Chapter 3

End Effector Cleanup

3.1 Introduction

Real human motion often contains interactions between end effectors (hands and feet) and parts of the environment, and each of these interactions may be thought of as a kinematic constraint that requires the relevant end effector and the object to be in contact. For example, one of the most common kinematic constraints is a *footplant*, which requires part of a foot to remain stationary on the ground. Footplants are ubiquitous because they are present in almost any motion where a character uses its legs to support its weight. Regardless of the accuracy of the original motion capture data, constraints such as footplants can easily be violated. This is because the raw data is rarely used directly, but rather is first fit onto a rigid skeleton and then altered to satisfy the particular demands of the animation. Indeed, the data-driven synthesis techniques developed in Chapters 4–6 produce motions that in general violate kinematic end effector constraints. Even modest constraint violations can be quite distracting because they break the logical connection between the character's actions and its surroundings — for example, when a footplant is violated, the character appears to unnaturally penetrate or slide across the ground, an artifact known as *footskate*.

This chapter presents a new algorithm for enforcing kinematic constraints on end effectors. This algorithm is a crucial component of the motion models developed in the following chapters, as it provides a mechanism for automatically enforcing these constraints in synthesized motion. Our algorithm is appropriate for constraints that require part of an end effector to be stationary

(such as when standing on the ground or leaning on a table), to reach a particular point in space (such as when pushing a button or kicking a ball), or to maintain contact with an object for a prolonged duration (such as when carrying a box). It exactly enforces constraints while ensuring that no discontinuous changes are made to the motion, and it exclusively uses analytic methods involving a fixed amount of calculation for each frame. It is also simple and easy to implement, as it is based entirely on straightforward geometric manipulations. Finally, our algorithm processes an individual frame using only information from a fixed neighborhood of surrounding frames, which makes it useful for online applications that can accept a small time delay ($\frac{1}{4}$s–1s), such as animating background characters in a game or individuals in a crowd simulation.

While previous work [30, 31, 54, 20, 83] has assumed that skeletons are perfectly rigid, we allow bones to change length. We argue that rigid skeletons force one to make a precarious and unnecessary tradeoff between satisfying constraints exactly and avoiding sharp adjustments to joint parameters. While more specific reasons will be covered in detail later, our decision is unusual enough that we would like to explain up front why we believe it to be reasonable. First, since end effector 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 [50]) and animation packages (such as Poser, Maya, and 3D Studio Max). Third, variable bone lengths can be seamlessly incorporated into traditional algorithms for specifying how a skeleton drives a character's mesh [56]. Lastly, recent perceptual experiments by Harrison et al. [37] suggest that the amount of non-rigidity that we require in practice is difficult to detect even when viewers are specifically looking for changes in limb length.

In the remainder of this chapter, we provide the details of our algorithm (Section 3.2), present some results (Section 3.3), and conclude with a brief discussion (Section 3.4).

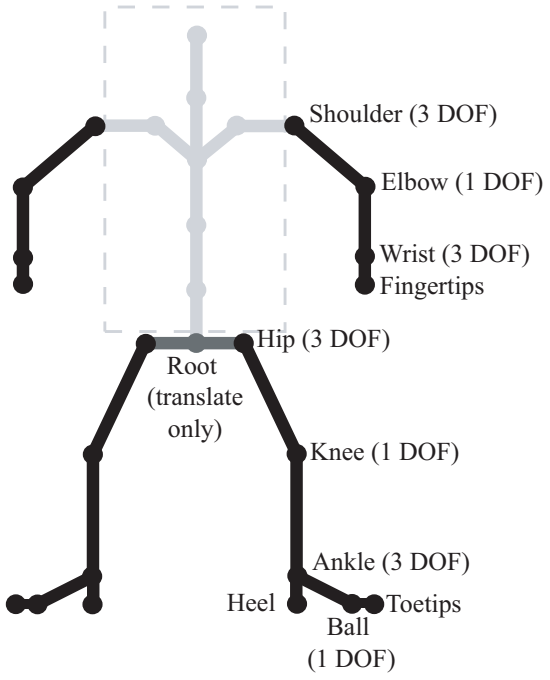

Figure 3.1: Skeleton model for end effector cleanup.

3.2 An End Effector Cleanup Method

3.2.1 Algorithm Overview

For the purpose of end effector cleanup, we model the body as shown in Figure 3.1. The orientations of the spine joints and the root are not altered, although we allow changes to the root position. While allowing the spine to bend would provide our solver with more flexibility, this would also require iterative optimization procedures [54] that lack the speed and robustness advantages of the analytic algorithm presented here. Each limb is treated as a 3 DOF joint (the shoulder/hip) followed by a 1 DOF joint (the elbow/knee) and a 3 DOF end effector (the wrist/ankle); extra degrees of freedom exhibited by the elbow/knee are left unchanged. The elbow and knee are prohibited from bending backwards, but no orientation limits are enforced for the other joints. The two segments of each limb are allowed to vary in length, but the end effectors are assumed to be rigid (e.g., the offsets of the heel and ball do not change).

