Synthesis of Complex Dynamic Character Motion from Simple Animations

C. Karen Liu Zoran Popović

University of Washington

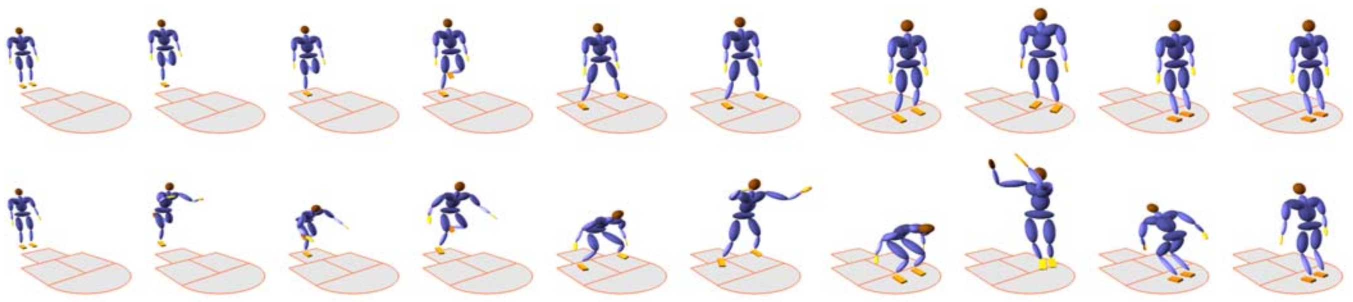
Figure 1 *Top:* Simple input animation depicting hopscotch (a popular child game consisting of hops, broad jumps and a spin jump). *Bottom:* Synthesized realistic hopscotch animation.

Abstract

In this paper we present a general method for rapid prototyping of realistic character motion. We solve for the natural motion from a simple animation provided by the animator. Our framework can be used to produce relatively complex realistic motion with little user effort.

We describe a novel constraint detection method that automatically determines different constraints on the character by analyzing the input motion. We show that realistic motion can be achieved by enforcing a small set of linear and angular momentum constraints. This simplified approach helps us avoid the complexities of computing muscle forces. Simpler dynamic constraints also allow us to generate animations of models with greater complexity, performing more intricate motions. Finally, we show that by learning a small set of key parameters that describe a character pose we can help a non-skilled animator rapidly create realistic character motion.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Animation, Animation w/Constraints, Physically Based Animation, Physically Based Modeling, Motion Transformation, Spacetime Constraints

1 Introduction

Generating realistic character animation remains one of the great challenges in computer graphics. To appear realistic, a character motion needs to satisfy the laws of physics, *and* stay within the space of naturally occurring movements. Simulated human models can move their muscles in many different ways to accomplish the same task, but only a small subset of these motions appears realistic. Creating such natural models of motion has proven to be an extremely difficult task, especially for characters as complex as humans.

Complex models of character dynamics are also difficult to control. Computing the correct dynamics requires an extensive mathematical infrastructure that often hinders artistic expressiveness. On the other hand, granting more control to animators provides greater expressive freedom often at the cost of realism because the burden of being physically correct falls into the animators' hands.

An ideal realistic character animation system should be able to synthesize natural motion of arbitrary characters, and, at the same time, provide the expressive power of keyframing. Furthermore, the system should allow non-skilled users to create realistic animations easily with minimal training. This paper strives to make a step in that direction. We present a novel approach for rapid prototyping of realistic character motion. We focus on the synthesis of highly dynamic movement such as jumping, kicking, running, and gymnastics. Less energetic motions such as walking or reaching are not addressed in this paper. Our system could be used by both experts and non-skilled animators alike.

The animator first creates a rough sketch of the desired animation. From this initial simple animation the system infers environmental constraints on the character motion (e.g. footsteps). We choose not to use the full character formulation of dynamics. In fact, we avoid solving for muscle forces altogether. Instead, we focus on determining more fundamental properties of realistic, highly dynamic motion and try to preserve these features throughout the process of animation. In particular, our system extracts the general patterns of linear and angular momentum transfer during dynamic motion and tries to preserve these patterns during animation. We ask the animator to fully specify a small set of specific keyframes. Since our target users are animators of all skilled levels, our system can also make suggestions for each of those keyframes. Finally, the animator can fine-tune the motion by specifying additional keyframes anywhere in the motion sequence.

The rest of the paper describes our approach in more detail. In Section 2, we discuss related work. Section 3 gives a short overview of our motion synthesis approach. Subsequent sections describe various aspects of our algorithms in more detail. In Section 9, we describe a collection of example animations generated by our system. Section 10 summarizes our contributions and outlines possible future research directions.

email: {karenliu,zoran}@cs.washington.edu

2 Related work

Synthesis of natural motion has its roots in a variety of research areas ranging from robotics and spacetime optimization to biomechanics and kinesiology.

Robot controller simulation has been successfully applied to the domain of realistic computer animation [Raibert and Hodgins 1991; van de Panne et al. 1994; van de Panne and Fiume 1993; Hodgins 1998]. Controllers drive actuator forces based on the current state of the environment. Actuator forces, through simulation, produce realistic motion. Once the controllers have been fine-tuned and synchronized to each other, a wide range of realistic animations can be produced, ranging from human running, diving [Hodgins et al. 1995], leaping and vaulting [Wooten 1998], to motion of non-human characters [van de Panne et al. 1994; Laszlo et al. 2000; Torkos and van de Panne 1998]. Although there have been some recent promising advances towards automatic controller synthesis [Hodgins and Pollard 1997; Faloutsos et al. 2001], creating controllers for a given task remains a difficult process. In addition, simulated robot controllers do not expose sufficient control to allow for expressive animations.

A number of researchers take the approach of modeling physical behavior on simpler machines, instead of full complex characters [Torkos and van de Panne 1998; Popović and Witkin 1999; van de Panne 1997; Pollard 1999; Discreet n. d.; Bruderlin and Calvert 1989]. In these methods, the physically modeled motion of simple machines is mapped onto the full character. Our simplified physics constraints are in part inspired by the idea that simpler physical models can approximate the behavior of more complex models. In contrast to the approach described by Popović [1999], we do not simplify the character, nor do we compute full dynamics of the simplified character. Instead, we compute significantly simpler momentum constraints directly on the complex character. Simpler and more general dynamics constraints allow us to synthesize realistic motion starting from highly unrealistic motion.

