Physical Touch-Up of Human Motions

Hyun Joon Shin Lucas Kovar Michael Gleicher
University of Wisconsin

Abstract

Many popular motion editing methods do not take physical principles into account potentially producing implausible motions. This paper introduces an efficient method for touching up edited motions to improve physical plausibility. We start by estimating a mass distribution consistent with reference motions known to be physically correct. The edited motion is then divided into ground and flight stages and adjusted to enforce appropriate physical laws for, respectively, zero moment point (ZMP) constraints and correct ballistic trajectory. Unlike previous methods, we do not solve a nonlinear optimization to calculate the adjustment. Instead, closed-form methods are used to construct a hierarchical displacement map which sequentially refines user-specified degrees of freedom at different scales. This is combined with standard methods for kinematic constraint enforcement, yielding an efficient and scalable editing method that allows users to model real human behaviors. The potential of our approach is demonstrated in a number of examples.

Keywords: Motion Editing, Animation of Articulated Figures, Motion Control, Physically-based Animation.

1 Introduction

Many motion editing techniques, including warping, filtering, and retargeting, do not explicitly incorporate physics. Important physical properties may therefore be lost in the editing process, making motions appear unrealistic. For example, if simple displacement mapping methods are used to alter a straight walk to follow a curving path [9], the character will remain vertical, whereas a person would lean toward the center of curvature to balance centripetal forces. In spite of such difficulties, however, these editing methods remain popular due to their simplicity, computational efficiency, and ease of control. Moreover, the generated motions are often *approximately* physically correct, and so it seems reasonable to simply adjust them so physical laws are obeyed.

Unfortunately, enforcing physical properties can be a difficult task. Since motion dynamics are governed by a system of nonlinear differential equations, solution methods are often computationally expensive. Moreover, there are many ways to adjust a motion so physical laws are satisfied, and yet only a subset yield natural-looking motions. Previous efforts have used the heuristic that a *minimal* change is the most natural and employed optimization techniques to simultaneously adjust many or all of a character's degrees of freedom. While this is capable of producing striking results, it comes at great computations expense, particularly since the optimization typically spans the entire motion. Additionally, optimizing over a large number of DOFs at once can make it difficult for an artist to take known human behaviors into account. For example, the artist may know a priori that balance adjustment should be handled by the upper body rather than the lower body.

This paper explores a different approach that seeks to retain the advantages of simplicity, efficiency, and control. Our approach starts with a preprocessing step that estimates the character's mass distribution using reference motions known to be physically plausible. The user may then edit a particular motion with standard methods (e.g., filtering to remove noise) and use our system to touch-up the results. Our system divides the edited motion into ground and flight stages and enforces appropriate physical laws in a multi-step process. At each step, the user specifies a portion of the body to be adjusted and a time scale over which changes should be introduced. Appropriate displacements are then computed independently on each frame using efficient closed-form methods, and the result is smoothed based on the time scale. While physical constraints may still fail to hold after a particular iteration, the motion is likely to be improved, and remaining errors can be reduced in subsequent iterations. Moreover, the computation is sufficiently fast that a reasonably lengthy motion (e.g., hundreds of frames) can be processed in its entirety at interactive rates, providing quick feedback to the user.

We note that even original, unedited motions are not necessarily physically correct. While these motions are based on observations of a real performer (who necessarily obeyed the laws of physics), errors are introduced in the capture process and in modelling the human as a rigid skeleton.

Therefore the skeletal movements themselves may not obey physical laws. However, it is reasonable to assume that at least some of these unedited motions *appear* to be physically plausible, and we therefore construct a physical model for the character (i.e., a mass distribution) based on these motions.

The remainder of the paper is organized as follows. First, Section 2 reviews related work. We then present our framework for physical touch-up in Section 3 and present some experimental results in Section 4. Finally, we conclude in Section 5 with a discussion of some of the advantages and limitations of our approach.

2 Related Work

One way to produce physically plausible motions is to generate them directly from physical simulation. Hodgins et al. [10] handcrafted controllers based on finite state machines and proportional-derivative servos to produce running, bicycling, and vaulting motions. Using similar methods, Wooten and Hodgins [26] simulated gymnastics motions such as flipping and tumbling, and Faloutsos et al. [6] simulated motions to preserve balance and recover from a fall. Bruderlin and Calvert combined physical simulation with empirically-motivated kinematic methods to generate walking [2] and running [3] motions. Liu and Popovic [15] demonstrated that motions dominated by dynamics, such as leaping or flipping, can be synthesized almost entirely from conservation laws, without computing joint torques explicitly. Fang and Pollard [7] improved the efficiency of this approach by differentiating aggregate force and torque in linear time. While these methods have been successful at producing certain kinds of motions (e.g., ballistic actions like diving), many other motions are beyond their reach. Moreover, a motion that is physically plausible may nonetheless appear unnaturally stiff and robotic, and it is difficult to encode stylistic attributes into physical simulation.

Motion capture, in contrast, can produce a rich variety of highly realistic motions. However, by itself motion capture offers no control over the motion, and so much effort has been invested in developing editing methods. Some of these have not explicitly taken physics into account, instead using signal processing methods [4, 25] or the enforcement of kinematic constraints [8, 13]. Our technique complements these methods, allowing one to touch-up an edited motion that breaks physical laws.

