Physically Based Motion Transformation

Zoran Popović Andrew Witkin*

Computer Science Department
Carnegie Mellon University

Abstract

We introduce a novel algorithm for transforming character animation sequences that preserves essential physical properties of the motion. By using the spacetime constraints dynamics formulation our algorithm maintains realism of the original motion sequence without sacrificing full user control of the editing process.

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

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

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; G.1.6 [Numerical Analysis]: Optimization.

Keywords: Human Body Simulation, Physically Based Animation, Animation with Constraints

1 Introduction

Controllable automatic synthesis of realistic character motion is a difficult problem. The motion of a character with many degrees of freedom (DOFs) needs to be consistent with the laws of physics. More importantly, in order for the motion to look realistic, the entire musculoskeletal structure must be taken into account. Controlling this complex motion generation process adds further difficulties. In this paper, we present a solution to the problem of generating both controllable and realistic character animations. Instead of motion synthesis, we take the approach of motion transformation. For example, we transform a human running sequence by restricting the range of motion for a knee joint to obtain a realistic run with a limp.

Any dynamically sound motion, such as captured motion or the result of a physical simulation, can be used as an input to our transformation algorithm. The first step of our algorithm constructs a simplified character model and fits the motion of the simplified model to the captured motion data. From this fitted motion we obtain a physical spacetime optimization solution that includes the body's mass properties, pose, and footprint constraints, muscles and the objective function. To edit the animation we modify the constraints and physical parameters of the model and other spacetime optimization parameters (e.g. limb geometry, footprint positions, objective function, gravity.) From this altered spacetime parameterization we compute a transformed motion sequence. Finally, we map the *motion change* of the simplified model back onto the original motion to produce a final animation sequence.

Once the spacetime model has been constructed from the input data, our algorithm can be turned into a motion library, since each change in the physical formulation of the model produces a new motion sequence. Thus, a captured motion sequence of a human run can be turned into a running motion library capable of generating all possible runs that fit the needs of the animator.

Our algorithm presents the first solution to the problem of editing captured motion taking dynamics into consideration. We also describe a novel methodology for mapping motion between characters with drastically different kinematic structure. In addition, we introduce a method for simplification of complex dynamic systems without losing the fundamental dynamic properties of motion.

The next section describes how this work relates to other research efforts. We follow with the algorithm outline, and proceed to describe each stage of the algorithm in detail. In Section 8 we report the results of the algorithm's application on two human captured motion sequences. We conclude the paper with the main contributions and future research directions.

2 Related Work

Forward dynamics methods compute motions of objects that obey the laws of physics. For rigid objects (e.g. [5, 4, 21]), or secondary motion of cloth (e.g. [12, 6]), forward dynamics techniques are ideal because obeying physical laws is synonymous with realism. However, active characters create motion with their own muscles. The specific motion of real creatures depends on their intricate musculoskeletal structure. Determining exact muscle forces that would make the animation look realistic is extremely difficult. In addition, with dynamics methods each animation frame depends on the previous frame (and consequently on all other preceding frames). The smallest change of dynamic properties of any single frame drastically affects all consecutive frames, resulting in lack of controllability.

Spacetime constraints approach effectively addresses the need for both realism and controllability of character motion [33, 9, 20, 28]. In the spacetime framework the user first specifies *pose constraints* that must be satisfied by the resulting motion sequence (e.g.

*now at Pixar Animation Studios.

the character pose at the beginning and end of the animation). In addition to these constraints, the user also specifies an *objective function* that is a metric of performance or style such as total power consumption of all of the character's muscles. The algorithm takes this spacetime specification and finds the motion trajectories that minimize the objective function while satisfying the constraints. High realism and intuitive control give this method great appeal. The downside, however, is that the current algorithms do not scale up to the complexity of characters one would like to animate. The time complexity of the spacetime formulation and convergence difficulties remain a huge impediment. Another problem of these methods is that they are extremely sensitive to the starting position of the optimization process — if optimization begins far away from the solution, the optimization methods often cannot converge to the optimal motion. As a result, spacetime optimization methods have not been successfully applied to automatic generation of human motion. Our work draws from the ideas of spacetime constraints and tries to address the issues that prevent this method from being applied to complex character models.

Robot controller design has also been applied to the domain of realistic computer animation (e.g. [26, 32, 31, 18]). These methods use controllers that drive the actuator forces based on the current state of the environment. These forces, in turn, produce desired motion. Intuitively, controllers can be thought of as a set of instinctual reflexes that control muscles and collectively produce a character's continuous motion. Once the controllers have been fine-tuned and synchronized to each other, this method can produce a wide range of expressive animations [26, 19]. Furthermore, a number of different animations can be created without any additional work, because the controllers adjust to the changes in the environment. Recently, van de Panne introduced an interesting method for generating motion from footsteps that includes some rudimentary physical properties[30]. Although a controller transformation algorithm has been reported [17], determining controllers that produce realistic character motion is extremely difficult, and has not been formalized.

