# Animation of Humanlike Characters: Dynamic Motion Filtering with a Physically Plausible Contact Model

Nancy S. Pollard and Paul S. A. Reitsma

Brown University*

## Abstract

Data-driven techniques for animation, where motion captured from a human performer is "played back" through an animated human character, are becoming increasingly popular in computer graphics. These techniques are appealing because the resulting motion often retains the natural feel of the original performance. Data-driven techniques have the disadvantage, however, that it is difficult to preserve this natural feel when the motion is edited, especially when it is edited by a software program adapting the motion for new uses (e.g. to animate a character traveling over uneven terrain). In fact, many current techniques for motion editing do not preserve basic laws of physics. We present a dynamic filtering technique for improving the physical realism of motion for animation. As a specific example, we show that this filter can improve the physical realism of automatically generated motion transitions. Our test case is the physically challenging transition contained in a double kick constructed by merging two single kicks.

## 1 Introduction

Motion editing is a "hot topic" in computer animation, especially animation of human characters. We have access to a growing base of realistic captured human motion data, and many researchers believe that this data will allow us to finally create believable digital humans. In a data-driven approach to animation, a continuous stream of motion is created by clipping, splicing, blending, and scaling existing motion sequences to allow the character to accomplish a specific set of goals. For example, a character's motion through an obstacle course may be created by assembling jumping, running, and climbing segments taken from a motion capture database.

Motion captured from human actors is physically realistic, at least within the accuracy of the measurements, but the process of blending, patching, and adjusting this motion introduces artifacts. Motion that has been edited is typically processed to reduce these artifacts. The tools that produce the best results are iterative: an artist will make repeated manual adjustments to achieve a desired effect, or an offline process will optimize the motion based on an objective such as minimal energy.

In certain domains, such as automatic animation of characters in virtual environments, an iterative approach is not appropriate. In a virtual environment, processing time is limited, and the character's motion may change at any time, due to user actions or events in the environment. The best we can hope for is to put the motion through one or more filters.

[1]email: [nsp|psar]@cs.brown.edu

The most common motion filters are kinematic. Kinematic filters adjust the pose of a character on a frame by frame basis to maintain constraints such as keeping the stance foot planted firmly on the ground or ensuring that the character's center of mass projects to a point within its base of support. More recently, dynamic filters and tracking controllers have begun to appear in the animation literature [14] [15]. Dynamic filters and tracking controllers can be used to maintain physical constraints, such as keeping torques within their limits and keeping required contact forces within a reasonable friction cone.

This paper describes a dynamic filter for processing motion for improved physical realism. Our contribution over existing techniques is to combine feed-forward tracking with a friction-based model of contact between the character and the environment. Such a model allows sliding when appropriate and rejects motions that would not be plausible for a physical system such as a robot to carry out. We show examples of success and failure of our filter with a physically challenging motion where a single kick has been altered to create a double kick (Figure 3).

## 2 Background

Iterative kinematic techniques for motion editing allow the user to adjust parameters of the motion, such as the pose at a specific frame, and propagate the effects of the adjustment through the entire motion. Gleicher [6] solves for the motion displacement that minimizes the difference from the source motion. Lee and Shin [8] use a hierarchical B-spline representation of motion to limit changes to user-specified frequency bands. Ko and Badler [7] use inverse kinematics to modify motion while considering balance. van de Panne [12] and the Character Studio software product [4] allow a user to control footprint locations. Others [11], [1], [10] have developed signal processing techniques for blending and altering example motion. These techniques largely rely on the artist's eye to ensure that the results appear physically plausible.

Constrained optimization of dynamic simulations was introduced to the graphics community by Witkin and Kass [13], who optimized motion for a jumping lamp. In the biomechanics community, Pandy has optimized lower body motion for maximal height jumping and for minimal energy walking. His models involved 54 muscles, 864 degrees of freedom, and use of parallel computers to keep computation

time to manageable levels. Popović and Witkin [9] show that optimization times on the order of minutes can be obtained for human motion when the optimization is performed on a simplified version of the character. These techniques result in motion that obeys physical laws (as well as possible), but they require search through a high dimensional space and are not suited for use in a virtual environment where a character's motion and goals may be changing.