The spacetime constraints framework, in contrast to simulation, casts the motion synthesis into a variational optimization problem [Witkin and Kass 1988; Cohen 1992; Liu et al. 1994; Rose et al. 1996]. The animator specifies an objective function that is a metric of performance or style (e.g. total power consumption of all of the character's muscles). The algorithm minimizes the objective function while satisfying the pose and Newtonian physics constraints across all animation frames. Optimal energy movement and intuitive control give this method great appeal. Unfortunately, for complex characters the Newtonian physics constraints are highly nonlinear, preventing the spacetime optimization from converging to a solution. Spacetime constraints are also highly sensitive to the starting position of the optimization — if the initial state is far away from the solution, the optimization often does not converge. To date, these drawbacks prevent spacetime constraints from being used in generating complex character motion. Our framework uses spacetime constraint optimization, but we circumvent its drawbacks by choosing a simpler set of dynamics constraints.

Realistic motion can also be obtained directly from the real world. Recently, a number of methods for editing motion capture data have been proposed [Witkin and Popović 1995; Bruderlin and Williams 1995; Gleicher 1998; Gleicher 1997; Gleicher and Litwinowicz 1998; Rose et al. 1998; Gleicher 2001; Lee and Shin 1999; Shin et al. 2001], including a few that try to preserve physical properties of the motion [Popović and Witkin 1999; Pollard and Reitsma 2001; Pollard and Behmaram-Mosavat 2000; Zordan and Hodgins 1999]. In general, these methods produce motion that does not deviate significantly from the input motion. Motion editing tools rely on the existence of captured motion that is similar to what the animator intends to create. Also, it is inherently difficult to introduce new expressive content into the animations, since most editing tools are designed to preserve the original motion features.

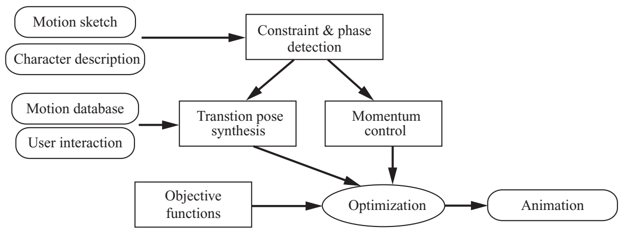

Figure 2 Algorithm overview.

Research in biomechanics and kinesiology provides a great source of information on the kinematic and dynamic behaviors of animals [Blickhan 1999; Hull 1991; Yeadon 1990; Alexander 1990; Alexander 1989; Pandy et al. 1990]. Their analysis of ground and flight stages helped us in designing realistic motion constraints. Blickhan and Full [1993] demonstrate the similarity in the multi-legged locomotion of kinematically different animals. They show striking similarities between a human run, a horse run and the monopod bounce (i.e. pogo-stick). This similarity motivates our approach of finding the least common denominator for a varied set of dynamic motions.

Our motion sketching approach to synthesizing motion was inspired by the effectiveness of Igarashi's sketching interface for 3d free-form design [1999], and the work on sketching realistic rigid-body motion [Popović et al. 2000].

3 Overview

Our system transforms simple animations into realistic character motion by applying laws of physics and the biomechanics domain knowledge. The input to our system consists of an articulated character with its mass distribution, and an arbitrary character animation containing values of joint angles on the character at each frame. Animators are free to provide input animations with an arbitrary level of detail. In all of our examples we started with rough low-quality animations. Figure 1 shows a synthesized realistic hopscotch motion and its original simple animation.

We frame the motion synthesis problem as a spacetime optimization. The unknowns to this problem include the values of joint angles at each frame, along with parameters that determine the behavior of angular and linear momentum. The entire synthesis process breaks down to four key stages (see figure 2):

Constraint and stage detection. Automatically detect environment constraints that correspond to user-intended motions, and separate the original motion sequence into constrained and unconstrained stages.

Transition pose generation. Establish transition poses between constrained and unconstrained animation stages.

Momentum control. Generate physical constraints according to the Newtonian laws and the biomechanics knowledge.

Objective function generation. Construct the objective function that favors motion that is smooth, similar to the input motion, and balanced when stationary.

Each stage improves specific aspects of the input motion sequences by introducing constraints or objective function components to the spacetime optimization problem. We then use a sequential quadratic programming method to find the optimal animation. In the subsequent sections we describe each of these steps in detail.

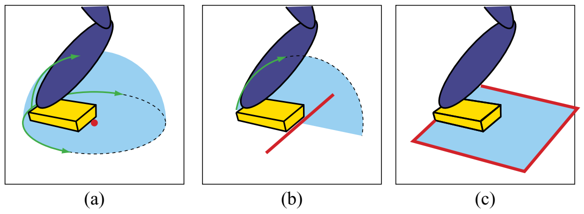

Figure 3 Positional constraint types: (a) A single positional constraint on a toe, (b) a line positional constraint on the front of the foot, and (c) a plane positional constraint. The green arrows indicate the free motion range.

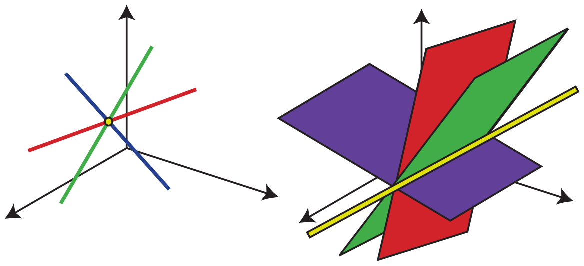

Figure 4 *Left:* A fixed-point positional constraint occurs at the intersection of three lines representing the solutions for the linear systems of three consecutive animation frames. *Right:* A fixed-line positional constraint occurs at the intersection of three planes.

4 Constraints and stage detection

From the standpoint of a user, each input motion sequence simply comprises two parts: the part that needs to be improved and the part that needs to be kept intact. For example, a user might wish that the hands of the character stay stationary on a high bar while our system makes the rest of the motion look realistic (figure 8). Moreover, users usually require a number of environmental restrictions on the movement of the character. For example, the feet should always remain above the ground. Violating these restrictions modifies the semantics of the animation that the user conveyed in the input animation. We represent these environmental restrictions with positional and sliding constraints. Since the rough sketch animation does not explicitly contain these requirements from users, our system automatically extracts the constraints from the input motion. In this section, we present an algorithm that automatically detects positional and sliding constraints from the original motion sequence.

4.1 Positional constraints detection

A positional constraint fixes a specific point on the character to a stationary location for a period of time. For example, when a character's heel touches the ground at landing, a positional constraint occurs on the heel across a number of frames. Violating positional constraints frequently causes undesirable artifacts such as feet sliding or penetrating the ground.