Another way to generate realistic motion is to attain it from the real world. Recent availability of real-time 3-D motion capture systems provides such an alternative. Motion capture systems use sensors to record absolute positions of key points on the character's body over a period of time. Not surprisingly, the resulting clip motion is convincing and rich with expressive detail that is almost impossible to generate by any computer methods. Although this type of animation is quite realistic, it yields highly unstructured and uncorrelated motion, even when converted to joint angles within a hierarchical character model.

Recently, a number of motion capture editing methods have been proposed [34, 8, 15, 14, 16, 27]. These methods don't generate motion "from scratch" like earlier described methods, but transform existing motion sequences. Even though some of the methods solve for the entire transformed motion sequence (and thus use the term spacetime), they do not include any notion of dynamics. Recently Gleicher [16] introduced a method for remapping captured motion onto drastically different characters. While his method is capable of producing many interesting motions, it has no means of making the motion physically realistic. For example, a tall and lanky character would utilize his muscles (and therefore move) in very different ways than a more compact character. A property of all motion editing methods that ignore inherent dynamics is that while they can effectively transform motion by small amounts, larger deformations reveal undesirable, unrealistic artifacts.

An alternative approach to editing realistic motion sequences is to extract the physical model from captured data, and perform all editing on the computer model instead. Spacetime optimization is a good candidate for this task. Of course, the primary problem of spacetime methods is that they do not guarantee a solution for

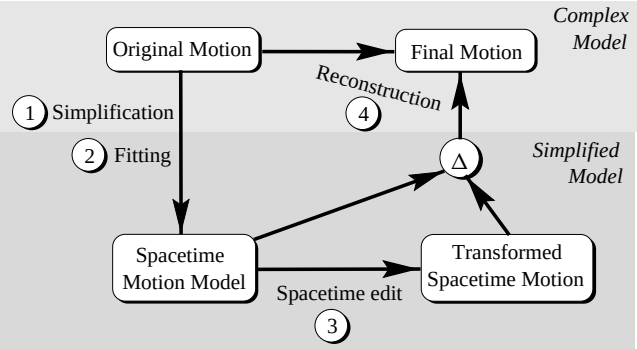

Figure 1: Algorithm outline.

motion problems of complex characters such as humans. We circumvent this issue by developing smaller, abstracted models. This model simplification is motivated by biomechanics research [7]. Blickhan and Full demonstrate the similarity in the multi-legged locomotion of kinematically different animals. They show striking similarities between a human run, a horse run and the monopode bounce (i.e. pogostick). This similarity motivates our approach to reducing the DOF count of complex kinematic structures such as humans.

Biomechanics also studies the postulated optimality of motion in nature [1, 2, 24]. There has been a considerable amount of work in the area of human performance in sports [22]. This work also reaffirms that spacetime optimization is a good choice for realistic motion synthesis.

3 Algorithm Outline

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

At its core our algorithm uses spacetime optimization because the spacetime formulation maintains the dynamic integrity of motion and provides intuitive motion control. Because such methods have not been shown to be feasible for human motion models, we must also find a way to simplify the character model.

The entire transformation process breaks down to four main stages (Figure 1):

Character Simplification. Create an abstract character model containing the minimal number of degrees of freedom necessary to capture the essence of the input motion. Map the input motion onto the simplified model.

Spacetime Motion Fitting. Find the spacetime optimization problem whose solution closely matches the simplified character motion.

Spacetime Edit. Change spacetime motion parameters, introduce new pose constraints, change the character kinematics, objective function, etc.

Motion Reconstruction. Remap the change in motion introduced by the spacetime edit onto the original motion to produce the final animation.

The *character simplification* and *spacetime motion fitting* stages require a significant amount of human intervention. However, once

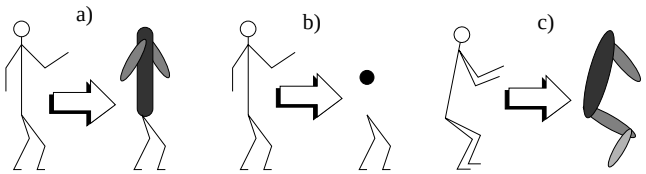

Figure 2: Kinematic character simplification: a) elbows and spine are abstracted away, b) upper body reduced to the center of mass, c) symmetric movement abstraction.

the spacetime model is computed it can be reused to generate a wide range of different animations. The *spacetime edit* and *motion reconstruction* stages are fully automated. They also take much less time to compute than the first two stages, which enables computation of transformed motion sequences at near-interactive speeds.

4 Character Simplification

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

- DOF reduction improves performance and facilitates convergence of the spacetime optimization.
- creation of an abstract model that contains only DOFs essential for the given motion captures the more fundamental properties of the body movement. As a result, detailed motion in the input sequence will be preserved during the transformation process.

Simplified models capture the minimum amount of structure necessary for the input motion task, and therefore capture the "essence" of the input motion. Subsequent motion transformations modify this abstract representation while preserving the specific feel and uniqueness of the original motion. Our simplification process draws from certain ideas in the biomechanics research [7]. We take the view that, abstractly speaking, highly dynamic natural motion is created by "throwing the mass around," or changing the *relative position* of body mass. With this in mind a human arm with more than 10 DOFs can be represented by a rigid object with only three shoulder DOFs without losing much of the mass displacement ability. Simplification of certain body parts also depends on the type of the input motion. For example, while the above mentioned arm simplification may work well for the human run motion, it would not appropriately represent the ball-throwing motion.