Other motion editing methods have been based around physical principles. Rose et al [21] generated realistic transitions by minimizing joint torques. Ko and Badler [11] and Dasgupta and Nakamura [5] attempted to preserve the dynamic validity of a motion by requiring a character's zero moment point to remain inside its support polygon. Yamane and Nakamura [27] and Pollard and Reitsma [18] used physical simulation to produce motions that tracked a reference motion while adhering to constraints on joint torque and environmental contacts. These efforts all relied on expensive numerical solution methods methods, whereas we sacrifice optimality and some physical rigor for greatly improved efficiency. To reduce the cost of optimization, Popović and Witkin [19] extracted a low-DOF character model capturing essential dynamical features and then optimized this simpler construct to satisfy user-specified constraints. Pollard [17] adopted a similar approach, but used a simpler editing method to achieve interactive rates. We maintain computational efficiency while operating directly on the human character. Zordan and Hodgins [28] used motion capture data to drive physical simulations in which a character can strike to specific locations and maintain balance while subject to impulsive forces.

The most similar work to our own is that of Tak et al. [22], who used an iterative algorithm based on Kalman filters to enforce kinematic constraints, dynamic balance, and joint strength limits. As with our work, they computed adjustments on a per-frame basis and filtered the results, allowing interactive motion editing. Also, joint strength limits could be used to control the form of the output motion. While we believe their system could be adapted to reproduce our results, our approach is significantly different: rather than numerically optimizing over all DOFs simultaneously, we compute closed-form adjustments to individual DOFs to improve efficiency and controllability.

To determine a character's mass distribution, previous editing methods have either used average distributions from the biomechanics literature or explicitly altered masses to achieve certain effects [10, 19, 15, 11, 22]. To our knowledge, our work is the first attempt to calculate the mass distribution of a captured motion for animation purposes.

3 A Framework for Physical Touch-up

3.1 Overview

Depending on whether the body is airborne or in contact with the ground, it is subject to different external forces and hence different physical laws apply. We address these flight and ground stages separately. To determine whether or not the character is airborne on a particular frame, we assume the motion is labelled with kinematic constraints specifying joint contact with the ground. We ensure that these constraints are enforced along with the physical constraints (e.g, footplants are maintained after physical touch-up). Since the physical laws for flight frames and ground frames are quite different, accurate labelling of the kinematic constraints is quite important. We employ automated methods [1] and manually adjust the results.

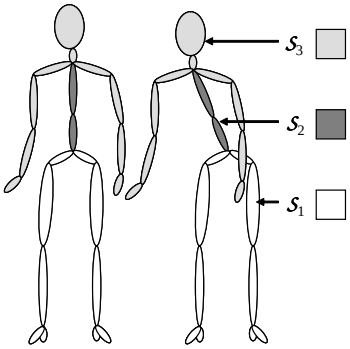

Figure 1. An example of link grouping: the lower body is fixed, the stomach and chest are to be rotated, and the head and arms keep their global orientations.

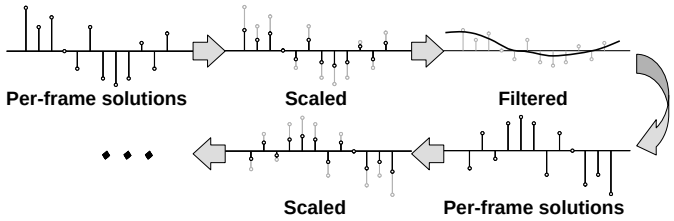

Figure 2. A schematic of a single iteration. Adjustments are calculated independently on each frame, scaled, and filtered.

When the body is airborne, the only external force acting on it is gravity, and hence its center of mass must follow a parabolic trajectory and its total angular momentum must be conserved. Mathematically, these constraints may be respectively expressed as

$$\mathbf{c}_M(t) = \frac{\sum_i m_i \mathbf{r}_i(t)}{\sum_i m_i} = C_1 t^2 + C_2 t + C_3, \quad \text{and} \quad (1)$$

$$\mathbf{h}_{CM} = \sum_i m_i \left(\mathbf{r}_i - \mathbf{c}_M\right) \times \dot{\mathbf{r}}_i = D, \quad (2)$$

where $\mathbf{c}_M$ is the center of mass, $\mathbf{h}_{CM}$ is the angular momentum, m_i and $\mathbf{r}_i$ are respectively the mass and the position of the center of mass for the i^{th} link, and C_1, C_2, C_3, and D are constants.

When the body is in contact with the ground, in addition to gravity it is also subject to ground reaction forces, i.e., normal forces and friction. There must be a point $\mathbf{Z}$ in the convex hull of the contact region (called the *support polygon*) where the net torque due to these ground forces is zero. This point is called the *zero moment point* (ZMP) [23]. At the ZMP the sum of the torques for each body link equals the total torque due to gravity:

$$\sum_i m_i(\mathbf{r_i} - \mathbf{Z}) \times \ddot{\mathbf{r}}_\mathbf{i} = \sum_i (\mathbf{r_i} - \mathbf{Z}) \times m_i \mathbf{g}, \quad (3)$$

where $\mathbf{g}$ is the gravity vector. Assuming the floor plane is $y = 0$, the ZMP can be given explicitly by

$$Z_x = \frac{\sum_i m_i(\ddot{r}_{i_y} - g_y) r_{i_x} - \sum_i m_i(\ddot{r}_{i_x} - g_x) r_{i_y}}{\sum_i m_i(\ddot{r}_{i_y} - g_y)},$$
$$Z_y = 0, \quad \text{and} \quad (4)$$
$$Z_z = \frac{\sum_i m_i(\ddot{r}_{i_y} - g_y) r_{i_z} - \sum_i m_i(\ddot{r}_{i_z} - g_z) r_{i_y}}{\sum_i m_i(\ddot{r}_{i_y} - g_y)}.$$

The accelerations terms in this equation must be computed with some care, as any noise in the motion will be magnified considerably through differentiation. In our experiments, we first applied a gaussian filter of width 5 to the motion and then used finite differences.

