamental level, ground penetration during leg swing occurs because smooth changes in joint angle parameters do not modify the bend of the knee. Other approaches model smoothness in terms of joint parameters, and would have the same limitations. Similarly, pose edits would not be matched when possible by other systems, as they cannot account for the interdependency between ground contacts and target poses.

These limitations result from most motion editing systems being designed to focus on the constraints of the motion, rather than the pose of the character, and these systems excel at meeting such constraints. Footskate correction is an example that is specifically

constraint–centric. In contrast, pose–centric editing specifically aims to match target poses, only prioritizing contact preservation above it to keep the motion plausible with respect to the ground. The pose–centric editing system also explicitly keeps end effector trajectories above the ground, avoiding the ground penetration that can occur with systems that only aim to keep joint motion smooth between contacts.

Relationship to Animator Workflows: In principle, existing IK tools can be used to apply edits similar to the pose–centric edits applied in the examples shown in this chapter. To do this, IK rigs can be set up on the legs of the character, and their effect on the leg stance can be animated on and off over time to fix contact error. This has to be balanced with changes that the animator would need to make to the root motion to allow the pose edits to be met. This can be exceptionally tedious and time–consuming, however.

To support this claim, the pose–centric editing system was described or demonstrated to a number of professional animators and a software designer who designs commercial motion editing tools. They all indicated that they would prefer to animate new motion than to use available tools to make similar edits, due to the practical difficulty of using existing tools.

In addition, four animators were asked to recreate the edit shown in Figure 6.1 using any commercial tools of their choice. Three of the animators stated that the change would be too difficult in practice and did not attempt it. The fourth did attempt to make the edit, but gave up after over three hours of work. In this time, he had manually tracked the root and increased the stride length for about two seconds of motion, but had not addressed contact preservation at all. In contrast, the author edited the 5 second motion for Figure 1 in about five minutes using the pose–centric editing system; the majority of this time was spent using Maya's pose editing tools to set the target poses.

Assumptions and Limitations: The pose–centric editing approach preserves contacts by applying a ground–parallel rigid transformation and vertical translation to the foot motion. While this prevents users from editing the foot stance during contact, it allows them to quickly make full body edits without having to position the feet into plausible configurations, which can be tedious. The system also assumes that the specified pose edits have plausible meaning with respect to the motion, relying on the user to specify good target poses. Finally, the approach assumes there are no pre–existing contact errors in the original motion. If such errors exist, the system preserves them. Footskate correction can be used before applying pose–centric editing to easily correct such errors.

As the pose–centric editing system focuses on the application of geometric pose changes to a full motion, user control over timing and the type of interpolation used has not not been implemented. Control over these aspects of motion would likely be needed for general–purpose use. In addition, the current implementation uses poses that are specified for the entire body; this could be generalized to handle poses that affect only portions of the body, potentially with encoded timing variation, such as that included in the staggered poses system (Chapter 7).

6.8 Future Work and Conclusion

This chapter has presented an efficient algorithm for applying pose–centric edits to character motion while preserving both ground contact constraints and plausible trajectories on limbs that make contact. While the approach can handle a wide variety of pose–centric edits, it could be extended in a variety of ways. When editing vertical height during flight phases, it would be more plausible to preserve the parabolic motion constraint on the center of mass. Because the pose–centric editing system explicitly preserves the motion of the foot during a ground contact to allow for quick high–level changes to be made,

users can not change the skeletal parameters of the foot during the duration of a ground contact; the approach could be extended to allow this when needed. Another potential direction is the development of an online version of the algorithm that is capable of applying pose–centric changes, given a sufficient temporal window of upcoming motion. Finally, the system relies on users to specify meaningful pose edits, it would be interesting to investigate algorithmic techniques for assisting novice users with the creation of target poses that remain meaningful in the context of the full motion.

Pose–centric editing modifies both the root trajectory and contact locations in the environment to match user–specified target poses. The solution is closed form, allowing it to scale well with respect to motion length and the complexity of the character. This approach allows users to quickly apply pose–centric edits without requiring them to explicitly manage the interdependency between root motion and pose constraints, the preservation of ground contact, or the plausibility of vertical end effector motion. By allowing users to focus solely on performance–related aspects of movement, the system enables users to make high–level changes to the expressive style of a motion, similar to those made by keyframe animators, that have previously been infeasible in motion editing applications.

Chapter 7

Staggered Poses for Modeling Coordinated Timing

This chapter introduces staggered poses—a representation of character motion that explicitly encodes coordinated timing among movement extrema in different parts of a character's body. This representation allows sparse, pose–based editing controls to be used for editing coordinated timing among different body parts for stylistic control. The staggered pose representation supports the editing of new motion by generalizing keyframe–based workflows to retain high–level pose–centric control after local timing and transitions have been specified using splines. For densely sampled motion, such as recorded motion capture data, an algorithm is presented that locates joint–centric extrema on individual body parts and combines them into staggered poses. Motion detail is then represented using splines and displacement maps. These techniques, taken together, enable the coordinated editing of pose and timing relationships among joint–centric extrema in dense motion data.

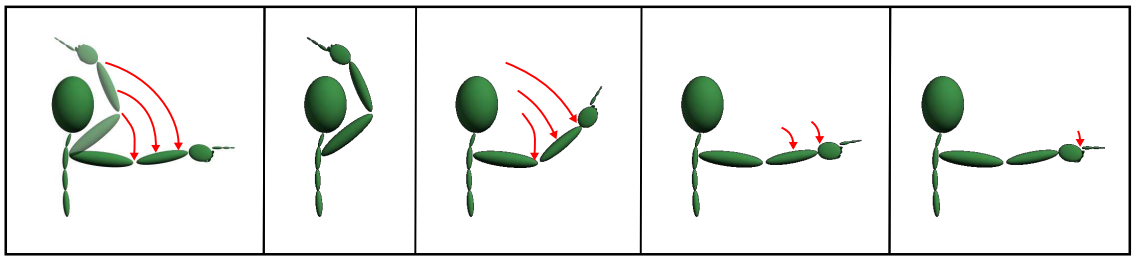
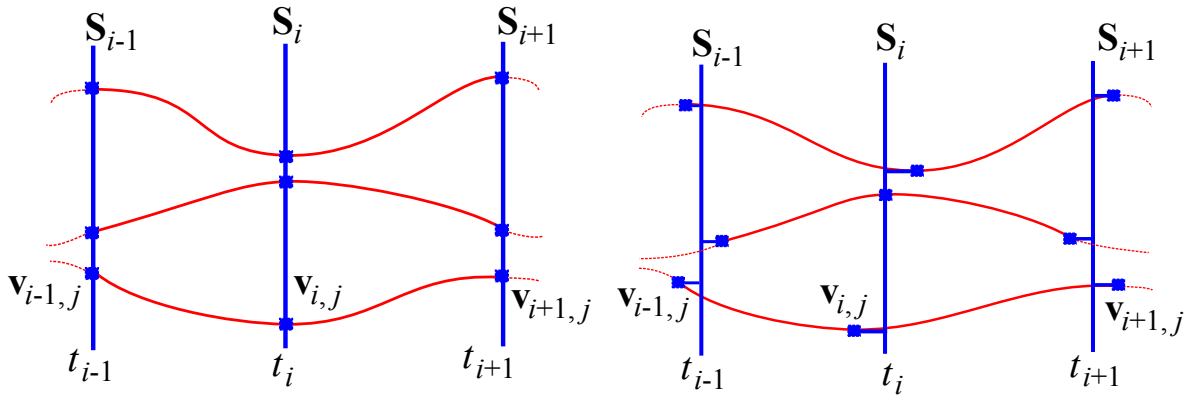


Figure 7.1: In contrast to traditional poses (left), staggered poses explicitly represent the timing relationships among coordinated body parts (images 2–5), which capture important nuances of character motion.

7.1 Motivation and Overview

An important part of a character's motion is the timing that specifies how a character moves from one pose to another. For example, speed of transition is important for conveying mass or character thought [84, 85]. Simply interpolating from pose to pose appears over–constrained and unnatural. Animators have long observed that a motion begins at the hips and propagates out to the extremities [84]. To quote Disney, "Things don't come to a stop all at once, guys; first there is one part, and then another" ([167], p.59). This idea, which is used to model *overlapping action*, is illustrated in Figure 7.1, in which each joint of the arm comes to rest after its parent joint. In addition, coordinated movement in different limbs is not exactly aligned in time; *asymmetrical action* is another important part of believable motion [32]. Despite the importance of timing variation in modeling believable movement, current animation and motion editing tools provide no support for working with these relationships as a whole.

Traditional poses represent the geometric state of the character and are interpolated to create movement. Tools for editing these geometric poses require a large number of similar poses to capture timing variation. If any significant geometric change needs to be made, each of the similar poses created to model the timing variation must be updated. Spline editors and other knot–editing tools allow users to create timing variation at a



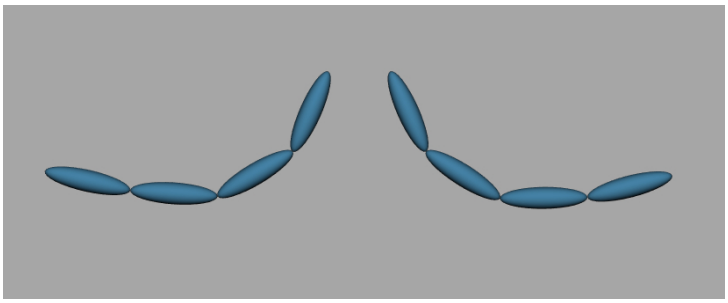
(a) (b)

Figure 7.2: Pose–based editing systems align all coordinated knots in time (a). Staggered poses (b) incorporate a timing refinement for each knot that encodes timing variation.

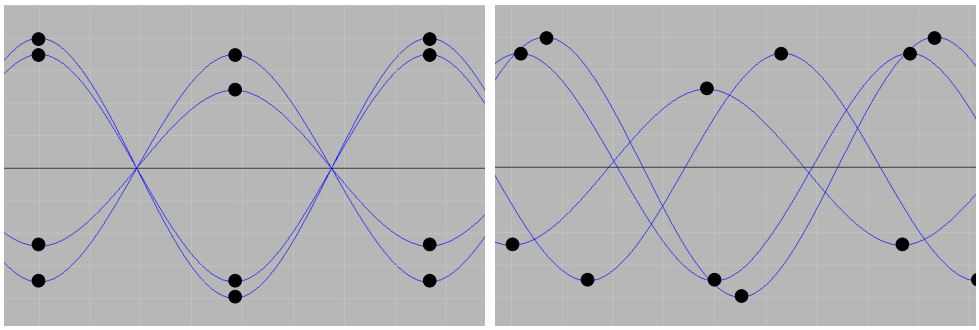
low level by moving individual knots in the time domain, but making these changes comes with the cost of losing the ability to geometrically alter coordinated extrema as a single pose. Again, users must work across multiple frames to introduce any significant geometric change. This chapter introduces *staggered poses*, a representation of motion that explicitly encodes timing variation among coordinated motion extrema. This enables geometric pose–based editing of coordinated extrema that have varied timing as well as the procedural editing of these timing relationships.

Staggered poses generalize traditional poses by allowing key values of different degrees of freedom to be specified at slightly different times. Relative to a traditional pose (Figure 7.2a), this representation staggers the timing, using explicitly encoded *timing refinements* (Figure 7.2b). The timing relationships among these key values determine how the character will pass through the extreme values of the pose and are important for modeling believable propagation of force and intention through a body.

In keyframe animation that represents motion as interpolation between sequential poses, each pose specifies one knot for each degree of freedom, and splines typically specify the interpolated values between the knots. In practice, poses are not often explicitly



(a)



(b) (c)

Figure 7.3: When refining motion from a sequence of rough poses (b) to add timing variation (c), coordination becomes visually confusing in low–level spline–based tools. Even this simple oscillating system with four degrees of freedom (a) quickly becomes disorganized when timing is refined.

represented, as animators need to break the temporal alignment of knots to create believable motion. In typical workflows, animators will quickly create poses and block them out in time to sketch out a motion (Figure 7.3b). The lack of explicit representation of timing refinement among the coordinated knots in each pose has a profound effect on animator workflows. From an interface standpoint, it is difficult to even locate coordinated extreme values, even for simple systems (Figure 7.3c). This challenge grows in complexity as the number of controls increases. In addition, adding this timing variation eliminates the presence of a representative extreme pose at any given time, causing the coordinated extreme values to no longer be editable as a single geometric entity. Staggered poses, by encoding explicit timing relationships among coordinated extreme values,

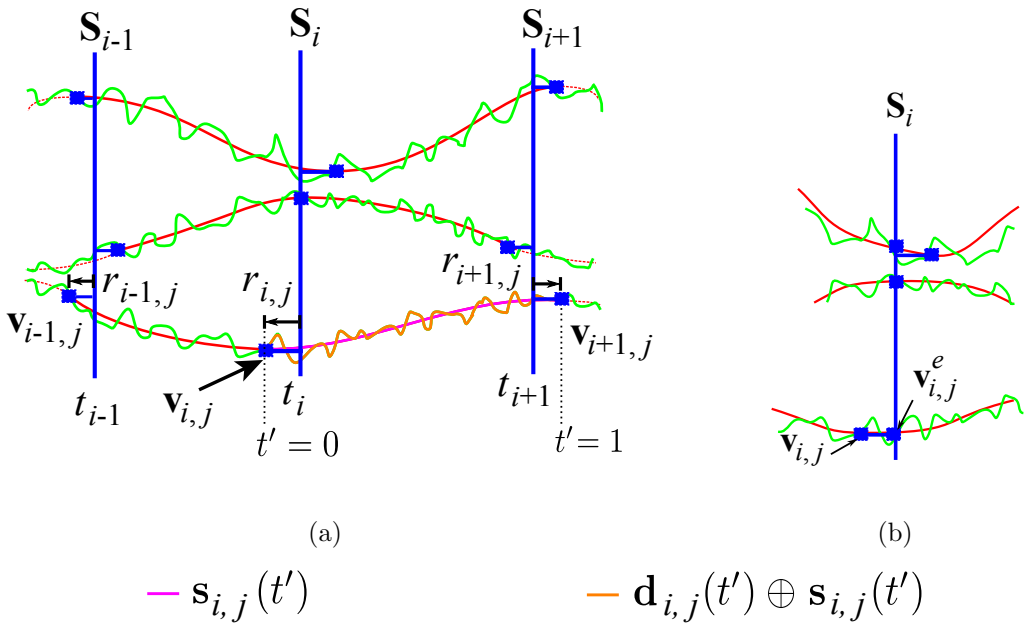


Figure 7.4: a) The staggered pose representation models transitions as a composition of splines (red) and displacement maps (green) to provide low–frequency editing controls that preserve detail. b) To facilitate direct geometric editing of poses, each staggered pose includes an additional knot at the pose time; this knot and the knot at the extreme value are mutually constrained to maintain continuity.

do allow for geometric pose–based editing of the coordinated extrema *after timing has been refined*, while retaining the mutual coordination among the knots that is important for representing believable transfer of motion from body part to body part.

As in keyframe animation, the staggered poses system use splines to interpolate between sequential extreme values and to provide concise user control over transitions from one extreme value to the next. These splines are defined over a normalized time domain between the refined times of each pair of sequential staggered poses (red, in Figure 7.4a). To support the representation of motion with arbitrary high–frequency detail, a displacement map represents an offset relative to the value interpolated by the spline (green, in Figure 7.4b). This facilitates the use of staggered poses as a high–level interface for edit-

ing coordinated extreme values in densely sampled data, such as recorded motion capture data. This also generalizes layered displacement maps [20, 179]; each temporal region between a pair of adjacent knots can be "interpolated" using a stack of concatenated splines and displacement maps. As interpolation does not exactly meet environmental contact constraints, a final displacement map represents an inverse kinematic solution that modifies the value to meet any such constraints[1]. To facilitate direct geometric pose editing in the presence of timing refinements, an additional knot is used at the central time of the pose (Figure 7.4b); this extra knot and the extreme value knot are mutually constrained during editing to maintain continuity.

7.1.1 Approach

The staggered poses editing system uses a custom motion representation that uses staggered poses as its foundation. Users can create motion from scratch, or they can work with recorded motion data that is uniformly and densely sampled. Pose–based editing is accomplished by making geometric changes to coordinated extrema as a single unit using existing forward kinematic or inverse kinematic posing tools. Users can edit coordinated pose extrema directly, which have slightly different times, or they can directly edit the true pose at the staggered pose time. The system mutually constrains extreme value knots and editing knots to maintain local continuity. For stylistic control of timing, tools are provided that apply procedural timing variation using a combination of random values to model asymmetric action and succession patterns to model a specific form of overlapping action. These stylistic timing controls are important, as animators often exaggerate such timing [175].

To support the editing of dense articulated character motion data in this keyframe–like setting, the system locates coordinated joint–centric extrema in human motion. It

[1]It remains as future work to extend the approach presented in Chapter 6 to preserve contacts when changing timing variation

combines joint–centric extrema associated with different body parts via temporal filtering to create a pose probability signal that indicates the presence of extreme poses. These probabilities are used to create staggered pose motion representations that include timing variation among coordinated joint–centric extrema, as well as splines and displacement maps for modeling transitions.

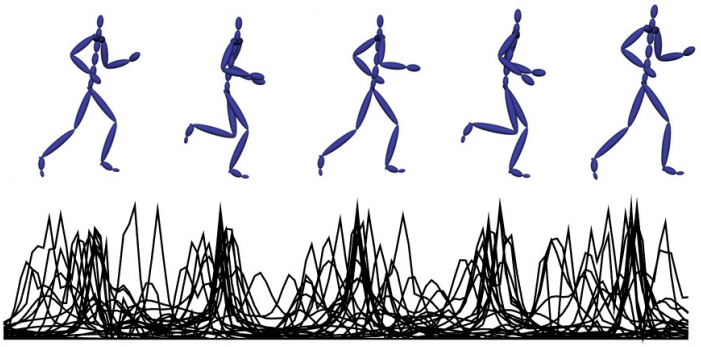


Figure 7.5: In recorded human motion, joint–centric motion extrema tend to cluster when different limbs move in a coordinated way.

7.1.2 Coordinated Joint–Centric Extrema in Human Motion

In many examples of human motion, joint–centric extrema tend to cluster in time. Figure 7.5 illustrates this using plots of the pseudo–curvature measure for each joint in a running motion[2]. Recall from Chapter 5 that the peaks of these plots indicate extreme joint configurations and can be used to identify joint–centric motion extrema. The coordination among the joint–centric extrema is apparent in the clustering of the peaks; the local spread among the peaks indicates timing variation among coordinated body parts. Character stance near the center of these clusters, overlaid in Figure 7.5, is similar to that in extreme poses used by animators. In practice, motion with significant coordination,

[2]Significant filtering of the motion was applied to account for the noise–sensitivity of the pseudo–curvature measure.

such as large intentional movement or response to physical contacts, exhibits the greatest degree of coordination among joint–centric extrema. Passive resting motion, such as ambient change in a static stance, has less coordination among joint–centric extrema.

To find staggered poses in human motion data, joint–centric extrema are first identified using any of the techniques presented in Chapter 5. These are then combined into staggered poses, in which the timing variation among the clustered joint extrema is explicitly represented as timing refinements in each staggered pose. As a result, *the staggered pose itself models a relationship among different joint–centric extrema.* Splines are then fit to the motion between each sequential pair of staggered poses, and approximation error not accounted for by the splines is stored in a motion displacement map.

7.1.3 Overview

The core contribution presented in this chapter is a new representation for motion that explicitly represents relative timing relationships among coordinated degrees of freedom (Section 7.2). The key advantage of this representation is that it enables new editing tools that allow for a sparse set of controls to be used for editing coordinated timing among different degrees of freedom; these tools are presented in Sections 7.3 and 7.4. First, Section 7.3 describes pose–based editing tools that allow the user to alter geometric aspects of motion without damaging either the temporal relationships among body parts or motion details. Second, Section 7.4 introduces temporal editing tools that alter the timing relationships among degrees of freedom as a coordinated unit, allowing for procedural control over some stylistic aspects of movement, such as asymmetric action and overlap.

In concurrent work, complex phase has been proposed an an alternative method for modeling overlapping action [66]. Section 7.5 compares the use of timing variation and phase variation, from both a mathematical perspective and using specific motion examples.

Section 7.6 presents an approach for creating staggered pose representations from existing motion, including recorded motion capture data. This approach uses joint–centric extrema to find central times for staggered poses, and the timing variation among the extrema is used to represent the timing refinements. Overall, these techniques allow for effective geometric editing of detailed motion, as they free the user from explicitly maintaining coordinated temporal relationships during pose–based editing, while enabling high–level temporal adjustments for stylistic control.

7.2 Representing Motion with Staggered Poses

The staggered pose motion representation models motion using a sparse sequence of key events; each of these events is modeled as a staggered pose that captures not only the values of the character's control variables, but also the local timing relationships among them. Effective geometric editing of the character's changing stance is facilitated by the sparseness of the staggered poses, while stylistic timing refinements are made by adjusting the timing refinements associated with the staggered pose.

A staggered pose can contain values for an arbitrary set of control variables. In the case of an articulated figure, these parameters can consist of the components of the different motion representations presented in Chapter 4. For the results presented in this chapter, the articulated skeletal motion representation is used.

Motion is represented as a sequence of staggered poses $\mathbf{S}_i = (t_i, \{(r_{i,j}, \mathbf{v}_{i,j})\}, \{\mathbf{v}^e_{i,j}\})$, where t_i is the *central time* associated with the staggered pose; these are illustrated in Figure 7.4a. For each staggered pose, each control variable j has an associated value $\mathbf{v}_{i,j}$ and *timing refinement* $r_{i,j}$. The equivalent knot in a traditional spline representation is $(t_i + r_{i,j}, \mathbf{v}_{i,j})$. Each refined knot can optionally have an associated *editing knot* value $\mathbf{v}^e_{i,j}$; these are used during interactive pose editing, which will be described below. For articulated character motion, each value $\mathbf{v}_{i,j}$ and editing knot value $\mathbf{v}^e_{i,j}$ generically represents

a position in space, direction vector, or quaternion rotation, as appropriate.

To support interpolation control while retaining the ability to model arbitrary motion detail, the staggered poses editing system uses local splines $\mathbf{s}_{i,j}(t')$ and displacement maps $\mathbf{d}_{i,j}(t')$ (see Figure 7.4a). These are defined on a local, normalized time domain between each pair of staggered poses; by convention $\mathbf{s}_{i,j}(t')$ and $\mathbf{d}_{i,j}(t')$ specify motion data that connects $\mathbf{S}_i$ to $\mathbf{S}_{i+1}$. As t' represents a local normalized time domain, it is given by the expression $t' = (t - (t_i + r_{i,j}))/((t_{i+1} + r_{i+1,j}) - (t_i + r_{i,j}))$. The key benefit of defining these splines and displacement maps with respect to the staggered poses is that *the central times and timing refinements of the staggered pose can be updated using editing tools without requiring explicit updates to the splines and displacement maps.*

To evaluate a control variable $\mathbf{v}_j(t)$ at time t, the system identifies the staggered poses $\mathbf{S}_i$ and $\mathbf{S}_{i+1}$ whose refined times $t_i + r_{i,j}$ and $t_{i+1} + r_{i+1,j}$ bound t. The normalized time t' is then computed and the value $\mathbf{d}_{i,j}(t') \oplus \mathbf{s}_{i,j}(t')$ is returned for $\mathbf{v}_j(t)$. This approach generalizes to multiple splines and displacement maps to support layered representations.

The static character state $\{\mathbf{v}_{i_j}\}$ obtained by using the extreme values for each variable in a pose is referred to as the *non–staggered pose*. The non–staggered pose will never actually exist in the motion, unless all of the timing refinements $r_{i,j}$ are zero. This is different from the *pose at the staggered pose time*, which is obtained by performing interpolation to compute the values for all variables at time t_i, resulting in the state $\{\mathbf{v}(t_i)\}$. Conceptually, a non–staggered pose often appears as an exaggerated version of the pose at the staggered pose time, as it brings all extrema into temporal correlation. The control variable values of the pose at the staggered pose time are generally less extreme than the values at the extrema, resulting in a less extreme appearance.

The editing knot values $\mathbf{v}^e_{i,j}$ are used to support direct editing of the pose at the staggered pose time; these are illustrated in Figure 7.4b . In addition to mutually constraining the relative values of the editing knot and the knot representing the extremum, the staggered poses editing system also constrains the intermediate spline tangents for

continuity. The editing tools include two approaches for geometric pose editing; each approach directly manipulates a different set of knots. Retiming tools change the timing refinements. Details of these tools are presented in Sections 7.3 and 7.4.

Spline interpolation of sparse poses cannot represent arbitrary detail. This detail is important when editing pre–existing motion or for the potential development of algorithms that create local displacement texture. In addition, motion constraints such as foot plants cannot be modeled using interpolation without a dense sampling of poses. The displacement maps are capable of capturing all these types of detail.

In the staggered poses system, constraints are labeled on end effectors with associated joint chains. The constraint enforcement displacement map has values for each control variable on which such constraints depend. These are nonzero for times during which a constraint is active. To avoid discontinuities, they also smoothly attenuate to zero in a small temporal window surrounding the constraint. The contents of the constraint–enforcement displacement map are determined using an IK solver, similarly to footskate correction [75]. Note that this approach has the same shortcomings as other approaches that enforce contact constraints after applying an edit, as described in Chapter 6. It remains as future work to implement the approach described in Chapter 6 within the staggered poses system.