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

Character simplification is performed manually. We apply three basic principles during this process:

DOF removal. Some body parts are fused together removing DOFs that link them together. Elbow and wrist DOFs are usually removed for running and walking motion sequences where they have little impact on the motion.

Node subtree removal. In some cases of high-energy motion the entire subtree of the character hierarchy can be replaced with a single object, usually a mass point with three translational DOFs. For example, the upper body of a human character can be reduced to a mass point for various jumping motion sequences where the upper body catapults in the direction of the jump.

Exploit symmetric movement. Broad jump motions contain inherent symmetry since both legs move in unison. Thus, we can abstract both legs with one, turning the character into a monopode.

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

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

4.1 Handles

Handles are multi-valued time-varying functions that can be evaluated on both complex and simplified character models. All handles depend on the character pose defined by the vector of values for each DOF $\mathbf{q}(t_i)$. They represent intuitive measurements of various body properties such as 3D point positions, 3D directions, distance between two assigned body points.

Some 3D position handles are simply points on the character's body (e.g. the foot-ground contact position). Others, like the center of mass position handle, depend on a much larger set of DOFs. Direction handles are often used to represent the orientation of the character. When kinematic topology has been drastically changed during the simplification, it is often useful to use distance handles to correlate various body points.

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

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

$$E_d = [\mathbf{h}_o(\mathbf{q}_o(t_i)) - \mathbf{h}_s(\mathbf{q}_s(t_i))]^2 \quad (1)$$

$$\min_{\mathbf{q}_s(t_i)} E_d \quad (2)$$

for each frame t_i. This process is equivalent to solving an inverse kinematics problem for each time frame of the animation. Naturally, there should be *at least* as many handles as there are DOFs in the simplified character. That way the simplified motion is fully determined by $\mathbf{h}_s(\mathbf{q}_s(t))$.

5 Spacetime Motion Fitting

Handles help us map the original motion onto the simplified character. However, the resulting motion is no longer dynamically correct. Before we can edit the motion with spacetime constraints we need to create not only dynamically correct but also realistic motion of the simplified model. In other words, we need to find the spacetime optimization problem whose solution comes very close

to the simplified model motion we computed in section 4.1. Section 5.1 describes the spacetime constraints formulation of motion. Subsequent sections describe our approach to finding the appropriate muscles, spacetime constraints and the objective function which would yield the motion closely matching the input sequence.

5.1 Spacetime Constraints Formulation

We obtain the body dimensions and mass distributions from biomechanics sources [11, 23]. All other concepts of spacetime optimization have their intuitive counterparts in real-life.

A *character* is an object performing motion of its own accord. It has a finite number of kinematic DOFs and a number of *muscles*. DOFs usually represent joint angles of the character's extremities, while muscles exert forces or torques on different parts of the body, thus actuating locomotion. Given that both body and muscle DOFs change through time we refer to them collectively as $\mathbf{q}(t)$, or separately as kinematic $\mathbf{q}_k(t)$, and muscle DOFs $\mathbf{q}_m(t)$.

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

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

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

Finally, we also need to ensure dynamic correctness of the motion. We do this by constraining the acceleration of each DOF. Intuitively, we make sure that $F = ma$ holds for all degrees of freedom at all times. In this document we call such constraints *dynamics constraints* ($\mathbf{C}_d$). As long as these constraints are satisfied, we know that the resulting motion is physically possible, given the muscles' ability to generate forces.

When motion is defined in this way, it straightforwardly maps onto a non-linearly constrained optimization problem: we optimize the objective function $E(\mathbf{q}(t), t)$ parameterized in space and time, subject to the pose, mechanical and dynamics constraints:

$$\min_{\mathbf{q}(t)} E(\mathbf{q}(t), t) \qquad \text{subject to} \qquad \begin{cases} \mathbf{C}_p(\mathbf{q}(t), t) = 0 \\ \mathbf{C}_m(\mathbf{q}(t), t) = 0 \\ \mathbf{C}_d(\mathbf{q}(t), t) = 0 \end{cases} \tag{3}$$

This optimization is a variational calculus problem, as we solve for functions, not values. Such problems are solved by continuous optimization methods, which require computation of first derivatives of all constraints and the objective function. We use a sparse SQP [13] method to solve spacetime optimization problems.

The following sections describe specifics of determining muscles, constraints and the objective function that produce realistic motion and closely match the input animation.

5.2 Muscles

Muscles are the primary source of character locomotion. The biomechanics community has developed a number of complex muscle models, which closely match empirical data [29, 10, 3]. While these models tend to be very accurate, their complexity makes them difficult to differentiate and use in full body optimizations. Since our character model is drastically simplified, it would not make much sense to apply realistic muscles on simplified kinematic structure. Instead, we use simple structures that account for entire muscle groups, yet still induce forces onto DOFs similar to those of real muscles.