To detect positional constraints, we need to find all points on the body that stay fixed in space for some period of time. We would also like to determine if these points are isolated in space or if they lie on a line or a plane in order to detect constraints at a finer granularity (figure 3). For example, when the character takes off in a jumping motion, a plane of positional constraints on the bottom of the foot is detected first, followed by a single point constraint when the character transfers from standing on the entire foot to being on its toes.

Because each body part of the character is a rigid body, we reduce the problem of finding the constrained points on the whole character to finding constrained points on each body part. To illustrate our approach, we describe the movement of a body point through time. Let $\mathbf{W}_i$ be the matrix that transforms a point in a local coordinate frame $\mathbf{p}$ to its world position $\mathbf{x}_i$ at time i, or

$$\mathbf{x}_i = \mathbf{W}_i\mathbf{p}. \tag{1}$$

At time $i+1$, $\mathbf{p}$ will be transformed to $\mathbf{W}_{i+1}\mathbf{W}_i^{-1}\mathbf{W}_i\mathbf{p}$. We then define

$$\mathbf{T}_{i+1} = \mathbf{W}_{i+1}\mathbf{W}_i^{-1}, \tag{2}$$

as the transformation that brings $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$. A positional constraint on $\mathbf{p}$ from time 1 to time n implies that $\mathbf{T}_1$ through $\mathbf{T}_n$ all bring $\mathbf{p}$ to the same global position or

$$\mathbf{T}_i\mathbf{x}_i = \mathbf{x}_i \qquad \text{or} \qquad (\mathbf{T}_i - \mathbf{I})\mathbf{x}_i = \mathbf{0}, \tag{3}$$

for $i = 1 \ldots n$. The solution for each time i is the eigenvector for $\mathbf{T}_i$ corresponding to the unit eigenvalue. Because $\mathbf{T}_i$ is an affine matrix, it can be written as

$$\mathbf{T}_i = \begin{bmatrix} \widehat{\mathbf{T}}_i & \mathbf{b}_i \\ 0 & 1 \end{bmatrix} \tag{4}$$

so that we can reformulate equation (3) as a linear system

$$(\widehat{\mathbf{T}}_i - \mathbf{I})\hat{\mathbf{x}}_i = -\mathbf{b}_i. \tag{5}$$

The linear system in equation (5) is not always consistent, so we solve it in a least-squares sense. Depending on the rank of $(\widehat{\mathbf{T}}_i - \mathbf{I})$, the solution for each time i can be represented as a point, a line or a plane.

If the intersection of the geometries representing $\hat{\mathbf{x}}_1$ through $\hat{\mathbf{x}}_n$, $\mathbf{X}$, exists and falls on the body, then we define a constraint that fixes $\mathbf{p}$ to $\mathbf{X}$. In other words, we establish position constraints where there exists a collection of points that remains stationary over a time period $1 \ldots n$ (figure 4).

Our algorithm can be fine-tuned by modifying following parameters:

Minimal frames required (n) The intersection has to exist across at least n frames to be considered a positional constraint.

Tolerance of intersections (ε) Two solutions are considered intersecting if the distance between them is less than ε.

Constrainable body parts Constraint detection is performed only on a subset of body parts where users are interested in finding positional constraints. This is useful when we want to detect constraints only on specific body parts in the case when the entire character stands still.

4.2 Sliding constraints

Sliding constraints are a generalization of positional constraints. Instead of fixing a point on the character to a single world coordinate, a sliding constraint limits the point's motion to a particular line or plane in world space. For example, a figure skater is free to change the location of the foot on the ice as long as it slides along the same line. Thus, we need to solve for both the constrained body points and the line or plane to which they are constrained.

We describe the algorithm for finding a line constraint. The plane constraint is computed analogously. We want to find a body point $\mathbf{p}$ that is constrained to a line $\mathbf{l}$. Instead of establishing a closed-form solution, we construct a least-squares problem, where unknowns are the parameters for $\mathbf{p}$ and $\mathbf{l}$

$$min_{\mathbf{p},\mathbf{l}} \sum_i Dist(\mathbf{T}_i\mathbf{W}_i\mathbf{p}, \mathbf{l}) \tag{6}$$

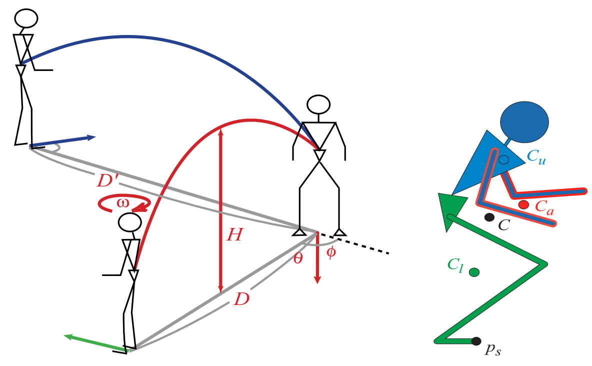

Figure 5 *Left:* The input parameters of a training example in the motion database include flight distance (D), flight height (H), previous flight distance (D'), takeoff angle (θ), landing angle (ϕ), spin angle (ω), feet speed at takeoff (δ_r, δ_l) and landing (η_r, η_l), and horizontal average speed (v). *Right:* The KNN algorithm selects the k most similar examples based on the input parameters and outputs a simplified representation of a character. The simplified representation consists of centers of mass for the lower body (C_l), upper body (C_u), and arms (C_a), all relative to the center of support (p_s).

In other words, we minimize the sum of distances between the $\mathbf{x}_i$ and line $\mathbf{l}$ at each frame i.

Because the definition of a plane sliding constraint subsumes line sliding constraints, and a line sliding constraint subsumes a point constraint, we perform our constraint detection in the order of decreasing restriction. First we solve for position constraints and then line sliding constraints and finally plane sliding constraints.

Although we tried to design our constraint detection to be as general as possible, in practice the rough sketch motion is rarely detailed enough to be able to find the exact location of the constraint. For example, the animator often leaves the entire character static during the time when character is on the ground. In that case, our constraint detection would find constraints on each body part. In those situations, we allow the animator to select specific body parts that should be tested for constraints. For example, we only select the feet of the character to be detected in the hopscotch example. This approach would also not fare well on extremely noisy data such as poor-quality motion capture.

4.3 Stage detection

Given the list of detected constraints, the system separates the original animation into unconstrained (flight) and constrained (ground) stages. We draw the distinction between unconstrained and constrained stages because the physics and biomechanics rules in the air are different from those on the ground. During the unconstrained stage, gravity is the only external force acting upon the character.