A tracking controller is used by Zordan [15], who enhances simple PD tracking with a balance controller and collision simulations for convincing behavior in actions such as drumming and punching. Dynamic tracking was used by Yamane and Nakamura [14], who place virtual links at locations of ground contact, and find a least squares solution for acceleration of the constrained system to best match the motion they are tracking. The contribution of our work over theirs is to present an alternative approach that models character-environment contact as contact with friction and allows complex interactions such as shifting the weight and pivoting to be easily handled.

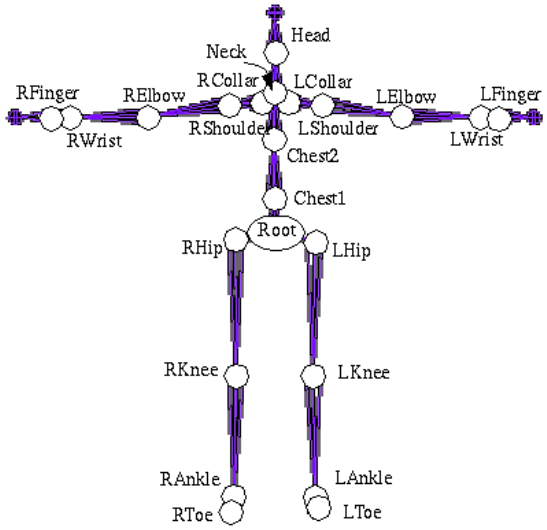


Figure 1: Our character has 22 ball joints (shown as white circles) that are assumed to be actuated. The root (large white oval) has six unactuated degrees of freedom.

## 3 Problem Setup

The examples in this paper draw on a motion editing system we have developed. This section describes how motion was captured, edited, and fit to a physical model. It also describes our model of contact between the character and the environment.

### 3.1 Motion Capture Data

The motion in our examples was captured using an optical setup. Some 40-50 reflecting markers were placed on the actor, and the motion of these markers was tracked in 3D space by 8 cameras placed around the edges of the room. The motion was captured at 60 frames per second and processed to fit a skeletal representation of the actor composed of rigid links and 22 ball joints (Figure 1). The total number of degrees of freedom of the system was 72 (6 for translation and rotation of the root, and 3 for each ball joint). Rotations were expressed using Euler angles. The motion was filtered to remove noise and outliers from the measured data.

### 3.2 Editing the Motion Data

The motion capture data was edited to assemble longer sequences from small clips of motion capture data. Transitions from one motion clip to the next were created automatically. The transition software takes as input two separate motions (e.g. the single kicks in the first two rows of Figure 3). It aligns these motions along the time axis, selects a transition boundary, and creates a smooth seam around the transition boundary by fading out the first motion and fading in the second over a fixed window of time.

The segue from the first motion to the second is done using an ease-in/ease-out blend of Euler angle parameters, treating each axis of each joint as a separate signal. These signals are separated into frequency bands in the manner described by Bruderlin and Williams [1], and each band is blended separately. Higher frequency bands are blended over shorter time intervals in a manner similar to Burt and Adelson [2]. The final signal is reconstructed from the blended frequency bands.

Our examples show two "takes" of a single roundhouse kick edited to form a double roundhouse kick (Figure 3). To create double kicks, the blending system first locates each kick in the source motions by looking for acceleration spikes in the data. It finds the best match between poses after the kick in the first motion and before the kick in the second motion, thus ensuring that both kicks are present in the resulting motion. To introduce some variability, the time associated with the transition boundary in each of the two motions is randomly altered by a small number of frames in either direction.

Because the character does not go through a home position between kicks, a straightforward transition from the first motion to the second can exhibit anomalies such as odd body rotation and foot sliding. So that foot sliding can be reduced (see Section 4.3), we store desired position data for the stance foot as part of the motion editing process. This position data is extracted from a simple splice transition where the toes in the first and second motions are aligned at the transition boundary.