7.3 Geometric Pose Edits with Staggered Poses

The geometric pose editing tools allow users to manipulate coordinated motion extrema with timing variation as a single unit. By changing the staggered poses that the character passes through, these tools can be used, for example, to change the style of a character's stance, position the limbs in new configurations, or scale the magnitude of a movement. The staggered poses editing system provides two approaches for editing coordinated extrema; direct editing of the parameter values at the central time of the staggered pose

and direct editing of all extreme values via the non–staggered pose.

The first approach to geometric editing allows users to edit the pose at the staggered pose time. The editing knots $\mathbf{v}^e_{i,j}$ represent the state of the character at time t_i, and by mutually constraining them to the extreme value knots $\mathbf{v}_{i,j}$, editing changes are applied to the extreme knots as equal changes. This constraint avoids the introduction of short, large accelerations that could occur if two knots close together in time differ significantly in value. To enforce this constraint, the updated extreme value knot is set to

$$\hat{\mathbf{v}}_{i,j} = (\hat{\mathbf{v}}^e_{i,j} \ominus \mathbf{v}^e_{i,j}) \oplus \mathbf{v}_{i,j},$$

given the new value $\hat{\mathbf{v}}^e_{i,j}$ of the editing knot. This is illustrated in Figure 7.6a.

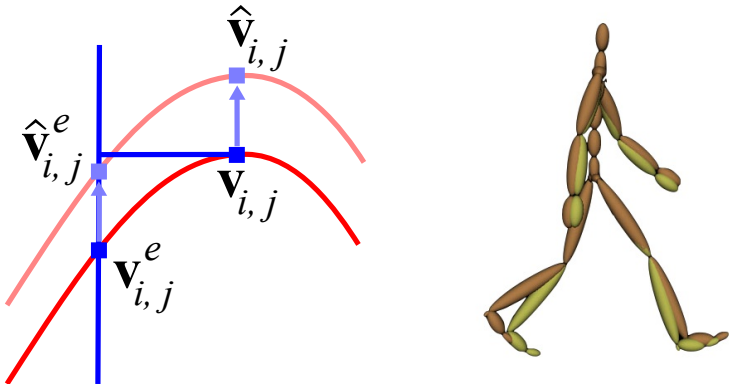
The second approach to geometric editing is possible with or without the presence of the editing knots $\mathbf{v}^e_{i,j}$. To directly edit all extreme values, the staggered poses editing system temporarily collapses all timing refinements to zero, allows the user to edit the non–staggered pose, and then restores the timing refinements. The non–staggered pose typically appears as a slightly exaggerated version of the true pose at the staggered pose time, as any parameters with timing refinements will usually have more extreme values. This is shown in Figure 7.6b. The yellow pose is the true pose that the character passes through at the central time of the staggered pose. The orange pose is the non–staggered pose, which has all extreme values collapsed to a static state for editing.

When editing knots exist, the stagered poses editing system offsets their values as

$$\hat{\mathbf{v}}^e_{i,j} = (\hat{\mathbf{v}}_{i,j} \ominus \mathbf{v}_{i,j}) \oplus \mathbf{v}^e_{i,j}$$

to maintain the mutual value constraint. This approach is useful when coordinated extreme values have meaningful bounds. For example, adjusting a joint near its rotation limit is easier when the extreme value knot is manipulated, as users can directly manipulate the state of the character when the joints are all in an extreme configuration.

As many edits will push poses to more extreme or less extreme values, the staggered poses editing system provides the ability to edit one pose relative to another pose. This



(a) (b)

Figure 7.6: Editing knots and extreme value knots are mutually constrained during interactive editing to avoid the introduction of sudden accelerations. Any change in value is copied from one to the other (a). The non–staggered pose, shown in orange, appears exaggerated relative to the true pose at the staggered pose time, which is shown in yellow (b).

can be any pose of the user's choosing, such as a neighboring pose for local exaggeration or an average pose for global exaggeration. The user has control of a weight value w_r that specifies the degree of offset; the system updates each value as[3]

$$\hat{\mathbf{v}}_{i,j} = w_r(\mathbf{v}_{i,j} \ominus \mathbf{v}^r_j) \oplus \mathbf{v}_{i,j},$$

given the relative pose values $\mathbf{v}^r_j$. As this operates on arbitrary knot values, it can be applied to either the pose at the staggered pose time or the non–staggered pose.

If users create or delete poses, the system updates the splines and displacement maps to maintain the locally normalized time domains. If done straightforwardly, knot removal would change the motion, as a change in the knots of the spline will change the interpolation evaluation. To compensate for this, the system updates the displacement map to preserve the overall values of the motion as the character moves from one pose to the next.

[3]For orientations, premultiplication by w_r represents a rotation scale rather than component–wise quaternion multiplication.

7.4 Procedural Editing of Timing in Staggered Poses

The temporal editing tools of the staggered poses editing system allow users to specify how timing varies among different body parts. The simplest form of temporal editing is retiming each staggered pose as a coordinated unit; this simply involves updating t_i, the central time of the staggered pose. This can be effective for changing perceived energy or weight, as it is similar to how animators set timing for traditional pose representations. As timing refinements, splines, and displacement maps are defined relative to the central time of the staggered poses, no other explicit change to the motion representation is needed.

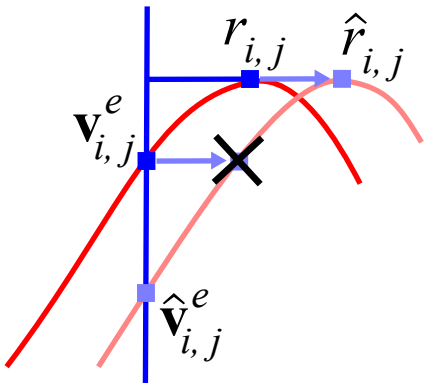


Figure 7.7: Changing a timing refinement requires spline evaluation to change the value of an editing knot, should one be present.

The staggered poses editing system provides two tools for applying *procedural* staggered timing edits; one applies succession–based offsets, as a specific form of overlap, and the other applies random offsets to model asymmetric action. As originally presented by Neff and Fiume [125], succession follows an incremental profile down the limb. Let each control parameter j have a depth d_j in the character hierarchy, relative to some user–selected joint. Given a base timing offset s, the succession tool applies the edit by updating the refinements of the staggered pose, for each joint below the user–selected

joint, using

$$\hat{r}_{i,j} = r_{i,j} + sd_j,$$

where $\hat{r}_{i,j}$ denotes the updated timing refinement.

The staggered poses editing system generalizes this idea to create a new nonlinear succession pattern that is capable of modeling a greater variety of stylistic movement. This generalized update is

$$\hat{r}_{i,j} = r_{i,j} + sd_i^l,$$

where l controls how nonlinear the timing is. For $l = 1$, this is the incremental succession of Neff and Fiume. Increasing l causes a limb to appear less constrained, with inertia having a greater apparent effect on the movement; this creates a cartoony feel to the timing.

For random offsets, the system applies timing refinement updates using

$$\hat{r}_{i,j} = r_{i,j} + rand(-\tau, \tau);$$

τ is a parameter for bounding the random timing. Random offsets are applied to model asymmetric timing among different limbs. The system allows users to select a set of joints and apply the effect only to those joints. Succession and random timing variation can also be combined to both stagger the timing of different limbs relative to each other and to apply a succession pattern down each limb.

As with any retiming algorithm, care must be taken to avoid large nonlinear time warps, as this can introduce sudden acceleration. The system warns users when timing refinements are set large enough to be closer to a neighboring staggered pose than the staggered pose with which the knot is associated.

When timing refinements are changed and editing knots are present, the system updates the editing knots to avoid sudden accelerations. As shown in Figure 7.7, the system first moves the editing knot to the global time $t_i + (\hat{r}_{i,j} - r_{i,j})$. This, however, is not an appropriate time for direct editing of pose value; the system must create an appropriate

editing knot at time t_i. To do this, the system evaluate the splines and displacement maps at time t_i to determine the new editing knot value $\hat{\mathbf{v}}^e_{i,j}$. It then replaces the existing editing knot with the new editing knot at time t_i with value $\hat{\mathbf{v}}^e_{i,j}$. Similarly, if an editing tool requires a new editing knot to be created, the system inserts it at t_i by evaluating splines and displacement maps. In each case, the system updates the displacement map to compensate for changes in spline interpolation.

7.5 Staggered Timing and Staggered Phase

The staggered pose representation models timing variation using time domain offsets that are specified relative to the central pose time. In concurrent work, the phase of complex–valued representations of motion has been used to model overlap in oscillatory motion [66]. This section compares the use of time delays and phase delays, considering how they affect a sampled motion signal and demonstrating how the two approaches change the quality of movement for a set of simple motion examples.

7.5.1 The Effect of Phase Delays on Timing

To apply a phase delay to a motion signal $f(t)$, a complex representation of the signal, $z(t)$, can be constructed by using the Hilbert transform to generate the imaginary component of the signal [66]. To apply a phase delay by a phase angle θ, each entry of the signal can be multiplied by $z_\theta = e^{\mathbf{i}\theta}$, resulting in the phase–delayed signal $z'(t) = z_\theta z(t)$.

In complex arithmetic, multiplication by a unit length complex number is equivalent to rotation in the complex plane. As z_θ is unit length, applying a phase delay is equivalent to rotating the original complex signal $z(t)$ in the complex plane. Specifically, the application of a phase delay can be written as

$$\begin{bmatrix} \Re(z'(t)) \\ \Im(z'(t)) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \Re(z(t)) \\ \Im(z(t)) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} f(t) \\ H(f(t)) \end{bmatrix},$$

where $H(f(t))$ represents the Hilbert transform of $f(t)$. The phase–delayed motion, $f'(t)$, is represented in the real component of $z'(t)$, which is

$$f'(t) = \cos(\theta)f(t) - \sin(\theta)H(f(t)). \tag{7.1}$$

Therefore, application of *a phase delay attenuates each frequency component by* $\cos(\theta)$ *and adds a contribution of the Hilbert transform of the signal, attenuated by* $\sin(\theta)$.

For pure oscillatory motion, i.e. a sinusoid, the phase shift represents a phase shift in the traditional sense, and the shape (value vs. time plot) of the motion will be the same, but shifted in time. However, *for non–oscillatory motion, each frequency band will be shifted by a different amount*, which can change the shape of the motion. This effect will be demonstrated for different types of movement in the next section.

In terms of equation 7.1, the complex representation of a sinusoidal motion will trace a path on a circle in the complex plane whose radius is the amplitude of the sinusoid, and Equation 7.1 will remain sinusoidal after rotation. However, for a general signal, $H(f(t))$ can have a dramatically different shape, and since Equation 7.1 applies a portion of $H(f(t))$ to $f(t)$, the final shape of the motion will, in general, be different.

7.5.2 Examples

To explicitly compare the use of timing and complex phase for modeling timing variation, a three–link pendulum with coordinated motion will be modified using both time–based and complex phased–based delays. Both smooth motion and non–smooth motion will be considered, as phase–based delays modify each type of motion in a different way. For smooth motion, the initial motion is sinusoidal. For non–smooth motion, the initial motion is piecewise linear.

The extreme poses of the pendulum motion, along with a semi–transparent frame between them, is shown in Figure 7.8. The motion has a period of 200 frames, and it is sampled at a rate of 120 frames per second. For the sinusoidal version of the motion,

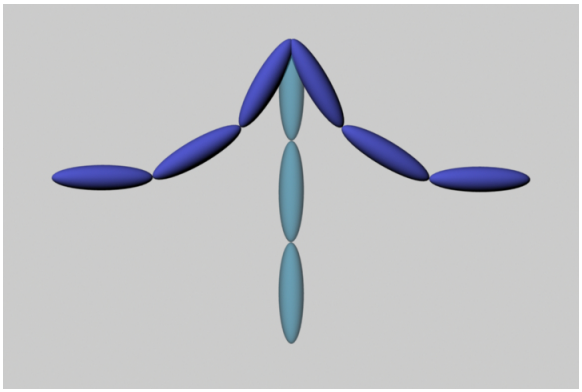


Figure 7.8: The extreme poses, along with a semi–transparent in–between pose, of the pendulum motion used to compare timing and phase delays.

the extreme poses represent the times when the rotation parameters are at their extreme values. For the piecewise linear version of the motion, each transition from one pose to the next is linearly interpolated using joint angles. To apply phase delays, the Hilbert transform implementation in the *SciPy*[4] scientific computing library is used.

In Figures 7.9 and 7.10, the three plots in each image show the values of the rotation parameters over the duration of the motion. The top plot corresponds to the top link of the pendulum, the middle plot to the middle link, and the bottom plot to the lower link. When time or phase delays are applied, the amount of delay is multiplied by the depth of the link in the hierarchy.

Figure 7.9a shows the initial sinusoidal pendulum motion plots. Figure 7.9b shows the same motion, with a time delay of fifteen frames applied, and Figure 7.9c shows the motion with a time delay of thirty frames. Figure 7.9d shows the result from applying a phase delay of 0.5 radians, which is qualitatively similar to the fifteen frame time delay. Figure 7.9e shows the motion with a phase delay of 1.0 radians applied; this is qualitatively similar to the time delay of thirty frames.

Note that the phase delays, in this case, preserve the shape of the motion plots,

[4]http://www.scipy.org

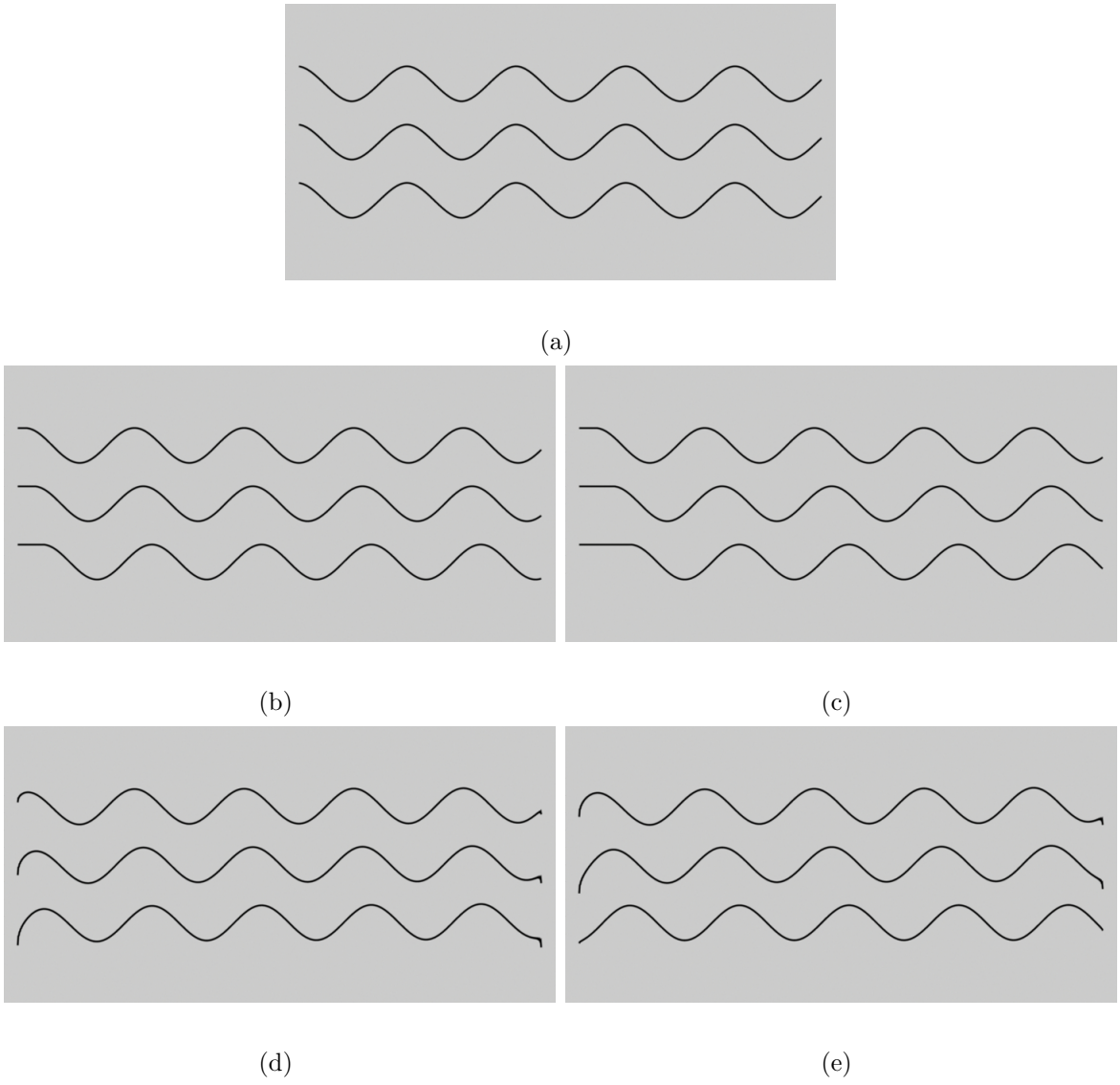
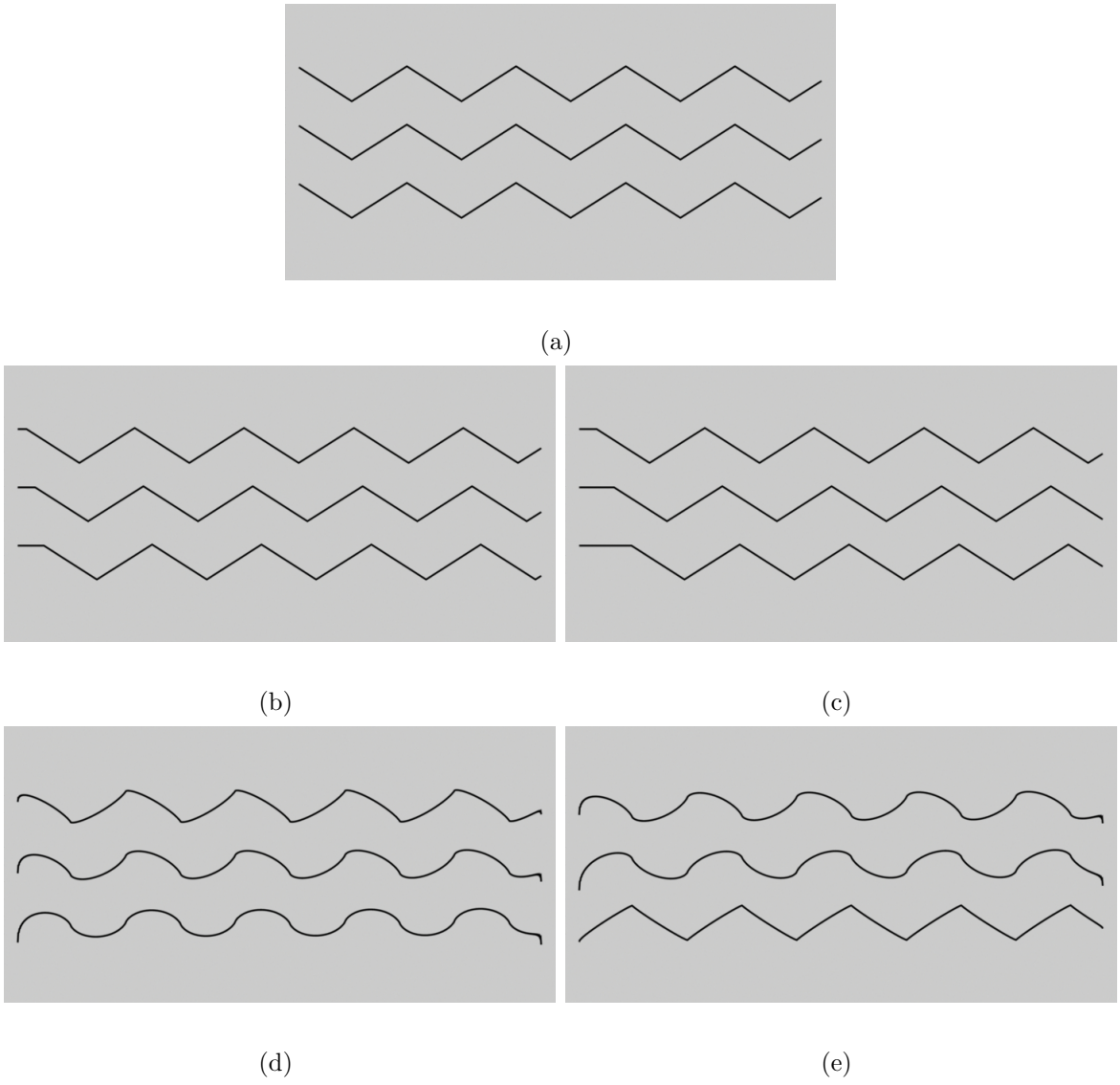


Figure 7.9: Time and phase delays applied in a succession pattern to the rotation signals of the sinusoidal pendulum motion. Original motion (a), time delay of 15 frames (b), time delay of 30 frames (c), phase delay of 0.5 radians (d), phase delay of 1.0 radians (e).



(a)

(b)

(c)

(d)

(e)

Figure 7.10: Time and phase delays applied in a succession pattern to the rotation signals of the linear pendulum motion. Original motion (a), time delay of 15 frames (b), time delay of 30 frames (c), phase delay of 0.5 radians (d), phase delay of 1.0 radians (e).

keeping them similar to the shape of the time delays, with the exception of the beginning and end of the motion. At both the beginning and end, the signals are distorted in shape, as the time–limited motion is not strictly periodic. This, in effect, causes the pendulum to both start and finish with faster motion that it has in the initial motion. Time delays, in contrast, hold the pendulum still at the beginning, as the implementation reuses the first frame's value to pad the signal.

The same succession edits are applied to the linear pendulum motion, and the results are shown in Figure 7.10. In this case, note that the phase delay significantly changes the shape of the rotation signals, as each frequency component is modified individually. In Figure 7.10e, the lower motion appears to be close to piecewise linear, but out of phase with respect to the original motion. This is due the phase delay being close to π (3.0), and the frequency components are approximately aligned, but out of phase.

Overall, the following observations hold: Neither time delays nor phase delays handle the beginning and end of the motion well, in general, due to incomplete context information. Phase delays create sudden accelerations near the beginning and end if the motion is not strictly periodic. Phase delays work well when motion is close to an oscillatory sinusoid, and, in this case, the shape of the motion plots will be approximately preserved. This is not true when the motion is far from sinusoidal, as the shape of the motion plots can be significantly changed.

7.6 Staggered Poses in Dense Motion Data

To support the use of staggered poses for pose–centric editing of existing motion data, an algorithm has been developed that locates staggered poses and creates a staggered pose motion representation. This algorithm takes a top–down approach, first locating the staggered poses, then fitting splines, and finally measuring displacement maps that capture remaining motion detail. As the degree of editing fidelity needed can vary based

on the particular editing problem, the algorithm prioritizes the staggered poses using a heuristic measure and allow users to optionally cull the set of poses to produce fewer editing controls. Figure 7.5 includes poses at the times chosen using this algorithm, which correspond to large–scale changes in movement due to contact and release with the ground.

7.6.1 Determining Pose Times and Refinements

To locate a sequence of prioritized candidate pose times, the staggered pose identification algorithm explicitly looks for coordinated extrema and heuristically prioritizes the resulting staggered poses. The likelihood of a staggered pose existing at any given frame is modeled probabilistically to allow for the incorporation of different measures, each of which is expressed as a probability $p_i(t)$. A weighted combination of these probabilities is used to determine the central times of the staggered poses, and these priorities are biased, based in part on how much each limb moves. A low–pass filter f is used to combine probability peaks that are close in time; the result is referred to as the aggregate *pose probability signal* $P(t)$. Global evidence measures, such as changes in labeled constraints, can be included as additional probability signals, as these often indicate the likely presence of a pose.

If motion extrema identification measures, such as those presented in Chapter 5, are used as probability signals, the maxima of the pose probability signal $P(t)$ represent strong evidence for correlated extrema at those times. Therefore, the maxima of $P(t)$ indicate a good times for a staggered pose to be created. The peaks of individual pose probabilities $p_i(t)$ are indicative of joint extrema times. Temporal differences between the joint extrema times and the central time of the staggered pose lead directly to timing refinements. To prioritize the staggered poses, the system biases them by both $P(t)$ and the distance to neighboring maxima of $P(t)$. These ideas are summarized in Figure 7.11.

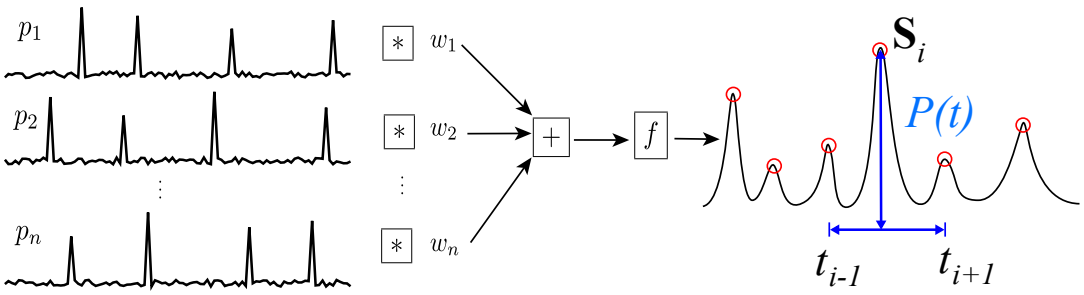


Figure 7.11: To locate central staggered pose times, signals that measure motion extrema are combined, the result is filtered to combine close peaks into an overall probability, and maxima of the resulting measure are selected. The staggered poses that occur at these maxima are prioritized by both overall probability and temporal locality.