While a user may directly provide target positions and orientations for an end effector, it is often convenient to supply more general restrictions on end effector configuration. We discuss in detail the case of plant constraints on end effector tips. These are constraints that require either the heel, ball, wrist, or fingertips to remain in a fixed position during some time interval. In general the plant position is unspecified, although it may be constrained to lie on a planar surface. Given constraints of this form, our system determines appropriate positions for the end effector tips (Section 3.2.2) and corresponding end effectors configurations (Section 3.2.3). Note that plant constraints are a natural generalization of footplant constraints, for which the end effector tip is the heel or ball of a foot and the planar surface is the ground. Constraints on the very tips of the toes are not considered, although we do ensure that the toes don't penetrate the ground as a result of other constraints being satisfied[1]. Other kinds of restrictions on end effector configuration, such as ones used to avoid object penetration, are also not discussed. If such constraints exist, then we assume that an outside process will provide our algorithm with appropriate end effector positions and orientations.

The basic idea behind our algorithm is simple. Frames of motion are processed sequentially, so both fixed-duration motions and continuous streams of motion can be handled. To place end effectors in given positions and orientations, an analytic inverse kinematics (IK) algorithm is used to adjust individual limbs. This IK algorithm creates smooth changes as long the target end effector configurations vary smoothly, and in particular damps unnaturally fast changes that can occur when a limb is near full extension. To ensure that smooth changes are made across constraint boundaries, our algorithm incorporates information from the surrounding neighborhood of frames when adjusting a particular frame.

Our approach is divided into five steps. The first two steps assume that constraints are specified as plant constraints on end effector tips; if target end effector positions and orientations are supplied directly, then these steps may be skipped.

1. Determine a position for each plant constraint.

[1]Toetip constraints are quite rare in practice. For example, a "tiptoeing" motion really involves the weight being supported on the toes *and* the end of the ball, and hence the correct result can be obtained simply by planting the ball.

2. For each constrained frame, compute global positions and orientations for the end effectors that satisfy the constraint positions found in the previous step. Intuitively, the end effector is "chopped off" and arranged so the plant constraints are satisfied. Care is taken to ensure that target end effector configurations change continuously when constraints switch on and off.

3. Calculate where the root should be placed so the target end effector positions can be reached by extending or contracting the limbs. This goal is balanced with ensuring that the adjustments made to the root are smooth, so in certain cases the target end effector positions may be slightly out of reach. If so, the residual distance is covered in the following step by stretching the limbs.

4. Using the root position found in the previous step, adjust each limb as necessary so target end effector configurations are reached. As much of the adjustment as possible is accomplished by modifying joint orientations, but when a limb is near full extension its length may be adjusted to avoid sharp changes to the knee/elbow.

5. The previous steps have ensured that smooth adjustments are made as long as at least one constraint exists on an end effector. However, there may still be discontinuities when an end effector switches from zero constraints to one or more constraints or vice-versa. This is avoided by smoothly dissipating adjustments into unconstrained frames.

Each step relies on results from the previous step over a neighborhood of nearby frames (Figure 3.2). Larger neighborhoods provide smoother results, but for online applications this comes at the expense of greater delay.

The remainder of this section describes each step of the algorithm and then concludes with a discussion regarding efficient implementation.