### 3.3 The Physical Model

A physical model of the character was constructed from the motion capture skeleton. This physical model consists of mass and an inertia tensor for each body part and is derived

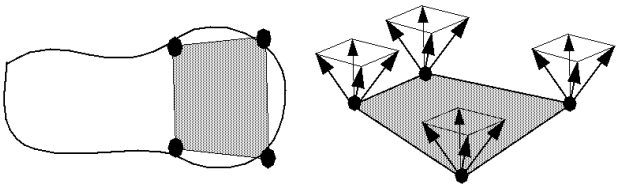

Figure 2: Example of contact force basis vectors. (Left) Assume the entire toe is in contact with the ground. The contact area is sampled with points on the boundary of the toe geometry, and (Right) force basis vectors are generated to span the friction pyramid at each contact point. This example has sixteen force basis vectors, four for each of the four sample contact points. If coefficients for all sixteen basis vectors are positive, the force applied by the foot will fall within given friction limits.

from the total mass of the actor, a fit of a geometric model to the skeleton, and body part density information measured from cadavers[3].

### 3.4 Contact Forces

A Coulomb model of friction is assumed for contact between the character and its environment. At each frame of the motion, the system checks for collisions between the character and its environment and selects a discrete set of contact points to represent each contact region. A set of basis forces is constructed at each contact point to approximate the friction cone at that point with a friction pyramid (Figure 2). External forces are assumed to only be applied along these basis directions.

## 4 Filtering the Motion

Given motion data that has been captured and edited as described above, the job of the dynamic filter is to keep the feet planted on the ground and filter out impossible forces and torques. If the foot should pivot or slide, we allow it to do so. If the motion cannot be achieved using legal forces and feedforward control, the filter fails to correct the motion (i.e. the character falls).

To filter the motion, we start out in an initial state extracted from the data and simulate forward in time. At each step, our goal is to find accelerations and applied forces that (1) move the character toward its state in the next frame of the measured motion, (2) require zero root force and torque, as these joints are not actuated, and (3) use forces only within the friction cones, if desired.

Section 4.1 outlines the dynamics equations we use. Section 4.2 describes a filter that tracks joint angles as well as possible, computing contact forces along the way. Section 4.3 describes how position tracking is added, to keep the character's stance foot fixed to the ground, for example.

### 4.1 Dynamics Equations

The equations of motion with no external forces except gravity are

$$Q = H\ddot{q} + C \tag{1}$$

where $H$ is the mass matrix, $C$ includes velocity effects and gravity terms, $\ddot{q}$ are accelerations of system state parameters, and $Q$ are generalized forces.

We add the effect of contact forces as follows:

$$Q = H\ddot{q} + C + Mf \tag{2}$$

where $f$ is a vector of force coefficients and $M$ maps force coefficients to generalized forces based on the current contact geometry. The size of $f$ and the interpretation of each of the coefficients vary depending on how the character contacts the environment. Figure 2 shows how basis forces were computed for the examples in this paper.

Vector $C$ and matrices $H$ and $M$ are computed by adapting the technique described in [5]: given the character's current state ($q$, $\dot{q}$), $C$ is set to the value of $Q$ computed when $f$ and $\ddot{q}$ are zero; each column $i$ of $H$ is set to the value $(Q-C)$ when $f$ is zero and $\ddot{q}$ is $\delta_i$ (value 1 for element $i$ and 0 for all other elements); each column $i$ of $M$ is set to the value $(Q - C)$ when $\ddot{q}$ is zero and $f$ is $\delta_i$.

### 4.2 Pose Tracking

Motion is tracked by numerically integrating generalized accelerations over time. Generalized accelerations are computed with two goals in mind: closely match the original motion and eliminate undesirable (physically impossible) root forces and torques.

The tracking process has four steps. The first step is to calculate desired accelerations $\ddot{q}_{DES}(t)$ that will result in close tracking of the given motion and recovery from tracking errors. This $\ddot{q}_{DES}(t)$ will be unrealistic if it requires forces and torques to be applied at the root, which is unactuated. Root generalized forces $Q_{root}$ are computed from $\ddot{q}_{DES}(t)$. Forces at the contact points are then computed to reduce or eliminate $Q_{root}$. Accelerations $\ddot{q}_{DES}(t)$ are then adjusted to eliminate any root forces and torques that remain. These four tasks are described below.