Local Evidence Measures:

The presence of motion extrema on individual joints can be used as evidence that a staggered pose should be located nearby. By aggregating the measures used to identify joint–centric motion extrema (Chapter 5), the times when joint–centric motion extrema tend to cluster can be found.

To do this, any of the joint–centric measures presented in Chapter 5 can be used, provided it is normalized to create a valid probability. In the examples presented in this chapter, the pseudo–curvature measure is used, and the methods presented in Chapter 5 are used to associate joint motion with parent joint rotation. To address the noise sensitivity of the pseudo–curvature measure, iterated Laplacian filtering is used to smooth the joint motion.[5] The pseudo–curvature measure associated with each joint is then used as a probability signal when determining $P(t)$. The motion extrema that are selected from the measure are used to find timing refinements for each joint.

[5]The pseudo–curvature measure was chosen for historical reasons, as the other techniques presented in Chapter 5 had not been developed at the time the staggered poses system was developed.

Global Evidence Measures:

Change in environmental contact constraints can be used as additional evidence for the presence of a staggered pose. The start and end times of constraints, such as the moment of contact or release, are often indicative of extreme poses in human motion. At these times, the discontinuous change in contact often has a corresponding acceleration that propagates through the body. This is modeled as a binary probability signal that is one only at times that constraints become active or inactive.

Aggregating Evidence:

The local and global probability metrics are used to create the aggregate pose probability signal by taking a weighted sum of probability signals and filtering. Weighting the results positively biases signals associated with limbs that are moving most significantly in space. Applying a low–pass filter allows peaks of probability to positively bias the overall probability of adjacent frames; this has the effect of combining closely–clustered peaks of overall probability into a single peak. Given a set of pose probability measures $p_i(t)$, the overall pose probability is

$$P(t) = f\left(\frac{\sum_i w_i p_i(t)}{\sum_i w_i}\right),$$

where f is a low–pass filter. A tent filter with a filter width of 0.15 seconds is used for the low–pass filter.

Weight values are set using both limb length and motion magnitude. The expression for this is

$$w_i = \sum_j |\mathbf{t}_i||\mathbf{m}_j|.$$

In this expression, $\mathbf{t}_j$ is the joint position in the articulated skeletal motion representation. $|\mathbf{m}_j|$ represents the arc length of the joint trajectory $\mathbf{m}_j$. For constraints, w_i is set to a nonzero value for motion in which change in constraint state is indicative of extreme poses.

Prioritizing the Staggered Poses:

Candidate staggered poses are created at all local maxima of the pose probability signal. Users have the option of interactively culling this set of staggered poses, which are prioritized using the probability signal. To favor both regular temporal sampling and poses with a high probability, a weight $w(\mathbf{S}_i)$ is defined for each staggered pose $\mathbf{S}_i$. This weight is

$$w(\mathbf{S}_i) = P(t_i)(t_{i+1} - t_{i-1});$$

the second half of the term positively biases staggered poses that have more distant neighbors, as they are isolated more in time. The staggered poses are sorted using $w(\mathbf{S}_i)$, and the highest–weighted percentage are displayed to the user. Users adjust a slider to select a desired number of high priority poses or specify a time interval of interest from which to choose the staggered poses. Finally, they can individually select staggered poses from the candidate set.

Timing Refinements:

Timing refinements are set by searching for joint–centric extrema that occur near the central time of the staggered pose. Only those joint–centric extrema that are closer to a given staggered pose than the neighboring staggered poses are considered for inclusion. If no joint–centric extreme exists, the timing refinements are set to zero.

7.6.2 Fitting Splines

Given a sequence of staggered poses $\mathbf{S}_i$, cubic splines are fit to the motion samples by minimizing squared distance for translation and squared angular distance on S^3 for rotation. The first problem has a least–squares linear solution. As cubic quaternion splines defined using recursive spherical linear interpolation are used for rotation, the second problem requires a nonlinear optimization (recursive spherical linear interpolation) with nonlin-

ear constraints (unit length quaternions). This problem is solved as an unconstrained optimization that models the constraints using an additional objective function term. A general—purpose nonlinear optimization solver is then used to find the intermediate quaternion rotations.

Given a sequence of quaternion orientations $\mathbf{q}_i$ at corresponding times t_i, where $t_1 = 0$ and $t_n = 1$, the solver fits a cubic orientation spline $\mathbf{q}(t)$, where cubic interpolation is defined using recursive spherical linear interpolation [156]. Denote this orientation spline as $\mathbf{q}(t) = \mathbf{s}(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3, \hat{\mathbf{q}}_4, t)$ for "control point" orientations $\hat{\mathbf{q}}_1$, $\hat{\mathbf{q}}_2$, $\hat{\mathbf{q}}_3$, and $\hat{\mathbf{q}}_4$. For endpoint interpolating splines, $\hat{\mathbf{q}}_1 = \mathbf{q}_1$, $\hat{\mathbf{q}}_4 = \mathbf{q}_n$, and $\hat{\mathbf{q}}_2$ and $\hat{\mathbf{q}}_3$, the "tangent" orientations, are free parameters. The optimization problem is

$$f(\hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3) = w_f f_f(\hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3) + w_c f_c(\hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3).$$

The first term, f_f, adjusts the tangent orientations to fit the spline to the data samples by minimizing the average squared angular difference on S^3:

$$f_f(\hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3) = \frac{1}{n-2} \sum_{k=2}^{n-1} cos^{-1}(\mathbf{q}_k \cdot \mathbf{s}(\mathbf{q}_1, \hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3, \mathbf{q}_n, t_k))^2.$$

The second term models the unit length constraint on quaternions that represent orientations. This is

$$f_c(\hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3) = (\|\hat{\mathbf{q}}_2\| - 1)^2 + (\|\hat{\mathbf{q}}_3\| - 1)^2.$$

The system minimizes $f(\hat{\mathbf{q}}_2, \hat{\mathbf{q}}_3)$ using the Nelder–Mead downhill simplex method [140]. Linear rotation splines are used as the initial solution. This optimization will generally not reach zero, and some error will be contributed by the second term. To account for this, $\hat{\mathbf{q}}_2$ and $\hat{\mathbf{q}}_3$ are then normalized. In the staggered poses system, $w_f = 1000 w_c$, as this relationship tends to produce quaternions that are close to unity in magnitude while resulting in splines that approximate the data well.

7.6.3 Displacement Maps

Given an interpolated spline value $\mathbf{s}_{i,j}(t')$ at normalized time t', as well as the actual value $\mathbf{v}_{i,j}(t')$, the value of the displacement map is simply

$$\mathbf{d}_{i,j}(t') = \mathbf{v}_{i,j}(t') \ominus \mathbf{s}_{i,j}(t').$$

7.7 Examples

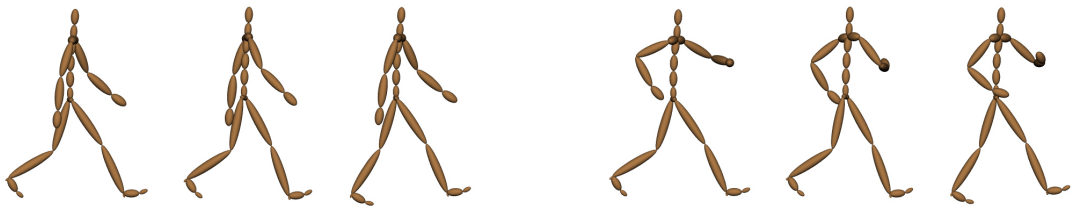
The staggered poses motion representation and associated motion editing tools have been implemented within the commercial animation system *Maya*. In comparison to pose–based editing, staggered poses require fewer poses to capture important aspects of timing variation, which allows users to work with fewer geometric controls when it is important to have direct editing control over the extreme configurations of each joint. In comparison to the use of independent splines, staggered poses can be retimed and changed in geometric value without requiring users to explicitly manage coordinated timing relationships that are only implicitly represented in a set of splines.

The staggered poses editing tools have been used to edit a number of motions, primarily to introduce stylistic effects, such as pose exaggeration and successive timing. An example edit of a walking motion is demonstrated in Figures 7.12 and 7.13. In this example, geometric edits are applied to change how the arms are held in the non–staggered poses. Nonlinear succession is then applied to loosen up the arm movement. Subtleties of transition are carried along in the retimed splines and displacement maps.

Figure 7.14 includes frames from a lifting and carrying motion. This motion was edited to increase the apparent effort exerted by the character. The bend of the back has been changed, and the arms are held higher. A subtle succession–based time delay pattern has been introduced to loosen up the walking portion of the motion

In Figure 7.14, the jumping motion has been edited to exaggerate the depth of the jump. In addition, successive timing has been added to loosen up the motion, giving the

character an appearance of feeling subtly less constrained.



(a) (b)

Figure 7.12: The staggered poses selected for editing (a) and the staggered poses after editing (b).

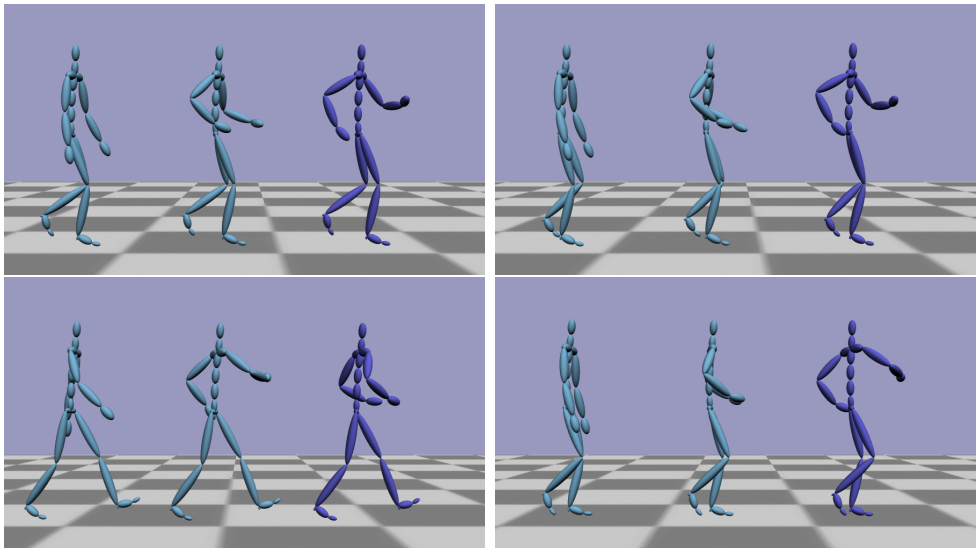


Figure 7.13: A walk (blue, left) is edited to change the extreme poses (blue, middle), and succession is then applied to create a cartoony feel to the movement (purple, right).

7.8 Discussion

An immediate question is how much effect the use of joint–centric motion extrema has on editing in comparison to general locations in time. Motion extreme–centric control is important when specific control is needed, as position and timing of a motion extreme

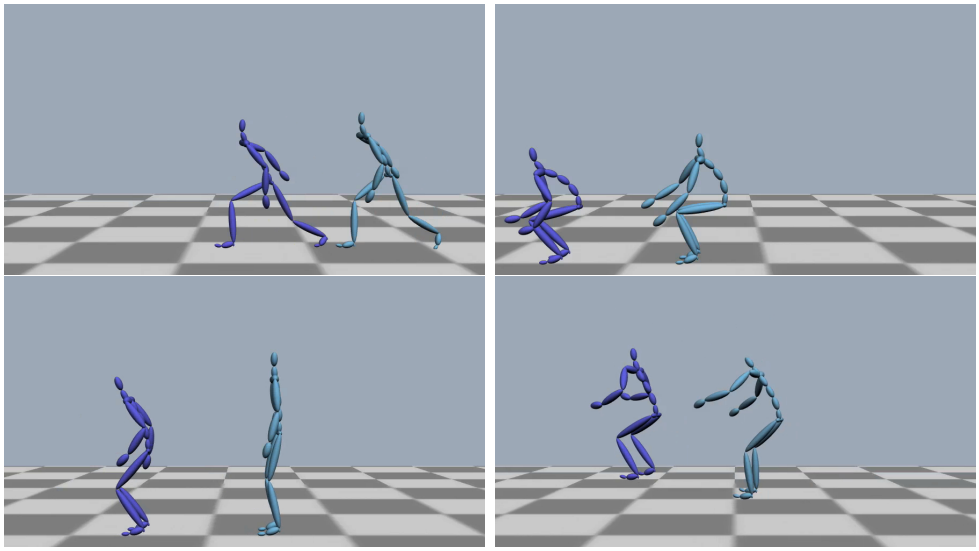


Figure 7.14: A jump (blue) is edited to increase the depth of the jump, and succession timing is added to create a looser feel to the motion.

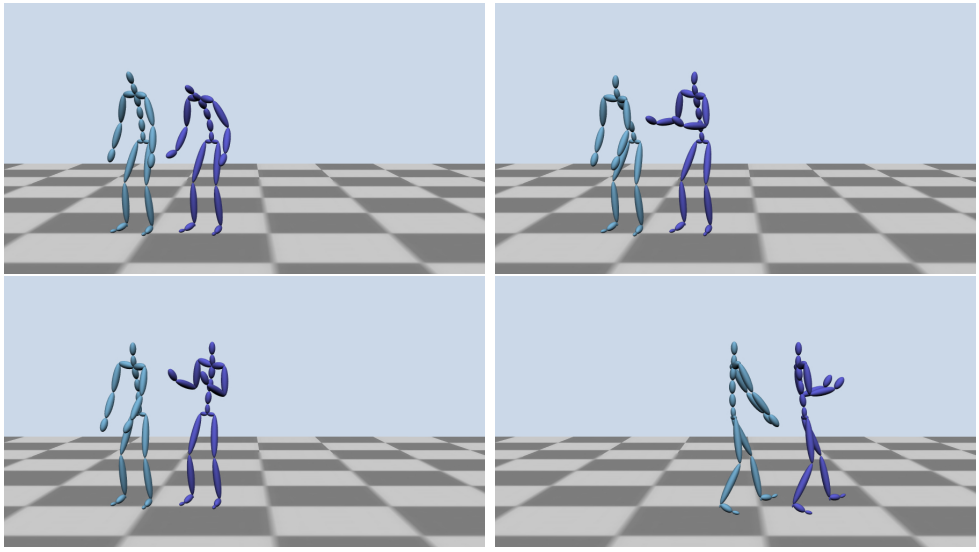


Figure 7.15: The motion of a character lifting and carrying an object (blue) is edited to increase the apparent effort exerted in carrying the object (purple).

have a direct effect on how we identify important events in a motion [182]. In addition, users must often be cognizant of extreme values, as they often reach believable limits of body shape for a given motion. As was illustrated in Chapter 1, editing at points in time away from motion extrema provides only indirect control over how much a part of the body moves.

Staggered poses, by explicitly representing the relative timing relationships among coordinated motion extrema, provide concise direct control over the extreme configurations of different body parts. In comparison to motion editing with layers of splines and displacement maps, the staggered poses representation provides all the power of layered editing, but with the addition of sparse extreme–based controls. Layered editing approaches also require users to specify temporal layering bounds; by smoothly adjusting all values with respect to neighboring motion extrema, the staggered poses system does not impose such a requirement.

When applying temporal edits to recorded human motion, care must be taken during motion processing to avoid artifacts associated with correlated noise in topologically adjacent joints. Many motion capture pipelines filter marker positions and fit skeletons to these filtered signals. If sequential joints are nearly linear, as is often the case with the back, fitting algorithms can create small noise contributions to the rotations that can have a perceptual effect on the overall body motion. However, the noise on one joint will correlate with the noise on child joints to approximately cancel out its effect on descendent joints, and the effect is usually not perceptible. De–correlating these signals in time removes this cancelation, and short high–frequency artifacts might appear in descendent joints. Appropriate filtering of rotation data before editing can be used to avoid this problem [93].

7.9 Future Work and Conclusion

This chapter has introduced staggered poses, a new motion representation that explicitly encodes timing variation among coordinated degrees of freedom. This allows users to make sparse geometric changes to coordinated extreme values with timing variation and to apply procedural timing variation, while preserving low–level detail of movement. One key benefit of this approach is that pose–centric editing can be applied to refined motion without introducing new undesired movement extrema and without requiring users to manage coordination in timing variation among many degrees of freedom. A second benefit is that common refinements to timing can be procedurally applied, rather than manually specified through low–level knot manipulation. The applicability of the staggered pose motion representation to making high–level geometric changes or timing changes to coordinated extreme values has been demonstrated using recorded motion capture data. To facilitate this, an algorithm has been introduced that locates likely extreme poses with timing variation by searching for coordinated motion extrema. While this algorithm could be used to find extreme poses alone, this is equivalent to using an aggregate measure on the full body, as was presented in Chapter 5, with low–pass filtering.

Going forward, it would be useful to investigate how staggered pose representations can be integrated with pose control graphs and motion graphs as concise descriptions of dense motion data. While the staggered poses system has been implemented within the commercial animation system *Maya*, the current interface does not yet support interactive editing of low–level components such as splines and displacement maps. A additional area for interface development would be the implementation of interfaces that assist users with understanding how and why specific joint–centric extrema get associated with a particular staggered pose. Finally, it would be interesting to investigate how coordinated timing variation can be used to refine or analyze the motion of simulated deformable objects or facial models.

Chapter 8

Spatial Exaggeration of Articulated Character Motion

This chapter introduces a form of procedural motion stylization that is based on the exaggeration of extreme poses. This technique changes a motion such that the extreme poses, or extreme configurations of individual joints, are more distinct with respect to each other. As the technique uses these extreme poses or extreme joint configurations as a foundation for applying exaggeration, the spatial extremeness measure presented in Chapter 5 is used to find them as motion extrema in existing motion data. When exaggerating a motion, the extreme poses or extreme joint configurations are geometrically altered to change their distance to each other with respect to a stance–centric, Cartesian distance metric. This results in an overall approach to procedural motion stylization that produces a motion that is similar to the original, but spatially exaggerated or more subtle. This chapter demonstrates the approach being applied to the body motion as a whole, when full body poses are used, and to the motion of each joint individually, when extreme joint configurations are used.

8.1 Motivation and Overview

Animators create visually compelling character motion by identifying and exaggerating important visual qualities of movement. Graphically significant aspects of movement can be concisely presented using *extreme poses*—poses in which the body is in a geometrically extreme configuration [177]. As such, animators often use extreme poses to create an initial pass at a motion, and the geometric relationships between adjacent extreme poses play a key role in determining the extent to which a character moves while carrying out an action. In addition, the relative distance among body parts, which is maximal at extreme poses, has been shown to be perceptually significant in terms of how we perceive continuous human motion as a sequence of events [183]. Despite the importance of extreme poses in representing the graphical aspects of movement in a concise, meaningful way, existing approaches to editing motion style do not take them into account.

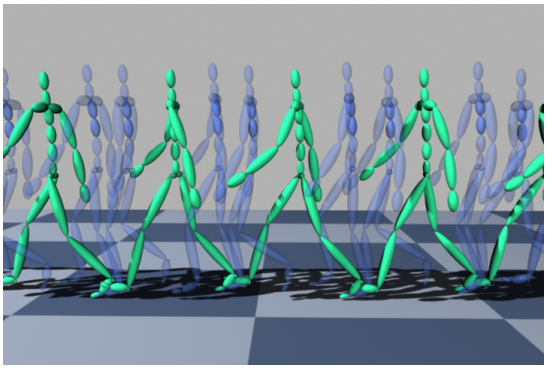
Motion style can be described in a variety of ways; animators often describe it in terms of visual terminology and the classical animation principles [167, 84]. A key aspect of creating stylized animated movement is the idea of exaggeration—choosing some aspect of the character's form or movement and making it read stronger. Key to effective exaggeration is choosing important aspects of form or movement and knowing how to effectively change them. For example, timing, geometric shape, and the overlap of different actions can each be exaggerated to achieve different effects. This chapter focuses on the exaggeration of one specific aspect of motion—geometric stance—using extreme poses of the full body or extreme configurations of each joint as an algorithmic foundation.

The Laban Movement Analysis (LMA) community has also developed a detailed system for describing aesthetic qualities of movement [81]. Key to the LMA approach is breaking down motion into specific qualities such as *effort*, *shape*, and *phrasing*, which describe either specific qualities of stance and timing or qualities about how they relate to each other. Spatial exaggeration is grounded in the same notion that *specific aspects of motion can be independently described*, and the spatial exaggeration algorithm modi-

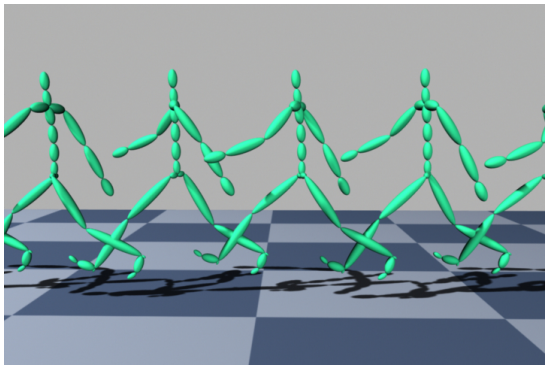
fies the LMA notion of *space*—how much space around the character is occupied while carrying out a movement. Spatial exaggeration also relates to the LMA concept of the *kinesphere*. The kinesphere represents the size and shape of the region of space around a character in which they move while carrying out an action. The exaggeration of extreme poses effectively increases the size of the character's kinesphere.

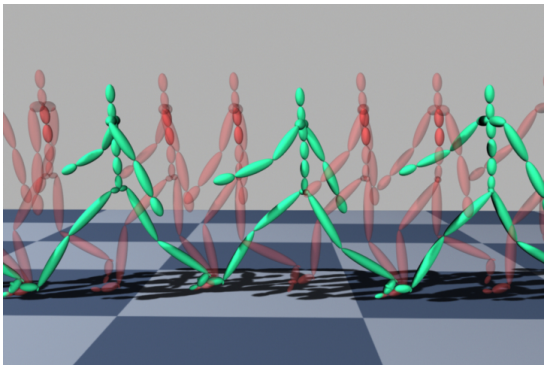
While existing stylization algorithms can incorporate quantitative style models [18, 56] or apply procedural stylistic changes [104, 173], the descriptive models used by animators and the LMA community indicate that style can be modeled in terms of specific qualitative aspects of movement; this idea has been investigated for user control over style when synthesizing new motion [27, 125]. One of the most significant visual aspects of style is how much the body moves in space with respect to itself; in this chapter, I refer to this as *spatial extent*. If the extreme poses of a motion are significantly different in terms of geometric stance, the motion will involve a great deal of local body movement—movement that is independent of how the character changes location in the environment. Such a motion has more spatial extent than a motion in which the extreme poses are less distinct. Spatial exaggeration, by making extreme poses more distinct with respect to each other, increases the spatial extent of the motion. The interdependency between the geometric relationship among extreme poses and how much the body moves further motivates their use as a foundation for spatial exaggeration.

To apply spatial exaggeration, extreme poses or extreme joint configurations must be chosen. To do this, maxima of spatial extremeness are used to select extreme poses or joint–centric motion extrema, as these predictably correlate to extreme poses and extreme joint configurations, respectively (Chapter 5). To exaggerate the motion, the selected extreme poses or extreme joint configurations are changed. This is encapsulated using a simple, high–level parameterization of spatial exaggeration that enables quick use, making the the approach accessible to unskilled users. As it is based on a foundation of extreme poses, skilled animators can modify the algorithmically generated

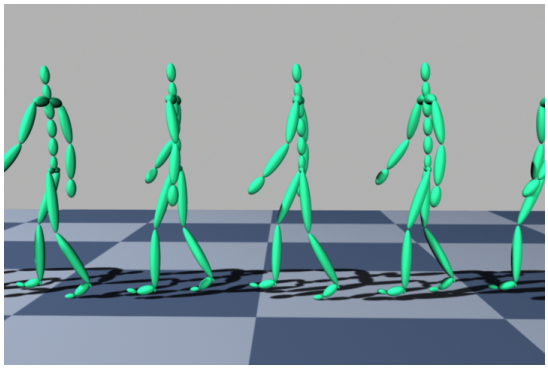


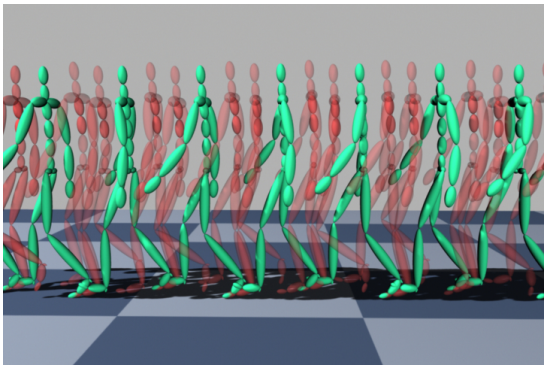
(a)





(b) (c)





(d) (e)

Figure 8.1: Spatial exaggeration uses the extreme poses (green, a) of a motion (blue, a). Increasing the geometric distance among extreme poses (b) creates an exaggerated version of the motion (red, c). Decreasing the geometric distance (d) creates a more subtle version of the motion (red, e).

results by refining the resulting pose edits. This is a key difference from other procedural approaches, as the algorithms presented here use motion extrema as editing controls, which are amenable to further editing by users. Existing approaches do not use such a conceptually meaningful foundation.

