Footskate Cleanup for Motion Capture Editing

Lucas Kovar * John Schreiner * Michael Gleicher *

University of Wisconsin, Madison

Abstract

While motion capture is commonplace in character animation, often the raw motion data itself is not used. Rather, it is first fit onto a skeleton and then edited to satisfy the particular demands of the animation. This process can introduce artifacts into the motion. One particularly distracting artifact is when the character's feet move when they ought to remain planted, a condition known as footskate. In this paper we present a simple, efficient algorithm for removing footskate. Our algorithm exactly satisfies footplant constraints without introducing disagreeable artifacts.

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

Keywords: motion capture, motion editing, inverse kinematics

1 Introduction

Real human motion usually contains *footplants*, which are periods of time when a foot or part thereof remains in a fixed position on the ground. Processed motion capture data may fail to accurately reproduce these footplants, introducing an artifact known as *footskate*. Since footplants are often the primary connection between a character and the surrounding environment, even small amounts of footskate can destroy the realism of a motion.

Footskate may be introduced to a motion in several ways. Sometimes the raw motion data itself is imperfect; for example, a sensor may be miscalibrated. In such cases the motion can often still be salvaged, but the footplants may be lost in the process. Footskate can also be added even when the raw data is faithful to the true motion. In standard animation pipelines motion data is usually mapped onto an articulated figure called a *skeleton*. Since a real human is not rigid, this mapping process can fail to fully preserve footplants. Also, skeletal motion data is often edited in order to adapt to the particular needs of an animation. Representative editing operations include warping [Witkin and Popović 1995], retargeting [Gleicher 1998], path editing [Gleicher 2001], transition generation [Rose et al. 1996], and various signal processing algorithms [Bruderlin and Williams 1995]. These editing operations fundamentally involve adding only low-frequency changes to a motion, so high-frequency details like crisp footplants are either lost entirely or only approximately preserved.

In light of this, there is a general need for an algorithm for enforcing footplant constraints in skeletal motion capture data. In this paper we introduce such an algorithm. Our method is simple and efficient in that it requires no nonlinear optimizations and performs a fixed amount of calculation on every frame. It is robust in that it is stable for near-singular joint configurations and degrades smoothly as the distance from a footplant constraint grows. Finally, the adjustment applied to a given frame depends only on a fixed neighborhood of surrounding frames. This makes our algorithm useful for online applications that can accept a small time delay ($\frac{1}{2}$–1 second), such as animating background characters in a game or individuals in a crowd simulation.

Most previous work has assumed rigid skeletons and balanced a precarious tradeoff between satisfying constraints precisely and avoiding sharp adjustments to joint rotations. We will demonstrate that a perfectly rigid skeleton can make it quite difficult to reliably meet both of these goals. For this reason we allow small changes in the bone lengths of the skeleton. While the specific reasons for choosing this approach will be covered in detail later, this decision is unusual enough that we would like to explain up front why we believe it to be reasonable. First, since footplant cleanup is intended as a postprocess, editing operations which require fixed bone lengths are still usable. Second, support for variable bone lengths exists in a variety of standard data formats (such as BVH, BVA, Acclaim, and HTR [Lander 1998]) and animation packages (such as Poser, Maya, and 3D Studio Max). Finally, variable bone lengths can be seamlessly incorporated into traditional algorithms for specifying how a skeleton drives a character's mesh [Lewis et al. 2000].

The remainder of this paper is divided into four sections. In Section 2, we define the footskate cleanup problem more precisely. In Section 3, we review related work. We then describe our algorithm in Section 4, present results in Section 5, and conclude with a brief discussion in Section 6.

2 The Footskate Cleanup Problem

A skeletal motion is a function

$$M(t) = (\mathbf{p_R}(t), \mathbf{q_0}(t), \ldots, \mathbf{q_k}(t), \mathbf{o_0}(t), \ldots, \mathbf{o_k}(t)) \quad (1)$$

where $\mathbf{p_R}$ is a 3-vector indicating the position of the root joint, $\mathbf{q_i}$ is the quaternion specifying the orientation of the ith joint in its parent's coordinate system, and $\mathbf{o_i}$ is a 3-vector defining the offset of the ith joint in its parent's coordinate system. We assume our data is a regular sampling of M into frames $\mathcal{F}_i = M(i)$. Only the lower body of the skeleton is relevant to our work; it is represented with a standard model that is shown in Figure 1.

The footskate cleanup problem may now be stated as follows: given a set of skeletal motion data annotated with footplant constraints, enforce these constraints while disturbing the original motion as

* {kovar,jmschrei,gleicher}@cs.wisc.edu, http://www.cs.wisc.edu/graphics

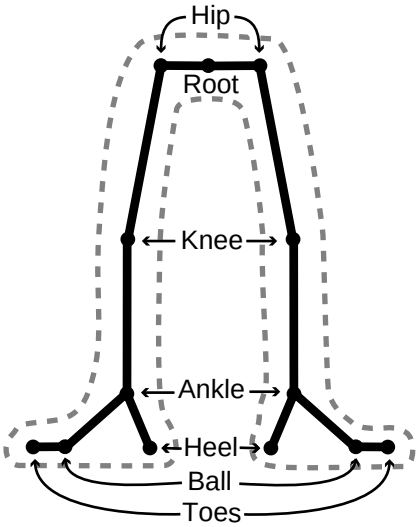

Figure 1: Our model of the lower body. Our method allows the heel and/or ball of each foot to be planted over a set of frames. The toes are dealt with independently to prevent floor penetration.

little as possible. A footplant constraint consists of requiring either the heel or the ball to be planted over a set of frames. It is possible for both the heel and ball of a foot to be planted simultaneously. A "disturbance" is any visually-distracting artifact not already present in the motion. In particular, while smooth changes generally preserve the integrity of a motion, sharp changes tend to be quite noticeable even if they are numerically small. It is for this reason that we allow the legs to stretch: as we will show in Section 4, sometimes a smooth change to limb length is the best way to avoid a sharp change in joint position or orientation.

Since we don't consider constraints on the very tips of the toes, motions like certain ballet movements are beyond the reach of our framework. However, such motions are quite rare. Consider, for example, that a typical "tiptoeing" motion really involves the weight being supported on the toes *and* the end of the ball. In such cases we need only plant the ball to produce the correct result. While we don't allow constraints on the toes, we do ensure that they don't penetrate the ground as a result of other constraints being satisfied.