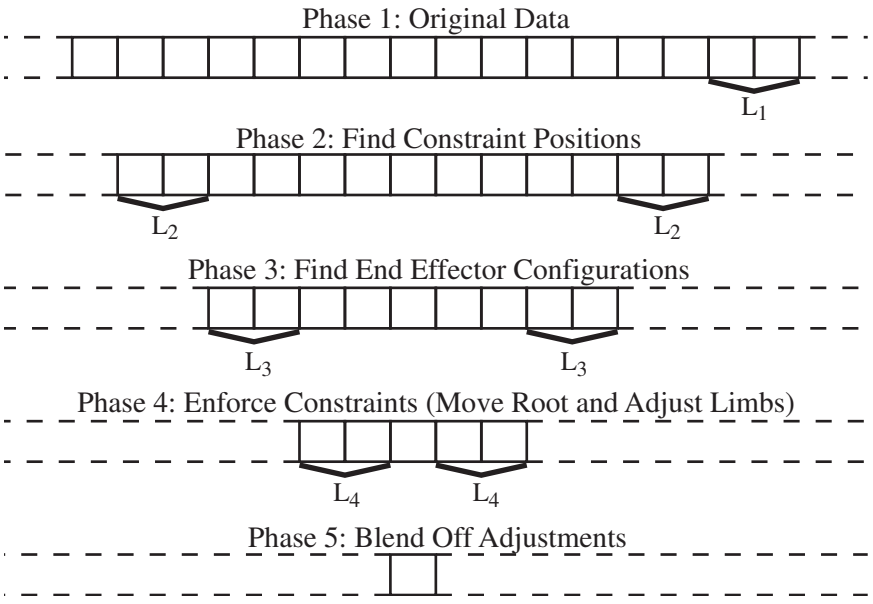

Figure 3.2: The phases of the constraint enforcement algorithm. First, plant positions are determined. Next, end effector configurations are found that satisfy these constraints. The root position and limb parameters are then adjusted to place the end effectors in these configurations. Finally, changes are dissipated into unconstrained frames. Each phase requires results from the previous phase over a neighborhood of surrounding frames; for example, in order to find the ankle configuration at frame $\mathbf{M_i}$, the constraint positions for the heel and/or ball must be known from $\mathbf{M_{i-L_2}}$ to $\mathbf{M_{i+L_2}}$.

3.2.2 Finding Plant Constraint Positions

Each frame may have up to eight plant constraints, one for each heel, ball, wrist, and fingertips. In some cases the location of a plant will be known a priori; for example, the character may be required to step on a pedal. If this is not the case, then plant locations must be computed.

Each end effector has two positions in its local coordinate system that may be planted. For an ankle, these positions correspond to the heel and the ball; for a wrist, they correspond to the wrist itself and the fingertips. Consider the problem of finding a position on frame $\mathbf{M_i}$ for a plant constraint on joint J_1 that spans n_{J_1} frames. Let J_2 be the other joint for that end effector, and let $\mathbf{o_1}$ and $\mathbf{o_2}$ be, respectively, the offset of J_1 and J_2 in the end effector's coordinate system (Figure 3.3). Since the motion is processed sequentially, the positions of all constraints existing on previous frames are already known, so it is sufficient to handle the case where the constraint starts on $\mathbf{M}_i$.

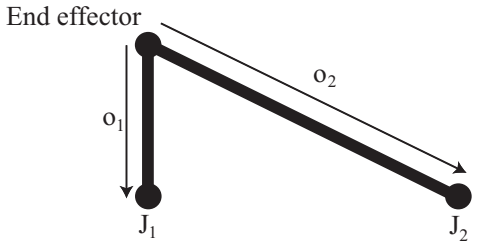

Figure 3.3: Diagram of an end effector. If the end effector is a hand and J_1 is the wrist, then $\|\mathbf{o_1}\| = 0$.

First, assume that J_2 either is not constrained on $\mathbf{M_i}$ or has a plant constraint for which a location has not yet been determined. The plant position $\mathbf{c_1}$ for J_1 is then found by averaging the global position of J_1 over the next $min(L_1, n_{J_1})$ frames and, if the plant is required to be in a plane, projecting the result onto this plane. L_1 is a user-defined constant.

Now assume that J_2 also has a plant constraint on $\mathbf{M_i}$ and that the position $\mathbf{c_2}$ of this plant is known. Since the hands and feet are treated as rigid objects, J_1 must be planted a distance $\|\mathbf{o}_1 - \mathbf{o_2}\|$ from J_2. Our strategy is to place J_1 in a position consistent with the average orientation of the end effector during the frames when both J_1 and J_2 are constrained. Let m be either the number of frames when both J_1 and J_2 are constrained or a user-defined constant L_1, whichever is smaller. Also, let $\mathbf{Q_j}$ be the global orientation of the end effector on frame $\mathbf{M_j}$, where we use capital letters to distinguish global orientations from local orientations. An average orientation $\overline{\mathbf{Q}}$ is defined by computing

$$\overline{\mathbf{Q}} = \frac{1}{1+m} \sum_{j=i}^{i+m} \mathbf{Q}_j \tag{3.1}$$

and then normalizing the result so $\overline{\mathbf{Q}}$ is a unit quaternion. $\mathbf{Q_j}$ is replaced with its antipode whenever the vector dot product of $\mathbf{Q_j}$ and $\mathbf{Q_i}$ is negative, which ensures that all of the $\mathbf{Q_j}$ are on the same hemisphere of the unit quaternion sphere [55]. We experimented with alternative (and slower) methods for averaging orientations based on the logarithmic map [68], but the differences were negligible. The plant location for J_1 is set to

$$\mathbf{c_1} = \mathbf{c_2} + \overline{\mathbf{Q}}(\mathbf{o_1} - \mathbf{o_2}). \tag{3.2}$$