A spatial exaggeration edit to full–body motion is illustrated in Figure 8.1 for a human walk. The extreme poses used to apply exaggeration are shown in green (a). To exaggerate the motion, the poses are changed to make them more distinct (b), resulting in a motion with wider stance and more arm swing (c). The motion can also be made more subtle by making extreme poses less distinct (d), which decreases leg stance width and arm swing (e). In general, spatial exaggeration of full–body motion provides control over the dominant coordinated movement that occurs in a motion, and exaggeration of individual joint motion provides unified control over how much each part of the body moves.

Overview: The key contribution presented in this chapter is the first approach to the exaggeration of character motion that is based on the use of extreme poses—the poses used by animators when designing new motion. To do this, a model of spatial extent—how much the character changes stance over time—is presented that is based on the geometric distinctiveness of extreme poses or extreme joint configurations (Section 8.2). I then present techniques for modifying the spatial extent of a motion that alter the geometric relationships among temporally adjacent extreme poses or extreme joint configurations (Section 8.3). Spatial exaggeration can be applied to a motion using full–body poses or individually for each joint; these approaches are compared in Section 8.4 for a variety of motion examples. Spatial exaggeration is also compared to procedural motion stylization using filters, the results show that spatial exaggeration using a Cartesian approach to modifying poses better retains natural body stance and lacks the high–frequency content that filter based approaches can introduce.

8.2 A Model of Spatial Extent

To develop an algorithm for spatial exaggeration, this section presents a model of spatial extent that quantitatively models how much the body moves in space with respect to itself, *independently of how the character changes location in the environment.* This model, when applied to the full body, should represent the spatial relationships among extreme poses. When applied to each joint individually, spatial extent represents how much that joint moves with respect to itself. For example, motion with significantly different extreme poses or extreme joint configurations should have greater spatial extent than a motion with extreme poses or extreme joint configurations that are less different.

8.2.1 Pose–Centric Spatial Extent

The pose–centric model for spatial extent incorporates the difference among all temporally adjacent pairs of extreme poses. Given a set of extreme pose time $\{t_i\}$, spatial extent is formally modeled as

$$S = \sum_{i=1}^{N-1} ||\mathbf{P}(t_{i+1}) \ominus \mathbf{P}(t_i)||, \tag{8.1}$$

where $||\mathbf{P}(t_{i+1}) \ominus \mathbf{P}(t_i)||$ represents a chosen pose distance metric, and the summation is taken over the sparse set of N extreme poses.

Two metrics are considered for pose–centric spatial extent. The first is the root centric Cartesian metric given in Equation 5.3, which was presented in Chapter 5. The second is a rotation–based metric, which measures the total change in angles needed to get from one pose to another; this is

$$||\mathbf{P}(t_{i+1}) \ominus \mathbf{P}(t_i)|| = \sum_j |2 * \cos^{-1}(\theta_{i,j})|, \tag{8.2}$$

where $\theta_{i,j}$ is the scalar component of $\mathbf{q}_{i+1,j}\mathbf{q}_{i,j}^*$.

In terms of body movement, if temporally adjacent pairs of extreme poses are similar, the value of the distance metric between them will be small, and Equation 8.1 will be

small, indicating that the motion has low spatial extent. Conversely, if adjacent extreme poses are significantly different, Equation 8.1 will be larger, indicating that the motion has higher spatial extent. The key technique for applying spatial exaggeration is to modify each pair of temporally adjacent extreme poses to deliberately increase or decrease Equation 8.1.

8.2.2 Joint–Centric Spatial Extent

The joint–centric model for spatial extent considers each joint individually. Given a set of times $\{t_{i,j}\}$ that indicate when the extreme configurations of joint j occur, the spatial extent measure for that joint is

$$S_j = \sum_{i=1}^{N-1} ||\mathbf{P}_j(t_{i+1}) \ominus \mathbf{P}_j(t_i)||, \tag{8.3}$$

where $\mathbf{P}_j$ represents only the state of joint j and the distance metric only considers the change in state of that joint.

Again, both a Cartesian metric and a rotation–centric metric will be considered. The Cartesian metric is the same as that used in Chapter 5, which is given in Equation 5.2. The rotation–centric metric is

$$||\mathbf{P}_j(t_{i+1}) \ominus \mathbf{P}_j(t_i)|| = |2 * \cos^{-1}(\theta_{i,j})|. \tag{8.4}$$

In this case, spatial exaggeration is applied *individually for each joint*, as each joint can have different times at which it is in extreme configurations. Therefore, Equation 8.3 will be increased individually for each joint when extreme joint configurations are changed.

8.3 Applying Spatial Exaggeration

The key idea of spatial exaggeration is to make extreme poses more distinct—that is, more extreme. Conversely, making them less extreme should result in a more subtle version of the given motion. This idea has been formally stated for motion of the full

body in Equation 8.1 and for the motion of individual joints in Equation 8.3. This section describes how to exaggerate a motion or make it more subtle by modifying the spatial extent as stated in Equations 8.1 and 8.3. This is done by explicitly changing the extreme poses or extreme joint configurations to increase or decrease the spatial extent measures.

The editing of spatial extent is parameterized so that it acts, intuitively, as a scale factor e on how much the character moves. If e is one, the motion should remain unchanged. If $e > 1$, the motion should be exaggerated in terms of how much it moves in space. If $e < 1$, the motion should be more subtle.

8.3.1 Joint Angle Exaggeration

When editing spatial extent using a joint angle metric (Equations 8.2 and 8.4), the goal is to either increase or decrease the rotation of each joint needed to move from one extreme pose or extreme joint configuration to the next. This section presents the expression for modifying one extreme pose's joint angles with respect to another extreme pose, and this is then generalized to consider both the prior and subsequent extreme poses.

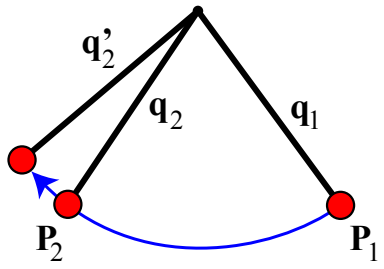


Figure 8.2: For rotation–based exaggeration, joint orientations are changed to explicitly increase the rotation angle needed to move from one extreme pose or extreme joint configuration to the next.

The expression for modifying the spatial extent of one pose relative to another is presented using a single joint, as illustrated in Figure 8.2. In this example, $\mathbf{P}_2$ is modified

relative to $\mathbf{P}_1$; these have joint orientations $\mathbf{q}_2$ and $\mathbf{q}_1$, respectively. The rotation to get to $\mathbf{P}_2$ from $\mathbf{P}_1$ is $\hat{\mathbf{q}} = \mathbf{q}_2\mathbf{q}_1^*$. Raising this to the power e, where e is the exaggeration parameter, scales the corresponding rotation angle; i.e. applying a partial rotation from $\mathbf{P}_1$ to $\mathbf{P}_2$ gives the new orientation for $\mathbf{P}_2$ as

$$\mathbf{q}_2' = (\mathbf{q}_2\mathbf{q}_1^*)^e\,\mathbf{q}_1.$$

If e is one, no change is applied. If $e > 1$, the angle from $\mathbf{q}_1$ to $\mathbf{q}_2$ increases, exaggerating the movement. If $e < 1$, the movement from $\mathbf{P}_1$ to $\mathbf{P}_2$ becomes more subtle.

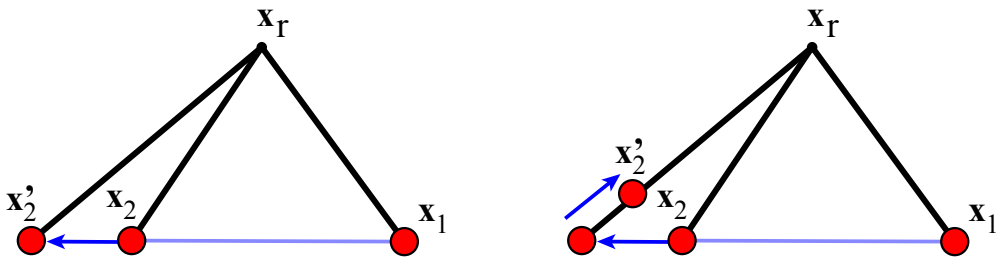
To generalize this to include both the prior and subsequent extreme poses, let the sequence of extreme poses or extreme joint configurations be $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$.. The exaggeration of $\mathbf{P}_2$ with respect to $\mathbf{P}_1$ is $\mathbf{q}_a = (\mathbf{q}_2\mathbf{q}_1^*)^e\,\mathbf{q}_1$, and the exaggeration with respect to $\mathbf{P}_3$ is $\mathbf{q}_b = (\mathbf{q}_3\mathbf{q}_1^*)^e\,\mathbf{q}_1$. These are combined using spherical linear interpolation, and the final expression is

$$\mathbf{q}_2' = (\mathbf{q}_a\mathbf{q}_b^*)^b\,\mathbf{q}_1.$$

The interpolation parameter b is referred to as the bias, which, by default, is set to 0.5 to equally combine the two exaggeration effects. Increasing b exaggerates $\mathbf{P}_2$ more with respect to $\mathbf{P}_1$, and decreasing b exaggerates $\mathbf{P}_2$ more with respect to $\mathbf{P}_1$.

8.3.2 Cartesian Space Exaggeration

Cartesian space exaggeration modifies how much the body moves with respect to itself using a Cartesian distance metric. As the Cartesian motion representations presented in Chapter 4 explicitly encode motion using Cartesian distance, the Cartesian approaches to motion exaggeration use these motion representations. For the full body, the global Cartesian motion representation is used, but changes to joint positions are applied with respect to the root to isolate the change in stance from how the character moves in the environment. For joint–centric exaggeration, the local Cartesian motion representation is used.



(a) (b)

Figure 8.3: Cartesian exaggeration explicitly increases the spatial distance each joint moves as the character moves from one extreme pose to the next. In a, the extreme position of a pendulum ($\mathbf{x}_2$) is moved away from its prior extreme position ($\mathbf{x}_1$), relative to the root joint ($\mathbf{x}_r$). To avoid the introduction of stretching, the system then corrects limb length (b).

A single–link pendulum is used to illustrate full body motion exaggeration in Figure 8.3. The left extreme position, $\mathbf{x}_2$, is moved away from the right extreme position, $\mathbf{x}_1$. This is done relative to the root position, shown here as the unmoving position $\mathbf{x}_r$. In general, each extreme pose will have a different root position ($\mathbf{x}_r(t_1)$ and $\mathbf{x}_r(t_2)$). Using s to indicate the amount of change applied, full body exaggeration of extreme poses is given by

$$\mathbf{x}'_2 = \mathbf{x}_2 + s\left((\mathbf{x}_2 - \mathbf{x}_r(t_2)) - (\mathbf{x}_1 - \mathbf{x}_r(t_1))\right).$$

This is equivalent to shifting $\mathbf{x}_2$ away from $\mathbf{x}_1$ (blue, Figure 8.3a), causing the joint to move more in space. The system then corrects the length of the limb (Figure 8.3b). While this appears qualitatively similar to rotating the joint for a single link pendulum, for multi–link structures such as a character body, this explicitly avoids the accumulation of exaggeration effects from parent joints to child joints.

When exaggerating the motion of each joint individually, the same general approach is taken. However, as the local Cartesian motion representation encodes joint motion

with respect to the parent joint, the expression for exaggeration is

$$\mathbf{t}'_s = \mathbf{t}_2 + s(\mathbf{t}_2 - \mathbf{t}_1).$$

This is again followed by limb length correction.

As described above, these approaches change the position of each joint relative to only one adjacent extreme pose or extreme joint configuration. To take both adjacent extreme poses or extreme joint configurations into account when exaggerating a pose, a weighted average of the change due to each is used.

For full body motion, let $\mathbf{x}_a$ be the modified position of $\mathbf{x}_2$ due to the position in the prior extreme pose $\mathbf{x}_1$, and let $\mathbf{x}_b$ the modified position of $\mathbf{x}_2$ due to the position in the subsequent extreme pose $\mathbf{x}_3$. The overall exaggeration, before length correction, is then

$$\mathbf{x}'_2 = b\mathbf{x}_a + (1 - b)\mathbf{x}_b,$$

where b again denotes the bias, which, again, is set by default to 0.5.

For joint–centric motion exaggeration that considers both adjacent extreme poses, the expression is

$$\mathbf{t}'_2 = b\mathbf{t}_a + (1 - b)\mathbf{t}_b,$$

where $\mathbf{t}_a$ is the exaggeration of $\mathbf{t}_2$ with respect to the prior extreme joint configuration $\mathbf{t}_1$ and $\mathbf{t}_b$ is the exaggeration of $\mathbf{t}_2$ with respect to the subsequent extreme joint configuration $\mathbf{t}_3$.

For the user interface, s is mapped to to an exaggeration parameter e that acts analogously to a geometric scaling effect. To accomplish this, $s = e - 1$. If $e = 1$, no change results. If $e > 1$, the motion becomes exaggerated. If $e < 1$, the motion becomes more subtle.

In the Cartesian approaches to motion exaggeration, the change in limb length in the modified poses could potentially be used to model another form of exaggeration, as limb length exaggeration is commonly applied by animators to emphasize important movements. However, the limb length scaling that results from this approach is a side effect

and is *not based on any animation principles*. Furthermore, it was not found to be aesthetically appealing. However, a principled approach to exaggerating limb length would be an interesting complement these algorithms, as they have been designed to create compelling exaggeration of the character's movement without changing the dimensions of the body.

8.3.3 Joint Limits and Application to the Full Motion

Spatial exaggeration can potentially rotate joints beyond plausible limits. A variety of approaches exist that return a joint to its nearest valid state. The spatial exaggeration system optionally enforces joint angle limits when needed, but it has been found to often be unnecessary when applying Cartesian exaggeration, as the modified extreme poses tend to remain plausible. To enforce joint limits on any labeled joints, the system measures how much each joint changes in orientation when applying the spatial extent edit, and if the edited orientation is past the limit, it attenuates the change to set it at the limit. Other schemes for enforcing joint limits or body configuration constraints could be applied here as well.

As spatial exaggeration specifies changes to extreme poses or extreme joint configurations, these changes must be mapped to the entire motion in a robust way that respects ground contact. Motion displacement maps [20, 179] can be used, as is common in commercial software, but this can cause ground contact error. Approaches such as footskate correction can be used to correct for this [74], but, as illustrated in Chapter 6, this correction can change the motion to partially undo pose changes applied by displacement maps. For full–body exaggeration, the system presented in Chapter 6 is used to apply the changes that have been specified on the extreme poses while preserving important ground interaction such as contacts. This approach is especially useful for spatial exaggeration using extreme poses, as it is robust and efficient, and it allows users to work only with the high level parameterization.

For joint–centric spatial exaggeration, the algorithm presented in Chapter 6 is modified slightly. Instead of applying target poses in the first step, the displacement map applies changes specified on the joint–centric extrema. For the root motion step, only the approximation steps are used, as there are no full–body target poses to use for refining the motion. In practice, this approximates the joint–centric spatial extent edit well, although some joints will not exactly meet the spatial extent edits. This is acceptable, as the intention of the spatial extent edit is specified using high–level parameters, and users do not directly specify how each joint should move.

8.4 Examples

To demonstrate how spatial exaggeration modifies articulated character motion, the approaches presented in this chapter will be applied to both a simple articulated body, for illustration purposes, and to recorded character motion. Both joint–centric and full–body exaggeration will be considered, as well as both Cartesian and rotation–based exaggeration. For the character motion examples, both exaggerated versions of the motion and more subtle versions will be shown. All motion examples have been sampled at 120 frames per second.

Multi–Link Pendulum:

To compare Cartesian and rotation–based exaggeration, Figure 8.4 shows the effect of full–body exaggeration on a multi–link pendulum with coordinated motion. To keep the motion coordinated, all parts of the articulated model reach extreme configurations at the same time. Two frames from the original motion are shown in Figures 8.4a and 8.4b. In Figures 8.4c and 8.4d, Cartesian exaggeration is applied with $e = 2$. Note that the pendulum has an overall wider swing, but the basic overall shape of the articulated system remains similar to the original. In Figures 8.4e and 8.4f, Cartesian exaggeration

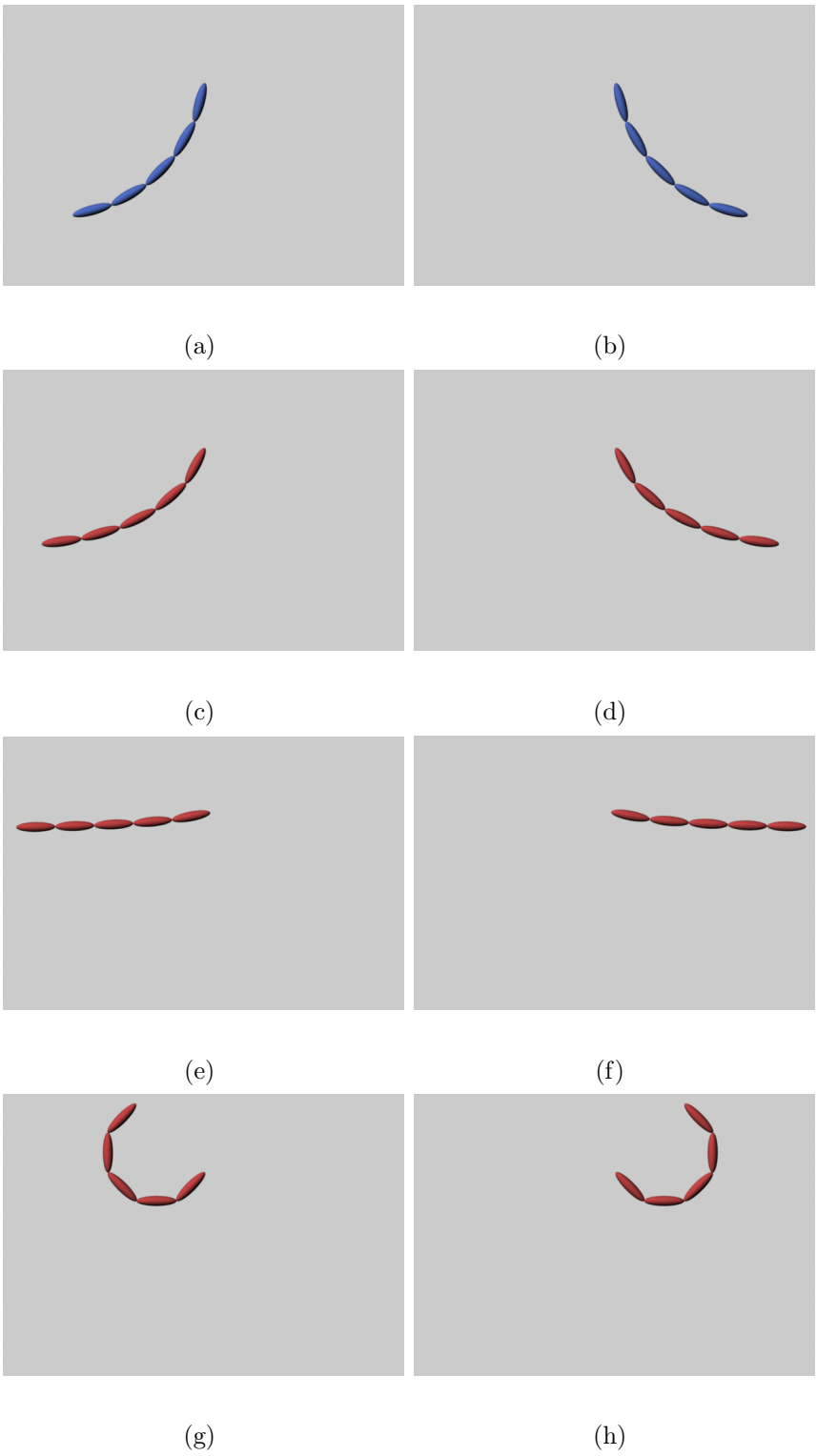


Figure 8.4: The motion of a pendulum is exaggerated. Original motion (a, b), Cartesian exaggeration with $e = 2$ (c, d), Cartesian exaggeration with $e = 20$ (e, f), rotation exaggeration with $e = 2$ (g, h).

is applied with $e = 20$. For this large value, the general motion remains similar to the original, but the swing is significantly wider. As e approaches infinity, the pendulum will swing out to nearly straight configurations.

Figures 8.4g and 8.4h demonstrate full-body rotation–based exaggeration for $e = 2$. Note that the pendulum starts to swing back over the top, as each joint's motion accumulates the exaggeration effects applied to all ancestor joints. For larger values of e, rotation–based exaggeration creates implausible configurations in which many parts the articulated model pass through each other.

Multi–Link Pendulum with Succession:

To compare full–body and joint–centric exaggeration, Figure 8.5 shows the effect of each type of exaggeration on a multi–link pendulum with *uncoordinated* motion. In this example (Figures 8.5a and 8.5b), a succession pattern has been applied to the original pendulum motion used in Figure 8.4 before exaggeration is applied. In each example, $e = 2$.

In Figures 8.5c and 8.5d, full–body Cartesian exaggeration is applied to the swinging pendulum motion. Again, the general shape of the articulated model remains the same, and the general movement has a greater swing to it. Figures 8.5e and 8.5f show the effect of full–body rotation–based exaggeration. Again, rotation–based exaggeration can create a larger increase in the swinging movement, and significantly large values of e will create implausible configurations.

Figures 8.5g and 8.5h show joint–centric Cartesian exaggeration. Again, the swing is wider, but the general shape of the articulated model's state has been changed. In particular, what was originally a smooth increase in rotation from one link to the next becomes an alternating pattern. This is due to different links having different amounts of exaggeration applied at any given particular time. For links that are near extreme, a large amount of exaggeration will be applied, and for links that are between two extremes, the

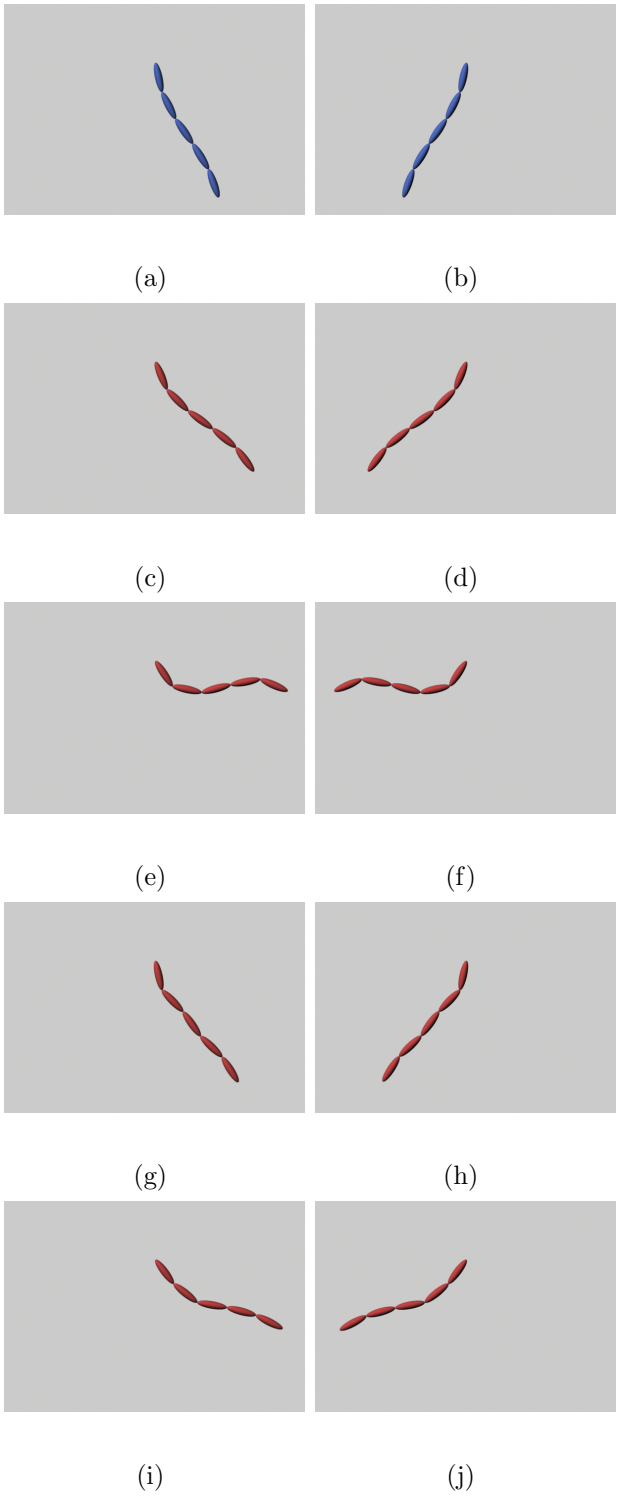


Figure 8.5: The motion of a pendulum with succession is exaggerated. Original motion (a, b), full–body Cartesian exaggeration (c, d), full–body rotation exaggeration (e, f), joint–centric Cartesian exaggeration (g, h), joint–centric rotation exaggeration (i, j).

exaggeration will be close to zero. Joint–centric rotation–based exaggeration (Figures 8.5i and 8.5j) has a similar, but less prevalent, effect.