We use *generalized muscle forces* Q to represent the abstract muscle. These muscles apply accelerations directly onto DOFs, much like robotic servo-motors positioned at joints apply forces on robotic limbs. Having a generalized muscle at each character DOF presents the minimum set of muscles that ensures the full range of character motion. Unfortunately, the ability to apply arbitrary generalized force onto each joint is a poor model of natural muscles. For example, sudden non-smooth muscle forces generate extremely jerky, unnatural motion much like motion generated by bang-bang controllers [25].

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

To circumvent the problems of simple generalized force muscles described above, yet still maintain a simple and differentiable muscle model, we use a damped servo model often used in robotic simulations [26, 19]. Each kinematic DOF q_{k_i} has a corresponding damped generalized muscle force

$$Q_{k_i} = k_s(q_{k_i} - q_{m_i}) - k_d(\dot{q}_{k_i} - \dot{q}_{m_i}) \tag{4}$$

where q_{m_i} is the additional muscle DOF that is often interpreted as the desired value of q_{k_i}. Differentiation with respect to DOFs and their velocities is straightforward. This formulation does not exhibit scaling problems since q_{m_i} is of the same scale as q_{k_i}, and the velocity dependent damping encourages smoothness. To further ensure smooth muscle forces akin to those found in nature, we always include a muscle smoothness metric within the objective function (see Section 5.4.)

5.3 Constraints

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

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

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

5.4 Objective Function

There has been much research into the optimality of motion in nature [1, 2, 24]. We, however, avoid guessing the right objective function altogether. Instead, we rely on the fact that the starting motion is very close to the optimum. At first, our objective measured the deviation from the original motion (E_d) as described by Equation 1. We also include the muscle smoothness objective component $E_m = \ddot{\mathbf{q}}_m^2$ at all times. The spacetime objective is a weighted sum of the two objective components.

$$E = w_d \int E_d + \int E_m \tag{5}$$

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

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

6 Spacetime Edit

A spacetime constraints parameterization provides powerful and intuitive control of many aspects of the dynamic animation: pose and environment constraints, explicit kinematic and dynamic properties of the character, and the objective function.

By changing existing constraints the user can rearrange foot placements both in space and time. For example, a human run sequence can be changed into a zig-zag run on an uphill slope by moving the floor contact constraints wider apart and progressively elevating them. The constraint timing can also be changed: extending the floor contact time duration of one leg creates an animation that gives the appearance of favoring one leg. We can also introduce new obstacles along the running path, producing new constraints that, for example, require legs to clear a specified height during the flight phase of the run. We can also affect the environment of the run by changing the gravity constant, producing a human running sequence on the moon surface, for example.

Changes can also be made on the character model itself. We can change the limb dimensions or their mass distribution characteristics, and observe the resulting dynamic change of the motion. We can remove body parts, restrict various DOFs to specific ranges, or remove DOFs altogether, effectively placing certain body parts in a cast. For example, we can create different gimpy run sequences by shortening the leg, making one leg heavier, reducing the range of motion for the knee DOF, removing the knee DOF. Various muscle properties of the character can also affect the look of transformed motion. We can limit the force output of the muscles, forcing the character to compensate by using other muscles.

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

After each edit we re-solve the spacetime optimization problem and produce a new transformed animation. Since the optimization starting point is near the desired solution, and all dynamic constraints are satisfied at the outset, optimization converges rapidly. In practice, while the initial spacetime optimization may take more than 15 minutes to converge, the spacetime optimizations during the editing process take less than two minutes.

7 Motion Reconstruction

In order to create the transformed animation of the full character model, we reconstruct the final motion from the original motion and two simplified spacetime motions. We apply the transformation to the original sequence so that we modify the fundamental dynamic properties of motion, while preserving the specific intricate details in the original.

The reconstruction relies on both spacetime constraints and motion handles as described in Section 4.1. All spacetime constraints are mapped to their full character equivalents. For example, foot placement constraints are mapped onto foot constraints of the full character. Having completed the spacetime editing stage, we have three distinct sets of handles[1]

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

We define the final motion handles as

$$\mathbf{h}_f(\mathbf{q}_f) = \mathbf{h}_o(\mathbf{q}_o) + (\mathbf{h}_t(\mathbf{q}_t) - \mathbf{h}_s(\mathbf{q}_s)) \tag{6}$$

essentially displacing the original handles by the difference between the two spacetime solutions. Since the right side of the equation is known, it would seem that solving for the inverse-kinematics-like problem of finding $\mathbf{q}_f$ that satisfies equation 6 would complete the reconstruction. Unfortunately, the number of handles is *considerably smaller* than the number of DOFs in the full character, so this problem is highly under-determined, and we cannot directly solve for $\mathbf{q}_f$ without accounting for the extra DOFs.