When the ZMP is outside of the support polygon, the resulting physical inconsistency often manifests itself as a lack of balance, and correcting the ZMP tends to place the character back into balance. For example, say we start with a straight walking motion and use a displacement map to make it follow a curving path, as described in [9]. Enforcing the ZMP constraint makes the character lean toward the center of curvature, as would happen in real life. Similarly, if a transition between walking and running is created via simple blending methods, then enforcing the ZMP constraint will make the character lean forward during the transition, which is also what happens in reality. This is *not* to suggest, however, that satisfying the ZMP constraint is identical to being in balance. Indeed, if a real person trips and falls, their ZMP still remains inside the support polygon.

Since the human body has many degrees of freedom and only a small number of physical constraints exist on any particular frame, there are many ways to adjust the body so these constraints are satisfied. We provide the user with three types of high-level control over the form of the adjustment:

1. **Body shape ($\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$).** The skeleton is divided into three sets of connected links (Figure 1): those which are not to be altered ($\mathcal{S}_1$), those which may have their position and orientation changed ($\mathcal{S}_2$), and those which may be translated but must retain the original global orientation ($\mathcal{S}_3$). The boundary between $\mathcal{S}_1$ and $\mathcal{S}_2$ and boundary between $\mathcal{S}_2$ and $\mathcal{S}_3$ both consist of a single joint, and $\mathcal{S}_1$ and $\mathcal{S}_3$ are disconnected. This division into link sets allows a user to operate on logical units.

2. **Adjustment time scale (σ).** To control the frequency content of the adjustment, the user specifies a time

scale σ over which changes are to be introduced. This allows the user to model, for example, a sway versus a lean.

3. **Adjustment damping factor (ρ).** Instead of satisfying the constraints completely, the user may choose to only use a portion ρ of a calculated adjustment and leave the remaining error for subsequent adjustments. This allows different movements to be layered on top of each other.

Thus, to alter the motion the user specifies a rough division of the skeleton into groups that should change coherently, plus two numbers that roughly correspond to meaningful quantities (namely, the frequency at which adjustment should occur and the fraction of physical error that should be removed at this step).

Our algorithm proceeds by building two sequences of displacement maps, one for ground stages and one for flight stages, that adjust the motion to better satisfy the appropriate physical laws. A single displacement map is constructed as follows (Figure 2). First, based on the user's choice of $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, an adjustment is computed independently on each frame to satisfy the constraints as much as possible. These adjustments are then scaled according to ρ. Finally, to make these adjustments coherent, they are filtered using a smoothing kernel whose width is determined by σ. To filter orientation data, we use the method proposed by Lee and Shin [14].

The rest of this section presents the details of our approach. In Section 3.2, we explain how the character's mass distribution is determined. In Sections 3.3 and 3.4, we describe how adjustments are calculated for ground frames and flight frames, respectively.

3.2 Mass Distribution Estimation

The physical laws we are concerned with involve quantities like the center of mass, angular momentum, and the ZMP, all of which depend upon the mass distribution of the character. However, the motion data itself contains no mass information. We can nonetheless leverage this data by using original, unedited motions to fit a mass distribution to the character such that the relevant physical laws are satisfied. Using this calculated mass distribution has the advantage that applying our touch-up method to an unedited motion will produce little to no change, as one would expect intuitively[1]. In modelling the mass distribution of the character, we assume all the mass of a link is concentrated at its center of mass. In practice, we have found that this simplification does not introduce a significant a error in comparison to using a larger set of mass points; see Section 5.

Since motions typically have many more ground frames than flight frames, we use the ZMP constraints to determine the mass distribution. We find values for the m_i's such that, to the extent possible, the ZMP is inside the support polygon when the character is in contact with the ground. Specifically, if $f(\mathbf{m}, t)$ is the distance of the ZMP from the support polygon on frame t, we compute

$$\min_{\mathbf{m}} \sum_{t \in \mathcal{F}_G} f(\mathbf{m}, t), \tag{5}$$

where $\mathcal{F}_G$ is the set of ground frames. This is a nonlinear optimization, and so an iterative numerical procedure is required. We use the conjugate gradient method [20], with the initial guess for the masses based on measured average values [24]. This computation is done as a preprocess, so there are no speed penalties incurred when the user interacts with the system.

3.3 Enforcing Ground Frame Constraints

To determine if a particular ground frame is physically plausible, we first need to calculate its ZMP. Because the ZMP equation involves acceleration terms, in general adjustments to one frame influence the ZMP on other frames. However, since the per-frame displacements are filtered, we ultimately generate a smooth displacement map, and hence the magnitude of the acceleration will in general be small relative to the overall amplitude of the displacement. For this reason, we assume that the change in acceleration is small enough to ignore, allowing us to satisfy the ZMP constraint on a frame-by-frame basis. Moreover, since the motion is touched-up through several iterations, the error at an iteration due to this assumption can be compensated by the next iteration.

To adjust a particular frame, we first apply a rotation to the links in $\mathcal{S}_2$ and $\mathcal{S}_3$. The global orientation of the links in $\mathcal{S}_3$ are then maintained by applying the inverse of this rotation to the joint forming the boundary between $\mathcal{S}_2$ and $\mathcal{S}_3$. This process requires the following parameters to be determined: the center of rotation for the initial rotation, the axis of this rotation, and the rotation angle. If $\mathcal{S}_1$ contains joints that are to remain fixed on the ground, then the center of rotation is coincident with the joint bordering $\mathcal{S}_1$ and $\mathcal{S}_2$. Otherwise, to preserve kinematic constraints we place the center of rotation closer to the ground, namely, at the projection of the ZMP onto the support polygon.

We now must determine the rotation axis and angle. Refer to Figure 3. If we temporarily assume that the axis of rotation is the z-axis, then a rotation by θ yields a new ZMP

[1] Note that an *average* mass distribution, such as one would find in the biomechanics literature, may not have this property. In particular, there is no reason to believe that the performer is of average build.