Footplant constraints may be identified automatically using techniques like those proposed by Bindiganavale [Bindiganavale and Badler 1998; Bindiganavale 2000]. However, such methods can be unreliable if the original data is noisy or the motion does not have clear footplants, such as a runner skidding to a halt or a dancer shuffling. Since our algorithm has no way of knowing if a particular footplant constraint *really* exists, artifacts may be introduced if the constraints are incorrectly specified — a foot may slide into a plant or appear stuck to the ground too long. In light of this, we use an automated algorithm to identify footplants and then edit the results by hand. In the context of the entire animation pipeline, we have not found this user intervention unduly taxing.

Our footskate cleanup algorithm modifies the position (but not orientation) of the root and the orientation of every other joint that is not an end effector. It also allows changes to the offsets of the knees and ankles or, equivalently, the lengths of the thigh and shin. The foot itself (defined as the ankle together with the heel and ball) remains rigid. While changes are allowed for every rotational degree of freedom in the hips and knees, the knee and ball are adjusted as if they were 1 DOF hinge joints. Any other degrees of freedom exhibited by these joints are unchanged.

Our algorithm is not restricted to flat surfaces; sloped or uneven surfaces are also acceptable. However, while we can place precise footplants on such terrain, the algorithm does not adapt other motion qualities like how the character's weight is loaded.

3 Related Work

Inverse kinematics (IK) is the general problem of determining the values of skeletal parameters that place end effectors (or, more generally, joints) at specified locations with specified orientations. The use of IK to drive articulated figure animation dates back to some of the earliest systems, and problems like footskate have been a concern from the beginning [Korein and Badler 1982; Girard and Maciejewski 1985].

There are two broad classes of solutions to inverse kinematics problems: analytic and numerical. Numerical solutions are the most general and are able to handle sophisticated constraints with complex articulated figures. Welman [1989] presented a good survey of computational techniques. Though powerful, numerical IK algorithms suffer from many drawbacks. Since they require solving systems of nonlinear equations, the solutions are computationally expensive. Convergence is difficult to guarantee, and as the IK problem is inherently ill-conditioned [Maciejewski 1990], the algorithms can be unstable.

Analytic IK algorithms, in contrast, find closed-form solutions in fixed amounts of time. However, these algorithms only exist for comparatively simple skeletal configurations. One popular algorithm [Tolani et al. 2000] provides a solution for a 7 DOF human limb consisting of a 3 DOF rotational base joint, a 1 DOF central joint, and a 3 DOF end effector. This algorithm has been incorporated into more complex IK solvers [Lee and Shin 1999; Shin et al. 2001]. As we will discuss in detail in Section 4, this algorithm can produce artifacts as a result of the nonlinear relationship between limb length and angle of the central joint. Our work presents a variant that addresses this difficulty.

One large class of related work involves using IK to match end-effector goals that are specified directly from raw motion capture data. Bodenheimer et al. [1997] fit raw data to a skeleton using an optimization-based numerical IK algorithm that attempted to simultaneously match both measured joint orientations and positions. Choi and Ko [2000] presented an online algorithm based on inverse rate control that was specifically geared toward tracking end-effector positions. Shin et al. [2001] extended this approach to meet positional constraints that were smoothly phased in and out based on proximity to important objects in the environment. Many commercial motion processing systems give a user control over how closely skeletal motion tracks the end-effector positions in the original data [Kaydara Corporation 2001; Stripinis 2001; House of Moves Studios 2001; Vicon Corporation 2001].

These approaches are all fundamentally different from ours in that they involve tracking a data source; that is, they assume the existence of ground truth end effector locations. Hence there are no provisions for editing the original data, as there is no corresponding ground truth for the adjusted motion. Since we explicitly label footplant constraints, our technique works whether or not there are explicit target positions for the end effectors.

Like ourselves, other researchers have used per-frame IK to enforce labelled constraints [Rose et al. 1998] and have blended the resulting adjustments into unconstrained parts of the motion [Monzani et al. 2000]. While the types of constraints addressed in these works are more general than ours, this comes at the expense of using numerical IK techniques that are both slower than ours and possess fewer guarantees about the continuity of the solutions.

One way to combat the instabilities of numerical IK is to consider larger windows of frames. Gleicher cast retargeting as a large optimization problem that enforced constraints over the entire motion simultaneously [Gleicher 1997; Gleicher 1998]. Smoothness in the motion adjustment, termed a *displacement map*, was enforced by defining the displacement map for each degree of freedom as a spline and optimizing over the values of the knots (a similar technique was used by Rose et al. [1996]). The use of splines means that constraints can only be approximately enforced. Also, as this considers the entire motion, it is both computationally expensive and inherently offline.

Lee and Shin [1999] presented an alternate retargeting method that built a displacement map out of a hierarchy of splines. As this involved repeated performing IK over the entire motion, and as this IK involved a nonlinear minimization, it lacks the efficiency of our method. Moreover, part of their IK algorithm involved the problematic single-limb IK solution mentioned in the third paragraph of this section.

4 A Footskate Cleanup Method

The basic idea behind our method is simple. We process a motion sequentially, allowing us to handle both motions of a fixed size and continuous streams of motion. All footplant constraints are satisfied using an analytic IK algorithm. To maintain continuity between adjacent constraints, this algorithm incorporates information from nearby constrained frames. To maintain continuity when a constraint is turned on or off, we blend off the adjustments created by the IK algorithm.

Our approach is divided into five phases:

1. Determine a position for each plant constraint.
2. For each constrained frame, compute global positions and orientations for the ankles that satisfy the constraint positions found in the previous step. Intuitively, we are "chopping off" the foot and setting it so the constraints are satisfied. Care is taken to ensure that adjustments to the original ankle configurations change continuously between constrained frames. Maintaining continuity into unconstrained frames is handled later.
3. Calculate where the root should be placed so the ankle positions found in the previous step can be reached by extending the legs. This goal is balanced with ensuring that the adjustments made to the root are smooth, so in certain cases the target ankle positions may be slightly out of reach. If so, the residual distance may be covered by stretching the legs.
4. For each constrained ankle, adjust the leg so the ankle ends up in the configuration found in step 2, using the root position found in step 3. As much of the adjustment as possible is accomplished by modifying joint rotations, but in certain cases the leg's length is adjusted in order to avoid sharp changes in orientation.
5. Steps 2-4 ensured that we added only smooth adjustments to skeletal parameters when satisfying constraints. However, there may still be discontinuities when constraints switch on and off. In this final step we maintain smoothness in surrounding frames by smoothly dissipating the adjustments.