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

$$\begin{array}{ll} \min_{\mathbf{q}_f} & E_{dm}(\mathbf{q}_o, \mathbf{q}_f) \\ \text{subject to} & \left\{ \begin{array}{l} \mathbf{C}(\mathbf{q}) = 0 \\ \mathbf{h}_f(\mathbf{q}_f) = \mathbf{h}_o(\mathbf{q}_o) + (\mathbf{h}_t(\mathbf{q}_t) - \mathbf{h}_s(\mathbf{q}_s)) \end{array} \right. \end{array} \tag{7}$$

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

We first formulate a measure of closeness to the original motion. A simple objective function that measures the deviation of each DOF $E_{dd} = (\mathbf{q}_f - \mathbf{q}_o)^2$ produces undesirable results. Each DOF needs to be carefully scaled both with respect to what it measures (joint angles measure radians, translational DOFs measure meters), and with respect to its importance within the character hierarchy. For example, the change of the hip joint DOF affects the overall motion significantly more than the same amount of change applied to the ankle joint. In order to avoid these problems, we designed a completely new objective E_{dm} that measures the amount of displaced mass between the two poses.

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

7.1 Minimum Displaced Mass

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

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

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

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

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

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

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

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

This notation allows us to simplify E_k

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

where tr() is a matrix trace operator and the mass matrix tensor $\mathbf{M}_i$ of node i is computed as the integral over body points $\mathbf{x}_j$ of outer products scaled by the node mass m_i

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

Note that because $\overline{\mathbf{W}}_i \mathbf{M}_i \overline{\mathbf{W}}_i^T$ is a constant expression (it does not depend on $\mathbf{q}$) we remove it from the final expression.

We also compute derivatives with respect to kinematic DOFs

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

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

Since the reconstruction process is performed separately for each frame the resulting motion may, on occasion, appear non-smooth. We correct this by defining intervals of animation where improved smoothness is required. The problem defined in Equation 7 is then solved collectively over the entire interval with the added smoothness objective $E_{smooth} = \ddot{\mathbf{q}}^2$.

Jump		Run	
Name	Minutes	Name	Minutes
Fitting	21	Fitting	16
Twist	1.6	Wide	0.9
Diagonal	1.4	Crossed	1.3
Unbalanced	0.6	Limp	1.7
Obstacle	1.4	Short-leg Limp	1.9
		Moon	1.5
		Neptune	1.3

Figure 3: SGI Octane computation times for the run and the jump motion transformations.

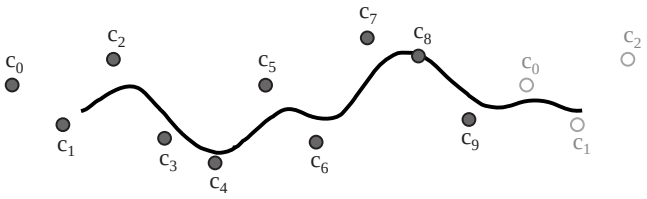

Figure 4: Representation of the cyclical B-spline DOFs.

8 Results

We have created two different motion libraries from which we generated a number of different animations. We used high detail (120Hz) motion capture data as input sequences. During the fitting stage, we used drastically different simplification approaches for the libraries in order to show the versatility of our algorithm.

All of the described motion sequences would be difficult to create with existing motion editing tools. While it is conceivable that a number of constraints could be introduced to enhance realism, for some sequences it would require an overwhelming amount of work, on par with creating a realistic motion sequence from scratch with keyframing. In contrast, our approach requires minimal number of intuitive changes for each transformed sequence.

8.1 Human Run

We extracted a single gait from a human run motion sequence, and made all DOFs cyclic so that the motion could be concatenated into a continuous run sequence of arbitrary length. We represented each DOF with a cyclical B-spline, where the first three coefficients influence the beginning and the end of the DOF values (Figure 4). This formulation forces endpoint values and their time derivatives to coincide.

In the character simplification stage we removed all hand, foot and elbow DOFs reducing the entire arm into one rigid object. Similarly, the foot and shin are represented with a single object. We also preserved only a few upper body DOFs. The torso, head and shoulders have been fused into a single object that has a quaternion ball joint at the waist. Each shoulder has a single Euler hinge joint that allows the arm to move back and forth. This is a significant reduction from the original configuration with a ball joint at each shoulder. Hip joints are the only ball joints that have been preserved during the simplification. (Figure 5). We should note that reducing the arms to a single object with only one DOF significantly restricts the amount of change each resulting motion can undertake. If we wanted to allow greater ability to alter the movement of arms, we would allow for additional DOFs at each shoulder.

In order to map the human motion onto the biped we introduce several types of handles:

Foot contact points. On the human character the floor contact points for each foot are located at the ball of the foot. On the

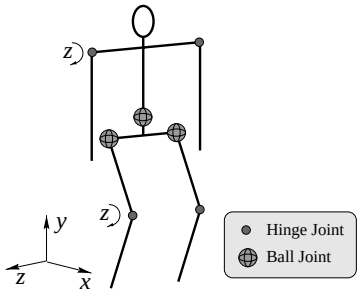

Figure 5: Biped: Simplified character for the human run motion library.

biped these points are shin endpoints. The handles are tracked throughout the entire animation, not just while the foot is on the ground.

Character mass center. The mass centers for the human and the biped follow the same trajectories.

Upper body mass center. We define the upper body as the character subtree rooted at the waist joint. Again, the mass centers follow the same trajectories.

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

Torso orientation. The body direction is tracked the orientation of the lower abdomen node.

Shoulder orientation. The shoulder orientation handle is defined as the unit vector formed by the two points located at the left and right shoulder.