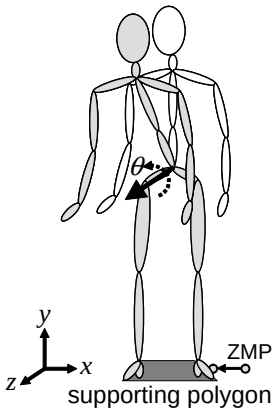

Figure 3. Adjusting a single frame to satisfy the ZMP constraint. Note that the axis of rotation is perpendicular to the line from the ZMP to its projection onto the support polygon.

$\mathbf{Z}'$ as follows:

$$\begin{aligned} Z'_x &= a_1 \cos\theta - a_2 \sin\theta + a_3, \text{ and} \\ Z'_z &= a_4 \cos\theta + a_5 \sin\theta + a_6, \end{aligned} \tag{6}$$

where $a_1, \cdots, a_6$ are coefficients (given in the Appendix) determined by m_i, $\mathbf{r}_i$, $\ddot{\mathbf{r}}_i$ and the rotation centers. The change in the Z_z is sufficiently smaller than the change in Z_x that it may be neglected (see Appendix). Using this approximation, it is clear that the rotation axis should be pointed perpendicular to the direction we would like the ZMP to move. Hence the rotation axis is set parallel to the ground plane and orthogonal to the line connecting the current ZMP position to its projection onto the support polygon. Once the rotation axis is chosen, we first transform the coordinate frame to align the rotation axis into z-axis. The rotation angle is then easily computed from Equation (6):

$$\arctan\left(\frac{\pm\sqrt{a_1^2 + a_2^2 - (Z'_x - a_3)^2}}{Z'_x - a_3}\right) - \arctan\left(\frac{a_2}{a_1}\right), \tag{7}$$

where $\mathbf{Z}'$ is now the projection of the ZMP onto the support polygon. Of the two solutions, we choose the one with the smaller absolute value, and to place θ in the correct quadrant we compute the inverse tangent with the $atan2$ function in the C standard library. If the value in the square root is negative, then the constraint cannot be satisfied exactly; physically, this means that there is no rotation placing the ZMP exactly at $\mathbf{Z}'$. In this case we set the square root to zero, which has the effect of placing the ZMP as close as possible to its projection. We then transform the aligned coordinate frame back to its initial coordinate frame.

Kinematic constraints such as footplants may be violated as a result of adding the displacement map. To make sure they are once again satisfied, we employ the method of [12], because of its efficiency. While this change may cause the ZMP constraints to be broken (and they may be broken anyway, due to scaling and filtering of the per-frame adjustments), the motion is likely to still be improved overall, and additional iterations can be used to remove remaining errors.

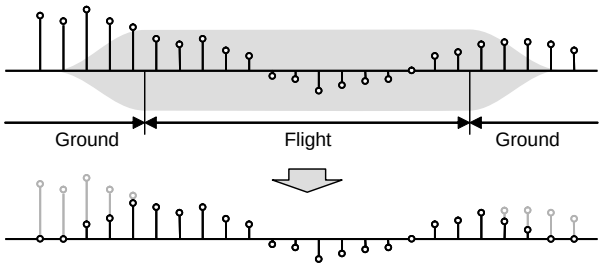

Figure 4. For ground frames near a flight stage, adjustments are computed as for the flight frames and blended to zero.

3.4 Enforcing Flight Frame Constraints

We touch-up flight frames through two sequential adjustments, the first to enforce angular momentum conservation and the second to make the center of mass follow a parabolic trajectory. To facilitate simple, closed-form solutions, the link set $\mathcal{S}_3$ is assumed empty. Also, to make ground and flight stages connect more smoothly, we have found it useful to adjust small neighborhoods of nearby ground frames as if they were part of the flight stage. The adjustments on these ground frames are smoothly blended to zero as depicted in Figure 4. In our experiments we used 15 frames ($\frac{1}{2}$ second), although this value may be changed by the user. Altering these ground frames may violate kinematic constraints; they are once again enforced using the method in [12].

3.4.1 Conservation of Angular Momentum

We adjust each flight frame so its angular momentum is the average angular momentum $\overline{\mathbf{h}}_{CM}$ over the surrounding block of flight frames. The adjustment is achieved by rotating the links in $\mathcal{S}_2$. We denote the center of rotation as $\mathbf{p}_{rot}$. If $\mathcal{S}_1$ is empty, $\mathbf{p}_{rot}$ is the center of mass of the entire body. Otherwise, $\mathbf{p}_{rot}$ is coincident with the joint at the boundary of $\mathcal{S}_1$ and $\mathcal{S}_2$.

Let $\mathbf{r}_c = \mathbf{p}_{rot} - \mathbf{c}_M$. Then the angular momentum $\mathbf{h}_{p_{rot}}$ about $\mathbf{p}_{rot}$ may be expressed in terms of $\mathbf{h}_{CM}$ and the linear momentum $\mathbf{p}$ as follows:

$$\mathbf{h}_{p_{rot}} = \mathbf{h}_{CM} - \mathbf{r}_c \times \mathbf{p}.$$

The target angular momentum $\mathbf{h}^*_{p_{rot}}$ about the center of rotation is then given by $\mathbf{h}^*_{p_{rot}} = \overline{\mathbf{h}}_{CM} - \mathbf{r}_c \times \mathbf{p}$. Using the same reasoning as in Section 3.3, we assume the velocities $\dot{\mathbf{r}}_\mathbf{i}$ are approximately unchanged by the adjustment. Under this assumption, $\mathbf{p}$ is unchanged, and hence the right hand side is entirely calculable from known quantities. To compute the rotation that should be applied to the links in $\mathcal{S}_2$, we use the fact that