5 Transition poses

A transition pose separates constrained and unconstrained stages. We ask animators to specify these specific poses because all other non-transition frames are more directly controlled by the realism constraints. Transition poses also tend to be interesting from the animator's perspective. The following two sections describe a learned estimator that suggests a pose at each transition frame. We also describe a set of tools that allow users to position the character at the phase transitions.

5.1 Suggesting learned poses

Our pose estimator predicts poses at transition frames based on the input motion sequence. The estimator is a K-nearest neighbor (KNN) algorithm. The training consists of storing a specific set of parameters about the transition poses for each example motion. The examples can be generated by the animators or captured from real world. We also incrementally update the database by inserting motions as they are specified by the user during the animation process. The training input parameters include flight distance, flight height, previous flight distance, takeoff angle, landing angle, spin angle, foot speed at takeoff and landing, and average horizontal speed (figure 5). To compute an appropriate distance between training examples, we scale each input parameter by its natural bounds. The estimator predicts a simplified representation of the transition poses at the constraint release (takeoff) and constraint creation (landing). For each pose, the characteristics of the target motion must match the prediction. The output representation consists of three mass points: center of mass (COM) of the lower body, COM of the upper body, and COM of two arms (figure 5). The locations of mass points are stored relative to character's center of support. This makes our parameterization invariant of the global transformation.

To predict a candidate pose from the input, the KNN algorithm selects the k most similar examples from the motion database ($k = 3$, in our implementation). The estimator computes the candidate pose, which consists of the positions of the three mass points, by interpolating the poses of the selected neighbors, weighted by their similarities to the input.

We reconstruct a full character pose from three mass points by solving an inverse kinematics (IK) problem, constraining the three mass points to values returned by KNN, while minimizing the deviation between the suggested and original poses. We set the initial states of the unknown DOFs equivalent to the DOFs of the poses from the nearest training-set example, so that the solution pose keeps some plausible details from the database examples.

There are a number of advantages to estimating a small set of pose parameters. Joint angles themselves are poor estimators of the pose since they are not uniformly scaled. Furthermore, with our representation we can use the same motion database to learn the poses of characters with drastically different skeletal structures. We use a simple formula to scale the estimated relative mass points positions to accommodate a different skeleton. Let $\mathbf{C}_A$ be one of character A's COM output parameters from the learning algorithm and $\overline{\mathbf{C}}_A$ be the corresponding COM for the default pose. Suppose we know $\overline{\mathbf{C}}_B$, the corresponding default pose of another skeletal structure, we can compute $\mathbf{C}_B$ as follows:

$$\mathbf{C}_B = \overline{\mathbf{C}}_B + (\mathbf{C}_A - \overline{\mathbf{C}}_A)\frac{\|\overline{\mathbf{C}}_B\|_2}{\|\overline{\mathbf{C}}_A\|_2} \tag{7}$$

Intuitively, we displace the center of mass parameter by the rescaled difference between the default and suggested pose of the character in the database. We also found that this parameterization allows us to learn poses from a relatively small set of examples. We used a database of about 50 motion captured constrained and unconstrained segments to synthesize all of animations described in Section 9.

5.2 User pose adjustment

Because the estimated poses do not always meet the needs of the animation, we allow animators to directly modify transition poses. Our posing tool gives the user both fine-grain and high-level control of the pose edit process. Users can directly modify the learned mass points by dragging them to a new location. The IK solver adjusts the joint DOFs accordingly. For greater control, the animator can modify the transition pose directly. The user can also customize the importance of the similarity between the current pose and its corresponding pose in the original sketch. By fine-tuning transition poses animators can impart expression and style to the overall motion.

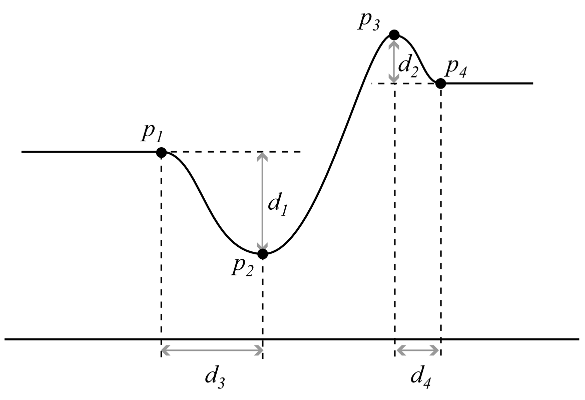

Figure 6 The general angular momentum pattern modeled after biomechanics data. During the unconstrained stage (left of p_1 and right of p_4), the angular momentum is constant. During the constrained stage (between p_1 and p_4), the curve is smooth, p_2 is lower than p_1, d_2 is less than d_1, and p_2 and p_3 are on opposite sides of p_4.

Animators are also free to introduce new keyframes anywhere in the animation to refine more detailed aspects of motion. However, when an animator creates too many of these constraints, the spacetime optimization problem becomes over-constrained. At that point, the animator has the option of turning momentum constraints into soft constraints. The optimization honors all of the animator's constraints while trying to satisfy soft realism constraints as much as possible. This approach provides graceful degradation of realism in the event that the animator's keyframe poses force the character into unrealistic movement.

6 Controlling the character momentum

The transition poses constrain the motion at a few key points of the animation. In a sense, they provide scaffolding for the motion, whereas dynamic constraints ensure realistic motion during each animation segment. We achieve this realism by formulating constraints on the behavior of the character's linear and angular momentum. We derive these constraints from the laws of physics and biomechanics domain knowledge.

Linear momentum determines the location of COM at each frame. The computation of angular momentum involves different body parts and their moments of inertia relative to the center of mass (for computation of angular momentum on an articulated character see Appendix A). The constraints on linear and angular momentum are different for unconstrained and constrained stages, and we discuss them separately.

6.1 Momentum during unconstrained stages

Since gravity is the only external force acting on the unconstrained character, the following equation holds

$$d\mathbf{P}(\mathbf{q})/dt = m\ddot{\mathbf{C}}(\mathbf{q}) = m\mathbf{g}, \tag{8}$$

where $\mathbf{C}$ is character's center of mass, and $\mathbf{q}$ are character's degrees of freedom.