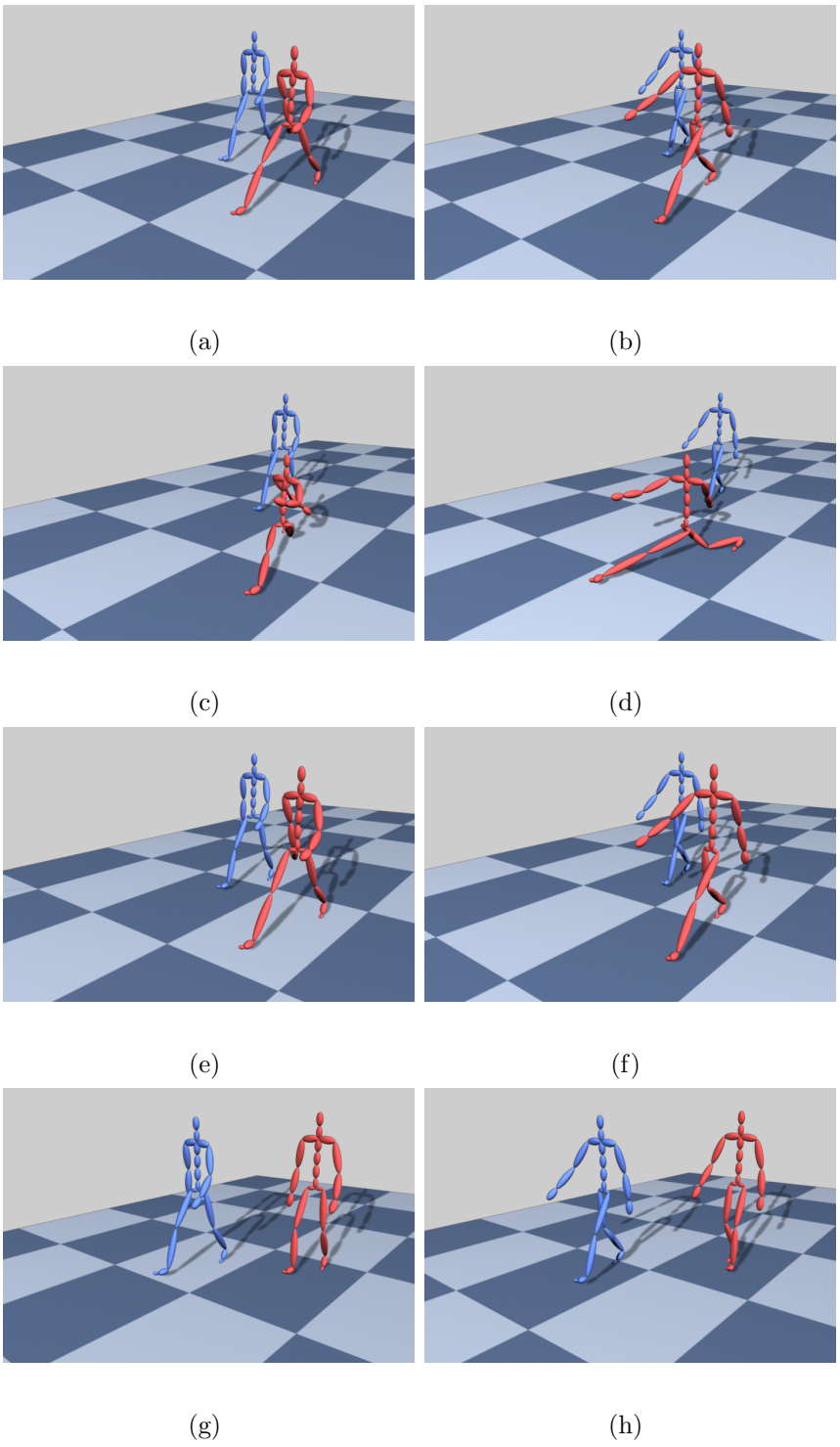
Overall, full–body rotation–based exaggeration works well when e is bounded, and full–body Cartesian exaggeration works well for these examples for any positive value of e. Joint–centric exaggeration can introduce subtle changes in the overall shape of the model, due to the uncoordinated nature of joint–centric exaggeration. For this reason, the full–body exaggeration approaches are generally preferable.

Walking:

Figure 8.6 illustrates different types of exaggeration applied to a walk. In this and all subsequent examples, the blue character represents the original motion, and the red character represents the exaggerated (or more subtle) motion. For exaggerated motion examples, $e = 2$. For more subtle motion examples, $e = 0.5$.

Figures 8.6a and 8.6b shows the effect of full–body Cartesian exaggeration. Note that both the leg stance width and arm swing are wider, and the character remains in plausible configurations. Figures 8.6c and 8.6d shows full–body rotation–based exaggeration. Unlike Cartesian exaggeration, rotation exaggeration results in some joint configurations that are possible, but not likely to occur in human motion. In particular the shoulders and hips rotate into very unlikely configurations. This is due to the increase of subtle twisting motion that would not likely occur if a character were acting in an exaggerated manner. This is a common problem when applying rotation–based exaggeration to human motion, and, in general, rotation–based exaggeration performs best for unconstrained systems such the multi–link pendulum. For this reason, Cartesian exaggeration is generally preferable for human motion.

Figures 8.6e and 8.6f shows joint–centric Cartesian exaggeration applied to the walk. This has a similar effect as the full–body Cartesian exaggeration (Figures 8.6a and 8.6b), as the coordination in the walking motion causes similar exaggeration effects to be ap-

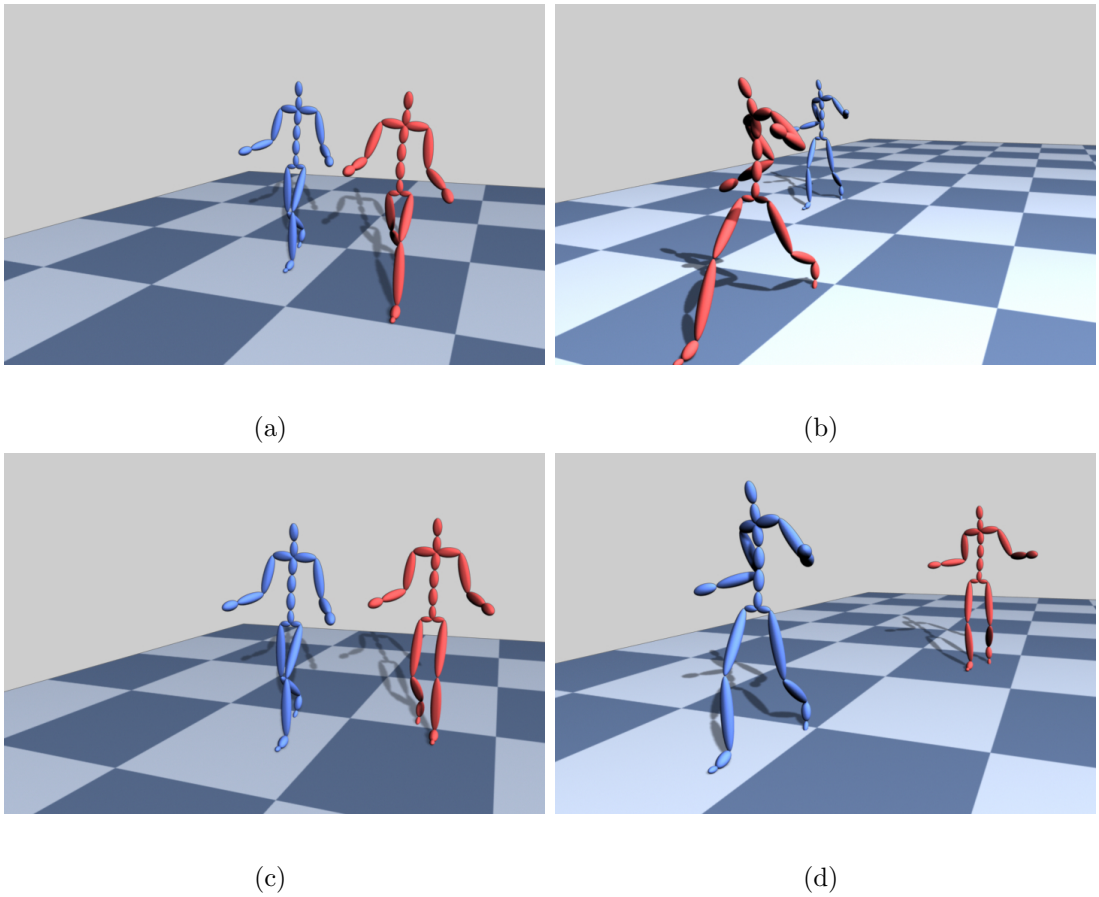


(a) (b) (c) (d) (e) (f) (g) (h)

Figure 8.6: Exaggeration of the walk using different techniques. Full–body Cartesian exaggeration with $e = 2$ (a, b), full–body rotation exaggeration with $e = 2$ (c, d), joint–centric Cartesian exaggeration with $e = 2$ (e, f), full–body Cartesian exaggeration with $e = 0.5$ (g, h). Original motion is shown in blue.

plied.

A more subtle version of the walk is shown in Figures 8.6g and 8.6h. In this example, the legs have a shorter stance width and the arms swing less, creating walking motion that is similar to the original, but with more subtle movement.



(a) (b)

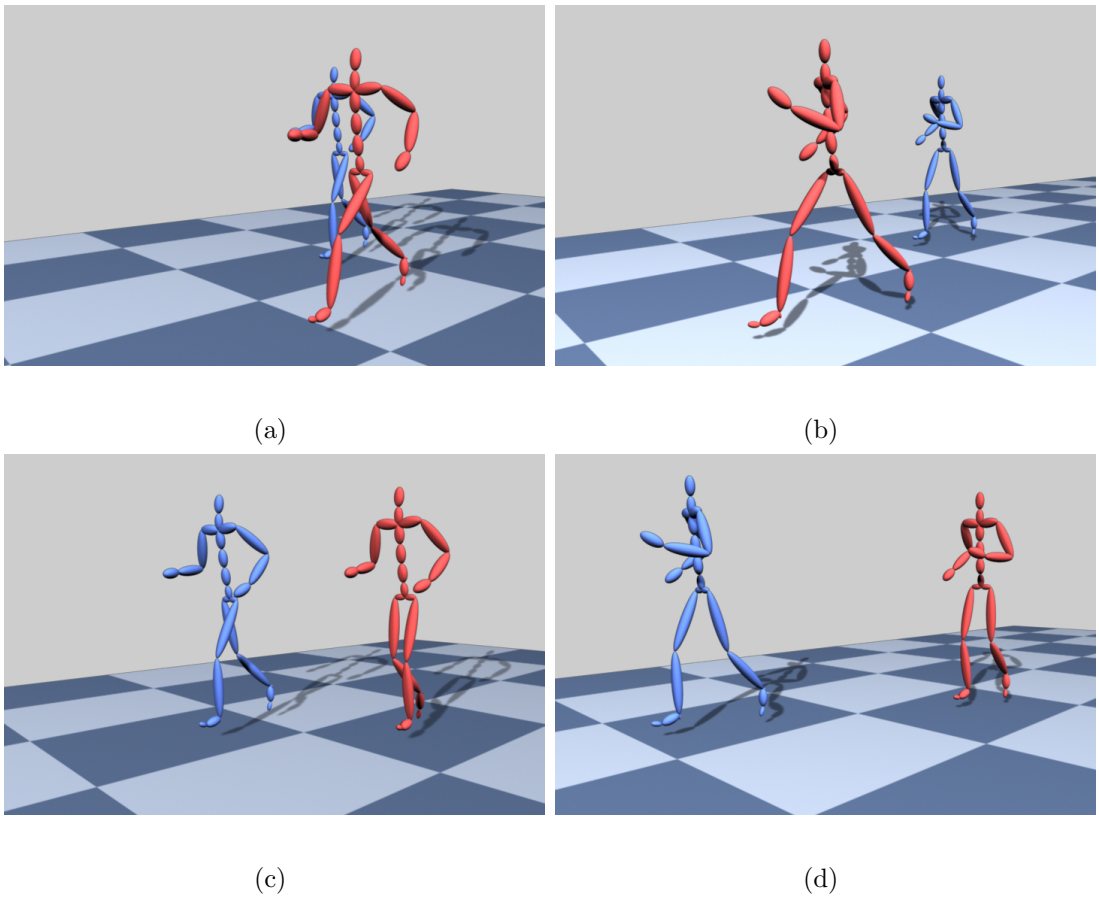
(c) (d)

Figure 8.7: Full–body Cartesian exaggeration of stylized walking motion. a, b: $e = 2$. c, d: $e = 0.5$.

Stylized Walk:

Figure 8.7 shows exaggerated and more subtle versions of a stylized walk using full–body Cartesian exaggeration. In Figures 8.7a and 8.7b, the character twists more and makes stronger gestures. As with the neutral walk, the stance width increases, causing the character to walk a greater distance in the environment.

In Figures 8.7c and 8.7d, the character twists less and makes more sublte arm gestures. As stance width is again decreased for this more subtle version of the motion, the character travels less distance in the environment.



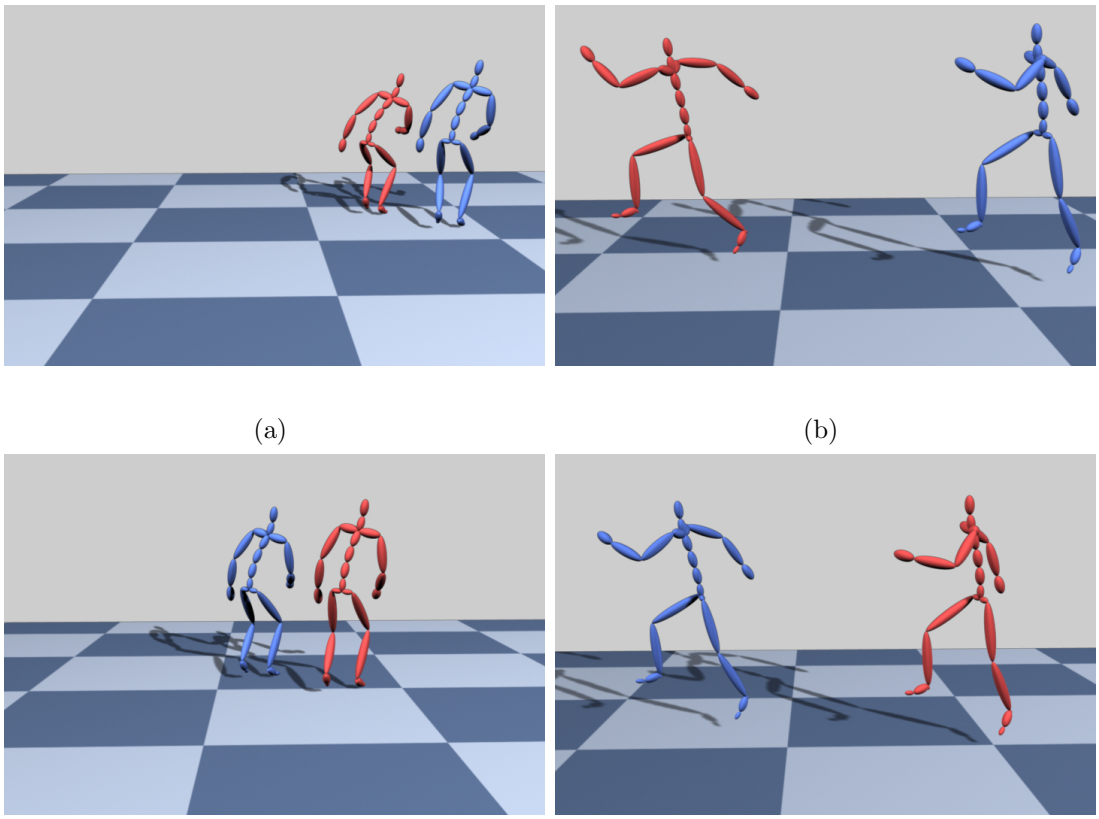
(a) (b)

(c) (d)

Figure 8.8: Full–body Cartesian exaggeration of a run. a, b: $e = 2$. c, d: $e = 0.5$.

Run:

In Figure 8.8, exaggerated and more subtle versions of a run are shown. For this example, the exaggerated motion (Figures 8.8a and 8.8b) appears as though the character is exerting greater effort. Similarly, the subtle version (Figures 8.8c and 8.8d) appears as though less effort is exerted. This is generally true for most athletic motion, as the greater or lesser change in stance corresponds to greater or lesser body movement, and therefore, greater or lesser energy exertion.



(a) (b)

(c) (d)

Figure 8.9: Full–body Cartesian exaggeration of a jump. a, b: $e = 2$. c, d: $e = 0.5$.

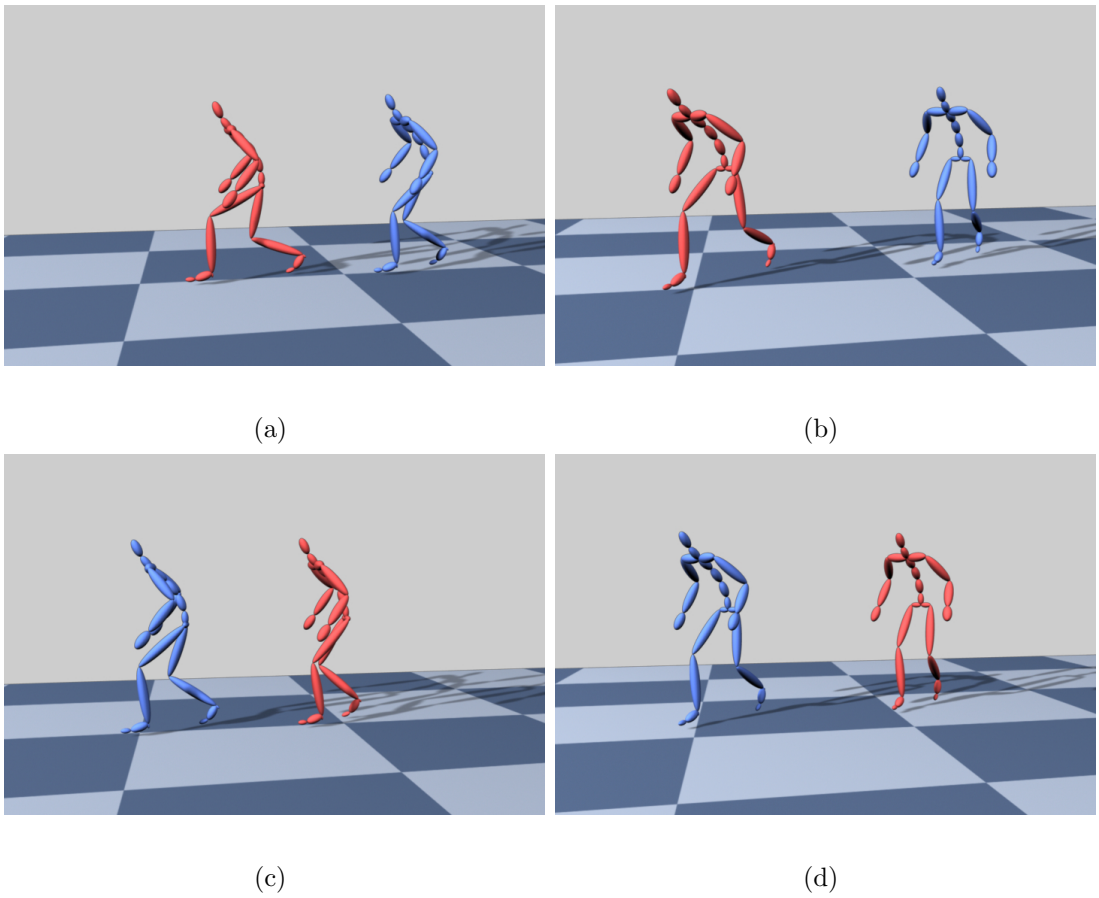
Jump:

In Figure 8.9, exaggerated (a, b) and more subtle (c, d) versions of a jump are shown. The spatial exaggeration system effectively increases leg stride and the swing of the arms that occurs during the jump. It does not, however, increase the height of the jump, as it only considers body stance. Developing a trajectory–centric approach to exaggeration would be a useful complement to the spatial exaggeration approach for examples such as this.

Sneaking:

For the sneaking motion shown in Figure 8.10, the full–body exaggeration (a, b)

increases the extent to which the character crouches and rises while walking. Similarly, the more subtle version of the motion (c, d) decreases this.



(a) (b)

(c) (d)

Figure 8.10: Full–body Cartesian exaggeration of a sneaking walk. a, b: $e = 2$. c, d: $e = 0.5$.

Comparison to Filter–Based Stylization:

Spatial exaggeration is compared to another procedural stylization algorithm, the cartoon animation filter [173], in Figure 8.11. In this example, root translation is removed to illustrate the effect of each algorithm in isolation. As the cartoon filter introduces new content in the form of anticipation and follow–though, it can introduce high–frequency content that is not present in the original motion, which can cause it to appear jerky. In contrast, spatial exaggeration exaggerates the movement that already exists.

The algorithms also differ in that spatial exaggeration can use a stance–centric Carte-

sian parameterization of the motion, while the cartoon filter modifies intrinsic parameters, i.e. joint angles. As noted above, Cartesian exaggeration performs better at creating plausible exaggerated stance in the movement, while the cartoon filter, which uses joint angles, can easily create unnatural poses, such as seen in the legs in Figure 8.11.

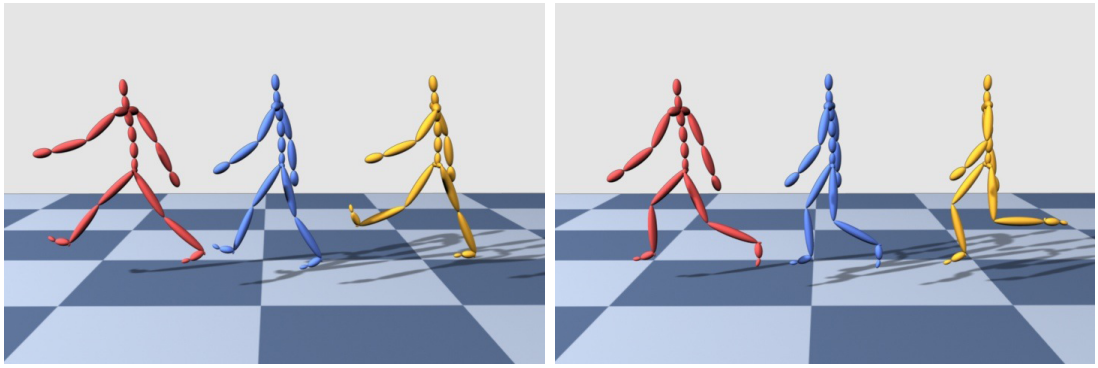


Figure 8.11: For the walk, full–body Cartesian spatial exaggeration (red) is compared to the cartoon animation filter (orange). Unlike the cartoon filter, spatial exaggeration does not introduce high–frequency content and results in more plausible body stance. Root translation is removed in this example to compare the effect of each algorithm in isolation.

8.5 Discussion

This chapter has presented an approach to applying spatial exaggeration to articulated character motion. This section describes the assumptions made by this model of exaggeration, how the implementation could be extended, and how the approach relates to other forms of procedural motion stylization.

Assumptions of the Spatial Extent Model: Spatial exaggeration assumes that exaggeration of movement can be entirely determined using only extreme poses or the times of extreme joint configurations. It might be possible to model spatial extent using

a different approach that does not rely on the selection of a meaningful sparse set of poses or joint configurations.

The spatial exaggeration approach also applies spatial exaggeration using only character poses or joint configurations. While this models exaggeration of body movement well, it does not model potential changes to the motion of the character within the environment. For example, spatial exaggeration does not capture variation of movement in free rigid bodies, as exaggeration is applied relative to the root position. It would be useful to extend these ideas to such motion by considering geometric extrema of a trajectory in space.

In addition, exaggeration is modeled in terms of an articulated control skeleton. This cannot capture and modify variation in geometric deformation. It would useful to consider an arbitrary collection of surface points for pose selection and develop a form of spatial exaggeration that geometrically deforms surface geometry.

Limitations of the System: In the implementation presented here, the exaggeration parameter values are only considered at specific times—the times of the extreme poses. If users wish to slowly turn on exaggeration over the course of a motion, they must control the variation of this parameter at these specific times. It would be useful to add a continuous attenuation parameter to provide control over how much exaggeration is applied at any time.

In addition, it would be useful to provide control over how much spatial exaggeration is applied to different parts of the body. Applying it to limbs that make ground contact can significantly speed up how fast a character moves in the environment, and users might not want to have this effect correlated to how much free limbs, such as swinging arms, are exaggerated.