We also defined two mechanical constraints corresponding to the foot-ground contact events. These constraints need to be mechanical since the floor exerts forces onto the body keeping the foot in place. We also constrained the forces produced by the floor constraints so that they are non-negative in the upward direction. This effectively indicates that the floor provides forces which prevent floor penetration, but not floor separation. In addition, we provided a muscle for each kinematic DOF of the biped character. The muscle coefficients were identical for each muscle: $k_s = 4.0$ and $k_d = -0.04$. Our experiments showed that different coefficients affected the speed of convergence for the optimization, but not the resulting motion itself.

We used the objective function described in Section 5.4. Optimizations during the spacetime model fit stage took about 21 minutes to compute on an SGI Octane. Once the spacetime model was completed we created numerous realistic animations by editing various spacetime formulation parameters.

Wide Footsteps. We repositioned the footprint mechanical constraints to be wider apart so that the character would have to leap significantly farther to the side at each step. We kept the constraint time intervals unchanged. Since each step now covered more distance, the overall resulting motion had leaps of smaller height (Figure 8). The appropriate change in the upper body orientation was apparent in the resulting motion.

Crossed Footsteps. We moved footprints to the opposite side of the body, forcing the character to twist at each step, criss-crossing the feet. Again we kept the constraint time intervals unchanged. We also introduced the "slippery floor" objective component, which penalized the component of the floor reaction forces in the floor plane. This effectively forced the character to rely less on the floor friction during landing.

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

Neptune Run. We also increased the Earth's gravity by tenfold to see how the running sequence would adjust to such extreme gravitational field[2]. Naturally, no human muscle forces could possibly produce a running motion under such extreme gravitational field. Consequently we removed any existing bounds on the muscle actuation forces. The resulting flight phase of the run was so low to the ground that the running character had the appearance of speed walking.

Limp Run. We removed the left knee DOF, creating the appearance of the leg being put in a cast. We also reduced the duration of the left footstep mechanical constraint, while we increased the duration of the right footstep. The advent of a straight leg introduced a new problem.

The stiff leg can move either to the outside or to the inside during the flight phase to avoid hitting the ground. These are two distinct local minima. If a leg were to move towards the inside, a more dynamically stable option, it would inevitably collide with the other leg. To prevent this, we included the objective component which penalizes for the closeness of the two foot points during the small time period just as the stiff leg leaves the ground. Effectively, we are biasing the solution towards the one where the legs do not go through each other. We do not need to include this objective during the entire animation since we only need to intervene at the specific point of the solution space bifurcation. We note that this problem would not be solved with the collision detection since the choice to go inside or outside occurs far before the actual collision. In the final motion sequence the character leans to the side and swings the right leg in a more dramatic fashion, creating a realistic (albeit painful) limp run.

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

8.2 Broad Jump

We created the broad jump motion library to explore how far we could simplify the character without losing the dynamic essence of the jump. In order to demonstrate the power and the flexibility of the simplification tools, we used drastically different simplification approach than the one used on the human running motion (Figure 6).

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

Figure 6: Full and simplified characters for the human run and broad jump.

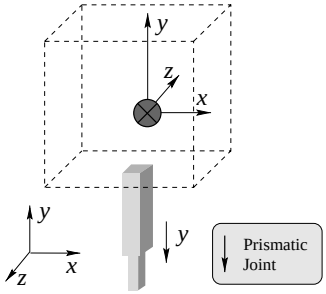

Figure 7: Hopper: Simplified character for the broad jump motion transformation.

The entire upper body structure is reduced to a single mass point. The mass point moves with three prismatic muscles that push off from the rest of the body (Figure 7). Since the legs move together during a broad jump, we turned them into a single leg. Although it was not necessary, we also turned the knee hinge joint into a prismatic joint, to show that even with this completely changed character model the dynamic properties of the broad jump are preserved. The simplified character (hopper) has *ten* DOFs of which six are the global position and orientation of the model. It does not contain any angular joints.

The broad jump motion sequence lasts more than two seconds, which is more than twice as long as the run gait cycle. Since the sequence duration is linearly proportional to the size of the problem it would appear that optimization would take considerably longer. However, due to fewer DOFs, the spacetime fitting optimization converges within 16 minutes. Subsequent transformation optimizations all finish within 2 minutes on an SGI Octane (Figure 3).

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

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

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

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

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

Torso orientation. We preserved the body orientation by tracking the orientation of the lower abdomen character node. For both the human and the hopper character, this is the direction of the unit vector pointing down the forward direction in the local frame of the lower abdomen node.

In addition to handles, we also defined a number of constraints. Again, we used mechanical constraints for the foot-ground constraints. We also specified the initial and the final upright pose constraint. These constraints are needed to make the endpoints of the animation invariant. We also restricted the freedom of movement for the upper body mass point, with simple bound constraints in each dimension. Without these constraints, the upper body mass point would be free to move outside the range of motion of a human upper body center of mass. The rest of this section describes each transformed jump in more detail.

Twist Jump. We introduced the torso orientation pose constraint at the end of the animation, which mandates a 90 degree turn. The output motion clearly shows the change in the anticipation and the introduction of the body twist during the flight stage. The landing stage has also changed to accommodate the sideways landing position.