The computation for a single frame at each step in general requires the results from the previous step over a neighborhood of nearby frames; see Figure 2. Larger neighborhoods provide smoother results at the expense of more computation and, for online applications, greater delay.

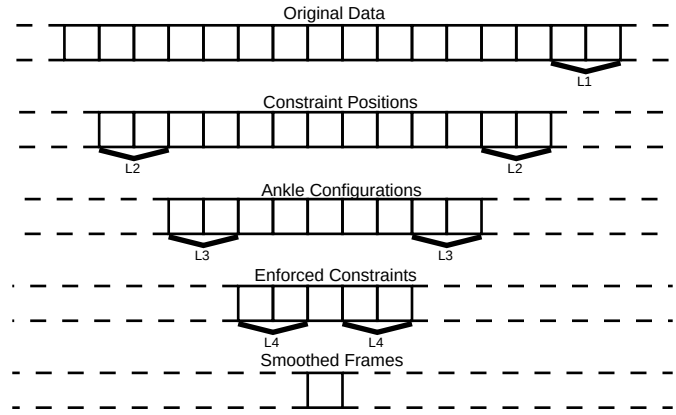

Figure 2: The five phases of the footskate cleanup algorithm. First constraint positions are determined. Next ankle configurations are found that satisfy these constraints. The algorithm then adjusts the root position and leg parameters to place the ankles in these target configurations. Finally, changes are blended off into unconstrained frames to smooth out discontinuities. Each step requires results from the previous step over a neighborhood of frames.

In the remainder of this section we will describe each step of the algorithm and then conclude with a discussion regarding efficient implementation.

4.1 Constraint Positions

Each frame may have plant constraints on the left heel, right heel, left ball, and/or right ball. In some applications the location of each plant is known a priori — for example, the character may be required to step on a pedal. In this case no calculation needs to be performed. Other applications will only require that the joint be planted somewhere on the ground. Since the motion is processed sequentially, when handling frame $\mathcal{F}_i$ the positions are known for all constraints active on the previous frame $\mathcal{F}_{i-1}$. $\mathcal{F}_{-1}$ is defined as having no constraints. There are two scenarios to consider. If a joint J constrained in $\mathcal{F}_i$ is also constrained in $\mathcal{F}_{i-1}$, the constraint position in $\mathcal{F}_i$ is the same as in $\mathcal{F}_{i-1}$. Otherwise the position of J is averaged over the next L_1 frames. The result is projected onto the ground (which may or may not be flat) to the point $\mathbf{c_J}$. Since the foot is treated as a rigid piece, we must next check whether the other joint J' on the same foot is also constrained and has a known position $\mathbf{c_{J'}}$. If so, $\mathbf{c_J}$ is relocated to the closest point on the ground where $\mathbf{c_J}$ and $\mathbf{c_{J'}}$ are the correct distance apart.

4.2 Ankle Position and Orientation

The second phase of the algorithm selects global ankle positions and orientations for $\mathcal{F}_i$ that satisfy the constraints found in the previous step. Care must be taken to ensure that adjustments to the ankle configurations vary continuously. This step assumes known constraint positions for all frames $\mathcal{F}_{i-L_2}$ to $\mathcal{F}_{i+L_2}$.

An ankle A is called *doubly constrained* if both its heel and ball have plant constraints, *singly constrained* if only one of its child joints is constrained, and *unconstrained* otherwise. We first address the case where A is doubly constrained with the heel constraint at $\mathbf{c_H}$ and the ball constraint at $\mathbf{c_B}$. In this case A has one degree of freedom in that when the heel and ball are at the constraint locations, the foot may rotate about the vector $\mathbf{c_B} - \mathbf{c_H}$. The amount

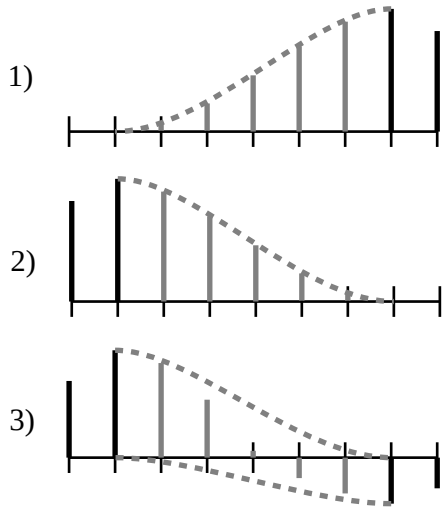

Figure 3: The three cases to consider when blending off an adjustment. Thick black lines represent original adjustments to the motion and thick grey lines represent new adjustments added to blend off the originals. 1) There is a change in the forward direction but not the backward. 2) There is a change in the backward direction but not the forward. 3) There are changes in both directions.

of rotation is commonly called the *foot roll*. If desired, the roll may be set explicitly, perhaps to orient the foot flat on the floor. This could be useful if the character is to traverse uneven terrain. Alternatively, the foot roll may be determined by selecting the roll angle that gives A the global orientation closest to its original value. Define $Rot(\mathbf{a}, \mathbf{b})$ as the shortest rotation that aligns $\mathbf{a}$ with $\mathbf{b}$. Also, let $\mathbf{p_H}$ and $\mathbf{p_B}$ be the original global positions of the heel and ball. Then the target global orientation of A is

$$\mathbf{Q_A}' = Rot(\mathbf{p_B} - \mathbf{p_H}, \mathbf{c_B} - \mathbf{c_H}) * \mathbf{Q_A} \tag{2}$$

where $\mathbf{Q_A}$ is the original global orientation. Capital $\mathbf{Q}$'s are used to distinguish global orientations from orientations relative to the parent coordinate system. The corresponding target global position is $\mathbf{p'_A} = \mathbf{c_H} - \mathbf{Q'_A}(\mathbf{o_H}))$. Here we use the shorthand $\mathbf{q}(\mathbf{v})$ to represent rotating a vector $\mathbf{v}$ by a quaternion $\mathbf{q}$.