#### 4.2.1 Computing Desired Accelerations

Desired acceleration $\ddot{q}_{DES}(t)$ includes root translational acceleration, root rotational acceleration, and joint rotational accelerations:

$$\ddot{q}_{DES}(t) = \begin{bmatrix} \ddot{\mathbf{x}}_{root,DES}(t) \\ \dot{\omega}_{root,DES}(t) \\ \dot{\omega}_{0,DES}(t) \\ \ldots \\ \dot{\omega}_{n,DES}(t) \end{bmatrix} \tag{3}$$

where $n$ is the number of actuated joints of the character. Desired acceleration is computed from the data and position

and velocity errors. For root translational acceleration $\ddot{\mathbf{x}}_{root}$:

$$\begin{aligned}\ddot{\mathbf{x}}_{root,DES}(t) &= \ddot{\mathbf{x}}_{root,DATA}(t) + \\ &\quad k_x(\mathbf{x}_{root,DATA}(t) - \mathbf{x}_{root}(t)) + \\ &\quad b_x(\dot{\mathbf{x}}_{root,DATA}(t) - \dot{\mathbf{x}}_{root}(t)) \end{aligned} \quad (4)$$

where $k_x$ and $b_x$ are stiffness and damping coefficients and $\mathbf{x}_{root,DATA}$, $\dot{\mathbf{x}}_{root,DATA}$, and $\ddot{\mathbf{x}}_{root,DATA}$ are the translational position, velocity, and acceleration of the root in the data we are tracking.

For root and joint angular acceleration $\dot{\omega}$:

$$\begin{aligned}\dot{\omega}_{i,DES}(t) &= \dot{\omega}_{i,DATA}(t) + \\ &\quad k_\omega(\hat{\mathbf{v}}\Delta\theta)_{i,CORR} + \\ &\quad b_\omega(\omega_{i,DATA}(t) - \omega_i(t)) \end{aligned} \quad (5)$$

where $k_\omega$ and $b_\omega$ are stiffness and damping coefficients and $\omega_{i,DATA}(t)$ and $\dot{\omega}_{i,DATA}(t)$ are the angular velocity and acceleration for the root or joint $i$ found in the data we are tracking. Parameter $(\hat{\mathbf{v}}\Delta\theta)_{i,CORR}$ is the rotation required to correct error at the root or at joint $i$, equivalent to $q_{i,DATA}(t)q_i^{-1}(t)$ in angle-weighted axis format.

The data we are tracking contains only position information. Velocities and accelerations are obtained using simple finite differences so that Euler integration with no constraints on accelerations would result in the original dataset.

#### 4.2.2 Computing Generalized Forces

Given $\ddot{q}_{DES}(t)$, root generalized forces are computed when $f = 0$:

$$Q_{root} = \begin{bmatrix} \mathbf{f}_{root}(t) \\ \tau_{root}(t) \end{bmatrix} = H_{root}\ddot{q}_{DES}(t) + C_{root} \quad (6)$$

where $Q_{root}$, $H_{root}$, and $C_{root}$ are the first six rows of $Q$, $H$, and $C$, and $\mathbf{f}_{root}(t)$ and $\tau_{root}(t)$ are the required force and torque at the root of the character.

#### 4.2.3 Computing Contact Forces to Support the Root

$Q_{root}$ in equation 6 represents undesirable forces and torques, because the root joint is not actuated. To track the data in a physically plausible way requires finding contact forces to eliminate $Q_{root}$ if possible. In other words, we seek contact forces to support the desired motion of the character. The following equation is solved for $f$:

$$M_{root}f = -Q_{root} \quad (7)$$

where $M_{root}$ is the first six rows of M.

If friction cone constraints are not important, equation 7 can be solved using least squares. If contact forces must fall within friction cones at the contacts, least squares cannot be used, because it may result in negative coefficients, which would violate friction cone constraints.

We have had good success with a simple, greedy iterative technique for computing $f$. At each step of the iteration, the single force coefficient that moves most directly toward a solution is modified. Coefficients can be either increased or decreased in an iteration step, but they are constrained to remain greater than or equal to zero. Iteration continues to within some small distance $\epsilon$ of the solution or until an iteration step increases the distance to the goal.