Relation to Geometric Laplacian Editing: Laplacian surface representations have become a popular representation for detail–preserving mesh editing [159]. Spatial exaggeration shares a similar motivation, in that it aims to make large–scale changes to motion, rather than a surface mesh. In both approaches, detail is preserved when large–scale edits are applied.

Spatial exaggeration differs, however, in that the geometric differences are modeled with respect to a sparse set of meaningful times, which are chosen using a motion extreme identification algorithm. In Laplacian mesh representations, each vertex is encoded with respect to its immediate topological neighbors, rather than a sparse set of surface extrema. This approach can be applied to motion representations. For example, Kim et al. use the Laplacian mesh editing approach to preserve detail when editing root trajectories in character motion. It would be interesting to develop surface editing tools that operate in a similar spirit to spatial exaggeration, in which meaningful surface extrema could be identified and edited with respect to neighboring surface extrema. For example, the iWires system allows users to directly manipulate related surface features [40]; such features could be modified procedurally as a step in this direction.

Relation to the Bilateral Filter: The bilateral filter [133] has recently become popular for image processing applications, as it has been designed to preserve important edges in an image while smoothing out detail. This could potentially be used to find spatial extrema, for example, by filtering out detail to find the most distinct times in the movement. Spatial exaggeration could then be applied to these times, similarly to how it is applied to the extreme poses found using an explicit search.

Other Models for Procedural Stylization: Spatial exaggeration only addresses one aspect of stylized exaggeration—change of body stance. This is intentional, as it allows for direct control over a specific qualitative aspect of motion. As it correlates

strongly to the exaggeration that animators can create in the earliest stages of character animation, it is a particularly important aspect of style. A practical implementation of these ideas would combine this type of exaggeration with other forms of specific control such as stance [127], or relative timing (Chapter 7), allowing users to make specific choices about *how* to stylize a motion. These different ideas could be combined, along with filtering techniques that add oscillation [104] or anticipation and follow–through [173], to create a suite of tools for making stylized changes to character movement.

8.6 Conclusion and Future Work

This chapter has introduced spatial extent as a model for exaggerating a specific aspect of character movement—how much the stance of the character changes over time. To do this, spatial extent edits are applied by changing the geometric relationships among temporally adjacent extreme poses or extreme joint configurations, which are selected using an algorithm for identifying motion extrema.

While this approach has been designed as a single edit to an entire motion, each pose change has bounded temporal support. It would be useful to extend it to an online algorithm that requires a limited window of upcoming frames. This approach to spatial exaggeration is based on a sequence of discrete changes at specific times; it would be interesting to pursue a continuous formulation. It would also be possible to use extreme poses in conjunction with data–driven pose models to add another form of style control.

Finally, it would be useful to combine spatial exaggeration with pose–centric versions of other procedural stylization algorithms that provide high level control over oscillatory response [104], anticipation and follow through [173], and temporal variation (Chapter 7) to create a unified approach for applying pose–centric style–based motion edits.

Chapter 9

Conclusion

This thesis has introduced motion extrema for use as a foundation for expressive motion editing. To do so, I have have focused on algorithms for identifying motion extrema, as well as applications that use them to edit existing motion, such as recorded motion capture data. The algorithms for identifying motion extrema use two approaches: finding the extrema of differential measures of motion data and finding times at which the character state is in a spatially extreme configuration. The editing applications take two approaches. The first approach, used in Chapters 6 and 7, allows for the direct editing of extreme–based pose representations by users. The second approach, used in Chapters 7 and 8, aims to develop high–level, procedural encapsulations of animation principles in terms of relationships among motion extrema.

The first application, pose–centric editing (Chapter 6), allows users to edit motion using a sparse set of extreme poses without requiring them to explicitly manage the relationship of the motion to the ground. This problem is modeled as a set of prioritized constraints, allowing pose edits to be met as best possible, subject to the requirements that ground contacts are preserved and ground–relative motion remains plausible. To solve this problem in closed–form, a greedy prioritized optimization solver is used; this enables pose–centric edits to be robustly applied at interactive rates.

The second application—staggered poses (Chapter 7)—presents a new pose representation that explicitly encodes timing variation among coordinated degrees of freedom. Users can directly edit these timing relationships, and the staggered poses system includes procedural tools for editing these timing relationships using techniques based on animation principles. In particular, the system can model overlapping action effects such as succession and asymmetric action. The staggered poses motion representation and associated algorithm for finding staggered poses in existing motion data allow for the editing of timing relationships in pre–existing motion.

Spatial exaggeration, the third system (Chapter 8), focuses on the procedural editing of a specific aspect of motion style, which I call spatial extent. Spatial extent models the geometric relationship among extreme poses specified at different times. The work in Chapter 8 investigates procedural algorithms for modifying these geometric relationships to make motion appear exaggerated or more subtle, in terms of how much the character's stance changes over time. This idea is considered on individual joints, as well as for the body as a whole. It is also considered using using both rotation–centric and Cartesian pose parameterizations.

Overall, the work in this thesis has investigated three types of problem. The first is the identification of a meaningful set of motion extrema that have qualities similar to the extreme poses created by keyframe animators (Chapter 5). The second is the development of techniques that enable users to edit motion using a sparse set of extreme character poses (Chapters 6 and 7). The third type of problem is the procedural editing of motion extrema using algorithms that express principles of movement in terms of relationships among motion extrema (Chapters 7 and 8). Each of these areas has been investigated at an exploratory level to demonstrate the overall viability of using motion extrema as a foundation for expressive motion editing. However, many areas for future work remain.

The remainder of this chapter will first reflect on what has been accomplished. It will

then propose a strategy for the future development of animation and editing applications that use motion extrema. Feedback from professional character animators and technical animators will be presented, and major directions of future work will be proposed.

9.1 Reflections

While the systems presented in this thesis have demonstrated the viability of using motion extrema to edit existing motion data, a number of questions remain, and a variety of improvements could be made. In this section, I will describe which animation principles have been considered in developing these systems, as well as how the various editing applications use different types of motion extrema. I also discuss the general capabilities of the techniques, their limitations, and suggest ideas for improvement.

9.1.1 Relationship to Principles of Movement

The techniques presented in this thesis have focused on two general areas of editing: pose and timing. However, a variety of other aspects of movement related to expressive performance could be considered. For example, shape change, particularly in the form of squash and stretch effects, can be especially effective for modeling expressive performance in cartoon–like motion [84]. As another example, the timing of how a character moves from one extreme pose to the next, whether in a smooth or jerky way, can convey how energetic a character appears.

The first editing application, pose–centric editing, focuses on enabling users to specify meaningful pose edits that can be automatically applied in a robust way. In particular, it has been designed to support the editing of a sparse set of extreme poses, which can be used as an effective set of editing controls (see Chapter 1). This application was motivated by a workflow commonly used by animators, in which they specify a small set of extreme poses when designing motion. The pose–centric editing application can

be considered the inverse of this idea, in that the high–level design of the character's performance can be changed after detail exists.

The primary focus of the staggered poses system is timing. When users edit parameter values in a staggered pose, any existing timing refinements will be preserved. With existing tools, animators commonly avoid setting timing variation patterns when creating new motion until the end of the creative process, because existing pose representations must be broken to allow for this timing variation to be specified. The staggered pose representation also allows for the development of procedural editing tools that encapsulate common motion principles related to overlapping action. The succession tool sets staggered timing patterns, and the randomization tool creates asymmetric action effects. In existing systems, users must explicitly specify these patterns on knots in low–level animation splines.

The third application, spatial exaggeration, focuses on exaggerating one specific aspect of movement—the geometric relationship among extreme poses. The spatial exaggeration system uses a set of related approaches to change the extreme poses, but it does not incorporate any timing changes, and it does not aim to be a comprehensive exaggeration system. Other aspects of motion could be exaggerated as well, such as different aspects of timing, a change in limb length, and geometric deformation, should the shape of the character be modeled.

The Classical Animation Principles

The classical animation principles were briefly described in Chapter 2 to describe the principles that motivate the ideas presented in this thesis. This section describes these principles more fully and reflects on how these principles relate to the work presented here.

Squash and Stretch: *Squash and Stretch* can be used in conjunction to deform a shape in a believable way, and animators often exaggerate the extent to which they

deform objects in motion. While the editing systems presented in this thesis do not model general character geometry, the idea of squash and stretch can also be applied to the change in stance of an articulated character with rigid limbs [84]. A change in body stance from a closed posture to an open posture can be exaggerated, for example. This directly relates to the spatial exaggeration algorithms described in Chapter 8, in that the techniques presented here explicitly change how much the stance at one time differs from stance at neighboring times. The exaggeration control gives the user control over how much this change is applied.

Anticipation: *Anticipation* refers to the tendency of an object in motion to have some kind of preparatory movement. I do not explicitly model anticipation in any of the systems presented in this thesis. However, spatial exaggeration will often cause existing anticipatory motion to be exaggerated, especially if it is the dominant motion of the character.

Straight Ahead and Pose to Pose: As mentioned in Chapter 1, *straight ahead* and *pose to pose* refer to two workflow strategies that animators can take when creating new motion. The approaches to motion editing presented in this thesis are fundamentally pose–based, as they have been designed with the goal of providing high–level editing controls at sparse moments in time. However, as these techniques smoothly map pre–existing detail between any pose–centric edits, they preserve the movement detail. This detail can contain the nuances of movement that animators create when animating in a straight ahead approach.

Follow Through and Overlapping Action: *Follow through* describes the opposite of anticipation. It models the idea that there will typically be some response movement to a primary motion. As with anticipation, the editing systems presented in this thesis do not explicitly model follow through. However, the spatial exaggeration algorithms presented in Chapter 8 exaggerate existing follow through movement if it is significantly dominant.

Overlapping action, described in Chapter 2, refers to the fact that one major body movement will typically start before the prior body movement finishes. The staggered pose model presented in Chapter 7 explicitly represents overlap using timing refinements. Succession and asymmetric action model specific types of overlapping action.

Slow–In and Slow–Out: *Slow–in* and *slow–out*, also known as *ease–in* and *ease–out*, refer to the tendency of character motion to remain smooth over time. I do not explicitly model this in the editing systems presented in this thesis. However, these systems preserve existing smoothness of motion when applying motion edits.

Arcs: Characters tend to move in such a way that joint trajectories make *arcs* in space. The editing systems presented here do not explicitly aim to change these arcs. However, pose–centric editing (Chapter 6) provides users with indirect control over the size of an arc in space, and the spatial exaggeration algorithms (Chapter 8) procedurally modify the size of motion arcs as a secondary effect of making poses more distinct.

Secondary Action: *Secondary action* refers to movement that occurs as a natural response to some primary movement. For example, when a limb moves, neighboring limbs tend to move with a timing delay rather remaining fully constrained to the primary limb movement. The staggered poses model explicitly represents the kinematic timing variation that typically occurs in character movement. This allows for high–level changes to be made to the timing of this secondary action.

Timing: *Timing*, also described in Chapter 2, refers to both the overall timing of a character performance and the timing characteristics of movement such as secondary action [175]. Both the pose–centric editing system (Chapter 6) and the spatial exaggeration system (Chapter 8) aim to preserve existing timing while providing direct editing control over geometric aspects of motion. In contrast, the staggered poses model (Chapter 7), by explicitly representing relative timing among different parts of the body, has been designed to provide timing–centric procedural edits.

Exaggeration: *Exaggeration* refers to the creative choice to take a particular aspect of movement or design and to make it stand out more. This is done with the intention of making that aspect of animation more readable to an audience. Different aspects of animation can be exaggerated, ranging from color to geometric design to movement. In terms of movement exaggeration, different specific aspects of movement can be exaggerated, such as overall timing, the relative timing among characters, and timing differences between different parts of the body.

This thesis includes procedural tools for exaggerating two specific aspects of motion. The staggered poses editing system (Chapter 7) allows for the procedural exaggeration of *asymmetric action* and *succession* effects, which model specific forms of timing variation among different body parts. The spatial exaggeration system (Chapter 8) models purely geometric aspects of character poses, focusing on the exaggeration of how much the body moves in space. This thesis focuses on the procedural exaggeration of specific aspects of motion in order to develop algorithms based on specific principles.

Appeal: While *appeal* would be difficult to model, as it is subjective, it remains a motivating idea for the work presented in this thesis. The techniques presented here have been designed with the goal of providing users with expressive performance control when editing existing motion data. From a user's perspective, the goal is to enable the creation of performances that are more appealing than those present in the original motion.

The remaining classical animation principles, *staging* and *solid drawing*, do not specifically relate to the techniques presented here. Staging refers to creative composition decisions, which remain the domain of the user. Solid drawing refers to a background skill needed for effective development for traditional animation.

9.1.2 Relationship to Motion Extrema

Each of the editing applications has been built to use motion extrema as an editing foundation. Some focus on extrema related to the full body in the form of poses; others

focus on extrema specific to joints. This section will summarize how each system uses motion extrema and describe additional ways in which they could make use of motion extrema. Many of the choices about which systems use which extrema are historical, as the motion extreme identification algorithms were developed in parallel with the various editing systems.

Pose–centric editing has been designed to enable interactive pose–centric edits that are specified using poses that have been algorithmically selected. In practice, the system can use any sparse set of poses, and it is not constrained to a particular pose selection algorithm. However, the overall goal is to use poses that are similar to the extreme poses created by keyframe animators when they block out motion. The results presented in Chapter 6 use poses selected as the minima of aggregate joint acceleration; this approach often requires users to refine the selection. A better approach would be to use spatially extreme poses, such as those chosen for spatial exaggeration, with the option to add additional poses, if needed. Finally, it would be good to extend the system to handle the robust application of edits on joint–centric motion extrema.

The staggered poses system focuses on the use of joint–centric motion extrema—the times at which the movement of individual joints changes significantly. While the results shown in Chapter 7 use motion trail curvature maxima, any of the joint–centric extreme identification algorithms could be used. The staggered pose representation itself models the temporal coordination among these joint–centric extrema, and therefore models *relationships among extrema of movement* on different parts of the body. From another viewpoint, the time at which a staggered pose is centered could be considered a time at which the full body is in an extreme pose.

Spatial exaggeration uses both full–body poses and joint–centric extrema. The principle of using spatially extreme body configurations was developed specifically to support this system, as encapsulating the use of poses as an editing foundation requires the selection to only include extreme poses. Poses selected as differential extrema can also

include poses that are not extreme; users would need to refine such selections, which would require significantly more user effort and break encapsulation.

Finally, while these systems have focused on extrema related to the full body or to individual joints, it would also be possible to consider extrema related to a portion of a character, such as a limb. Another possibility would be identifying extrema related to mesh-based motion representations.

9.1.3 Capabilities and Limitations

While the techniques presented in this thesis allow the style of motion to be changed, they do not, in general, allow the *type* of motion to be fundamentally changed. For example, changing a walk to a jump is outside the scope of each of these systems. This is due to many fundamental qualities of movement being tied to a specific type of motion. For example, the ground contact patterns in a walk are different from those in a run or jump. A broader set of techniques would be needed to accomplish these types of edits. However, the performance–related changes that can be applied to motion can be fairly significant, and the edit will remain plausible. It is noted, however, that large–scale edits will typically create motion that is not commonly made by real people. Instead, such movement will appear as an exaggerated version of realistic motion.

Balance and other physical effects, such as body lean, are not modeled by the editing systems presented in this thesis. While the editing results for the various systems do demonstrate these effects, these were specified by the user in the edits made to the poses. Adapting the ideas in this thesis to motion generated by physical simulation or physically–constrained optimization could lead to an editing system that allows for balance to be automatically maintained while a user edits poses.

Finally, it would be good to incorporate the editing techniques presented in this thesis into a general–purpose system. The systems presented in this thesis, while sharing similar motivations, were developed independently. It would be useful to combine them into a

general performance–centric motion editing system. It would also be useful to consider the inclusion of other techniques for motion editing.

9.2 Developing Editing Tools Using Motion Extrema

This section proposes a general approach that can be taken to develop motion editing tools that use motion extrema, based on the author's experience of developing the systems presented in this thesis. It is not strictly the approach that was taken while these systems were in development. However, in reflection, these are the common stages of development that took place for each system. It does not recommend particular approaches to *user interface design*, which has not been a focus of this work, but instead focuses on *algorithmic* design.

For each stage of the development process, I will first discuss how that step is useful when developing new extreme–based motion editing tools. I will then describe how each stage relates to the development of the three motion editing systems presented in this thesis.

9.2.1 Model the Editing Task

To model the editing task, it can be useful to choose one or more principles of movement or known workflow techniques. These principles or techniques can be used as a conceptual reference to model what is known about the user's goals and the approach the user would take when manually applying an edit. This modeling of the editing task can draw from the classical animation principles, the Laban Movement Analysis model, or other principles of movement or workflow strategies.

For example, the pose–centric editing and staggered poses systems each model the pose–to–pose animation technique, but they apply it to the editing of existing motion rather than the creation of new motion. Each of the three systems makes use of a sparse

set of extreme poses, and the spatial exaggeration system, in particular, aims to use only poses that are similar to the extreme poses created by keyframe animators. The extreme pose selection algorithm that is based on finding spatially extreme body configurations, in particular, has been designed with the goal of selecting poses that are similar to extreme poses.

The staggered pose system explicitly models overlapping action, one of the classical animation principles. The two procedural tools included with the staggered poses system model specific timing patterns: succession and asymmetric action.

The spatial exaggeration system focuses on the exaggeration of one specific aspect of motion—the geometric relationship among extreme poses. This has been motivated by observing how animators, when designing new motion, will typically work only with poses in initial stages. Animators will then work with these poses to exaggerate character stance and how it changes over time. The decoupling of this specific aspect of exaggeration from others was motivated by the classical discussion on animation exaggeration, in which a variety of different aspects of animation are discussed in terms of how they can be exaggerated [167].

9.2.2 Express the Editing Task in Terms of Motion Extrema

After a conceptual model of the editing task has been chosen, the next step is to relate it to extrema of motion. A variety of questions should be considered. For example, does the editing task relate to skeletal motion representations, facial motion, or geometric surface representations? Should the extrema be related to the full character, a portion of the character, or individual degrees of freedom? Is the use of motion extrema meant to be encapsulated within the editing algorithm, or should this be visible to the user? Finally, it might not be possible to encapsulate some editing tasks in terms of motion extrema. For example, continuous correlation effects [129] relate one part of a body to others, but do not consider any specific moments in time. Therefore, that approach would have little

to gain from expressing it in terms of a sparse set of motion extrema.

The pose–centric editing system has been designed to support the robust application of extreme pose edits to character motion that has important ground contact. While it uses full body poses, it does not specifically require any particular choice of extreme character pose. In addition, it supports non–extreme poses, but these can be less effective for expressing performance–centric motion edits. This system does not consider joint–centric motion extrema.

Staggered poses model the relative timing relationships among motion extrema on different parts of the body. While the staggered pose model itself supports arbitrary character parameterizations, the motion editing system presented here focuses on finding motion extrema on individual joints. Relationships among these extrema are created based on temporal proximity. The procedural tools for editing timing model specific timing patterns within a staggered pose. As such, these tools modify the relationships among these joint–centric motion extrema.

Spatial exaggeration considers both full–body motion extrema in the form of extreme poses and joint–centric motion extrema. It requires that only spatially extreme configurations be used as poses or joint–centric extrema. Furthermore, as the technique encapsulates the use of extreme poses or joint–centric extrema from the user, the extreme identification algorithm it uses must produce a sparse set that does not require user refinement. This is in contrast to the two other editing systems, in which it is intended that users directly interact with the extreme character poses.

9.2.3 Choose or Develop a Model for Motion Extrema

The next stage is to choose a model for motion extrema. Chapter 5 introduces two approaches for finding motion extrema, and a number of specific algorithms that use these principles were presented. It might be the case that one of these approaches will work for a new application, but it might also be the case that a new model and algorithm

will be needed. This is especially true if motion is represented in a form other than a skeletal hierarchy. For the systems presented here, the general goal is that motion extrema have similar qualities to the extreme poses created by keyframe animators.

Pose–centric editing uses minima of aggregate joint acceleration. This choice is not intrinsic to the algorithm, and it was made for simplicity when presenting the algorithmic material in Chapter 6. In addition, the flexibility of the pose–centric editing system is important, as users might want fine–grained control with extra poses, or they might want coarser control, using only a subset of the extreme poses.

The staggered poses system requires a joint–centric extreme identification algorithm. The specific examples shown in Chapter 7 use maxima of motion trail curvature. This measure is also used to select the central times of staggered poses. However, any other approach could be used, provided that it produces both joint–centric extrema and a continuous probability signal for each joint.

Spatial exaggeration specifically requires that the identification algorithm only select extreme poses. This is an example for which a new extreme identification technique was developed. Other techniques were not consistently suitable for this application, especially for full–body exaggeration.

9.2.4 Model the Editing Task Using the Model for Motion Extrema

After expressing the editing task in terms of motion extrema and choosing a model for motion extrema, the editing task must be explicitly related to the motion extrema. Specifically, when a user makes a change to edit the motion, either a direct change to the motion extrema or a change to a relationship among them must take place. For direct editing by the user, this is straightforward, although mapping the change to the motion can be nontrivial. For high–level procedural editing, this can be more complex, as it involves specifying how to change a relationship among the motion extrema.

In the pose–centric editing system, the extreme poses are given, as well as the user edits to these poses. To apply these changes, this system must modify the remainder of the motion such that the pose edits are applied, while constraining them to be plausible. As part of this method, the pose–centric edits might be modified to preserve ground contact.

The staggered poses system also allows users to directly apply geometric pose–based edits. The correlation between the extreme value knots and the editing knots that users work with when editing the true pose represents one type of relationship that is formally modeled by the editing system. Specifically, these knot pairs are constrained such the change in value to each is equal. This system also models relationships among motion extrema using the procedural tools that apply timing refinements. The succession and asymmetric action tools alter the values of the timing refinements within a staggered pose to produce specific timing patterns. The succession tool does this based on the skeletal topology. The asymmetric action tool does this based on a user selection of character joints.

Spatial exaggeration formally models spatial extent in terms of geometric relationships among motion extrema at different times. As users edit the exaggeration parameters, these geometric relationships are interactively applied by changing the motion extrema, whether they are extreme poses or extreme joint configurations. This causes the extreme poses, or extreme joint configurations, to appear more or less distinct in the resulting motion.

9.2.5 Address Practical Aspects of Real Data

Practical aspects of real world data must also be considered. This applies to the development of new motion extreme identification algorithms, as well as to new motion editing algorithms. This includes the application of any needed motion annotations, as well as accounting for noise in the data. The system design should also consider any re-

construction artifacts that might arise due to algorithmic limitations in motion recording software. Finally, any assumptions made in the extreme selection and editing algorithms might not hold true for certain types of motion.

In the systems presented in this thesis, the key motion annotation needed is ground contact. Two approaches have been used to identify ground contact. The staggered poses system searches for clusters of frames in which the end effector is held close to still. It relies on the user to correct contact annotation errors. The pose–centric editing and spatial exaggeration systems use per–frame labeling based on velocity thresholds and gap filling to reduce error. In practice, this robustly labels nearly all ground contacts for the motion examples used in this thesis. If other annotations are needed by an editing algorithm, some procedural approach to applying such annotations might need to be considered.

Some degree of noise exists in all recorded motion, as it is reconstructed from data recorded using physical devices. While motion reconstruction pipelines typically apply filtering to reduce this noise, in practice, the degree to which this is done varies significantly. The extreme identification algorithms presented in Chapter 5 address this by either filtering data or by using small temporal windows to identify local extrema. New extreme identification algorithms would likely need to use similar techniques to take this noise into account.

Noise can also effect how well editing algorithms preserve motion quality. For example, the retiming tools in the staggered poses system are sensitive to correlated noise in rotation parameters. This correlated noise results from the use of per–frame inverse kinematic solvers in motion recording software. However, as this skeletal fitting is done per–frame, applying timing refinements can de–correlate this noise, which can manifest itself as increased jerkiness in the motion. This can be avoided by filtering motion *after* fitting it to a skeletal parameterization, rather than before.

The motion itself might also break assumptions implicit in the extreme identification

and editing algorithms. For example, the extreme identification algorithms all search for extrema of some measure. In recorded motion, the character never stops moving; even in standing motion, there will be some subtle movement. Animators intentionally add this when creating animated motion of a still character [85]. However, if the body or joints are deliberately held still, there might not be specific times at which local extrema exist. The systems in this thesis do not consider this possibility, as they have been designed for editing recorded human motion data.

9.2.6 Address Practical Aspects of User Modeling

A key decision to be made is what algorithmic representation to present to the user. One specific choice to be made relates to extreme selection. In some cases, it would be good to expose this to the user, especially if it is intended that they directly edit a sparse set of motion extrema, such as extreme poses. In other cases, it would be better to encapsulate the extreme selection process. For example, spatial exaggeration encapsulates the use of extreme poses or extreme joint configurations. However, expert users might benefit from removing the encapsulation. For example, in spatial exaggeration, novice animators might want to work only with high–level parameters. Expert users, such as professional animators, would likely want to use these parameters to apply a procedural edit that results in motion that is close to their editing goal. They could then change the resulting poses or joint configurations to refine the algorithmic output.

Another goal to consider is the presentation of an intuitive algorithmic parameterization. A recurring example for the editing systems presented in this thesis is the set of parameters users interact with to edit a pose. In these systems, users interact with Maya's pose editing tools to change the poses. Maya's skeletal parameterization is then translated to the pose representation used by the editing systems, and the motion editing results are translated back to Maya's skeletal parameterization.