Next we consider the case where A is singly constrained. Call the constrained joint J, its global position $\mathbf{p_J}$, and its constraint location $\mathbf{c_J}$. Given *any* orientation for A, we can always satisfy the constraint simply by translating A so J is at its target position. Hence we could simply keep A's current orientation and translate it by $\mathbf{c_J} - \mathbf{p_J}$. However, this can lead to discontinuities if there is a nearby frame with A doubly constrained, since there A must in general be rotated to allow the heel and ball to meet the constraint positions. The solution is to blend off these added rotations. We define a blend function $\alpha(t)$ such that $\alpha(0) = 1$, $\alpha(1) = 0$, and $\frac{d\alpha}{dt}(0) = \frac{d\alpha}{dt}(1) = 0$. The latter two conditions ensure that the blends have C^1 continuity at the beginning and end. The unique cubic polynomial satisfying these conditions is

$$\alpha(t) = 2t^3 - 3t^2 + 1 \tag{3}$$

The algorithm proceeds as follows. We look forward and backward L_2 frames for the first frame in which A is unconstrained or doubly constrained. If we only encounter singly-constrained frames or encounter an unconstrained frame before any doubly constrained frames, we have a *miss* in that direction; otherwise we have a *hit*. If we have a hit in neither direction, we make no change — the adjustment is the identity. Otherwise there are three cases (see Figure 3):

1. We have a hit in the forward direction but not the backwards direction. Say that the adjustment we applied to the doubly-constrained frame F_{i+k} is $\delta_{\mathbf{Q_{i+k}}}$. Then the adjustment we seek is

$$\delta_{\mathbf{Q_i}} = slerp(\alpha(\frac{k+1}{L_2+1}), \mathbf{I_Q}, \delta_{\mathbf{Q_{i+k}}}), \tag{4}$$

   where $slerp$ is spherical linear interpolation and $\mathbf{I_Q}$ is the identity quaternion.

2. The same as (1), but interchanging backward and forward.

3. There are hits in both directions. In this case we compute blends for the forward and backward directions as in the previous two steps and blend the results. Let the doubly constrained frame in the forward direction be F_{i+k} and in the backward direction be F_{i-j}. If

$$\delta_{\mathbf{forward}} = slerp(\alpha(\frac{k+1}{L_2+1}), \mathbf{I_Q}, \delta_{\mathbf{Q_{i+k}}}) \tag{5}$$

   and

$$\delta_{\mathbf{backward}} = slerp(\alpha(\frac{j+1}{L_2+1}), \mathbf{I_Q}, \delta_{\mathbf{Q_{i-j}}}), \tag{6}$$

   then the blended rotation adjustment is

$$\delta_{\mathbf{Q_i}} = slerp(\alpha(\frac{j+1}{j+k+1}), \delta_{\mathbf{backward}}, \delta_{\mathbf{forward}}). \tag{7}$$

By applying the adjustment to the original global orientation, we obtain the target global orientation $\mathbf{Q'_A}$ for the singly-constrained ankle. It is possible that in this orientation the unconstrained joint penetrates the floor; if this is the case, we adjust $\mathbf{Q'_A}$ by the smallest rotation about the constrained joint that places the unconstrained joint above the floor. As before, the target global position is $\mathbf{p'_A} = \mathbf{c_J} - \mathbf{Q'_A}(\mathbf{o_J})$.

The final case to consider is when A is unconstrained. If an adjacent frame has A constrained, then in general A's configuration will be discontinuous between the two frames. However, since there are no constraints on A, we are free to adjust the parameters for the leg so as smooth out the discontinuity. This is simpler to accomplish once the constraints have been satisfied; it will be discussed in Section 4.5. For now we continue to focus on the issue of satisfying $\mathcal{F}'_i s$ constraints.

4.3 Root Placement

By this stage we have figured out target global positions and orientations for the constrained ankles in frames $\mathcal{F}_{i-L_3}$ to $\mathcal{F}_{i+L_3}$. The next step will adjust the legs to meet these targets. However, it may be that the legs cannot reach the target ankle positions even when fully extended. Hence we attempt to move the root by the smallest amount that makes the target ankle positions reachable.

If only one ankle is constrained, we calculate the maximum leg length l_{max} and compare with the distance l_{target} of the target from the root. If $l_{target} > l_{max}$, we project the root onto the sphere of radius l_{max} centered about $\mathbf{p'_A} - \mathbf{o_H}$, where $\mathbf{o_H}$ is the hip offset.

If both ankles have constrained children, then we must project onto the intersection of two spheres. This can be done as follows. First, for each ankle project $\mathbf{p_R}$ onto the appropriate sphere. If only one

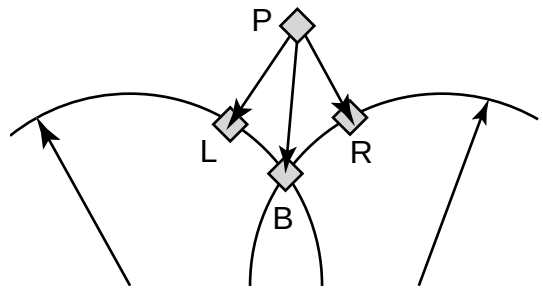

Figure 4: If a target position is to be reached by the root without stretching the leg, the root must be within the sphere centered on the target whose radius is equal to the length of the leg at full extension. If both ankles have target positions, then the root must be in the intersection of these two spheres. When constraints switch the projection may move discontinuously, even if the root is stationary. For example, projecting the root position (P) onto the sphere centered about the left ankle target at (L) produces a point distant from the corresponding projections for the right ankle target (R) and both targets (B).

result is in the $3D$ intersection of the spheres, then this is the projection. If both are in the $3D$ intersection, then select the closest. Otherwise the point should be projected onto the circle defined by the intersection of the surfaces of the spheres. Details on this straightforward operation are in Shin et al. [2001].

As long as the same constraints hold, these projection operations will preserve the continuity of the motion. However, a discontinuity may be introduced if a constraint switches on or off. This is because the projected root position can be very different in the single-ankle case and the double-ankle case (Figure 4), causing the root to "pop" from one frame to the next. Even small amounts of root popping can be quite distracting, as a translation of the root joint moves the entire body.