$$\mathbf{h}^*_{p_{rot}} = \mathbf{h}_{\mathcal{S}_1} + \mathbf{I}_{\mathcal{S}_2}\omega^*_{\mathcal{S}_2},$$

where $\mathbf{h}_{\mathcal{S}_1}$ is the contribution of the links in $\mathcal{S}_1$ to the angular momentum, $\mathbf{I}_{\mathcal{S}_\mathbf{2}}$ is the moment of inertia of the links in $\mathcal{S}_2$[2], and $\omega^*_{\mathcal{S}_2}$ is the desired angular velocity. Assuming that the necessary adjustment to the frame is reasonably small, $\mathbf{I}_{\mathcal{S}_\mathbf{2}}$ will remain approximately constant, and so we have

$$\omega^*_{\mathcal{S}_2} = {\mathbf{I}_{\mathcal{S}_\mathbf{2}}}^{-1}\left(\mathbf{h}^*_{p_{rot}} - \mathbf{h}_{\mathcal{S}_1}\right). \tag{8}$$

If the time between frames is Δt, the current orientation of $\mathcal{S}_2$ is $\mathbf{q}$, and the orientation on the previous frame was $\mathbf{q}'$, then we have the relationship

$$\mathbf{q} = \exp\left(\omega_{\mathcal{S}_2}\Delta t/2\right)\mathbf{q}',$$

and hence the rotation applied to $\mathcal{S}_2$ is

$$\Delta\mathbf{q} = \exp\left(\omega^*_{\mathcal{S}_2}\Delta t/2\right)\exp\left(-\omega_{\mathcal{S}_2}\Delta t/2\right), \tag{9}$$

where we compute $\omega_{\mathcal{S}_2}$ as $\mathbf{I}^{-\mathbf{1}}_{\mathcal{S}_\mathbf{2}}\mathbf{h}_{\mathcal{S}_\mathbf{2}}$.

3.4.2 Parabolic Center of Mass Trajectory

Within a block of flight frames, the center of mass must follow a parabolic path. We indirectly determine this parabola using the duration T of the flight stage and the total displacement $\mathbf{d}$ of the center of mass. The initial velocity at the start of the flight stage is

$$\mathbf{v}_0 = \frac{\mathbf{d}}{T} - \frac{\mathbf{g}T}{2},$$

and the desired trajectory $\mathbf{c}^*_M(t)$ of the center of mass is

$$\mathbf{c}^*_M(t) = \mathbf{c}_M(t_0) + \mathbf{v}_0 \cdot (t - t_0) + \frac{1}{2}\mathbf{g} \cdot (t - t_0)^2,$$

where t_0 is the time when the flight stage begins.

If $\mathcal{S}_\mathbf{1}$ is empty, then we simply translate the root so the center of mass is at the appropriate point. Otherwise we rotate $\mathcal{S}_\mathbf{2}$ about the joint bordering $\mathcal{S}_\mathbf{1}$ and $\mathcal{S}_\mathbf{2}$. We can write the center of mass as

$$\mathbf{c}_M(t) = \frac{1}{\sum_i m_i}\left(\mathbf{c}_{M_1}(t)\sum_{i\in\mathcal{S}_1} m_i + \mathbf{c}_{M_2}(t)\sum_{i\in\mathcal{S}_2} m_i\right), \tag{10}$$

[2]The moment of inertia of a set of mass points $\mathcal{S}$ is computed based on their configuration: $\mathbf{I}_\mathcal{S} = \sum_{i\in\mathcal{S}} m_i\mathbf{r}_i\mathbf{r}_i^T$, where $\mathbf{r}_i$ is the vector from the rotation center to ith point mass.

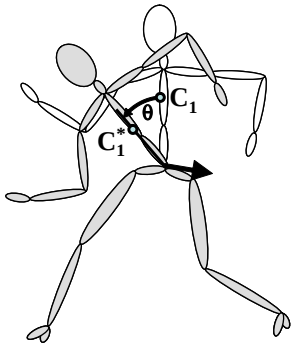

Figure 5. Rotating $\mathcal{S}_2$ to correct its center of mass.

where $\mathbf{c}_{M_1}(t)$ and $\mathbf{c}_{M_2}(t)$ are the centers of mass of $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. The goal position $\mathbf{c}^*_{M_2}$ of the center of mass of $\mathcal{S}_2$ is

$$\mathbf{c}^*_{M_2}(t) = \frac{1}{\sum_{i\in\mathcal{S}_2} m_i}\left(\mathbf{c}^*_M(t)\sum_i m_i - \mathbf{c}_{M_1}(t)\sum_{i\in\mathcal{S}_1} m_i\right).$$

$\mathcal{S}_2$ is rotated so $\mathbf{c}_{M_2}(t)$ is as close as possible to $\mathbf{c}^*_{M_2}$, as depicted in Figure 5.

4 Experimental Results

4.1 Parameter Settings

One of the advantages of our approach is that a user can explicitly model human behavior using a small number of parameters, and the method's computational efficiency provides a quick feedback loop for tuning. Moreover, particular sets of parameters can be reused for different motions. We primarily used two sets of default parameters, one for ground frames and the other for flight frames. For ground frames, parameters were based on the heuristic that overall balancing is accomplished by tilting the whole body while keeping the upper body relatively straight, with secondary balancing actions in the torso and a final fine-tuning performed with the arms. This translated into a sequence of three adjustments. Our first adjustment placed the lower body in $\mathcal{S}_2$ and the upper body in $\mathcal{S}_3$, and smoothing was done with a filter kernel 40 frames wide to mimic smooth behavior of lower body. The second adjustment placed the lower body in $\mathcal{S}_1$ and the upper body in $\mathcal{S}_2$, and it used a filter kernel 20 frames wide. Finally, the third adjustment placed the arms in $\mathcal{S}_2$ and the rest of the body in $\mathcal{S}_1$, and it used a 10-frame kernel since the arm should be able to introduce more rapid acceleration. The ρ's were respectively set to 0.5, 0.3, and 0.2.