#### 4.2.4 Adjusting Acceleration to Eliminate Root Forces

Once f has been computed, $\ddot{q}_{DES}(t)$ is altered to eliminate any root forces and torques that remain. The following equation is solved for $\Delta\ddot{q}$ using least squares:

$$H_{root}\Delta\ddot{q} = -Q_{root} - M_{root}f \quad (8)$$

The new acceleration is:

$$\ddot{q}'_{DES}(t) = \ddot{q}_{DES}(t) + \Delta\ddot{q} \quad (9)$$

Value $\ddot{q}'_{DES}(t)$ is used to integrate forward one frame of the animation.

### 4.3 Reference Point Tracking

Unconstrained tracking results in motion that is close to the reference motion and maintains physical plausibility as defined here. Frequently some aspects of the motion are more important than others, however, such as having the stance foot behave as though it is firmly connected to the ground. We track one or more points attached to the character with a stiffer system than that used to track joint angles.

Assume $c$ reference point position signals, extracted from the motion data or some other source and collected into a single vector $r_{DATA}(t)$:

$$r_{DATA}(t) = \begin{bmatrix} \mathbf{r}_0(t) \\ \mathbf{r}_1(t) \\ \ldots \\ \mathbf{r}_c(t) \end{bmatrix} \quad (10)$$

Derivatives $\dot{r}_{DATA}(t)$ and $\ddot{r}_{DATA}(t)$ are computed using finite differences.

The desired acceleration of the reference points $\ddot{r}_{DES}(t)$ is

$$\begin{aligned}\ddot{r}_{DES}(t) &= \ddot{r}_{DATA}(t) + \\ &\quad k_r(r_{DATA}(t) - r(t)) + \\ &\quad b_r(\dot{r}_{DATA}(t) - \dot{r}(t)) \end{aligned} \quad (11)$$

where $k_r$ and $b_r$ are stiffness and damping used for reference point tracking.

The expected acceleration of the reference points $\ddot{r}_{EXP}(t)$ is defined based on the Jacobian $J(q)$ relating state velocities to reference point velocities:

$$\dot{r}(t) = J(q)\dot{q}(t) \quad (12)$$

$$\ddot{r}_{EXP}(t) = \dot{J}(q)\dot{q}(t) + J\ddot{q}'_{DES}(t) \quad (13)$$

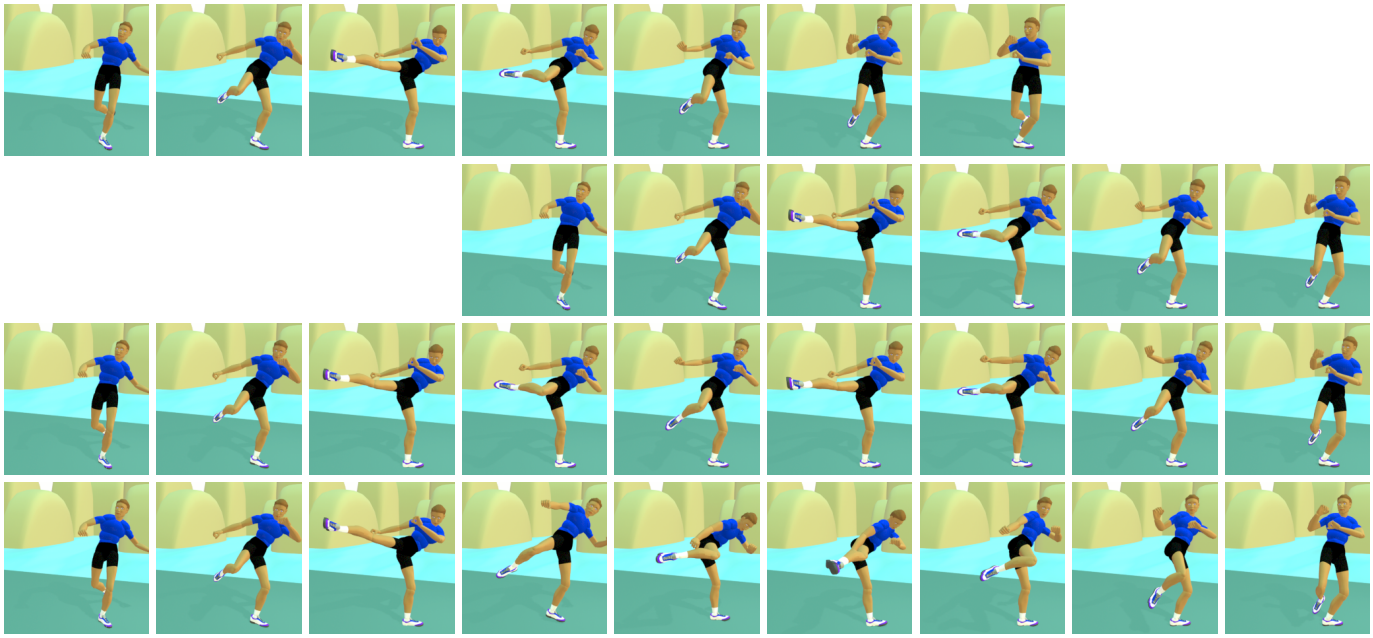

Figure 3: Filmstrips of a double kick created from a single kick. Frames are spaced at 0.167 second intervals, or 6 frames per second. (Rows 1 and 2) A motion containing a single kick is duplicated and aligned in time so that a transition from the first motion to the second would produce a double kick. (Row 3) The double kick from the first example, after filtering, looks very realistic. (Row 4) The double kick from the second example, after filtering, contains a hop (frame 4). The character continues to rotate while he is in the air, resulting in a second kick aimed in a different direction than the first (frame 6). By the end of the sequence, however (frame 9), the character has recovered and is tracking the given motion closely.

Desired state acceleration $\ddot{q}'_{DES}(t)$ is adjusted to reduce the error between the desired and expected reference point accelerations:

$$\Delta\ddot{q}' = J^{+}(q)\Delta\ddot{r} = J^{+}(q)(\ddot{r}_{DES} - \ddot{r}_{EXP}) \quad (14)$$

$$\ddot{q}''_{DES}(t) = \ddot{q}'_{DES}(t) + \Delta\ddot{q}' \quad (15)$$

where $J^{+}$ is the pseudoinverse of $J$. Value $\ddot{q}''_{DES}(t)$ is used to integrate forward one frame of the animation.

## 5 Results

We show results from two experiments at different levels of difficulty for the dynamic filter. Animations from these examples can be found at the following web site: http://www.cs.brown.edu/people/nsp/tracker.html.

The plots in Figures 4 and 5 show the angle from the vertical of applied forces (top) and the reference point tracking errors (bottom) under three scenarios:

- **Joint Filtering:** The character tracks the original motion. (Accelerations $\ddot{q}_{DES}(t)$ are computed from Equations 3 and 9. Forces at the contact points are not constrained.)
- **Position Filtering:** Joint Filtering plus tracking of the stance toe position, based on desired toe position data. (Accelerations $\ddot{q}_{DES}(t)$ are computed from Equations 3, 9, and 15. Forces at the contact points are not constrained.) Because additional forces must be applied to keep the foot in place, this step is often associated with applied forces that fall much further outside the friction cone than with Joint Filtering only (e.g. Figure 5, top). In this sense, Position Filtering makes the motion worse.
- **Friction Filtering:** Position Filtering plus the requirement that forces meet friction cone constraints. The horizontal position of the center of mass of the character was also tracked when this level of filtering was turned on. For center of mass tracking, we found that setting $r_{DATA}(t)$ and its derivatives to a mix of the center of mass and the center of support extracted from the original data gave us better results than tracking center of mass alone.

Stiffness and damping values for the examples are shown in Figure 6. Accelerations were integrated with timesteps of 0.0167 seconds to keep performance of the system interactive. Observed tracking errors could be decreased by reducing this timestep.

To test the performance of the filter, we assembled two single kicks to create a double kick where the character does not put his foot down between the two kicks (Figure 3). Many transitions were generated with random variation in transition parameters, and two were selected for display. For the first, Figure 4 and Figure 3, row 3, the required forces and initial foot tracking error were significant, but the filter was

able to fix these errors to create visually plausible motion.