Attempting to generate smooth root displacements subject to sphere-interiority constraints is a non-linearly constrained variational problem for which it is difficult to find a reliable and efficient solution. For this reason we adopt a simpler solution to root popping. Root translations are calculated independently for each frame in the neighborhood $\mathcal{F}_{i-L_3}$ to $\mathcal{F}_{i+L_3}$. We then compute a weighted average over the block of constrained frames that contains $\mathcal{F}_i$. While this filtering may leave the root outside of the sphere(s), the discrepancy is usually quite small, and we can still exactly meet the ankle position constraints by slightly stretching the legs.

As in the previous section, there is still an opportunity for root popping when a frame with no constraints borders a frame with at least one constraint. We address this possibility in 4.5.

4.4 Meeting the Target Ankle Configuration

Using the root position calculated in the previous step, all constraints on $\mathcal{F}_i$ can be satisfied by adjusting the hip, knee, and ankle parameters such that the target ankle positions and orientations are met. We use a modified version of a standard single-limb IK algorithm [Lee and Shin 1999; Tolani et al. 2000]; see Figure 5. We will first describe the original algorithm. The global positions of the hip, knee, ankle, and target ankle location are respectively $\mathbf{p_H}$, $\mathbf{p_K}$, $\mathbf{p_A}$, and $\mathbf{p_T}$. The lengths of the thigh and the shin are respectively l_t and l_s. Since the knee can only rotate about a single axis, we are also interested in the lengths $\tilde{l_t}$ and $\tilde{l_s}$ of the projections of the thigh and shin into the plane defined by this axis. The knee is first extended/contracted so the distance from the hip to the ankle equals the distance from the hip to the target. The final knee angle θ is chosen so as to satisfy the constraint that the knee must not bend

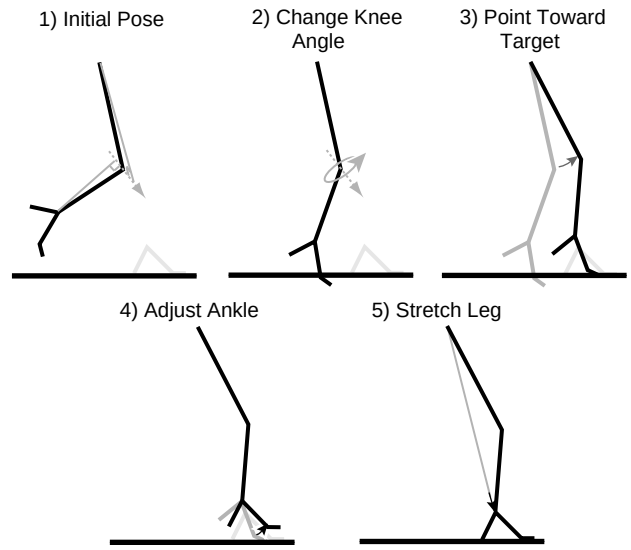

Figure 5: The five steps for attaining a target ankle position and orientation.

backwards. Using straightforward geometry, one can show

$$\theta = \arccos\left(\frac{l_t^2 + l_s^2 + 2\sqrt{l_t^2 - \tilde{l_t}^2}\sqrt{l_s^2 - \tilde{l_s}^2} - \|\mathbf{p_H} - \mathbf{p_T}\|^{\mathbf{2}}}{2\tilde{l_t}\tilde{l_s}}\right) \tag{8}$$

See Lee and Shin [1999] for a concise derivation.

The hip is now adjusted as follows. Let the new position of the ankle be $\tilde{\mathbf{p_A}}$. The hip is rotated by $Rot(\tilde{\mathbf{p_A}} - \mathbf{p_H}, \mathbf{p_T} - \mathbf{p_H})$ to point the leg toward the target. If the length of the leg matches the distance to the target, the ankle will end up at $\mathbf{p_T}$.

The hip may be rotated about the axis $\hat{\mathbf{n}} = \mathbf{p_T} - \mathbf{p_H}$ without changing the ankle position. There are a variety of ways to exploit this redundancy [Korein and Badler 1982]. In our implementation we chose to make the global ankle orientation as close as possible to the target value $\mathbf{Q'_A}$ found in Section 4.2. Let the current global ankle orientation (incorporating the adjustments to the hip and the knee) be $\mathbf{Q_A}$. Then we seek a rotation ϕ about $\hat{\mathbf{n}}$ such that if $\delta_{\mathbf{Q_H}} = (\mathbf{cos}\frac{\phi}{\mathbf{2}}, \hat{\mathbf{n}}\sin\frac{\phi}{\mathbf{2}})$, then $\delta_{\mathbf{Q_H}}\mathbf{Q_A}$ is as close as possible to $\mathbf{Q'_A}$, where the distance between two quaternions is determined by the corresponding geodesic in S^3. The solution, derived in Shin et al. [2001], is as follows. Writing $\mathbf{Q_A}' = (w', \mathbf{v}')$ and $\mathbf{Q_A} = (w, \mathbf{v})$, define

$$\beta = \arctan\left(\frac{ww' + \mathbf{v}\cdot\mathbf{v}'}{w\hat{\mathbf{n}}\cdot\mathbf{v}' - w'\hat{\mathbf{n}}\cdot\mathbf{v} + \mathbf{v}'\cdot(\hat{\mathbf{n}}\times\mathbf{v})}\right). \tag{9}$$

Then ϕ is either $-2\beta + \pi$ or $-2\beta - \pi$, depending on which value maximizes $(\delta_{\mathbf{Q_H}}\mathbf{Q_A}) \cdot \mathbf{Q'_A}$ (here the quaternions are treated as Euclidean 4-vectors).

Having adjusted the hip and the knee, the single-limb IK concludes by applying to the ankle the unique rotation that brings it to the target global orientation.