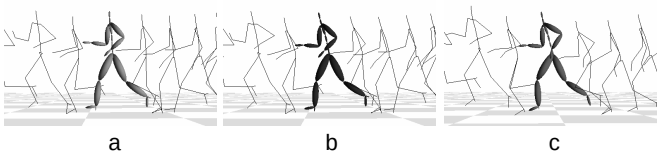

Figure 6. Transition from walking to jogging: (a) original, (b) touched up, and (c) captured.

During flight stages, we attempted to model tucking actions by separately modifying the lower body and the upper body. That is, first $\mathcal{S}_2$ had the lower body and $\mathcal{S}_1$ had the upper body, and then we swapped the link sets. A final adjustment was then made with all links in $\mathcal{S}_2$. We set the first two ρ's to 0.3 and the final ρ to 1. While the ground stage required relatively lengthy filtering windows to smooth over noise in the ZMP calculations, we were able to use smaller filtering windows for the flight stage: 10, 10, and 5, respectively.

Of course, a user can always tweak a set of default parameters to obtain specific behaviors. While the default parameters above worked well in our first, second, and fourth experiments, in the third experiment (Section 4.4) we wanted more of the adjustment in the upper body. We achieved this simply by swapping the order of the first two adjustments. The small number of parameters used by our system, combined with our grouping scheme, facilitates this sort of control. Moreover, since our system adjusts one DOF at a time, the user can reproduce a particular behavior in top-down manner; that is, one can first model the global behavior and then add smaller details. In our experiments, we used a PC with an Intel(R) Xeon 2.0 GHz processor and 1 GB memory.

4.2 Transition from Walking to Jogging

Our first experiment used a transition between walking and jogging created with standard signal-blending methods [16]. In real life, a person would lean forward during the transition so as not to be tipped backwards as they accelerate. However, since no physical principles were used when generating the transition, the character remains upright. Our method produced the expected lean. Figure 6 shows the result and compares it with a captured motion of a live performer transitioning from walking to jogging. This motion has 98 frames and it took 0.108 seconds to process this motion.

4.3 Path-Edited Walking

In our second experiment, we adjusted a straight walking motion to follow a curved path using the method of [9].

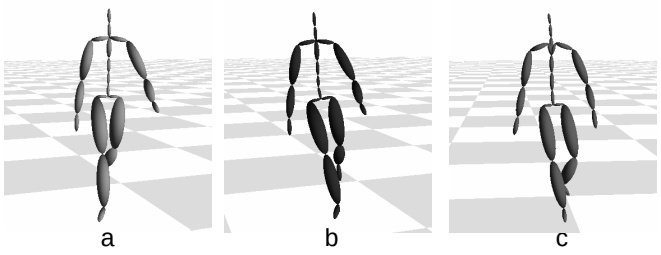

Figure 7. Curved walking: (a) original, (b) touched up, and (c) captured.

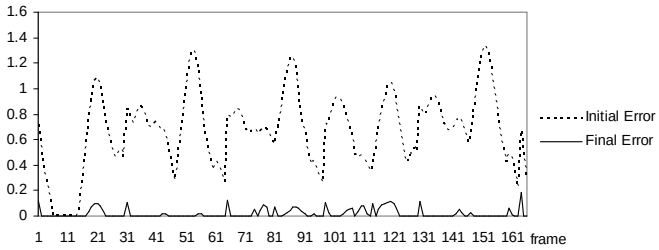

Figure 8. Distance of the ZMP from the supporting polygon before and after the touch-up.

The resulting motion is largely correct, except the character does not lean toward the center of curvature. In reality, centripetal forces would knock the character over. Our method adds the expected inward tilting (Figure 7) and produces a swing pattern in the upper body that compares favorably with a captured motion of a person walking in a curve. For this particular example, our method spent 0.172 seconds to process 111 frames.

Figure 8 illustrates the distance between the ZMP and the supporting polygon at each frame of both the initial motion and the touched-up motion. Here the character has been scaled so its height is 10 units. Figure 9 shows the average distance of ZMP from the supporting polygon at each iteration. We adjusted the lower body with five iterations, and iterated the touch-up for the upper body five times followed by two iterations for the arms. Then plot shows that the error gradually converges to zero.

4.4 Walking Uphill

Figure 10 shows a straight walking motion rotated so as to travel uphill. The motion is clearly physically implausible; the character is out of balance. We used our method to add a natural-looking forward lean to the walk. 0.211 sec-

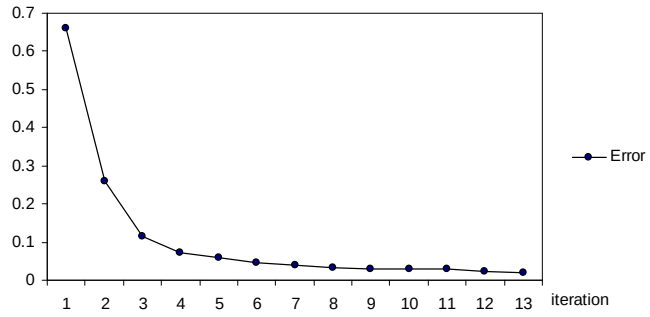

Figure 9. Average distance at each iteration.