During flight there are no external torques acting on the character, so the angular momentum is constant

$$d\mathbf{L}(\mathbf{q})/dt = \mathbf{0}. \tag{9}$$

The spacetime optimization enforces these two constraints during an unconstrained stage. Effectively, these constraints ensure that the center of mass falls into a parabolic trajectory and the joints move in such way that the angular momentum of the whole body remains constant (figure 6).

6.2 Momentum during constrained stages

Unlike the unconstrained stage where gravity is the only external force acting on the character, the momentum at a constrained stage results from a complex exchange of energy between the character and the environment constraints. We would like to avoid computing linear and angular momentum by complex physical simulation. Instead, we build an empirical model for the behavior of momentum based on biomechanics studies [Pandy et al. 1992; King 1999] and the analysis of motion capture data. We observe that the momentum during a constrained stage has a characteristic shape shown in figure 6. The figure shows a graph of an angular momentum component during a constrained stage between two unconstrained stages. Note that the angular momentum is constant during the two unconstrained stages.

During the constrained stage, the momentum transfers from one constant value to another. Natural dynamic systems achieve this transfer by first storing energy (momentum decreases), and then releasing it in a burst which causes a small overshoot at p_3.

The characteristic pattern of the linear momentum is the same with the exception that the linear momentum in the neighboring unconstrained stages has a slope mg instead of 0.

We try to capture all aspects of this curve by enforcing the following invariants:

- the curve is C^1–continuous at transition points p_1 and p_4
- p_2 is less than p_1
- d_1 is larger than d_2
- p_2 and p_3 are on the opposite sides of p_4

Since the momentum pattern during the constrained stage is fully determined by the control point vector $\mathbf{q}_m = [p_1, p_2, p_3, p_4]$, we formulate the linear and angular momentum constraints as

$$\mathbf{P}(\mathbf{q}) = S_l(\mathbf{q}_m) \tag{10}$$

$$\mathbf{L}(\mathbf{q}) = S_a(\mathbf{q}_m) \tag{11}$$

During optimization we solve for both $\mathbf{q}_m^j$ vectors for each constrained stage j and $\mathbf{q}^i$ for each time frame i enforcing the constraints in equations 8,9,10,11, as well as the inequality constraints on $\mathbf{q}_m^j$ governing the shape of the momentum curves.

7 Objective function

The momentum constraints enforce realism of the motion while detected constraints take into account the user intent and environmental restrictions. However, natural looking motion also requires natural joint movements, smoothness across frames, and static balance during stationary points of the animation. We formulate each as an objective function component.

Minimum mass displacement. To achieve natural joint movement, we use the minimum mass displacement metric [Popović and Witkin 1999]. Instead of comparing DOFs directly, this metric computes the integral of mass displacement over the character's body. This metric is loosely analogous to the measurement of power consumption. Our results show that minimum mass displacement presents its own merits in producing natural looking animations. For example, the compression on the body of the character before the unconstrained stages would not affect the lower body (knees, especially) without the minimum mass displacement as an objective function. Without it, the character tends to bend at the waist in order to lower the COM.

Minimal velocity of DOFs. Time coherence plays a major role in creating visually plausible animations. To account for the smoothness across frames, we define an objective function that minimizes the deviation of each DOF between two consecutive frames, effectively minimizing the velocity of each joint angle over the entire animation.

Static balance. The static balance is important during constrained stages when the character is standing still [Tak et al. 2000]. We measure balance by the distance between the COM and constraints when projected onto the plane normal to gravity.

The spacetime objective function is a weighted sum of the three objective components.

8 Putting it all together

All constraints and the objective function fit naturally within the spacetime framework. The unknowns of our system $\mathbf{Q}$ are the character DOFs $\mathbf{q}^i$ for each time i and the control point vectors for all constrained-stage momentum curves $\mathbf{q}^j_m$. The optimization needs to enforce three types of constraints:

Environment constraints($\mathbf{C}_e$). Constraint detection produces a collection of user-intended constraints that partition the motion into constrained and unconstrained stages.

Transition pose constraints($\mathbf{C}_p$). These constraints were defined between each motion phase either by our pose estimation method, or explicitly by the user.

Momentum constraints($\mathbf{C}_m$). During both unconstrained and constrained phases we dictate the behavior of the linear and angular momentum through constraints defined in equations 8,9,10,11.

The spacetime constraints formulation finds the unknowns $\mathbf{Q}$ that minimize the objective function while satisfying all the constraints:

$$\min_{\mathbf{Q}} \sum E_i(\mathbf{q}^i) \qquad \text{subject to} \qquad \left\{ \begin{array}{l} \mathbf{C}_e(\mathbf{Q}) = 0 \\ \mathbf{C}_p(\mathbf{Q}) = 0 \\ \mathbf{C}_m(\mathbf{Q}) = 0 \end{array} \right. \tag{12}$$

9 Results

We used our framework to generate a wide range of animations on a male, female and child figure, all of which comprise 51 DOFs, including 16 Euler rotations, 3 translations, and 32 quaternion rotations. We also created a three-legged creature with 58 DOFs. Inequality constraints enforce bounds on each DOF. We obtained the body dimensions and mass distributions from the biomechanics literature [de Leva 1996; Pearsall et al. 1994]. To start the animation process, the animator creates a simple animation as an input to the synthesis process. In some cases the animator also selects the parts of the body to be used for constraint detection (e.g. feet and hands). Once motion phases have been determined, the animator can also change the relative timing between each phase. In some cases, the animator adjusted the learned transition poses to achieve a desired effect. For the karate-kick animation, the animator also created an additional pose.

We solve our optimizations using SNOPT [Gill et al. 1996], a general nonlinearly-constrained optimization package. The optimization times depend on the duration of the motion sequence. All of the simple animations took only a few minutes to sketch. For all examples, the synthesis process took less than five minutes.

Broad jump. We synthesized a broad jump motion that clearly improves a crude input animation where only global translations of the character (3 out of 51 DOFs) are keyframed. The original animation is created by interpolating only 3 keyframes at takeoff, peak, and landing. The appropriate movement on the arms and legs results from enforcing the momentum constraints and learning the realistic transition poses.

Twist jumps. The input motion for this sequence consists of two jumps, each with a 90° turn. The output animation shows the preparation before each take-off. The character twists away from the turn to increase the potential energy so that it can generate enough angular momentum at take-off to accomplish the 90° turn.

Hopscotch. Much like for the broad jump example, the animator created an animation of a popular child game, consisting of hops, broad jumps and a spin jump. Each hop only requires 3 keyframes, each of which has fewer than 7 DOFs specified. This example shows that our system can deal with asymmetric motions by coordinating different parts of body to accommodate the momentum constraints. This example also demonstrates smooth transitions among different types of jump styles (figure 1).