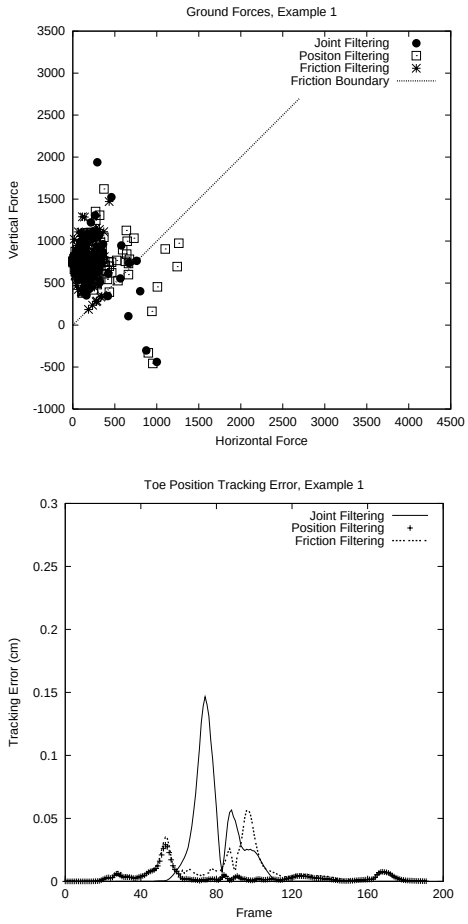


Figure 4: Modified data, double kick formed from two single kicks. (Top) Effects of filtering on applied forces. Horizontal force magnitude is plotted vs. vertical force for each frame of the motion. Forces below and to the right of the solid line require an unrealistic friction coefficient (greater than 1.0). Forces with Joint Filtering (filled circles) and Position Filtering (boxes) fall outside the friction cone boundary. When Friction Filtering is turned on (stars), the dynamic filter keeps ground contact forces within the friction cone at the contacts. (Bottom) Tracking errors for this motion are small for both Position and Friction Filtering.

The second example, Figure 5 and Figure 3, row 4, is more aggressive because the amount of time between kicks is less than in the first example. Here, the dynamic filter fails, in the sense that the character appears off balance in the corrected motion. The toe position error begins at nearly 30cm, twice that of the first example. Reducing this error generates large forces well outside the friction cone. Turning on friction cone constraints causes the character to hop and spin about 20 degrees around the vertical axis (Figure 3, row 4).

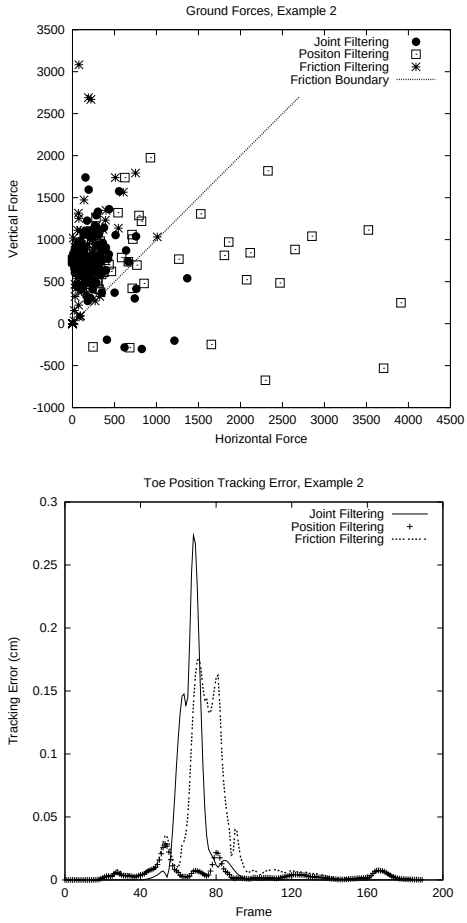


Figure 5: More difficult data, double kick formed from two single kicks. See Figure 4 for plot descriptions. Here, the foot sliding in the given data is very pronounced, and tracking the desired toe position requires inappropriately large forces to be applied to the foot (visible as the boxes in the top plot). The dynamic filter can correct these forces, bringing them inside the friction cone, at the cost of having the character take a hop in the middle of the motion. That hop is responsible for the resulting errors in foot tracking in the lower plot. (Also see Figure 3.)