Another example of choosing an intuitive algorithmic parameterization occurs in the

spatial exaggeration system. In this system, the exaggeration parameter e is mapped to changes in character poses in such a way that it appears as though the user is "scaling" how much the character moves.

User interfaces are also important to design well if the goal is to develop a practical motion editing system. Two examples of such systems are commercial software and software developed for studying new character animation interface designs. As user interface design is outside the scope of this thesis, it has not been considered in depth for the editing systems presented here. However, to describe the types of issues that would arise in designing effective user interfaces, the staggered pose representation will be considered briefly.

When editing a staggered pose, users can change parameter values and timing refinements. To practically work with this data representation, some visualization aid would be needed that would relate the timing refinement values to the structure of the character, as patterns in the timing refinements are related to this structure. The user might also want to visualize which procedural editing tools have been applied to which parts of the character body. Finally, some interactive interface would be needed for manually editing timing refinements; the existing system requires values to be typed in.

9.3 Feedback from Professional Animators

The editing systems presented in this thesis have been designed and developed to demonstrate the viability of using motion extrema as a foundation for expressive motion editing. As the focus of this work has been on algorithmic development, the user interfaces for these systems remain functional, yet sparse. Ideally, it would be good to develop user interfaces that provide functionality comparable to those that exist in commercial animation software. This would allow for the comparison of time cost and user satisfaction when various edits are applied. However, in the case of interactive character animation

tools, developing effective, interactive, user interfaces requires a significant development effort, and this remains beyond the scope of this thesis.

However, to gain feedback on the motion editing systems presented here and the results from Chapters 6, 7, and 8, these systems and results have been demonstrated to a variety of professional animators and technical directors[1]. The goal of these demonstrations has been to collect feedback, gauge interest in the ideas and approaches, and collect additional ideas for future work.

The systems were demonstrated to approximately twelve individuals to collect feedback[2]. The formality and length of the demonstrations and resulting discussions varied significantly, depending on time availability and the person's background and interest. All have professional film production experience, ranging from independent short films to major feature films. A number of these individuals also have software development or interface design experience. Specific animation backgrounds of these individuals include character animation, motion editing, character design, film direction, single–person film production, visual effects animation, crowd simulation, animation software development, and animation software design.

The remainder of this section will review comments made in a variety of areas. These are summarized as points of interest, critiques, and suggestions for future extensions.

9.3.1 Points of Interest

All individuals expressed interest in the editing techniques and how they could be applied to not only character motion editing, but also other applications related to both character motion and simulation data. This can be considered a biased point of feedback, however, as animation technology is constantly changing, and demonstrations of new technology

[1] In this context, the term technical director refers to an animator who is capable of developing new software, as needed, to create or edit motion.

[2] More than twelve people have been given demonstrations, but approximately twelve provided meaningful feedback.

are commonly met with interest by animation professionals. A more objective interpretation of this consistent interest is that no individuals felt the ideas included in these motion editing systems are bad ideas or headed in the wrong direction. A number of specific points are summarized below.

Professional character animators consistently expressed a need for simple, but specific, control over motion. A common opinion is that any procedural system, including the procedural editing tools included here, will likely allow them to get close to specific editing goals quickly. However, they would likely want to refine the output of any such system. A common concern about procedural systems, in general, is that systems which treat motion as a black box tend to create dense data that can not be refined in a practical way.

In asking for further detail about what type of simplified representation might be useful, a consistent suggestion was the simplification of dense motion data into animation splines with sparse knots that are organized such that users can find meaningful poses. This is consistent with how the staggered poses system identifies poses with associated timing relationships and represents them using a concise data structure. Admittedly, the idea of using a sparse set of extreme–based editing controls in each of these systems was motivated by the author's professional experience working with simulated motion data. Such data is also difficult to manually refine if it is not organized into a simplified, meaningful form.

Individuals with a film direction background were particularly interested in how the performance–related edits reflect the personality of the character. One interesting suggestion is the creation of a psychological model that would relate personality and individual motivations to specific types of motion edits.

Individuals with experience using simulators to create motion, such as physical simulation systems and crowd simulation systems, were particularly interested in the ability to choose a sparse set of meaningful poses for editing simulation output. These individ-

uals would like to see this idea adapted to parametric splines, since high–quality editing interfaces for real–valued splines are available in current animation systems.

Software designers, unsurprisingly, were especially interested in algorithmic capabilities related to applying pose–centric edits and procedural edits. Again, a specific interest was seeing the idea of intelligent extreme identification applied to the problem of simplifying dense motion data for practical editing control.

9.3.2 Critiques

The staggered pose idea requires a new mental model of how to organize data. Not all people find this immediately intuitive. This is likely due to the fact that it is a space–time data representation, and it can not be visualized as as single frame within a piece of motion. The challenge of understanding space–time data representations is not unexpected; student animators commonly cite spline editors as one of the most challenging tools to gain an intuitive understanding of, even though they remain one of the most commonly used interfaces for working with motion data. To address this challenge, an effective interface for visualizing and editing staggered poses would be necessary.

Another common request was to see the results of the motion edits applied to rigged characters, especially stylized characters. However, this opinion has not been consistent. Some individuals feel simplified representations, such as stick figures or the ellipsoid model used here, remain superior for evaluating motion, as rigs can hide problems in motion data. As an example, skinning–based rigs can be set up in such a way that some skeletal parameters might have a negligible effect on the motion of the character geometry. The perceptual tradeoff between demonstrating algorithmic motion on rigged characters or simplified skeletal models has not been well studied, however. It would be good to investigate this in more depth, as opinions expressed by both researchers and practitioners, in the author's experience, are typically strong, but not consistent.

9.3.3 Suggestions for Extensions

As evident from comments discussed above, the most common suggestion is the development of a motion extreme selection algorithm that could select meaningful knots in a dense motion data signal specified for a single real–valued parameter. It is likely that that this idea has been recurring due to the fact that, in current systems for editing recorded or simulated motion, users can directly modify the parameter samples that control specific parameters. However, due to data density, this remains tedious, and choosing meaningful knots would allow the dense motion data to be converted into an approximating spline representation that places knots in expected locations.

The application of procedural editing tools to other types of motion data is another commonly suggested extension. In particular, simulated deforming meshes and articulated rigid bodies have been mentioned as examples. While these applications have also been considered by the author as areas for future study, this feedback further motivates these applications.

Another commonly–mentioned extension is the idea of using motion extrema as a foundation for action identification and motion segmentation. This is an example of a higher–level data annotation application, and motion extrema could be used as a foundation for detecting recurring patterns in motion data. For example, the extreme poses that result from a walk, while varying slightly from individual to individual, are similar in structure across individuals. As an example application, a clustering algorithm could be applied to extreme poses for automatically segmenting and annotating large motion databases to label different types of movement.

9.4 Toward a User–Centric Evaluation

In developing new algorithms for identifying motion extrema and introducing motion editing systems that use motion extrema as a base representation, this thesis has shown

that motion extrema can be used as a simplified foundation for expressive motion editing. While this work has been focused on solving algorithmic challenges related to these problems, it remains as future work to more formally evaluate how well these approaches assist users with tasks related to expressive motion editing. This section describes a variety of issues that would need to be considered when designing and carrying out a user study that seeks to answer these questions, including user interface needs, evaluation criteria, and the selection of meaningful editing tasks.

An important consideration when designing a user evaluation is the target audience: who is the intended user? For animation systems, the experience of the user can significantly affect how much information should be presented and how the interface should be designed. For example, novice users would likely not want to know details about motion representations and how algorithms function. They would likely accept the limits of a given algorithm, and work with it in an exploratory manner. More specifically, algorithms should be presented using simplified, high–level parameterizations, such as those used in the procedural timing tools presented in Chapter 7 and the spatial exaggeration system (Chapter 8). User interfaces would need to hide complexity, such as in the inverse kinematic posing tools that were used to edit poses for many of the examples presented in Chapters 6 and 7.

Expert users would have significantly different needs, however. Expert users of animation software are accustomed to directly working with motion representations such as splines, constraints, and displacement maps. When procedural algorithms are used, such as those presented in Chapters 7 and 8, such users would expect to be able to refine the output, as they often have a specific desired result in mind. Simplified parameterizations for such algorithms, instead of being an exclusive set of controls for producing the final motion, should be presented as controls for assisting users to achieve a desired result in a more expressive way and at a faster pace. Manual refinement of algorithmic output would be expected. For example, if building a system to evaluate the effectiveness of the

spatial exaggeration system for expert users, the system should allow them to directly edit the resulting poses to refine the algorithm output instead of abstracting them away. Finally, evaluations should be carried out in a way that does not limit expert users from being able to take advantage of existing tools with which they are familiar. For example, such a system should integrate well with common tools such as spline editors, spreadsheet interfaces, and inverse kinematic posing tools. If such capabilities were not present, it could bias the feedback in a negative way, as it could appear to the user as a system that removes capabilities that they regularly rely on.

In conducting an evaluation, some metric needs to be defined to determine how well the motion editing system succeeds at meeting a particular goal. This metric can be defined in a variety of ways. For example, the time it takes a user to complete an editing task could be recorded and compared for different editing systems. Ease of use, which can be related to completion time, could also be evaluated, although such a metric might need to be measured subjectively using survey data reported by the users. In addition, user satisfaction can be considered, which would also require a subjective response from the users. In general, one or more of these criteria can be considered; they should be chosen to evaluate both the motion editing algorithm and the associated user interface. For example, interfaces for expert users might be optimized to allow users to work faster, while interfaces for novices might be optimized to produce high user satisfaction. Evaluation criteria should consider this choice of target user.

Of particular importance in the design of the evaluation is the selection of the editing task that the user will perform. For example, this could involve the user changing the performance of the character or changing the constraints that relate the motion of the character to the environment. Some editing tasks might require a variety of tools, including path editing tools or motion sequencing and concatenation tools. The chosen editing tasks should reflect the design goals of the algorithms or systems being evaluated. For much of the work presented in this thesis, appropriate editing tasks could require the

user to meet specific target poses or to induce particular timing changes.

Also of importance is the context in which the user is given direction. Different applications would require users to make changes to meet different needs. For example, motion that is edited for later on-demand reuse in interactive systems has different requirements than the more carefully–crafted changes that would be required for high–quality film use. In both cases, it would be useful to conduct a preliminary observational study to observe the types of changes users make in these settings, as well as how often they are required to make them. The resulting set of common editing tasks would then serve as a guide for determining how well particular motion editing algorithms, including those presented in this thesis, succeed at allowing users to quickly and concisely make required motion edits.

9.5 Future Directions

This thesis has investigated three types of problem to demonstrate the use of a sparse set of motion extrema as a foundation for expressive motion editing. These three problems are the identification of motion extrema (Chapter 5), the development of algorithms that support the direct editing of motion extrema (Chapters 6 and 7), and the development of procedural tools that modify relationships among motion extrema (Chapters 7 and 8). While each of Chapters 6–8 discuss future directions of study related to specific editing systems, this section instead aims to identify major new directions of study that are larger in scope than improving or extending a particular system.

A major challenge is gaining a formal understanding of the mental models of motion and common workflow techniques used by animators. While the systems presented here use ideas related to animator–centric motion models and workflows, animators, in practice, use many other workflows that have not been documented in a formal setting. To gain such an understanding, one could formally study how animators currently use

existing motion creation and motion editing tools to identify recurring patterns in their workflows.

Gaining a greater understanding of motion extrema and how they are perceived would be also be useful. Immediate directions include the development of other approaches for identifying motion extrema and the study of how well other existing motion editing algorithms might benefit from using a motion representation in that is expressed in terms of a sparse set of motion extrema. From a perceptual standpoint, it would be interesting to study how well these different techniques for finding motion extrema relate to event segmentation. While the two approaches used here to find motion extrema are similar to theories from the event segmentation literature that explain mechanisms of event detection [182, 183], it would be good to study, in a rigorous sense, how the mental models of movement that animators use when creating motion relate to such principles.

It would also be useful to investigate the formal modeling of other relationships among motion extrema. The staggered pose and spatial extent models describe two such relationships, but it is likely that other relationships could be used to develop new procedural motion editing tools. For example, the coordinated editing of similar poses that occur at different times has been demonstrated [60]; this could be expressed as a relationship among extreme poses or extended to joint–centric motion extrema.

As mentioned in the previous section, action identification and the segmentation of motion based on action annotations are also potential directions of future study. To do this, the annotation of a particular action could be based on the state and timing of motion extrema. Actions could be parameterized using the variation in these extreme poses among multiple examples of the same action. Applications related to this include motion understanding, database organization, and motion search.

It would also be good to look at formally modeling other expressive aspects of movement. The techniques in this thesis focus on pose–to–pose style editing, the modeling of overlapping action, and the modeling of pose–based motion exaggeration. As an example

of another possibility, other types of timing, such as the timing of how a character moves from one extreme pose to another, could be formalized and encapsulated into editing tools. Squash and stretch effects and the geometric shape of trajectories in space could also be considered.

A key area for future investigation is the application of motion editing based on motion extrema to different types of motion. The results presented in this thesis focus on recorded motion capture data, due to both its availability and the motivating application of making performance–related changes to recorded data. The editing systems presented here, while using different motion representations, as were presented in Chapter 4, do assume that character motion is represented using articulated control skeletons. Other types of data include both data that has a similar representation, but differs in how it is created, as well as data that has a fundamentally different representation.

Motion that has a similar form to articulated character motion can be directly used by the motion editing systems presented in this thesis, provided that it is densely and uniformly sampled. For example, simulated character motion, in which control signals drive an articulated rigid body simulation, can be directly used. In addition, keyframe animation explicitly authored by an animator can be used if the motion is sampled at a regular rate; tools for such resampling are included in all the systems presented here. Articulated motion from other types of simulation models can also be used. For example, crowd simulation software often applies a variety of motion edits such as retiming, retargeting, path edits, and inverse kinematic constraint enforcement. Provided that the output of these systems can be converted to motion on an articulated control skeleton, the motion would be directly amenable to editing using the techniques presented here.

Other types of motion data can have fundamentally different representations, in that they cannot be represented using simplified articulated control skeletons. For example, deforming mesh data, such as facial animation or the output of deformable object simulations, does not have any intrinsic hierarchical control skeleton. Existing algorithms for

detecting motion extrema, identifying and enforcing contact constraints, measuring and editing timing refinements, and modeling and changing spatial extent would need to be either modified substantially or replaced with new models that have knowledge of the motion representation.

Free body motion, such as a rigid bouncing ball, while directly editable, is not currently handled by the spatial exaggeration model, as there is no notion of intrinsic stance for a freely moving rigid object. A new model of spatial extent that considers world space trajectories would be required to allow spatial exaggeration to be applied to this type of motion. Finally, approximations of continuous phenomena, such as fluid simulation grids and particle systems, would also require substantially new algorithms for identifying motion extrema and contact constraints, as well as new techniques for applying edits. User interface design would be more important for this class of motion, as its high dimensionally precludes practical specification of direct edits by users.

The motion editing systems presented here use kinematic models to apply changes to the original motion data. While this isolates the problems studied here from ongoing challenges related to the development of stable, efficient physics–based models of motion, it would be good to allow users to have a choice of technique for *how* specific motion edits are applied. For example, the expressive motion edits modeled here could still be expressed using a sparse set of motion extrema, but physical models of motion could be developed to apply them using constrained optimization or optimal control algorithms.

Finally, another potential direction for study is the development of animation–centric tools, in which the focus is on the creation of new motion, rather than the editing of existing motion. Key to this would be the development of effective algorithmic encapsulations of workflow techniques and animation principles. For example, users might specify a small set of poses and apply expressive style transforms. Full motion could then be generated that includes automatic handling of environmental relationships, such as contacts and obstacles.

9.6 Final Thoughts

This thesis has demonstrated the viability of using motion extrema as a foundation for expressive motion editing. To do so, a number of extreme identification algorithms have been introduced; these techniques have been developed to identify full–body extreme poses, as well as extreme states of individual joints in motion. Three applications have been developed that allow users to apply expressive motion edits that use motion extrema as a foundation. Two of them enable direct editing by users, and two include tools for applying high–level edits that procedurally modify the relationships among motion extrema.

Looking forward, the overall goal of this line of work is to develop animation and motion editing tools that are approachable to novice users, but have additional, discoverable, editing capabilities that are appropriate for expert users. To do this, it would be necessary to formalize many animation principles and workflow techniques on solid computational foundations, and to build animation and motion editing algorithms that encapsulate these principles and techniques.

Appendix A

Constraining a Rotation to a Given Axis

Some algorithms presented in this thesis need to constrain a 3D rotation to be similar to the original, but about the world up axis. Doing so can change the effect of the rotation on points in space, as it is, in general, not possible to modify a rotation to be about another axis and have the same effect as the original. Instead, this Appendix presents a technique for finding a rotation about the the world up axis that has a similar effect as the original rotation, but does not increase the angle by which any given point is rotated.

Given a 3D rotation by θ about an axis $\mathbf{a}$ to constrain about the axis z, this technique finds a rotation angle $\hat{\theta}$ about the world up axis z that has a similar effect to the original rotation. To do this, choose a vector $\mathbf{b}$ in the xy plane and apply the rotation given by θ and $\mathbf{a}$. This is illustrated as $\mathbf{c} = R_{\theta,\mathbf{a}}(\mathbf{b})$ in Figure A.1. As $\mathbf{c}$ is not, in general, in the xy plane, project it to $\mathbf{d} = (c_x, c_y, 0)$. The resulting angle $\hat{\theta}$ is that between $\mathbf{b}$ and $\mathbf{d}$. Note that this angle is not unique, and depends on the choice of $\mathbf{b}$. Choose $\mathbf{b} = (a_x, a_y, 0)$, as this results in an angular contraction similar to that needed to constrain a 3D twist rotation of a leg to a 2D foot rotation in the ground plane.

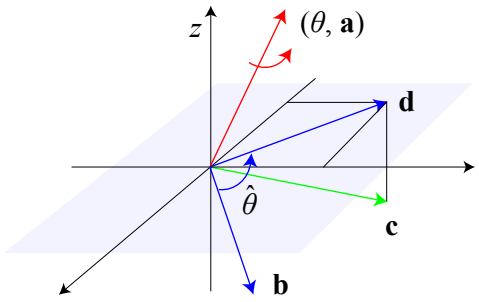


Figure A.1: To constrain a rotation to be about the up axis, the technique presented here finds a similar rotation about z.

Bibliography

[1] Yeuhi Abe, C. Karen Liu, and Zoran Popović. Momentum–Based Parameterization of Dynamic Character Motion. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 173–182, New York, NY, USA, 2004. ACM Press.

[2] Amr Ahmed, Farzin Mokhtarian, and Adrian Hilton. Enriching Animation Databases. In *Eurographics 2004 Short Papers*, 2004.

[3] Brian Allen, Derek Chu, Ari Shapiro, and Petros Faloutsos. On the Beat!: Timing and Tension for Dynamic Characters. In *Proceedings of SCA 2007*, pages 239–247, 2007.

[4] Kenji Amaya, Armin Bruderlin, and Tom Calvert. Emotion from Motion. In *Graphics Interface 1996*, 1996.

[5] Okan Arikan and D. A. Forsyth. Interactive Motion Generation from Examples. In *SIGGRAPH 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 483–490, New York, NY, USA, 2002. ACM Press.

[6] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion Synthesis from Annotations. *ACM Trans. Graph.*, 22(3):402–408, 2003.

[7] Okan Arikan, David A. Forsyth, and James F. O'Brien. Pushing People Around. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 59–66, New York, NY, USA, 2005. ACM Press.

[8] Golam Ashraf and Kok Cheong Wong. Generating Consistent Motion Transition via Decoupled Framespace Interpolation. In *Eurographics 2000*, 2000.

[9] Golam Ashraf and Kok Cheong Wong. Constrained Framespace Interpolation. In *Computer Animation 2001*, 2001.

[10] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action Synopsis: Pose Selection and Illustration. *ACM Trans. Graph.*, 24(3):667–676, 2005.

[11] Norman I. Badler, Ramamani Bindiganavale, John P. Granieri, Susanna Wei, and Xinmin Zhao. Posture Interpolation with Collision Avoidance. In *Proceedings of Computer Animation 1994*, pages 13–20, 1994.

[12] Paolo Baerlocher and Ronen Boulic. An Inverse Kinematics Architecture Enforcing an Arbitrary Number of Strict Priority Levels. *The Visual Computer*, 20(6):402–17, 2004.

[13] Jernej Barbič, Alla Safonova, Jia–Yu Pan, Christos Faloutsos, Jessica Hodgins, and Nancy Pollard. Segmenting Motion Capture Data into Distinct Behaviors. In *Proceedings of Graphics Interface 2004*, pages 185–194, 2004.

[14] Philippe Beaudoin, Stelian Coros, Michiel van de Panne, and Pierre Poulin. Motion–Motif Graphs. In *Proceedings of SCA 2008*, pages 117–126, 2008.

[15] Rama Bindiganavale and Norman I. Badler. Motion Abstraction and Mapping with Spatial Constraints. In *CAPTECH 1998: The Modeling and Motion Capture Techniques for Virtual Environments International Workshop*, pages 70–82, 1998.

[16] Leslie Bishko. The Uses and Abuses of Cartoon Style in Animation. *Animation Studies*, 2:24–35, 2007.

[17] Simon Bouvier-Zappa, Victor Ostromoukhov, and Pierre Poulin. Motion Cues for Illustration of Skeletal Motion Capture Data. In *Proceedings of NPAR*, pages 133–140, 2007.

[18] Matthew Brand and Aaron Hertzmann. Style Machines. In *SIGGRAPH 2000: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000.

[19] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. Turning to the Masters: Motion Capturing Cartoons. In *SIGGRAPH 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2002. ACM Press.

[20] Armin Bruderlin and Lance Williams. Motion Signal Processing. In *SIGGRAPH 1995: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 97–104, New York, NY, USA, 1995. ACM Press.

[21] Meeran Byun and Normal I. Badler. FacEMOTE: Qualitative Parametric Modifiers for Facial Animations. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 65–71, New York, NY, USA, 2002. ACM Press.

[22] Yong Cao, Petros Faloutsos, Eddie Kohler, and Frédéric Pighin. Real–Time Speech Motion Synthesis From Recorded Motions. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 345–353, New York, NY, USA, 2004. ACM Press.

[23] Yong Cao, Wen C. Tien, Petros Faloutsos, and Frédéric Pighin. Expressive Speech–Driven Facial Animation. *ACM Trans. Graph.*, 24(4):1283–1302, 2005.

[24] Marc Cardle, Loic Barthe, Stephen Brooks, and Peter Robinson. Music–Driven Motion Editing: Local Motion Transformations Guided by Music Analysis. In *Eurographics 2002*, 2002.

[25] John Chadwick, Dave Haumann, and Rick Parent. Layered Construction for Deformable Animated Characters. In *SIGGRAPH 1989: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, pages 243–252, 1989.

[26] Jinxiang Chai and Jessica Hodgins. Constraint–Based Motion Optimization Using a Statistical Dynamic Model. *ACM Trans. Graph.*, 26, July 2007.

[27] Diane Chi, Monica Costa, Liwei Zhao, and Norman Badler. The EMOTE Model for Effort and Shape. In *SIGGRAPH 2000: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 173–183, 2000.

[28] Kwang-Jin Choi and Hyeong-Seok Ko. Online Motion Retargeting. *The Journal of Visualization and Computer Animation*, 1:223–235, 2000.

[29] Min Gyu Choi, Jehee Lee, and Sung Yong Shin. Planning Biped Locomotion Using Motion Capture Data and Probabilistic Roadmaps. *ACM Trans. Graph.*, 22(2):182–203, 2003.

[30] Erika Chuang and Christoph Bregler. Mood Swings: Expressive Speech Animation. *ACM Trans. Graph.*, 24(2):331–347, 2005.

[31] Michael F. Cohen. Interactive Spacetime Control for Animation. In *SIGGRAPH 1992: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 293–302, New York, NY, USA, 1992. ACM Press.

[32] Michael Comet. Character Animation: Principles and Practice. *http://www.comet-cartoons.com/3ddocs/charanim/*, 1999.

[33] Seth Cooper, Aaron Hertzmann, and Zoran Popović. Active Learning for Real–Time Motion Controllers. *ACM Trans. Graph.*, 26, 2007.

[34] Christina de Juan and Bobby Bodenheimer. Cartoon Textures. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 267–276, New York, NY, USA, 2004. ACM Press.

[35] Christina de Juan and Bobby Bodenheimer. Re–Using Traditional Animation: Methods for Semi–Automatic Segmentation and Inbetweening. In *SCA 2006: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006.

[36] Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. Feature–Based Locomotion Controllers. *ACM Trans. Graph.*, 22(3), 2010.

[37] Zhigang Deng and Ulrich Neumann. eFASE: Expressive Facial Animation Synthesis and Editing with Phoneme–Isomap Controls. In *SCA 2006: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2006.

[38] Mira Dontcheva, Gary Yngve, and Zoran Popović. Motion Synthesis from Annotations. *ACM Trans. Graph.*, 22(3), 2003.

[39] Anthony C. Fang and Nancy S. Pollard. Efficient Synthesis of Physically Valid Human Motion. *ACM Trans. Graph.*, 22(3):417–426, 2003.