If the skeletal parameters are varied continuously, this algorithm produces continuous outputs. However, continuity alone does not guarantee that small changes in the initial conditions produce similarly small adjustments to the skeletal parameters. The culprit is the nonlinear relationship between leg length l (the distance from

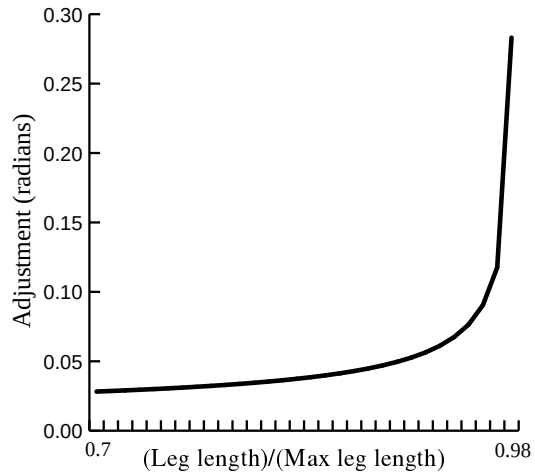

Figure 6: The adjustment in knee angle necessary to extend the leg by 1% of its maximum length versus the current length of the leg, expressed as a percentage of its maximum length.

the hip to the ankle) and knee angle θ. By the law of cosines, $l^2 = l_t^2 + l_s^2 - 2l_t l_s \cos\theta$. Since $\frac{dl}{d\theta}(\theta) = \frac{l_t l_s \sin\theta}{l}$ drops to zero as $\theta \to \pi$, small changes in leg length can require comparatively large changes in knee angle. Consider Figure 6, where we plot the knee angle adjustment necessary to extend the leg by 1% of its maximum length versus the current length of the leg. The sharp rise in the graph manifests itself as an unnaturally fast extension/contraction of the knee, which we term a "knee pop". Even a minor knee pop can stand out in a motion since it affects the shape of the entire leg.

The knee pop problem is fundamental to the geometry of the leg; attempting to extend a nearly-straight leg is a poor way to increase the reach of the ankle. The solution is to limit the amount we rotate the knee when the leg is near full extension. We refer to this as knee-damping. Knee-damping can be implemented as follows. Let $\rho \in (0, \pi)$ and define $f(x)$ as follows:

$$f(x) = \begin{array}{ll} 1, & x < \rho \\ \alpha\left(\frac{x-\rho}{\pi-\rho}\right) & \rho \le x < \pi \\ 0, & x \ge \pi \end{array} \tag{10}$$

Then if the original knee angle is θ_0 and the adjustment to it is Δ_θ, the final angle θ is

$$\theta = \theta_0 + \int_{\theta_0}^{\theta_0+\Delta_\theta} f(x)dx. \tag{11}$$

This has the effect of limiting the allowable rotation adjustment when the leg is near full extension.

When we use knee damping, in general the ankle may not able to reach its target position upon adjusting the knee. We might try to fix this by translating the root, but this would produce the problems discussed in the previous subsection: when constraints switch on and off we can have discontinuities in the root position. Our solution is to adjust the length of the leg. If the actual distance from the hip to the ankle is d and we require the distance to be d', then we replace the knee offset $\mathbf{o_K}$ with $(\frac{d'}{d})\mathbf{o_K}$ and the ankle offset $\mathbf{o_A}$ with $(\frac{d'}{d})\mathbf{o_A}$. With this length adjustment the ankle is always able to exactly reach its target position.

If the leg is bent, then the ankle-to-hip distance is only affected by a fraction of the length changes applied to the leg bones. However, we only adjust the bone lengths when the leg is nearly straight.

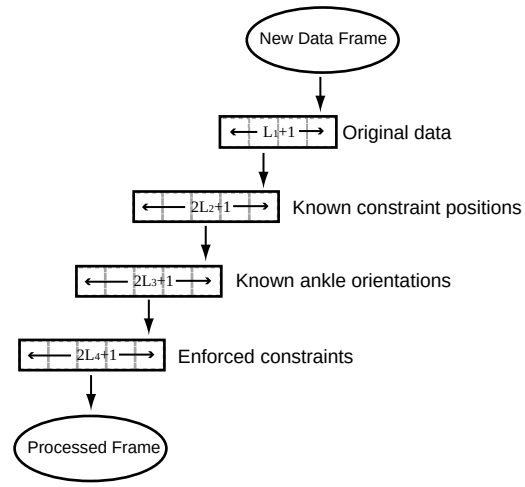

Figure 7: A schematic for an efficient online implementation of the footskate cleanup algorithm. Four buffers are maintained to store, respectively, original frame data, ankle configurations, constraint-enforced frames, and fully-processed frames. Each buffer contains all the information necessary to create a single entry in the next buffer. Hence every time a new frame enters the top buffer, a processed frame may be extracted from the bottom buffer.

Thus almost all of the length adjustment goes directly into changing the ankle-to-hip distance, and so usually only small length adjustments are necessary. Moreover, these length changes are smoother and smaller than the corresponding change to the knee and hence are less distracting.

4.5 Final Processing

So far the discussion has focused on enforcing constraints for a particular frame. While we don't introduce discontinuities when satisfying constraints that hold over multiple frames, discontinuities may still occur when a constraint becomes active or inactive. To correct this, we blend off the adjustments generated to enforce constraints. The method is similar to how we calculated smooth target ankle configurations in Section 4.2. We assume that all constraints have been solved in the neighborhood $\mathcal{F}_{i-L_4}$ to $\mathcal{F}_{i+L_4}$. For each skeletal parameter in $\mathcal{F}_i$ that was *not* adjusted to satisfy a constraint, we scan forwards and backwards in the neighborhood until we either run off the end or encounter a frame where that parameter was adjusted to enforce a constraint. If there are no such frames in either direction, the parameter is left unchanged. Otherwise we create a new adjustment for $\mathcal{F}_i$ by blending off the adjustment(s) we encountered in the scan. For parameters represented as quaternions, this is done exactly as in Section 4.2. For parameters represented by 3-vectors, we replace $slerp$ with $lerp$, which linearly interpolates the 3-vectors.

If a foot has at least one constraint, it is guaranteed to be above the floor after that constraint is satisfied. However, a foot with no constraints may still penetrate the floor. If this is the case, we translate the ankle by the smallest amount that places both the ball and the heel above the floor and use the algorithm of section 4.4 to adjust the rest of the leg accordingly. The continuity of the original motion is preserved as long as the floor is smooth.