Running. The input for the running motion sequence required keyframing of 7 DOFs. Originally, the upper body is completely stiff since no upper body DOFs were keyframed. The angular momentum constraint creates a counter-body movement by the shoulders and arms to counteract the angular momentum generated by the legs. In the synthesized animation the arms clearly twist to counter the leg movement.

Handspring. We generated a rough sketch of an advanced handspring motion on an uneven terrain (figure 7). This hazardous movement of landing and jumping with arms would be difficult to capture in the real world. The constraint detector successfully finds constraints on both the feet and hands. Since there were no handstands within the learning example database, the animator had to substantially modify the suggested handstand transitions poses. This example demonstrates that momentum constraints are general enough to capture the dynamics of movement regardless the orientation of the model.

High-bar. Figure 8 shows a character performing a high-bar gymnastic exercise. The two positional constraints on the hands during bar-contact time create a "hanging" constrained stage. The constraints of linear momentum and angular momentum apply equally to the bar-contact and the more common ground constrained stage.

Skating. This example demonstrates sliding constraint detection. The motion is similar to the 180° spin jump, except that the character slides a single leg along a straight line through the take-off phase and landing phase. The resulting motion resembles an ice-skating figure (figure 9).

Karate kick. The karate kick animation was created by an incremental synthesis process. First the animator created a simple side-jump animation. The first synthesis pass created a realistic side jump. The animator then introduced an additional keyframe at the peak of the jump indicating a leg kick. The second synthesis pass created the final karate-kick animation by enforcing the original constraints augmented by the additional mid-flight constraint.

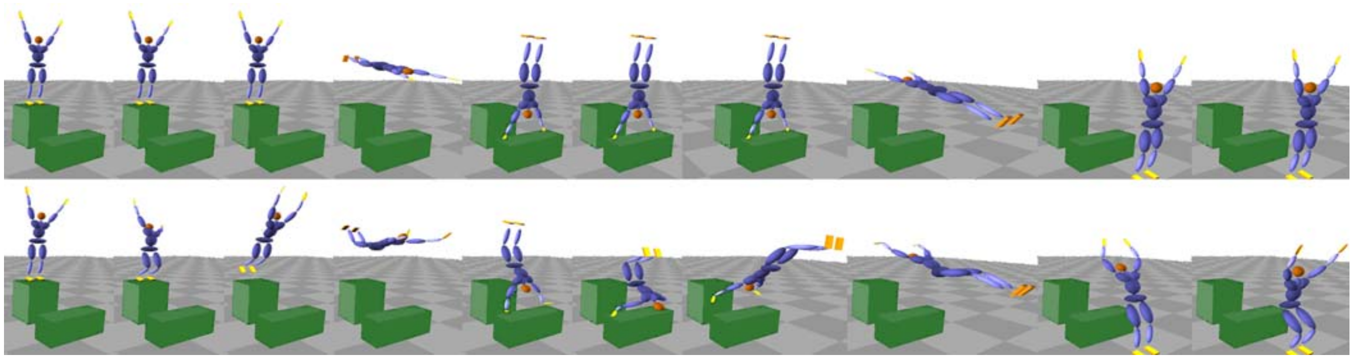

Figure 7 A handspring motion: simple and synthesized animation.

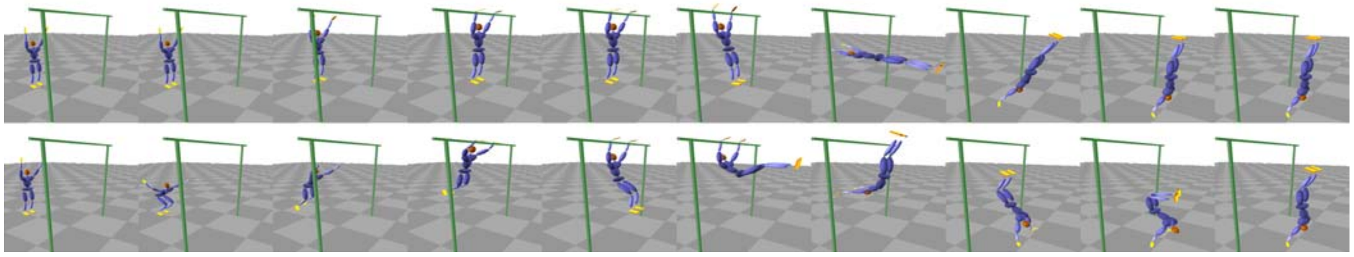

Figure 8 A high-bar gymnastic exercise: simple and synthesized animation.

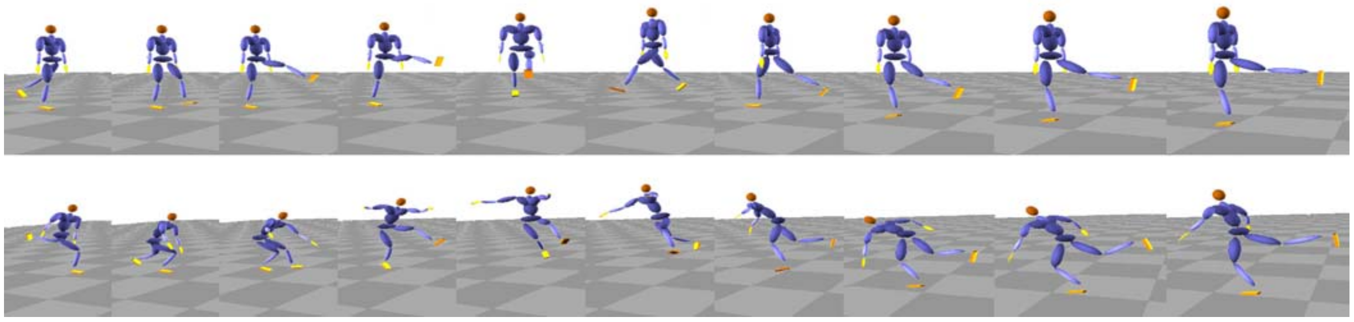

Figure 9 Jumping on ice-skates: simple and synthesized animation.