Diagonal Jump. We displaced the landing position to the side and constrained the torso orientation to point straight ahead at all times. This change realigned the push-off and anticipation stages in the direction of the jump. Since the jump length was increased, the entire resulting motion appears more impulsive.

Obstacle Jump. We raised the landing position and introduced a hurdle, which forces a raising of legs during the flight stage. As a result, the character push-off is more vertical, and the legs tuck in during the flight.

Unbalanced Jump. We removed the final pose constraint that imposed the upright position. In the resulting sequence, the character never uses its muscles to stand up upon landing, since this would require extra energy. Instead of straightening up the character tumbles forward giving the appearance of poor landing balance.

In conclusion, our experiments show that, despite the extreme simplification, the hopper spacetime model still encapsulates the realistic properties of the broad jump with surprising accuracy.

8.3 Limitations

Although our algorithm has been effective for a number of input motion sequences and for a wide spectrum of transformed motion animation sequences, a number of possibilities for improvement remain.

It appears that our methodology is best suited for the animation sequences containing high-energy, dynamic character movement. Other motion sequences that are more lethargic or kinematic, like picking up an object or getting up from the chair are not well suited for our dynamic transformation framework. Of course, it is still possible to apply our technique to such animations, but the benefits of a full-blown dynamics representation would not be large. In general, the non-realism for such motions is much less of an issue. The more a particular motion contains visible dynamic properties, the more suitable it would be for our motion transformation algorithm.

The main shortcoming of our approach is that large portions of the motion fitting algorithm stage are performed manually. We have found the simplification process quite intuitive. The simplification is performed only once per input motion sequence, so the effort spent by the motion library creator is amortized over the large number of possible transformed animation sequences. Nevertheless, automating this manual decision-making process would enable

on-the-fly construction of a physically based spacetime formulation from an input animation.

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

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

9 Conclusion

Our algorithm presents the first solution to the problem of editing captured motion that takes dynamics into consideration.

We also describe a novel methodology for mapping motion between characters with drastically different kinematic structures which broadens the applicability of motion capture data to animation of characters that do not exist in nature.

The paper also presents a method for control of complex dynamic systems with simpler ones. These ideas can be further developed into a realtime optimal robot control planner.

Lastly, these methods can be used for motion analysis in biomechanics and perhaps help prove the very ideas that motivated this algorithm.

10 Acknowledgements

This research was supported by the Schlumberger Foundation Fellowship and National Science Foundation award IRI9502464. The authors wish to thank R.J. Full for motivating certain biomechanics aspects of this work. We also thank Sebastian Grassia, Jovan Popović, Andrew Willmott, Elly Winner and our reviewers for the valuable comments during the preparation of this paper.

References

[1] R. M. Alexander. Optimum walking techniques for quadrupeds and bipeds. *J. Zool., London*, 192:97–117, 1980.

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

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

[4] David Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. In *Computer Graphics (SIGGRAPH 90 Proceedings)*, volume 24, pages 19–28, August 1990.

[5] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics (SIGGRAPH 94 Proceedings)*, July 1994.

[6] David Baraff and Andrew Witkin. Large steps in cloth simulation. In Michael Cohen, editor, *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 43–54, July 1998.

[7] R. Blickhan and R. J. Full. Similarity in multilegged locomotion: bouncing like a monopode. *J Comp. Physiol. A*, 173:509–517, 1993.

[8] Armin Bruderlin and Lance Williams. Motion signal processing. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, pages 97–104, August 1995.

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

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

[11] Paolo de Leva. Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *J. of Biomechanics*, 29(9):1223–1230, 1996.

[12] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In Michael Cohen, editor, *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 85–94, July 1998.

[13] P.E. Gill, M.A. Saunders, and W. Murray. SNOPT: An SQP algorithm for large-scale constrained optimization. Technical Report NA 96-2, University of California, San Diego, 1996.

[14] Michael Gleicher. Motion editing with spacetime constraints. In Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 139–148. ACM SIGGRAPH, April 1997. ISBN 0-89791-884-3.

[15] Michael Gleicher. Retargeting motion to new characters. In *Computer Graphics (SIGGRAPH 98 Proceedings)*, pages 33–42, July 1998.

[16] Michael Gleicher and Peter Litwinowicz. Constraint-based motion adaptation. *The Journal of Visualization and Computer Animation*, 9(2):65–94, 1998.

[17] J. K. Hodgins and N. S. Pollard. Adapting simulated behaviours for new characters. *SIGGRAPH 97*, pages 153–162, 1997.

[18] Jessica K. Hodgins. Animating human motion. *Scientific American*, 278(3):64–69, March 1998.

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

[20] Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. Hierarchical spacetime control. In *Computer Graphics (SIGGRAPH 94 Proceedings)*, July 1994.

[21] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. In *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 289–298, August 1988.

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

[23] D.J. Pearsall, J.G. Reid, and R. Ross. Inertial properties of the human trunk of males determined from magnetic resonance imaging. *Annals of Biomed. Eng.*, 22:692–706, 1994.