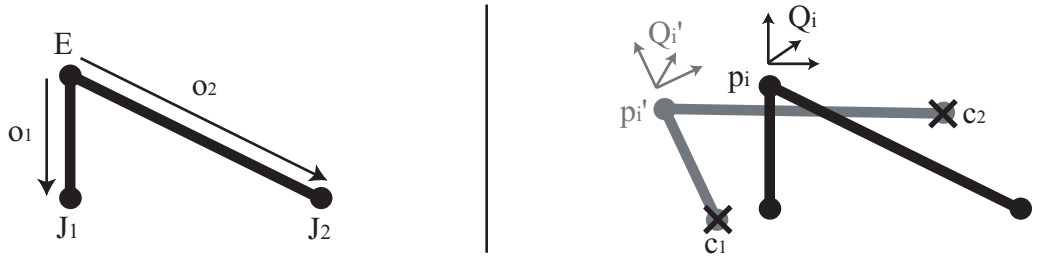

Figure 3.4: Left: Diagram of an end effector. Note that $\mathbf{o_1}$ and $\mathbf{o_2}$ are fixed. **Right:** The configuration $(\mathbf{p_i}, \mathbf{Q_i})$ of E on frame $\mathbf{M_i}$ is changed to $(\mathbf{p'_i}, \mathbf{Q'_i})$ so J_1 and J_2 are placed, respectively, at $\mathbf{c_1}$ and $\mathbf{c_2}$.

If $\mathbf{c_1}$ is constrained to lie in a plane, then it is rotated about $\mathbf{c_2}$ by the smallest rotation needed to place it in that plane.

3.2.3 Finding Target End Effector Configurations

Once locations for plant constraints are known, the next step is to find end effector configurations that satisfy these constraints. For example, if the heel and ball of the left foot are both planted, then the ankle must be set in a position and orientation that places these joints at their plant locations. We now discuss the problem of finding end effector configurations on frame $\mathbf{M}_i$ given plant positions for all constraints that are active from frames $\mathbf{M}_{i-L_2}$ to $\mathbf{M}_{i+L_2}$, where L_2 is a user-defined constant.

A foot is called *doubly constrained* if both the heel and ball are planted, *singly constrained* if only one of these joints are planted, and *unconstrained* otherwise. Similar definitions are used for hands. We now consider each of these scenarios for a given end effector E.

- E **is doubly constrained.** Let J_1 and J_2 be the joints that are to be planted, $\mathbf{o}_j$ be J_j's offset in $E's$ local coordinate system, and $\mathbf{c_j}$ be J_j's plant location (Figure 3.4). It is assumed that $\|\mathbf{o_2} - \mathbf{o_1}\| = \|\mathbf{c_2} - \mathbf{c_1}\|$. Also, let $\mathbf{p_i}$ and $\mathbf{Q_i}$ be the original global position and orientation of E on $\mathbf{M}_i$, and let $\mathbf{p'_i}$ and $\mathbf{Q'_i}$ be the (as yet undetermined) adjusted global position and orientation. When J_1 and J_2 are at their constraint locations, E is still free to rotate about the vector $\mathbf{c_2} - \mathbf{c_1}$. This degree of freedom is called the *roll*. If desired, the roll could be set explicitly, perhaps to orient a foot flat on the ground. Alternatively, the roll can be chosen

to minimize the difference between $\mathbf{Q_i}$ and $\mathbf{Q'_i}$. This can be done by finding the smallest rotation that, when applied to $\mathbf{Q_i}(\mathbf{o_2} - \mathbf{o_1})$, orients it in the same direction as $\mathbf{c_2} - \mathbf{c_1}$, where the magnitude of a rotation $\mathbf{q}$ is defined as the magnitude of its logarithmic map ($\|log(\mathbf{q})\|$). Define $\mathbf{rot}(\mathbf{a}, \mathbf{b})$ as a function that takes two vectors $\mathbf{a}$ and $\mathbf{b}$ and returns a rotation of $\cos^{-1}\left(\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}\right)$ degrees about $\frac{\mathbf{a}\times\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$. The closest orientation to $\mathbf{Q_i}$ that satisfies the plant constraints is then

$$\mathbf{Q'_i} = \mathbf{rot}\left(\mathbf{Q_i}\left(\mathbf{o_2} - \mathbf{o_1}\right), \mathbf{c_2} - \mathbf{c_1}\right)\mathbf{Q_i}. \tag{3.3}$$

With this orientation, the target global position of E is

$$\mathbf{p'_i} = \mathbf{c_1} - \mathbf{Q'_i o_1}. \tag{3.4}$$

- E **is singly constrained.** Let J_1 be the joint that is to be planted, with $\mathbf{o_1}$ and $\mathbf{c_1}$ defined as before. Given *any* orientation $\mathbf{Q'_i}$ for E, the plant constraint can be satisfied by setting $\mathbf{p'_i}$ as in Equation 3.4. Hence E could simply retain its original orientation and be translated by $\mathbf{c_1} - \mathbf{Q_i o_1}$. However, a discontinuity may exist if E is doubly constrained on $\mathbf{M_{i-1}}$ or $\mathbf{M_{i+1}}$, since in general E would have to be rotated to meet both plant constraints.