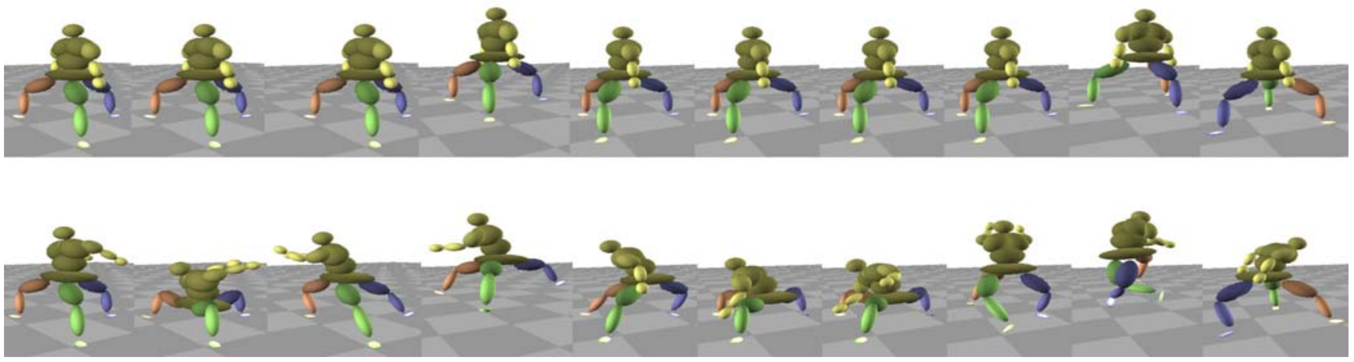

Figure 10 Spin jump by a three-legged creature: simple and synthesized animation.

Other humanoid characters. Our framework can also animate characters with different skeletal structures and mass distributions. We created a child character whose limbs are shorter and whose torso is relatively larger. Our algorithm successfully scaled down the poses learned from motion database to accommodate a different skeletal structure. In another example, we removed the left knee DOF of the character and generated a motion sequence of the twist jumps. The character had to drastically twist the pelvis while landing to place the stiff leg on the ground properly.

Non-humanoid characters. Our method also generalizes to non-humanoid characters. We created a strange three-legged creature and synthesized a number of animations experimenting with various locomotion gaits. In figure 10 we show the creature jumping to the side and doing a 180° turn jump.

10 Conclusion and future work

In this paper we present a general method for rapid prototyping of dynamic character motion that could be used by both experts and non-skilled animators. Animators can use our system to produce relatively complex realistic motion with little effort on behalf of the animator. Creating the initial simple animation often takes less than two minutes, while modification of the transition poses can be avoided completely by accepting suggestions presented by the learned-pose estimator. In addition, a skilled animator can fine-tune transition poses as well as add any number of additional keyframes to achieve desired details.

We show that realism can be approximated by a small number of constraints on the behavior of linear and angular momentum. This approach helps us avoid the complexities of Newtonian constraints. Since we don't solve for muscle forces, we also avoid computing the right distribution of muscle usage needed to produce natural-looking motion. Simpler dynamic constraints allow us to generate more complex animations in terms of the character model and motion description complexity. Simpler constraints also allow greater variability of the resulting motion. This feature enables the animator to add more expressive detail to the motion by providing additional keyframes.

We also show that by learning a small set of key parameters that describe a pose, we can create realistic transition poses. We show that the sample space can be populated either by realistic motion (e.g. motion capture data), or by storing each of the previously created poses. In both cases, a very small data set creates useful pose suggestions.

Our methods are best suited for synthesis of highly dynamic motion, since such motions are mainly governed by Newtonian physics. One clear future research direction would be to extend these methods so that they apply to low-energy character motion such as walking, reaching, or picking up an object.

Many aspects of our approach could be potentially applied to other animation problems, most notably realistic editing of motion capture data. The animator could start from a motion capture sequence instead of a simple animation. The automatic constraint detection methods could be useful in processing motion capture data, since they can accurately find the foot-ground contact points. Transforming a motion capture sequence by adding additional keyframes would create a new realistic animation keeping as much of the captured detail as possible.

It is worth noting that our synthesis approach does not fundamentally need to start with an input animation. A skilled animator could, alternatively, specify the environment constraints and keyframes explicitly. This is probably the most likely approach for using our algorithms in the production environment. To make our methods more accessible to a wider audience, we need to develop a more effective user interface. In the future, we hope to make our tools accessible to animators of varying skill levels.

11 Acknowledgments

We thank Jiwon Kim and Jia-chi Wu for their assistance with creating figures and videos for this paper. Special thanks goes to Gary Yngve and David Hsu for their invaluable comments on various versions of this document. We also thank Charles Gordon, Eugene Hsu, Adam Kirk, and Keith Grochow for their help with processing motion capture data. Funding and research facilities were provided by the University eof Washington RRF Grant 65-4803, UW Animation Research Labs, Microsoft Research, Intel, and NSF grant Ccr-0092970.

Appendix A

We compute the angular momentum of a body point **x** as

$$\mathbf{L} = \mu \mathbf{r} \times \dot{\mathbf{x}}, \tag{13}$$

where **r** is the vector between **x** and COM, $\dot{\mathbf{x}}$ is the velocity of **x** and μ is the mass of **x**.

To compute the net angular momentum of the whole body, we sum the angular momentum contributions for each body part (node) i, computed by integrating each body point $\mathbf{x}_j$:

$$\begin{aligned}\mathbf{L} &= \sum_i \iiint_j \mu_j(\mathbf{x}_j - C) \times (\dot{\mathbf{x}}_j - \dot{C})dxdydz \\ &= \sum_i cr(\mathbf{W}_i \mathbf{M}_i \dot{\mathbf{W}}_i^T) + \sum_i m_i C \times \mathbf{W}_i C_i \\ &\quad + \sum_i m_i \mathbf{W}_i C_i \times \dot{C} + \sum_i m_i C \times \dot{C}\end{aligned} \tag{14}$$

where C_i is the COM of the node i in its local coordinate frame.

We define operator $cr()$ that transforms a 3×3 matrix **A** to a 3×1 vector as follows:

$$cr(A) = \begin{bmatrix} a_{23} - a_{32} \\ a_{31} - a_{13} \\ a_{12} - a_{21} \end{bmatrix}$$

We compute the mass matrix tensor M_i of the node i as an integral of outer products over all body points x_j, scaled by the node mass m_i.

$$M_i = m_i \iiint_j x_j x_j^T dxdydz$$

References

ALEXANDER, R. M. 1989. Optimization and gaits in the locomotion of vertebrates. *Physiol. Rev. 69*, 1199–1227.

ALEXANDER, R. M. 1990. Optimum take-off techniques for high and long jumps. *Phil. Trans. R. Soc. Lond. 329*, 3–10.