[24] A. Pedotti, V. V. Krishnan, and L. Stark. Optimization of muscle-force sequencing in human locomotion. *Math. Biosci.*, 38:57–76, 1978.

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

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

[27] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications*, 18(5), September – October 1998.

[28] C. Rose, B. Guenter, B. Bodenheimer, and M. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Computer Graphics (SIGGRAPH 96 Proceedings)*, pages 147–154, 1996.

[29] A. Seireg and R. J. Arvikar. The prediction of muscular load sharing and joint forces in the lower extremities during walking. *J. Biomechanics*, 8:89–102, 1975.

[30] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–224, 1997.

[31] Michiel van de Panne and Eugene Fiume. Sensor-actuator networks. In *Computer Graphics (SIGGRAPH 93 Proceedings)*, volume 27, pages 335–342, August 1993.

[32] Michiel van de Panne and Eugene Fiume. Virtual wind-up toys. In *Proceedings of Graphics Interface 94*, May 1994.

[33] Andrew Witkin and Michael Kass. Spacetime constraints. In *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 159–168, August 1988.

[34] Andrew Witkin and Zoran Popović. Motion warping. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, August 1995.

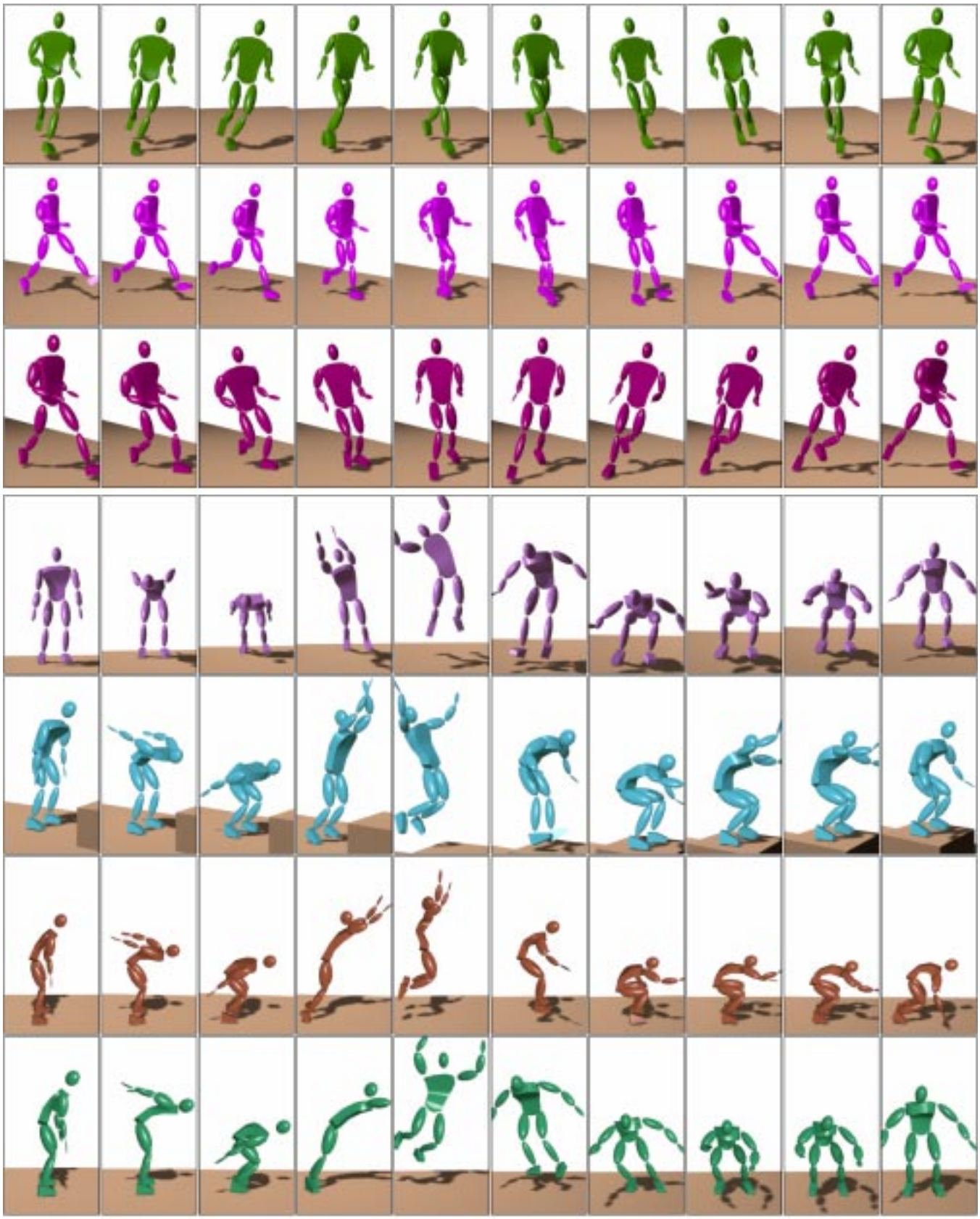

Figure 8: Frames from the crossed footsteps, limp, wide footsteps run; and the diagonal, obstacle, unbalanced, and twist jump.