## 6 Discussion

The initial motion in both examples exhibits unpleasant anomalies. The most noticeable anomalies are foot sliding and unrealistic motion of the upper body. When the kinematic fix of tracking stance toe position is applied, foot sliding is reduced, but the upper body motion is more disturbing. The unrealistic upper body motion is associated with unrealistic contact forces as shown in the force plots, and it is especially apparent in the second example, which requires large forces far outside the friction cone. The dynamic filter is able to constrain these forces to fall within the friction cone, at very little visual cost in the first example, but at the cost of a hop and extra pivot in the second example. It is the hop (not foot sliding) that is responsible for the increase in tracking error from Position Filter-

| | | | |
|---|---|---|---|
| $k_x$ | 4 | $b_x$ | 2 |
| $k_{\omega,root}$ | 1280 | $b_{\omega,root}$ | 40 |
| $k_\omega$ | 40 | $b_\omega$ | 7 |
| $k_{r,foot}$ | 1440 | $b_{r,foot}$ | 125 |
| $k_{r,com}$ | 8 | $b_{r,com}$ | 4 |

Figure 6: Stiffness and damping values used in the experiments. Units of stiffness are $\frac{1}{s^2}$. Units of damping are $\frac{1}{s}$.

ing to Friction Filtering in the second example. Arguably even the second example looks more realistic after filtering than before, but it is probably not the solution desired by the animator. (See the animations on the following web page to compare the dynamic appearance of these motions: http://www.cs.brown.edu/people/nsp/tracker.html.)

When friction constraints are turned on, the dynamic filter essentially clips extreme horizontal ground contact forces. Often these horizontal forces offset torques generated by the vertical forces supporting the character. For the dataset we used, these torques were substantial enough that turning on friction constraints sometimes caused the character to lose his balance. To correct this problem, we added center of mass tracking (so that both toe and center of mass positions were tracked), and a single parameter to nudge the path tracked by the center of mass toward the center of support. The addition of this parameter meant that the tracker was not fully automatic, but it dramatically increased the set of motions we were able to handle. An offline filtering process could do a quick search for this parameter. It is also possible that it could be set automatically in an on-line process with a small window of lookahead, where expected force characteristics of the motion over the next second were known, for example. We did not investigate this possibility.

Overall, we were pleased by the performance of the dynamic filter on this physically challenging motion and believe that dynamic filtering could help make captured motion data more appropriate as a target for robot motion. Further research is required to test the performance of the filter on a broader base of examples.

## Acknowledgments

Thanks to Jessica Hodgins and Victor Zordan for detailed comments on the paper. We also thank Acclaim studios for helping us to collect the motion capture data.

## References

[1] A. Bruderlin and L. Williams. Motion signal processing. In *SIGGRAPH 95 Proceedings*, Annual Conference Series, pages 97–104. ACM SIGGRAPH, Addison Wesley, August 1995.

[2] P. J. Burt and E. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, October 1983.

[3] W. T. Dempster and G. R. L. Gaughran. Properties of body segments based on size and weight. *American Journal of Anatomy*, 120:33–54, 1965.

[4] Discreet. *Character Studio*. http://www.discreet.com.

[5] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, Boston, MA, 1987.

[6] M. Gleicher. Retargetting motion to new characters. In *SIGGRAPH 98 Proceedings*, Annual Conference Series. ACM SIGGRAPH, ACM Press, August 1998.

[7] H. Ko and N. I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, pages 50–59, March 1996.

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

[9] Z. Popović and A. Witkin. Physically-based motion transformation. In *SIGGRAPH 99 Proceedings*, Annual Conference Series. ACM SIGGRAPH, ACM Press, August 1999.

[10] C. F. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, September/October:32–40, 1998.

[11] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. In *SIGGRAPH 95 Proceedings*, Annual Conference Series, pages 91–96. ACM SIGGRAPH, Addison Wesley, August 1995.

[12] M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, October 1997.

[13] A. Witkin and M. Kass. Spacetime constraints. In J. Dill, editor, *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 159–168, August 1988.

[14] K. Yamane and Y. Nakamura. Dynamics filter – concept and implementation of on-line motion generator for human figures. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 2000.

[15] V. B. Zordan and J. K. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. In *Eurographics Workshop on Animation and Simulation '99*, Milan, Italy, September 1999.