  To avoid this discontinuity, rotations applied to E on doubly constrained frames are blended off into nearby singly constrained frames. Let $\alpha(t)$ be a C^1 function such that $\alpha(0) = 1$, $\alpha(1) = 0$, $\frac{d\alpha}{dt}(0) = \frac{d\alpha}{dt}(1) = 0$. The unique cubic polynomial satisfying these conditions is

$$\alpha(t) = 2t^3 - 3t^2 + 1 \tag{3.5}$$

  The algorithm starts by looking forward and backward L_2 frames. For each direction, it finds the first frame where E is unconstrained or doubly constrained. If only singly constrained frames are encountered, or if an unconstrained frame is reached before any doubly constrained frames, then there is a *miss* in that direction; otherwise there is a *hit*. When both directions results in misses, then no changes are made, and $\mathbf{Q'_i} = \mathbf{Q_i}$. Otherwise there are three possibilities (Figure 3.5):

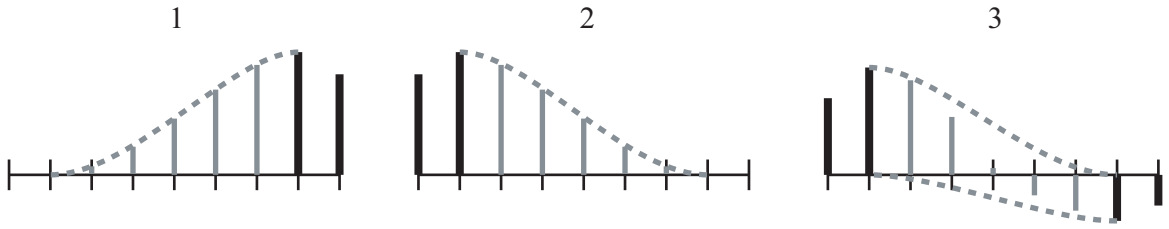

Figure 3.5: Blending off orientation adjustments from doubly constrained frames (thick black lines) to yield orientation adjustments on singly constrained frames (grey lines). There are three cases: 1) There is an adjustment in the forward direction but not the backward. 2) There is an adjustment in the backward direction but not the forward. 3) There are adjustments in both directions.

1. There is a hit going forward at frame $\mathbf{M_{i+k}}$, but a miss going backward. If the orientation adjustment applied to E on $\mathbf{M_{i+k}}$ is $\mathbf{\Delta_{Q_{i+k}}} = \mathbf{Q'_{i+k}Q_{i+k}^{-1}}$, then E's global orientation on the current frame is premultiplied by

$$\mathbf{\Delta_{Q_i}} = \mathbf{slerp}\left(\alpha\left(\frac{k}{L_2+1}\right), \mathbf{I_Q}, \mathbf{\Delta_{Q_{i+k}}}\right), \tag{3.6}$$

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

2. There is a hit going backward but not forward. This is handled in a manner similar to the previous case.

3. There are hits going forward at frame $\mathbf{M_{i+k_f}}$ and going backward at frame $\mathbf{M_{i-k_b}}$. In this case the orientation adjustments made to $\mathbf{M_{i+k_f}}$ and $\mathbf{M_{i-k_b}}$ are dissipated as in the previous two cases and combined. Let

$$\mathbf{\Delta_f} = \mathbf{slerp}\left(\alpha\left(\frac{k_f}{L_2+1}\right), \mathbf{I_Q}, \mathbf{\Delta_{Q_{i+k_f}}}\right) \tag{3.7}$$

   and

$$\mathbf{\Delta_b} = \mathbf{slerp}\left(\alpha\left(\frac{k_b}{L_2+1}\right), \mathbf{I_Q}, \mathbf{\Delta_{Q_{i-k_b}}}\right). \tag{3.8}$$

   Then E's global orientation on the current frame is premultiplied by

$$\mathbf{\Delta_{Q_i}} = \mathbf{slerp}\left(\alpha\left(\frac{k_b}{k_b+k_f}\right), \mathbf{\Delta_b}, \mathbf{\Delta_f}\right). \tag{3.9}$$

- E **is unconstrained.** Even though there are no constraints to satisfy, changes must still in general be made to E's configuration, since discontinuities will occur if plant constraints affecting E exist on $\mathbf{M_{i-1}}$ or $\mathbf{M_{i+1}}$. A discussion of this case is deferred until Section 3.2.6, which addresses the general problem of eliminating discontinuities when skeletal parameters switch from being constrained to being completely unconstrained.