BLICKHAN, R., AND FULL, R. J. 1993. Similarity in multilegged locomotion: bouncing like a monopode. *J Comp. Physiol. A. 173*, 509–517.

BLICKHAN, A. S. A. F. V. W. R. 1999. Dynamics of the long jump. *Jornal of Biomechanics 32*, 1259–1267.

BRUDERLIN, A., AND CALVERT, T. W. 1989. Goal-directed, dynamic animation of human walking. *Computer Graphics 23*, 3 (July), 233–242.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, 97–104.

COHEN, M. F. 1992. Interactive spacetime control for animation. In *Computer Graphics (SIGGRAPH 92 Proceedings)*, vol. 26, 293–302.

DE LEVA, P. 1996. Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *J. of Biomechanics 29*, 9, 1223–1230.

DISCREET. Character studio. *http://www.discreet.com/products/cs/*.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 251–260. ISBN 1-58113-292-1.

GILL, P., SAUNDERS, M., AND MURRAY, W. 1996. SNOPT: An SQP algorithm for large-scale constrained optimization. Tech. Rep. NA 96-2, University of California, San Diego.

GLEICHER, M., AND LITWINOWICZ, P. 1998. Constraint-based motion adaptation. *The Journal of Visualization and Computer Animation 9*, 2, 65–94.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *1997 Symposium on Interactive 3D Graphics*, M. Cohen and D. Zeltzer, Eds., ACM SIGGRAPH, 139–148. ISBN 0-89791-884-3.

GLEICHER, M. 1998. Retargeting motion to new characters. In *Computer Graphics (SIGGRAPH 98 Proceedings)*, 33–42.

GLEICHER, M. 2001. Motion path editing. In *2001 ACM Symposium on Interactive 3D Graphics*, 195–202. ISBN 1-58113-292-1.

HODGINS, J. K., AND POLLARD, N. S. 1997. Adapting simulated behaviors for new characters. *Proceedings of SIGGRAPH 97*, 153–162. ISBN 0-89791-896-7. Held in Los Angeles, California.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. *Proceedings of SIGGRAPH 95* (August), 71–78. ISBN 0-201-84776-0. Held in Los Angeles, California.

HODGINS, J. K. 1998. Animating human motion. *Scientific American 278*, 3 (Mar.), 64–69.

HULL, M. P. F. C. A. D. G. 1991. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *Journal of Biomechanical Engineering 114*, 450–460.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. *Proceedings of SIGGRAPH 99* (August), 409–416. ISBN 0-20148-560-5. Held in Los Angeles, California.

KING, D. 1999. Generating vertical velocity and angular momentum during skating jumps. *23rd Annual Meeting of the American Society of Biomechanics* (Oct).

LASZLO, J., VAN DE PANNE, M., AND FIUME, E. L. 2000. Interactive control for physically-based animation. *Proceedings of SIGGRAPH 2000* (July), 201–208. ISBN 1-58113-208-5.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Computer Graphics (SIGGRAPH 99 Proceedings)*.

LIU, Z., GORTLER, S. J., AND COHEN, M. F. 1994. Hierarchical spacetime control. In *Computer Graphics (SIGGRAPH 94 Proceedings)*.

PANDY, M., ZAJAC, F. E., SIM, E., AND LEVINE, W. S. 1990. An optimal control model of maximum-height human jumping. *J. Biomechanics 23*, 1185–1198.

PANDY, M., ANDERSON, F. C., AND HULL, D. G. 1992. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *J. of Biomech. Eng.* (Nov.), 450–460.

PEARSALL, D., REID, J., AND ROSS, R. 1994. Inertial properties of the human trunk of males determined from magnetic resonance imaging. *Annals of Biomed. Eng. 22*, 692–706.

POLLARD, N. S., AND BEHMARAM-MOSAVAT, F. 2000. Force-based motion editing for locomotion tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

POLLARD, N. S., AND REITSMA, P. S. A. 2001. Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model. In *Yale Workshop on Adaptive and Learning Systems*.

POLLARD, N. S. 1999. Simple machines for scaling human motion. In *Computer Animation and Simulation '99*, Eurographics, Milano, Italy. ISBN 3-211-83392-7.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *Computer Graphics (SIGGRAPH 99 Proceedings)*.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. P. 2000. Interactive manipulation of rigid body simulations. *Proceedings of SIGGRAPH 2000* (July), 209–218. ISBN 1-58113-208-5.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. In *Computer Graphics (SIGGRAPH 91 Proceedings)*, vol. 25, 349–358.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. 1996. Efficient generation of motion transitions using spacetime constraints. In *Computer Graphics (SIGGRAPH 96 Proceedings)*, 147–154.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications 18*, 5 (Sept. – Oct.).

SHIN, H. J., LEE, J., GLEICHER, M., AND SHIN, S. Y. 2001. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2 (April), 67–94. ISSN 0730-0301.

TAK, S., SONG, O.-Y., AND KO, H.-S. 2000. Motion balance filtering. In *Proceedings of the 21th European Conference on Computer Graphics (Eurographics-00)*, Blackwell Publishers, Cambridge, S. Coquillart and J. Duke, David, Eds., vol. 19, 3 of *Computer Graphics Forum*, 437–446.

TORKOS, N., AND VAN DE PANNE, M. 1998. Footprint-based quadruped motion synthesis. In *Graphics Interface '98*, 151–160. ISBN 0-9695338-6-1.

VAN DE PANNE, M., AND FIUME, E. 1993. Sensor-actuator networks. In *Computer Graphics (SIGGRAPH 93 Proceedings)*, vol. 27, 335–342.

VAN DE PANNE, M., KIM, R., AND FIUME, E. 1994. Virtual wind-up toys for animation. *Graphics Interface '94* (May), 208–215. Held in Banff, Alberta, Canada.

VAN DE PANNE, M. 1997. From footprints to animation. *Computer Graphics Forum 16*, 4, 211–224.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Computer Graphics (SIGGRAPH 88 Proceedings)*, vol. 22, 159–168.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion warping. In *Computer Graphics (SIGGRAPH 95 Proceedings)*.

WOOTEN, W. L. 1998. *Simulation of leaping, tumbling, landing, and balancing humans*. PhD thesis, Georgia Institute of Technology.

YEADON, M. R. 1990. The simulation of aerial momement - iii the determination of the angular momentum of the human body. *Journal of Biomechanics 23*, 75–83.

ZORDAN, V. B., AND HODGINS, J. K. 1999. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Animation and Simulation '99*, Eurographics, Milano, Italy. ISBN 3-211-83392-7.