The final step is to ensure that neither of the toes penetrate the floor. A positive toe rotation is defined as one that bends the toes up toward the top of the foot when the skeleton is in a normal standing posture. We apply the smallest positive rotation that places the toe above the floor. In general this depends on the shape of the floor. For flat surfaces, the toe position can be solved as follows. Let $\mathbf{n}$

be the normal defining the plane of rotation of the toes, c be the vertical distance from the position of the ball to the floor, and $\mathbf{v_T}$ be the location of the toes relative to the ball. Then what we seek is a vector $\mathbf{v'_T}$ that 1) has value c in the vertical direction (y axis), 2) has length $\|\mathbf{v_T}\|$, and 3) is orthogonal to $\mathbf{n}$. If we define

$$\eta = \sqrt{(n_x^2 + n_z^2)(\|\mathbf{v_T}\|^2 - c^2)} \tag{12}$$

then the solution to this system of equations is

$$\mathbf{v'_T} = (\frac{-cn_y \mp n_z\eta}{n_x^2 + n_z^2}, c, \frac{-cn_z \pm n_z\eta}{n_x^2 + n_z^2}) \tag{13}$$

where the sign is chosen such that $(\mathbf{v'_T} \times \mathbf{v_T}) \cdot \mathbf{n}$ is positive.

4.6 Efficient Implementation

As shown in Figure 2, processing a single frame involves a calculation over a neighborhood of nearby frames. Since these neighborhoods overlap, calculations will be needlessly repeated if each frame is processed completely independently. A more efficient implementation, depicted in Figure 7, is to maintain a series of buffers that hold information pertaining to the different steps of the algorithm. The first buffer holds $L_1 + 1$ frames of original data. Constraint locations are extracted from this and passed into a buffer of length $2L_2 + 1$. In this buffer the ankle configuration for the central frame can be calculated using the methods of Section 4.2. The result is then sent into the third buffer, which holds $2L_3 + 1$ frames. The root translation for the central frame is set as in Section 4.3 and then the leg parameters are adjusted according to Section 4.4. Finally, the constraint-solved frames are placed into a fourth buffer of length $2L_4 + 1$, where adjustments are blended off as described in Section 4.5. At the center of this final buffer is a completely processed frame.

Each buffer contains exactly the amount of information necessary to calculate a single entry in the next buffer. Hence every time a new data frame is added, one entry is added to the beginning of each buffer and removed from the end.

5 Results

In this section we present some sample results for our footskate cleanup algorithm. In our experiments L_1 was 10 frames (a third of a second) and L_2, L_3, and L_4 were 5 to 10 frames (a sixth to a third of a second). The value of ρ in equation 10 was 2.8 (about 160 degrees). Over all of our experiments, the maximum change in leg length on a given frame was 2.6% of the leg length at full extension.

Our algorithm generates precise footplants on the specified frames. Figure 8 compares an example of footskate with the corresponding footplant produced by our algorithm. To get a better sense of how successful our algorithm is, we have compared it to the corresponding algorithm in Kaydara's FILMBOX 3, a popular software package for processing motion capture. Data from an optical motion capture system was fit to a skeleton using both FILMBOX's default settings and the "reach-feet" option, which tracks end-effectors to reduce artifacts like footskate. We then applied our footskate cleanup algorithm to FILMBOX's default skeleton fit. Figure 9 shows the height of the right ball as a function of time for each motion. Due to errors in marker tracking and the skeleton-fitting, FILMBOX both introduced minor root-popping and knee-popping artifacts and generated small amounts of footskate. Our algorithm met the footplants exactly while avoiding such artifacts.

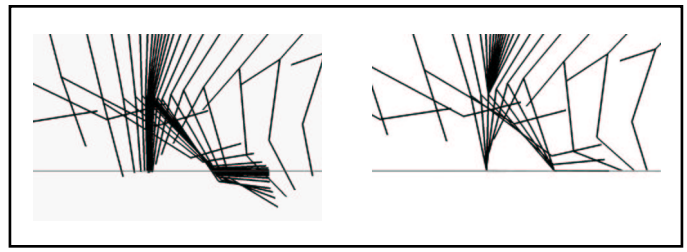

Figure 8: An example of footskate and the result after applying our algorithm. The pictures show the location of the right foot over a portion of a walking motion.

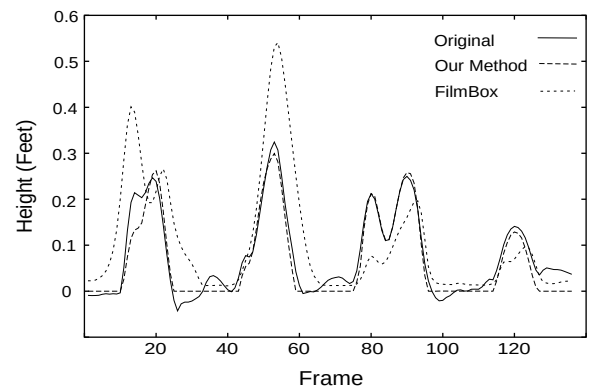

Figure 9: Three graphs of the height of the right ball in a segment of motion where the actor turned around in place several times. The solid plot shows skeletal data generated using FILMBOX's defaults for fitting marker data to a skeleton. The dotted plot had FILMBOX's "reach-feet" mode activated, which executes their footskate solution. The dashed plot shows the results of applying our algorithm to the default FILMBOX fit.

Our algorithm's comparative success is not terribly surprising since it was given more information. While FILMBOX only had marker data, our algorithm was supplied with the exact periods of time over which footplants were to occur. Hence it is able to incorporate additional timing information (when footplants begin and end) and geometric information (where the foot is supposed to go, determined by the shape of the ground).

We have tested our algorithm in a variety of contexts:

1. **Motion Salvage.** We received a motion from a motion capture studio that was damaged because the actor adjusted his pants during the shoot, thereby throwing off the calibration of some of the markers. We used our algorithm to salvage the motion by eliminating the considerable footskate while at the same time not adding any new visible artifacts. Figure 10 shows plots of the height of the right ball on the damaged motion before and after applying our algorithm.

2. **Retargeting.** With footskate cleanup, simple retargeting operations are possible. A scale factor is applied to the root translation data in the original motion according to the difference in bone lengths between the original skeleton and the target skeleton, and we enforce footplant constraints on the result. While this is not a general solution to the retargeting problem, in simple cases it suffices.