3.2.4 Root Placement

The target position of a constrained end effector may be sufficiently far away that it cannot be reached even when the limb containing that end effector is at full extension. This problem can be corrected by adjusting the root position so the limb is closer to the target. Let $\mathbf{o_B}$ be the positional offset of the base of a limb (i.e., the shoulder/hip) from the root on the current frame. Also, let $\mathbf{p_R}$ be the root position, $\mathbf{p_T}$ be the target position of the end effector, and l_{max} be the length of the limb at full extension. As shown in Figure 3.6, $\mathbf{p_T}$ is reachable only if $\|\mathbf{p_T} - (\mathbf{p_R} + \mathbf{o_B})\| \leq l_{max}$ — that is, only if the root is inside a sphere of radius l_{max} centered at $\mathbf{p_T} - \mathbf{o_B}$. More generally, if n end effectors are constrained, then the root must be within the intersection of n spheres. If this is not true, then the root must be projected onto the surface of this intersection region. We perform this projection with the algorithm of Shin et al. [83], which involves a simple sequence of projections onto spheres, intersections of two spheres (which are circles), and intersections of three spheres (which are sets of discrete points).

When constraints switch on and off, the sphere intersection region changes discontinuously. If the root is projected onto the intersection region independently on each frame, the root trajectory can therefore also become discontinuous (Figure 3.7). Even small discontinuities can be quite distracting, since movement of the root affects the entire body. At the same time, attempting to generate smooth root displacements subject to sphere-interiority constraints is a non-linearly constrained variational problem which is difficult to solve efficiently. Our algorithm instead computes root positions independently on each frame and then filters these positions with a gaussian kernel of width $2L_3 + 1$, where L_3 is a user-defined constant (note that this requires knowing the target

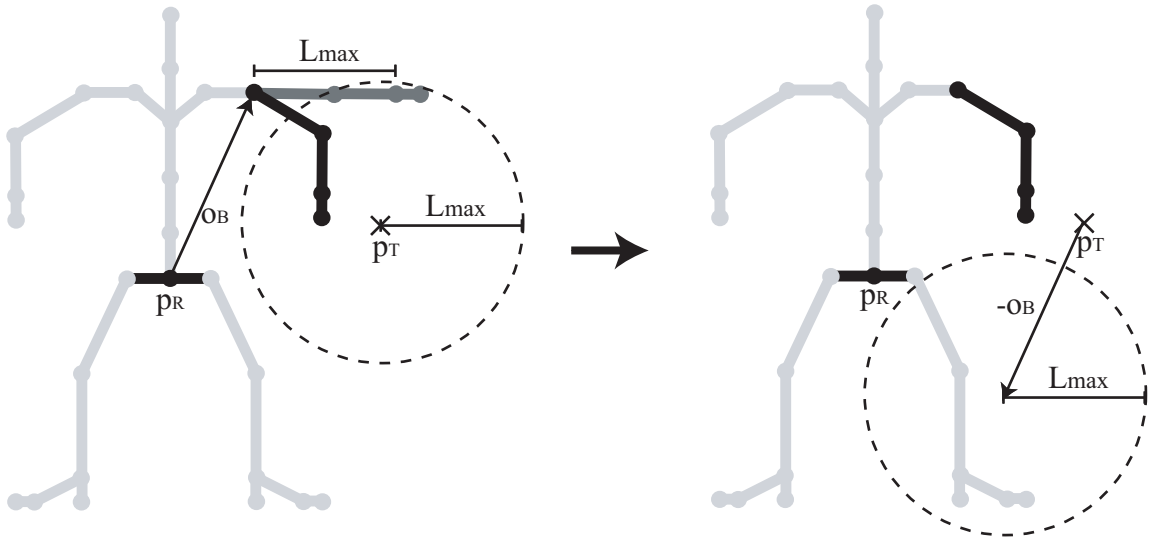

Figure 3.6: Left: For $\mathbf{p_T}$ to be reachable, the base of the limb must be within a sphere centered at $\mathbf{p_T}$ whose radius is l_{max}. **Right:** Equivalently, the root must be in a sphere of radius l_{max} centered at $\mathbf{p_T} - \mathbf{o_B}$

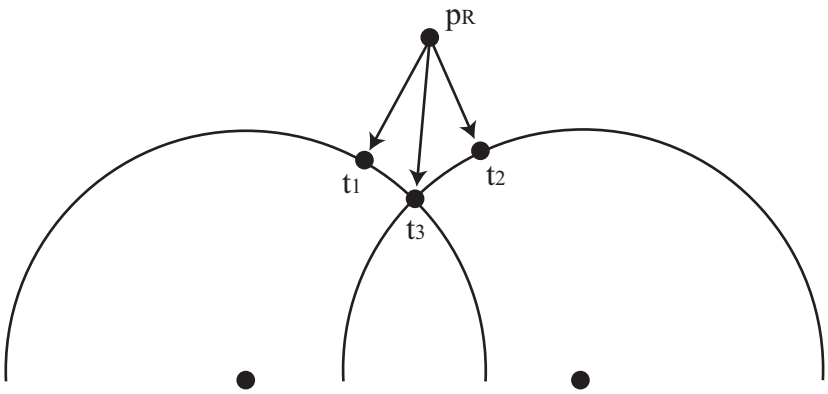

Figure 3.7: A constraint on one end effector causes the root to move to $\mathbf{t_1}$. If a second end effector were constrained instead, the root would project to $\mathbf{t_2}$. If both were constrained, the root would project to $\mathbf{t_3}$.