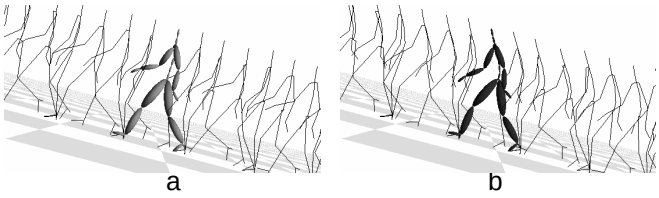

Figure 10. Climbing uphill: (a) original, and (b) touched up.

onds were required to touch-up this motion, which has 166 frames.

4.5 Curved Jumping

For our final experiment, we used path editing to change a straight jumping motion to one that followed a curved trajectory. Such a trajectory is clearly infeasible; see Figure 11. Our editing algorithm produced a new motion that started and ended in the same place but looked more realistic. Not only was the ballistic component of the motion plausible, but the landing stage exhibited a balancing action that one would expect from a human. For this experiment, our system spent 0.14 sec to process 91 frames.

5 Discussion

This paper has presented a method for touching up an edited motion to satisfy physical constraints. To control the process, the user specifies a sequence of adjustments that operate on different parts of the body and at different time scales, allowing explicit modelling of human movement. The algorithm itself is quite efficient, only requiring simple closed-form calculations, and our per-frame approach is suitable for online applications.

It should be emphasized that our system is not intended for demanding physics-based synthesis applications, such

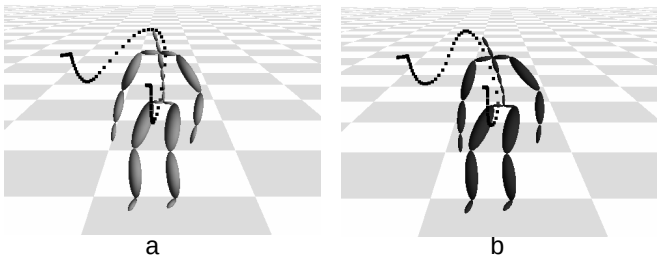

Figure 11. Curved jumping: (a) original, and (b) touched up.

as the one discussed in [15]. The simplicity and speed of our approach is made possible through approximations that assume the total necessary adjustment is reasonably small. Our system is geared primarily toward the addition of nuances like balance-preserving leans, which are of small amplitude compared to the overall motion, tedious to keyframe directly, and readily noticeable when absent.

We have represented each link of the character as a point mass located at the center of the link. More accurate representations may be desired. For example, it would be a straightforward extension to represent links as either larger clouds of points or solid geometric models. However, we have performed some experiments that suggest this will not yield qualitatively different results. We tried modelling each link as a box with a hundred points uniformly distributed in its interior, where each point had the same mass and the total mass was determined by the method of Section 3.2. For the motions used in our experiments, the differences between the angular momentum of the point-mass model and the box model were less than 2%, and the maximum difference in the computed ZMP position were less than 1% of the height of the character.

To produce realistic results, we have found it critical that the kinematic constraints be labelled accurately. Since the support polygon is determined by which joints are on the ground, mislabelling joint-ground contact states can yield significant inaccuracies in the ZMP projection. Also, the physical laws are very different for ground versus flight frames. For example, Equation (4) is technically undefined when the character is in flight — the force on the center of mass in the vertical direction is identical to gravity, and hence the denominators are zero. For motions with relatively clear changes in contact state (e.g., locomotion), we have found it to be straightforward for a human to accurately identify the kinematic constraints. For motions that are both highly dynamic and involve rapid changes in contact state (say, breakdancing), it is less clear how our approach would fare.

Acknowledgements

We would like to thank House of Moves for donating motion capture data. This work was supported in part by NSF grants CCR-9984506 and CCR-0204372.

References

[1] R. Bindiganavale and N. Badler. Motion abstraction and mapping with spatial constraints. In *Modelling and Motion Capture Techniques for Virtual Environments, CAPTECH'98*, pages 70–82, Nov. 1998.

[2] A. Bruderlin and T. Calvert. Goal-directed dynamic animation of human walking. In *Proceedings of ACM SIGGRAPH 89*, Annual Conference Series, pages 233–242. ACM SIGGRAPH, Aug. 1989.

[3] A. Bruderlin and T. Calvert. Knowledge-driven, interactive animation of human running. In *Graphics Interface*, pages 213–221. Canadian Human-Computer Communications Society, May 1996.

[4] A. Bruderlin and L. Williams. Motion signal processing. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, pages 97–104. ACM SIGGRAPH, Aug. 1995.

[5] A. Dasgupta and Y. Nakamura. Making feasible walking motion of humanoid robots from human motion capture data. In *the 1999 IEEE International Conference on Robotics & Automation*, pages 1044–1049, May 1999.

[6] P. Faloutsos, M. van de Panne, and D. Terzopoulos. The virtual stuntman: Dynamic characters with a repetoire of autonomous motor skills. *Computers and Graphics*, 25(6):933–953, 2001.

[7] A. C. Fang and N. S. Pollard. Efficient synthesis of physically vaild human motion. In *Proceedings of ACM SIGGRAPH 2003*, Annual Conference Series. ACM SIGGRAPH, 2003.

[8] M. Gleicher. Retargeting motion to new characters. In *Proceedings 0f ACM SIGGRAPH 98*, Annual Conference Series, pages 33–42. ACM SIGGRAPH, July 1998.

[9] M. Gleicher. Motion path editing. In *Proceedings 2001 ACM Symposium on Interactive 3D Graphics*. ACM, Mar. 2001.

[10] J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien. Animating human athletics. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, pages 71–78. ACM SIGGRAPH, Aug. 1995.

[11] H. Ko and N. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Application*, 16(2):50–59, 1996.

[12] L. Kovar, J. Schreiner, and M. Gleicher. Footskate cleanup for motion capture editing. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation 2002*. ACM SIGGRAPH, July 2002.

[13] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of ACM SIGGRAPH 99*, Annual Conference Series, pages 39–48. ACM SIGGRAPH, Aug. 1999.