[40] Ran Gal, Olga Sorkine, Niloy Mitra, and Daniel Cohen–Or. iWIRES: An Analyze–and–Edit Approach to Shape Manipulation. *ACM Trans. Graph.*, 28:33:1–33:10, July 2009.

[41] Michael Gleicher. Motion Editing with Spacetime Constraints. In *I3D 1997: Proceedings of the 1997 Symposium on Interactive 3D Graphics*, New York, NY, USA, 1997. ACM Press.

[42] Michael Gleicher. Retargetting Motion to New Characters. In *SIGGRAPH 1998: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 33–42, New York, NY, USA, 1998. ACM Press.

[43] Michael Gleicher. Comparative Analysis of Constraint–Based Motion Editing Methods. In *Workshop on Human Modeling and Analysis 2000*, 2000.

[44] Michael Gleicher. Comparative Analysis of Constraint–Based Motion Editing Methods. *Graphical Models*, 63(2):107–134, 2001.

[45] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap–Together Motion: Assembling Run–Time Animations. In *I3D 2003: Proceedings of the 2003 Symposium on Interactive 3D Graphics*, pages 181–188, New York, NY, USA, 2003. ACM Press.

[46] Eric Goldberg. *Character Animation Crash Course!* Silman–James Press, 2008.

[47] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style–Based Inverse Kinematics. *ACM Trans. Graph.*, 23(3):520–531, 2004.

[48] Shang Guo and James Robergé. A High–Level Control Mechanism for Human Locomotion Based on Parametric Frame Space Interpolation. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation 1996*, pages 95–107, New York, NY, USA, 1996. Springer-Verlag New York, Inc.

[49] Rachel Heck and Michael Gleicher. Parametric Motion Graphs. In *Proceedings of I3D 2007*, pages 129–136, 2007.

[50] Rachel Heck, Lucas Kovar, and Michael Gleicher. Splicing Upper–Body Actions with Locomotion. In *Proceedings of Eurographics 2006*, 2006.

[51] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real–Time Motion Retargeting to Highly Varied User–Created Morphologies. *ACM Trans. Graph.*, 27(3), 2008.

[52] Edmond Ho, Taku Komura, and Chiew–Lan Tai. Spatial Relationship Preserving Character Motion Adaptation. *ACM Trans. Graph.*, 29, July 2010.

[53] Berthold Horn. Closed–Form Solution of Absolute Orientation Using Unit Quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987.

[54] Eugene Hsu, Marco da Silva, and Jovan Popović. Guided Time Warping for Motion Editing. In *Proceedings of SCA*, 2007.

[55] Eugene Hsu, Sommer Gentry, and Jovan Popović. Example–Based Control of Human Motion. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 69–77, New York, NY, USA, 2004. ACM Press.

[56] Eugene Hsu, Kari Pulli, and Jovan Popović. Style Translation for Human Motion. *ACM Trans. Graph.*, 24(3), 2005.

[57] Ke–Sen Huang, Chun–Fa Chang, Yu–Yao Hsu, and Shi–Nine Yang. Key Probe: A Technique for Animation Keyframe Extraction. *The Visual Computer*, 21(8–10):532–541, 2005.

[58] Leslie Ikemoto, Okan Arikan, and David Forsyth. Knowing When to Put Your Foot Down. In *I3D 2006: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, New York, NY, USA, 2006. ACM Press.

[59] Leslie Ikemoto, Okan Arikan, and David Forsyth. Quick Transitions with Cached Multi–Way Blends. In *Proceedings of I3D 2007*, pages 145–151, 2007.

[60] Leslie Ikemoto, Okan Arikan, and David Forsyth. Generalizing Motion Edits with Gaussian Processes. *ACM Trans. Graph.*, 28(1), 2009.

[61] Leslie Ikemoto and David A. Forsyth. Enriching a Motion Collection by Transplanting Limbs. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 99–108, New York, NY, USA, 2004. ACM Press.

[62] Eakta Jain, Yaser Sheikh, and Jessica Hodgins. Leveraging the Talent of Hand Animators to Create Three–Dimensional Animation. In *Proceedings of SCA 2009*, pages 93–102, 2009.

[63] Sumit Jain and C. Karen Liu. Interactive Synthesis of Human–Object Interaction. In *Proceedings of SCA 2009*, pages 47–53, 2009.

[64] Eunjung Ju, Myung Geol Choi, Minji Park, Jehee Lee, Kang Hoon Lee, and Shigeo Takahashi. Morphable Crowds. *ACM Trans. Graph.*, 29, 2010.

[65] Paul Kanyuk. Brain Springs: Fast Physics for Large Crowds on WALL·E. *IEEE Computer Graphics and Applications*, 29:19–25, 2009.

[66] Michael Kass and John Anderson. Animating Oscillatory Motion with Overlap: Wiggly Splines. *ACM Trans. Graph.*, 27(3), 2008.

[67] Oussama Khatib, Luis Sentis, Jae–Heung Park, and James Warren. Whole Body Dynamic Behavior and Control of Human–Like Robots. *International Journal of Humanoid Robotics*, 1(1):1–15, 2004.

[68] Jong–Hyuk Kim, Jung–Ju Choi, Hyun Shin, and In-Kwon Lee. Anticipation Effect Generation for Character Animation. In Tomoyuki Nishita, Qunsheng Peng, and

Hans-Peter Seidel, editors, *Advances in Computer Graphics*, volume 4035 of *Lecture Notes in Computer Science*, pages 639–646. Springer Berlin / Heidelberg, 2006.

[69] Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. Synchronized Multi–Character Motion Editing. *ACM Trans. Graph.*, 28(3), 2009.

[70] Tae–hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic–Motion Synthesis Based on Motion–Beat Analysis. *ACM Trans. Graph.*, 22(3):392–401, 2003.

[71] Taku Komura, Howard Leung, and James Kuffner. Animating Reactive Motions for Biped Locomotion. In *VRST 2004*, 2004.

[72] Lucas Kovar and Michael Gleicher. Flexible Automatic Motion Blending with Registration Curves. In *SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

[73] Lucas Kovar and Michael Gleicher. Automatic Extraction and Parameterization of Motions in Large Data Sets. *ACM Trans. Graph.*, 23(3), 2004.

[74] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion Graphs. In *SIGGRAPH 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 473–482, New York, NY, USA, 2002. ACM Press.

[75] Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate Cleanup for Motion Capture Editing. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 97–104, New York, NY, USA, 2002. ACM Press.

[76] Paul G. Kry and Dinesh K. Pai. Interaction Capture and Synthesis. *ACM Trans. Graph.*, 25(3), 2006.

[77] R. Kulpa, F. Multon, and B. Arnaldi. Morphology–Independent Representation of Motions for Interactive Human–Like Animation. In *Eurographics 2005*, 2005.

[78] Ji-Yong Kwon and In-Kwon Lee. Rubber–Like Exaggeration for Character Animation. In *Proceedings of Pacific Graphics*, 2007.

[79] Taesoo Kwon, Kang Hoon Lee, Jehee Lee, and Shigeo Takahashi. Group Motion Editing. *ACM Trans. Graph.*, 27, August 2008.

[80] Taesoo Kwon and Sung Yong Shin. Motion Modeling for On–Line Locomotion Synthesis. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2005. ACM Press.

[81] Rudolph Laban. *The Mastery of Movement*. Plays, Inc., 1971.

[82] Yu-Chi Lai, Stephen Chenney, and ShaoHua Fan. Group Motion Graphs. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 281–290, New York, NY, USA, 2005. ACM Press.

[83] Alexis Lamouret and Michiel van de Panne. Motion Synthesis by Example. In *Proceedings of the 1996 Eurographics Workshop on Animation and Simulation*, 1996.

[84] John Lasseter. Principles of Traditional Animation Applied to 3D Computer Animation. In *SIGGRAPH 1987: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 35–44, New York, NY, USA, 1987. ACM Press.

[85] John Lasseter. Tricks to Animating Characters with a Computer. In *SIGGRAPH 1994 Course Notes*, 1994.

[86] Manfred Lau, Ziv Bar–Joseph, and James Kuffner. Modeling Spatial and Temporal Variation in Motion Data. *ACM Trans. Graph.*, 28(5), 2009.

[87] Benoît Le Callennec and Ronan Boulic. Interactive Motion Deformation with Prioritized Constraints. In *SCA 2004: Proceedings of the 2004 ACM SIG-*

GRAPH/Eurographics Symposium on Computer Animation, New York, NY, USA, 2004. ACM Press.

[88] Benoît Le Callennec and Ronan Boulic. Robust Kinematic Constraint Detection for Motion Data. In *SCA 2006: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2006. ACM Press.

[89] Hyun-Chul Lee and In-Kwon Lee. Automatic Synchronization of Background Music and Motion in Computer Animation. In *Proceedings of Eurographics 2005*, 2005.

[90] 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 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 491–500, New York, NY, USA, 2002. ACM Press.

[91] Jehee Lee and Kang Hoon Lee. Precomputing Avatar Behavior from Human Motion Data. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 79–87, New York, NY, USA, 2004. ACM Press.

[92] Jehee Lee and Sung Yong Shin. A Hierarchical Approach to Interactive Motion Editing for Human–Like Figures. In *SIGGRAPH 1999: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[93] Jehee Lee and Sung Yong Shin. General Construction of Time–Domain Filters for Orientation Data. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):119–128, 2002.

[94] Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. Motion Patches: Building Blocks for Virtual Environments Annotated with Motion Data. *ACM Trans. Graph.*, 25(3), 2006.

[95] Tong-Yee Lee, Chao-Hung Lin, Yu-Shuen Wang, and Tai-Guang Chen. Animation Key–Frame Extraction and Simplification Using Deformation Analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(4):478–486, 2008.

[96] Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. Motion Fields for Interactive Character Locomotion. *ACM Trans. Graph.*, 29, December 2010.

[97] Sergey Levine, Philipp Krähenbühl, Sebastian Thrun, and Vladlen Koltun. Gesture Controllers. *ACM Trans. Graph.*, 29, July 2010.

[98] Sergey Levine, Christian Theobalt, and Vladlen Koltun. Real–Time Prosody–Driven Synthesis of Body Language. *ACM Trans. Graph.*, 28, December 2009.

[99] Q. Li, W. Geng, T. Yu, X. Shen, N. Lau, and G. Yu. MotionMaster: Authoring and Choreographing Kung–Fu Motions by Sketch Drawings. In *Proceedings of SCA 2006*, pages 233–241, 2006.

[100] Shiyu Li, Masahiro Okuda, and Shin–ichi Takahashi. Embedded Key–Frame Extraction for CG Animation by Frame Decimation. In *Proceedings of IEEE Multimedia*, pages 1404–1407, 2005.

[101] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion Texture: A Two–Level Statistical Model for Character Motion Synthesis. *ACM Trans. Graph.*, 21(23), 2002.

[102] Yin Li, Michael Gleicher, Ying-Qing Xu, and Heung-Yeung Shum. Stylizing Motion with Drawings. In *SCA 2003: Proceedings of the 2003 ACM SIG-*

GRAPH/Eurographics Symposium on Computer Animation, pages 309–319, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[103] Ik Soo Lim and Daniel Thalmann. Key–Posture Extraction out of Human Motion Data by Curve Simplification. In *Proceedings of the IEEE Conf. on Engineering in Medicine and Biology*, 2001.

[104] Peter Litwinowicz. Inkwell: A $2\frac{1}{2}$D Animation System. In *Proceedings of SIGGRAPH*, pages 113–122, 1991.

[105] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning Physics–Based Motion Style with Nonlinear Inverse Optimization. *ACM Trans. Graph.*, 24(3):1071–1081, 2005.

[106] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Composition of Complex Optimal Multi–Character Motions. In *SCA 2006: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006.

[107] C. Karen Liu and Zoran Popović. Synthesis of Complex Dynamic Character Motion from Simple Animations. In *SIGGRAPH 2002: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 408–416, New York, NY, USA, 2002. ACM Press.

[108] Feng Liu, Yueting Zhuang, Fei Wu, and Yunhe Pan. 3D Motion Retrieval with Motion Index Tree. *Computer Vision and Image Understanding*, 94:265–284, 2003.

[109] Zicheng Liu and Michael F. Cohen. Keyframe Motion Optimization by Relaxing Speed and Timing. In *1995 Eurographics Workshop on Animation and Simulation*, 1995.

[110] Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. Hierarchical Spacetime Control. In *SIGGRAPH 1994: Proceedings of the 21st Annual Conference on*

Computer Graphics and Interactive Techniques, pages 35–42, New York, NY, USA, 1994. ACM Press.

[111] Noah Lockwood and Karan Singh. Biomechanically–Inspired Motion Path Editing. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 267–276, 2011.

[112] Jiyong Ma, Ron Cole, Bryan Pellom, Wayne Ward, and Barbara Wise. Accurate Automatic Visible Speech Synthesis of Arbitrary 3D Models Based on Concatenation of of Diviseme Motion Capture Data. *Computer Animation and Virtual Worlds*, 15(5):485–500, 2004.

[113] Jiyong Ma, Ron Cole, Bryan Pellom, Wayne Ward, and Barbara Wise. Accurate Visible Speech Synthesis Based on Concatenating Variable Length Motion Capture Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):266–276, 2006.

[114] Xiaohan Ma, Binh Huy Le, and Zhigang Deng. Style Learning and Transferring for Facial Animation Editing. In *Proceedings of SCA 2009*, pages 123–132, 2009.

[115] A. Majkoqska, Victor Zordan, and Petros Faloutsos. Automatic Splicing for Hand and Body Animations. In *SCA 2006: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2006.

[116] Anna Majkowska and Petros Faloutsos. Flipping with Physics: Motion Editing for Acrobatics. In *Proceedings of SCA 2007*, pages 35–44, 2007.

[117] J. McCann, N. S. Pollard, and S. Srinivasa. Physics–Based Motion Retiming. In *SCA 2006: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006.

[118] James McCann and Nancy Pollard. Responsive Characters from Motion Fragments. *ACM Trans. Graph.*, 26, July 2007.

[119] Jianyuan Min, Yen–Lin Chen, and Jinxiang Chai. Interactive Generation of Human Animation with Deformable Motion Models. *ACM Transactions on Graphics.*, 29(1), 2009.

[120] Jianyuan Min, Huajun Liu, and Jinxiang Chai. Synthesis and Editing of Personalized Stylistic Human Motion. In *I3D 2010: Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 39–46, 2010.

[121] Luis Molina Tanco and Adrian Hilton. Realistic Synthesis of Novel Human Movements from a Database of Motion Capture Examples. In *HUMO 2000: Proceedings of the IEEE Workshop on Human Motion*, 2000.

[122] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical Motion Interpolation. *ACM Trans. Graph.*, 24(3), 2005.

[123] Tomohiko Mukai and Shigeru Kuriyama. Pose–Timeline for Propagating Motion Edits. In *Proceedings of SCA 2009*, pages 113–122, 2009.

[124] Michael Neff and Eugene Fiume. Modeling Tension and Relaxation for Computer Animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 81–88, New York, NY, USA, 2002. ACM.

[125] Michael Neff and Eugene Fiume. Aesthetic Edits for Character Animation. In *SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 239–244, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[126] Michael Neff and Eugene Fiume. Artistically Based Computer Generation of Expressive Motion. In *Proceedings of the AISB 2004 Symposium on Language, Speech, and Gesture for Expressive Characters*, 2004.

[127] Michael Neff and Eugene Fiume. Methods for Exploring Expressive Stance. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2004. ACM Press.

[128] Michael Neff and Eugene Fiume. AER: Aesthetic Exploration and Refinement for Expressive Character Animation. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2005. ACM Press.

[129] Michael Neff and Yejin Kim. Interactive Editing of Motion Style Using Drives and Correlations. In *Proceedings of SCA 2009*, pages 103–112, 2009.

[130] 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, March 2008.

[131] Nam Nguyen, Nkenge Wheatland, David Brown, Brian Parise, C. Karen Liu, and Victor Zordan. Performance Capture with Physical Interaction. In *Proceedings of the SCA 2010*, pages 189–195, 2010.

[132] Masaki Oshita and Akifumi Makinouchi. A Dynamic Motion Control Technique for Human–Like Articulated Figures. In *Eurographics 2001*, 2001.

[133] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. Bilateral Filtering: Theory and Applications. *Computer Graphics and Vision*, 4(1):1–73, 2009.

[134] Sang Il Park, Hyun Joon Shin, Tae Hoon Kim, and Sung Yong Shin. On–Line Motion Blending for Real–Time Locomotion Generation. In *Computer Animation and Virtual Worlds*, 2004.

[135] Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. On–Line Locomotion Generation Based on Motion Blending. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2002.

[136] Manuel Peinado, Bruno Herbelin, Marcelo Wanderley, Benoit Le Callennec, Ronan Boulic, Daniel Thalmann, Daniel Méziat, and Escuela Politécnica. Towards Configurable Motion Capture with Prioritized Inverse Kinematics. In *Proceedings of the Third International Workshop on Virtual Rehabilitation*, pages 85–97, 2004.

[137] Ken Perlin. Real Time Responsive Animation with Personality. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):5–15, 1995.

[138] Ken Perlin and Athomas Goldberg. Improv: A System for Scripting Interactive Actors in Virtual Worlds. In *SIGGRAPH 1996: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 205–216, New York, NY, USA, 1996. ACM Press.

[139] Zoran Popović; and Andrew Witkin. Physically Based Motion Transformation. In *SIGGRAPH 1999: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 11–20, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[140] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, 1992.

[141] Katherine Pullen and Christoph Bregler. Animating by Multi–Level Sampling. In *Computer Animation 2000*, 2000.

[142] Katherine Pullen and Christoph Bregler. Synthesis of Cyclic Motions with Texture. In *Stanford University Technical Report*, 2002.

[143] Liu Ren, Gregory Shakhnarovich, Jessica K. Hodgins, Hanspeter Pfister, and Paul Viola. Learning Silhouette Features for Control of Human Motion. *ACM Trans. Graph.*, 24(4):1303–1331, 2005.

[144] Charles Rose, Bobby Bodenheimer, and Michael F. Cohen. Verbs and Adverbs: Multidimensional Motion Interpolation Using Radial Basis Functions. *Computer Graphics and Applications*, 18(5):32–40, 1998.

[145] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient Generation of Motion Transitions Using Spacetime Constraints. In *SIGGRAPH 1996: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 147–154, New York, NY, USA, 1996. ACM Press.

[146] Charles F. Rose, Peter-Pike J. Sloan, and Michael F. Cohen. Artist–Directed Inverse Kinematics Using Radial Basis Function Interpolation. In *Eurographics 2001*, 2001.

[147] Alla Safanova and Jessica K. Hodgins. Analyzing the Physical Correctness of Interpolated Human Motion. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2005. ACM Press.

[148] Alla Safonova and Jessica Hodgins. Construction and Optimal Search of Interpolated Motion Graphs. *ACM Trans. Graph.*, 26, July 2007.

[149] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. Synthesizing Physically Realistic Human Motion in Low–Dimensional, Behavior–Specific Spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.

[150] Luis Sentis and Oussama Khatib. Control of Free–Floating Humanoid Robots Through Task Prioritization. In *Proceedings of the IEEE International Conference in Robotics and Automation*, 2005.

[151] Ari Shapiro, Yong Cao, and Petros Faloutsos. Style Components. In *Proceedings of Graphics Interface 2006*, 2006.

[152] Hyun Joon Shin, Lucas Kovar, and Michael Gleicher. Physical Touch–Up of Human Motions. In *Pacific Graphics 2003*, 2003.

[153] 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.

[154] Hyun Joon Shin and Hyun Seok Oh. Fat Graphs: Constructing an Interactive Character with Continuous Controls. In *Proceedings of SCA 2006*, pages 291–298, 2006.

[155] Takaaki Shiratori, Atsushi Nakazawa, and Kutsushi Ikeuchi. Dancing–to–Music Character Animation. In *Proceedings of Eurographics 2006*, 2006.

[156] Ken Shoemake. Animating Rotation with Quaternion Curves. In *Proceedings of SIGGRAPH*, pages 245–254, 1985.

[157] Hubert Shum, Taku Komura, Masashi Shiraishi, and Shuntaro Yamazaki. Interaction Patches for Multi–Character Animation. *ACM Trans. Graph.*, 27, December 2008.

[158] Kwang Won Sok, Katsu Yamane, Jehee Lee, and Jessica Hodgins. Editing Dynamic Human Motions via Momentum and Force. In *Proceedings of SCA 2010*, pages 11–20, 2010.

[159] Olga Sorkine, Daniel Cohen-Or, Taron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 175–184, 2004.

[160] Stephen Mann and Nathan Litke and Tony Derose. A Coordinate Free Geometry ADT. Technical report, 1997.

[161] Matthew Stone, Doug DeCarlo, Insuk Oh, Christian Rodriguez, Adrian Stere, Alyssa Lees, and Chris Bregler. Speaking with Hands: Creating Animated Conversational Characters from Recordings of Human Performance. *ACM Trans. Graph.*, 23(3):506–513, 2004.

[162] Adnan Sulejmanpašić and Jovan Popović. Adaptation of Performed Ballistic Motion. *ACM Trans. Graph.*, 24(1):165–179, 2005.

[163] Mankyu Sung, Lucas Kovar, and Michael Gleicher. Fast and Accurate Goal–Directed Motion Synthesis for Crowds. In *SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 291–300, New York, NY, USA, 2005. ACM Press.

[164] Seyoon Tak and Hyeong-Seok Ko. A Physically–Based Motion Retargeting Filter. *ACM Trans. Graph.*, 24(1):98–117, 2005.

[165] G. Taubin. Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation. In *Proc. ICCV*, 1995.

[166] S. C. L. Terra and R. A. Metoyer. Performance Timing for Keyframe Animation. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2004. ACM Press.

[167] Frank Thomas and Ollie Johnston. *The Illusion of Life: Disney Animation*. Abbeville Press, 1981.

[168] Hideyuki Togawa and Masahiro Okuda. Position-Based Keyframe Selection for Human Motion Animation. In *Proceedings of the International Conference on Parallel and Distributed Systems*, 2005.

[169] Adrien Treuille, Yongjoon Lee, and Zoran Popović. Near–Optimal Character Animation with Continuous Control. *ACM Trans. Graph.*, 26, July 2007.

[170] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier Principles for Emotion–Based Human Figure Animation. In *SIGGRAPH 1995: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 91–96, New York, NY, USA, 1995. ACM Press.

[171] Kevin Wampler, Erik Andersen, Evan Herbst, Yongjoon Lee, and Zoran Popović. Character Animation in Two–Player Adversarial Games. *ACM Trans. Graph.*, 29, July 2010.

[172] Jing Wang and Bobby Bodenheimer. Computing the Duration of Motion Transitions: An Empirical Approach. In *SCA 2004: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 335–344, New York, NY, USA, 2004. ACM Press.

[173] Jue Wang, Steven M. Drucker, Maneesh Agrawala, and Michael F. Cohen. The Cartoon Animation Filter. *ACM Trans. Graph.*, 25(3), 2006.

[174] Z. Wang and Michiel van de Panne. "Walk to Here:" A Voice Driven Animation System. In *SCA 2006: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2006.

[175] Harold Whitaker and John Halas. *Timing for Animation*. Focal Press, 1981.

[176] Douglas J. Wiley and James K. Hahn. Interpolation Synthesis of Articulated Figure Motion. *Computer Graphics and Applications*, 17(6):39–45, 1997.

[177] Richard Williams. *The Animator's Survival Kit*. Faber and Faber, 2001.

[178] Andrew Witkin and Michael Kass. Spacetime Constraints. In *SIGGRAPH 1988: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, pages 159–168, New York, NY, USA, 1988. ACM Press.

[179] Andrew Witkin and Zoran Popovic. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108, New York, NY, USA, 1995. ACM Press.

[180] Yuting Ye and C. Karen Liu. Animating Responsive Characters with Dynamic Constraints in Near–Unactuated Coordinates. *ACM Trans. Graph.*, 27, December 2008.

[181] KangKang Yin and Dinesh K. Pai. FootsSee: An Interactive Animation System. In *SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003.

[182] Jeff Zacks. Using Movement and Intentions to Understand Simple Events. *Cognitive Science*, pages 979–1008, 2004.

[183] Jeffrey Zacks, Shawn Kumar, Richard Abrams, and Ritesh Mehta. Using Movement and Intentions to Understand Human Activity. *Cognition*, 112:201–216, 2009.

[184] Liming Zhao, Aline Normoyle, Sanjeev Khanna, and Alla Safonova. Automatic Construction of a Minimum Size Motion Graph. In *Proceedings of SCA 2009*, pages 27–35, 2009.

[185] Liming Zhao and Alla Safonova. Achieving Good Connectivity in Motion Graphs. *Graph. Models*, 71:139–152, July 2009.

[186] Victor B. Zordan and Jessica K. Hodgins. Tracking and Modifying Upper–Body Human Motion Data with Dynamic Simulation. In *Computer Animation and Simulation 1999*, 1999.

[187] Victor Brian Zordan and Jessica K. Hodgins. Motion Capture–Driven Simulations that Hit and React. In *SCA 2002: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 89–96, New York, NY, USA, 2002. ACM Press.

[188] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic Response for Motion Capture Animation. *ACM Trans. Graph.*, 24(3):697–701, 2005.