3. **Path Editing.** We have used our footskate cleanup algorithm to successfully satisfy footplant constraints after applying the path editing operation discussed in [Gleicher 2001]. Our algorithm is a simple, efficient alternative to the spacetime optimization used in the original paper.

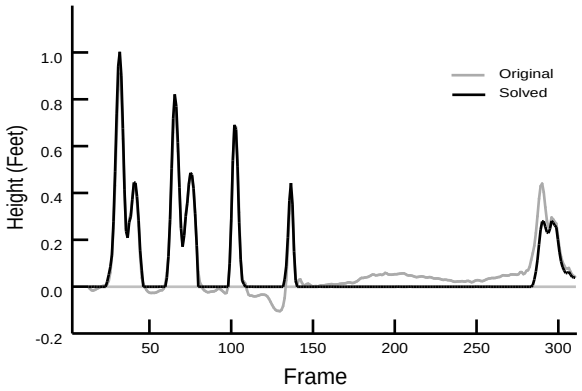

Figure 10: The height of the right heel as function of time in a motion damaged due to sensor calibration error (grey), along with the height of the right heel after running our algorithm on this motion (black).

Our algorithm is quite fast; processing a 445 frame motion with 673 constraints required only 0.176 seconds (0.0966 milliseconds per frame) on a 1.3 GHz Athlon personal computer.

6 Discussion

In this paper we have presented a simple, efficient, and online algorithm for precisely enforcing footplant constraints in a motion without adding visible artifacts. This algorithm is useful because it increases the utility of other editing operations which destroy footplants. Moreover, while in the past new editing algorithms have had to explicitly address footskate, our cleanup algorithm frees future algorithm designers from this additional burden.

We have focused exclusively on footplants, ignoring the possibility of also constraining the hand positions. It would only be a minor extension to satisfy hand constraints by treating the arms in the same manner as we treat the legs. The only step of the algorithm that would be affected is when the root position is calculated, as in general the root would have to be projected into the intersection of four spheres [Shin et al. 2001]. We chose not to do this for the following reasons. First, hand constraints are much rarer than foot constraints. Second, the hand is far more complicated than the foot in that the specific configuration of the fingers is typically important, whereas individual toes usually need not be modelled. Third, the torso is more of a factor in arm movement than in leg movement, and so the simple IK used in this paper is unlikely to suffice.

The success of our algorithm was made possible by taking the somewhat unusual measure of allowing small length changes in the skeleton's bones, whereas in most previous work the skeleton has been strictly rigid. This leads to the interesting general question of how various artifacts in a motion — footskate, over-stretched limbs, sudden changes in joint orientation, etc. — may be balanced so as to produce a desired change while minimizing visual disturbance.

Acknowledgements

Our work was made possible through generous motion data donations from Spectrum Studios (particularly Demian Gordon), House of Moves, and The Ohio State University. This work was supported in part by NSF grants CCR-9984506 and IIS-0097456, the Wisconsin Alumni Research Fund's University Industrial Relations program, equipment donations from IBM, NVidia, and Intel, and software donations from Discreet, Alias/Wavefront, and Pixar.

References

BINDIGANAVALE, R., AND BADLER, N. 1998. Motion abstraction and mapping with spatial constraints. In *Modelling and Motion Capture Techniques for Virtual Environments, CAPTECH'98*, 70–82.

BINDIGANAVALE, R. 2000. *Building parameterized action representations from observation*. PhD thesis, Department of Computer and Information Sciences, University of Pennsylvania.

BODENHEIMER, B., ROSE, C. F., ROSENTHAL, S., AND PELLA, J. 1997. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97*, 3–18.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proceedings of ACM SIGGRAPH 1995*, Annual Conference Series, ACM SIGGRAPH, 97–104.

CHOI, K.-J., AND KO, H.-S. 2000. On-line motion retargeting. *Journal of Visualization and Computer Animation 11*, 223–243.

GIRARD, M., AND MACIEJEWSKI, A. A. 1985. Computational modeling for the computer animation of legged figures. In *Proceedings of ACM SIGGRAPH 1985*, ACM SIGGRAPH, 263–270.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Proceedings 1997 Symposium on Interactive 3D Graphics*, M. Cohen and D. Zeltzer, Eds., ACM, 139–148.

GLEICHER, M. 1998. Retargeting motion to new characters. In *Proceedings 0f ACM SIGGRAPH 98*, Annual Conference Series, ACM SIGGRAPH, 33–42.

GLEICHER, M. 2001. Motion path editing. In *Proceedings 2001 ACM Symposium on Interactive 3D Graphics*, J. Hughes and C. Sequin, Eds., ACM.

HOUSE OF MOVES STUDIOS, 2001. Diva. Software Package.

KAYDARA CORPORATION, 2001. Filmbox 3.0. Software Package.

KOREIN, J. U., AND BADLER, N. I. 1982. Techniques for generating the goal-directed animation of articulated structures. *IEEE Computer Graphics & Applications 2*, 9 (November), 71–81.

LANDER, J. 1998. Working with motion capture file formats. *Game Developer 5*, 1 (January), 30–37.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of ACM SIGGRAPH 99* (Aug.), 39–48.

LEWIS, J., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, ACM SIGGRAPH, 165–172.

MACIEJEWSKI, A. A. 1990. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics & Applications 10*, 3 (May), 63–71.

MONZANI, J.-S., BAERLOCHER, P., BOULIC, R., AND THALMANN, D. 2000. Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum 19* (August).

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of ACM SIGGRAPH 1996*, Annual Conference Series, ACM SIGGRAPH, 147–154.

ROSE, C., COHEN, M., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comp. Graphics and application 18*, 5, 32–40.

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

STRIPINIS, D. 2001. Product review: Kaydara's filmbox 3.0. *Game Developer 8*, 12 (December), 8–9.

TOLANI, D., GOSWANMI, A., AND BADLER, N. 2000. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models 62*, 353–388.

VICON CORPORATION, 2001. Body builder. Software Package.

WELMAN, C. 1989. *Inverse Kinematics and Geometric Constriants for Articulated Figure Manipulation*. Master's thesis, Simon Fraser University.

WITKIN, A., AND POPOVIĆ, Z. 1995. Motion warping. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, ACM SIGGRAPH, 105–108.