[14] J. Lee and S. Y. Shin. General construction of time-domain filters for orientation data. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):119–128, 2002.

[15] C. K. Liu and Z. Popović. Synthesis of complex dynamic character motion from simple animations. In *Proceedings of ACM SIGGRAPH 2002*, Annual Conference Series. ACM SIGGRAPH, July 2002.

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

[17] N. Pollard. Simple machines for scaling human motion. In *Computer Animation and Simulation '99, Eurographics Animation Workshop*, pages 3–11, 1999.

[18] N. S. Pollard and P. S. A. Reitsma. Animation of human-like characters: Dynamic motion filtering with a physically plausible contact model. *Yale Workshop on Adaptive and Learning Systems*, 2001.

[19] Z. Popović and A. Witkin. Physically based motion transformation. In *Proceedings of ACM SIGGRAPH 99*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, Aug. 1999.

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, New York, 1992.

[21] C. Rose, B. Guenter, B. Bodenheimer, and M. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of ACM SIGGRAPH 1996*, Annual Conference Series, pages 147–154. ACM SIGGRAPH, Aug. 1996.

[22] S. Tak, O. Song, and H. Ko. Spacetime sweeping: An interactive dynamic constraints solver. In *Computer Animation 2002*, pages 261–270, June 2002.

[23] M. Vukobratovic and D. Juricic. Contribution to the synthesis of biped gait. *IEEE Trans. Bio-Med. Eng.*, BME-16(1):1–6, 1969.

[24] D. Winter. *Biomechanics and Motor Control of Human Movement*. Wiley, New York, 1990.

[25] A. Witkin and Z. Popović. Motion warping. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, pages 105–108. ACM SIGGRAPH, Aug. 1995.

[26] W. Wooten and J. Hodgins. Simulation of leaping, tumbling, landing, and balancing humans. In *IEEE International Conference on Robotics and Animation*, 2000.

[27] K. Yamane and Y. Nakamura. Dynamics filter – concept and implementation of on-line motion generator for human figures. In *the 2000 IEEE International Conference on Robotics & Automation*, pages 688–695, April 2000.

[28] V. Zordan and J. Hodgins. Motion capture-driven simulations that hit and react. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation 2002*. ACM SIGGRAPH, July 2002.

Appendix

The coefficients in Equation 6 are as follows:

$$
\begin{aligned}
\sigma a_1 = &\sum_{i\in\mathcal{S}_2}(\alpha_i k_{2_i} - \beta_i k_{1_i}) + \sum_{i\in\mathcal{S}_3}(\alpha_i k_{4_i} - \beta_i k_{3_i}) \\
\sigma a_2 = &-\sum_{i\in\mathcal{S}_2}(\alpha_i k_{1_i} + \beta_i k_{2_i}) + \sum_{i\in\mathcal{S}_3}(\alpha_i k_{3_i} + \beta_i k_{4_i}) \\
\sigma a_3 = &\sum_{i\in\mathcal{S}_1}(\alpha_i x_i - \beta_i y_i) + \sum_{i\in\mathcal{S}_2}(\alpha_i c_x^2 - \beta_i c_y^2) \\
&+ \sum_{i\in\mathcal{S}_3}(\alpha_i(x_i - k_{4_i}) - \beta_i(y_i - k_{3_i})) \\
\sigma a_4 = &-\sum_{i\in\mathcal{S}_2}(\gamma_i(y_i - c_y^2)) + \sum_{i\in\mathcal{S}_3}(\gamma_i(c_{i_y}^3 - c_y^2)) \\
\sigma a_5 = &-\sum_{i\in\mathcal{S}_2}(\gamma_i(x_i - c_x^2)) + \sum_{i\in\mathcal{S}_3}(\gamma_i(c_{i_x}^3 - c_x^2)) \\
\sigma a_6 = &\sum_i \alpha_i z_i - \sum_{i\in\mathcal{S}_1}(\gamma_i y_i) - \sum_{i\in\mathcal{S}_2}(\gamma_i c_y^2) \\
&- \sum_{i\in\mathcal{S}_3}(\beta_i(y_i - k_{3_1})),
\end{aligned}
$$

where

$$
\begin{aligned}
k_{1_i} &= y_i - c_y^2, \\
k_{2_i} &= x_i - c_x^2, \\
k_{3_i} &= c_{i_y}^3 - c_y^2, \\
k_{4_i} &= c_{i_x}^3 - c_x^2, \\
\sigma &= \sum_i \alpha_i, \\
\alpha_i &= m_i(\ddot{y}_i - g_y), \\
\beta_i &= m_i(\ddot{x}_i - g_x), \text{ and} \\
\gamma_i &= m_i(\ddot{z}_i - g_z).
\end{aligned}
$$

Here $\mathbf{c}_i^3$ is the rotation center of i^{th} link in $\mathcal{S}_3$ and $\mathbf{c}^2$ is that of $\mathcal{S}_2$.

The differential change of Z_x and Z_z is

$$
\begin{aligned}
\sigma\Delta Z_x &= \sum_i(\alpha_i \Delta x_i) - \sum_i(\beta_i \Delta y_i) \\
\sigma\Delta Z_z &= -\sum_i(\gamma_i \Delta y_i).
\end{aligned}
$$

Note that when the overall acceleration of the character is small, α is significantly larger than γ due to its g_y term. Also, for motions where the character is mostly upright (which is typically the case), Δx_i is significantly greater than Δy_i since the center of the rotation for the adjustment is aligned roughly vertically with the center of mass. Hence in typical situations the dominant change in the ZMP is due to ΔZ_x, and ΔZ_z may be neglected. In practice, we have found that even for uncommon motions such as flipping, it is safe to ignore ΔZ_z.