configurations of all constrained end effectors on frames $\mathbf{M_{i-L_3}}$ through $\mathbf{M_{i+L_3}}$). While this filtering ensures that the root path is smooth, it may also leave the root outside of the sphere intersection on some frames. However, this discrepancy is likely to be small, and end effector positions can still be exactly reached by stretching the limbs.

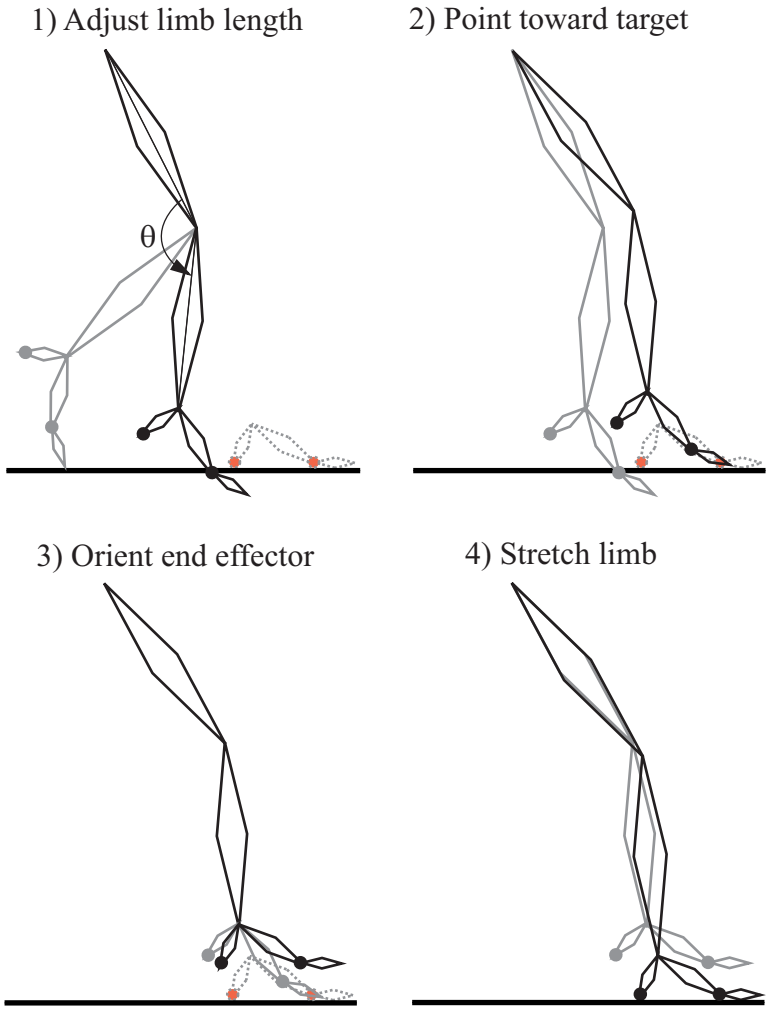

Figure 3.8: Steps in our single-limb IK algorithm.

3.2.5 Meeting the Target End Effector Configuration

Once the root position has been fixed, each constrained limb can be independently adjusted to place its end effector in the target configuration. This is accomplished with a modified version of a standard single-limb IK algorithm [89, 54] (see Figure 3.8). We first discuss the original algorithm for the case of the leg; the arm is handled similarly. Let the global positions of the hip and ankle on frame $\mathbf{M_i}$ respectively be $\mathbf{p_H}$ and $\mathbf{p_A}$, and let the target ankle position be $\mathbf{p_T}$. The first step is to rotate the knee so $\|\mathbf{p_A} - \mathbf{p_H}\| = \|\mathbf{p_T} - \mathbf{p_H}\|$. In general, the plane of rotation for the knee will not be the same as the plane defined by the thigh and shin. If l_1 and l_2 denote the length of the

upper and lower leg bones and l_1' and l_2' denote the lengths of these bones when projected onto the plane of rotation, it can be shown that the desired knee angle θ is

$$\theta = \arccos\left(\frac{l_1^2 + l_2^2 + 2\sqrt{l_1^2 - {l_1'}^2}\sqrt{l_2^2 - {l_2'}^2} - \|\mathbf{p_H} - \mathbf{p_T}\|^2}{2l_1'l_2'}\right), \tag{3.10}$$

where the solution is chosen so $0 \leq \theta \leq \pi$. See Lee and Shin [54] for a concise derivation. Once the knee has been adjusted, the ankle is in a new position $\mathbf{p}'_\mathbf{A}$. To move the ankle to the target position, the hip is next rotated so $(\mathbf{p}'_\mathbf{A} - \mathbf{p_H})$ is oriented in the same direction as $(\mathbf{p}'_\mathbf{T} - \mathbf{p_H})$, using the $\mathbf{rot}$ function defined in Section 3.2.3. Finally, the ankle is rotated to meet the target orientation.

Since the limb has 7 DOF and the target end effector configuration only specifies 6 constraints (3 for position, 3 for orientation), there is a remaining DOF that can be varied without breaking these constraints. This DOF corresponds to a rotation $\mathbf{Q}_\phi$ of the hip by ϕ degrees about the unit axis $\hat{\mathbf{n}} = \frac{\mathbf{p_T} - \mathbf{p_H}}{\|\mathbf{p_T} - \mathbf{p_H}\|}$, followed by a compensating rotation of the ankle. There are several ways of exploiting this extra DOF, such as enforcing joint limits on the hip and ankle or placing the knee as close as possible to a user-defined position [89]. Our algorithm adjusts this DOF so the ankle and hip are closer to their original orientations relative to their parent coordinate systems. Specifically, note that when $\phi = 0$ the hip is rotated as little as possible, and for some value $\phi = \phi_0$ the ankle is rotated as little as possible. A reasonable value of ϕ is then $\lambda\phi_0$ for some weight $\lambda \in [0, 1]$; in our implementation we use $\lambda = 0.5$.

The only remaining problem is to compute ϕ_0. When $\phi = 0$, let $\mathbf{\Delta Q}$ be the rotation that must be applied to the ankle so it is in the target orientation $\mathbf{Q}'_\mathbf{A}$. If the quaternion coordinates of $\mathbf{\Delta Q}$ are written as $(w, \mathbf{v})$, where w is a scalar and $\mathbf{v}$ is a 3-vector, then

$$\phi_0 = \arctan\left(\frac{w}{\hat{\mathbf{n}} \cdot \mathbf{v}}\right) \pm \pi, \tag{3.11}$$

where the sign is chosen to maximize the dot product $\mathbf{\Delta Q} \cdot \mathbf{Q}_\phi$ (treating $\mathbf{\Delta Q}$ and $\mathbf{Q}_\phi$ as ordinary 4-vectors). See Shin et al. [83] for a detailed derivation.

If the limb parameters and target end effector configurations vary continuously, then this algorithm produces a continuous sequence of adjustments. However, when the limb is nearly straight

a small change in target position can require a much larger change in knee/elbow angle. The is because limb length $L = \|\mathbf{p_A} - \mathbf{p_H}\|$ is a nonlinear function of knee/elbow angle: by the law of cosines,

$$L^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos\theta, \tag{3.12}$$

and differentiating this equation yields

$$\frac{dL}{d\theta} = \frac{l_1 l_2 \sin\theta}{L}, \tag{3.13}$$

which drops to 0 as $\theta \to \pi$. The amount that θ must be changed to produce a given change in limb length therefore grows dramatically as the starting limb length approaches its maximum value. For example, Figure 3.9 shows the knee/elbow angle adjustment necessary to extend the limb by 1% of its maximum length, as a function of the current length of the limb. The sharp rise in the graph manifests itself as an unnaturally fast extension or contraction of the knee/elbow, which we call a "pop". Even a minor pop can stand out because it affects the shape of the entire limb, and pops occur quite often because in many common motions (e.g., walking, reaching, or any motion where the character is standing) some of the limbs spend a great deal of time near full extension.

To eliminate popping artifacts, the IK algorithm is modified to limit knee/elbow rotation when the limb is near full extension. We refer to this as knee/elbow damping. Let $\rho \in (0, \pi)$ and define $f(x)$ as

$$\begin{array}{rcll} f(x) & = & 1, & x < \rho \\ & & \alpha\left(\frac{x-\rho}{\pi-\rho}\right) & \rho \le x < \pi \\ & & 0, & x \ge \pi \end{array} \tag{3.14}$$

where α is defined as in Equation 3.5. Then if the original knee/elbow angle is θ_0 and the adjustment to it is Δ_θ, the final angle θ is

$$\theta = \theta_0 + \int_{\theta_0}^{\theta_0 + \Delta_\theta} f(x)dx. \tag{3.15}$$

In our experiments we set $\rho = 135^\circ$, at which point the limb length is about 85% of its maximum value.

As a result of knee/elbow damping, the end effector may not able to reach its target position. This can be corrected by adjusting the length of the limb. If the limb length is L after rotating

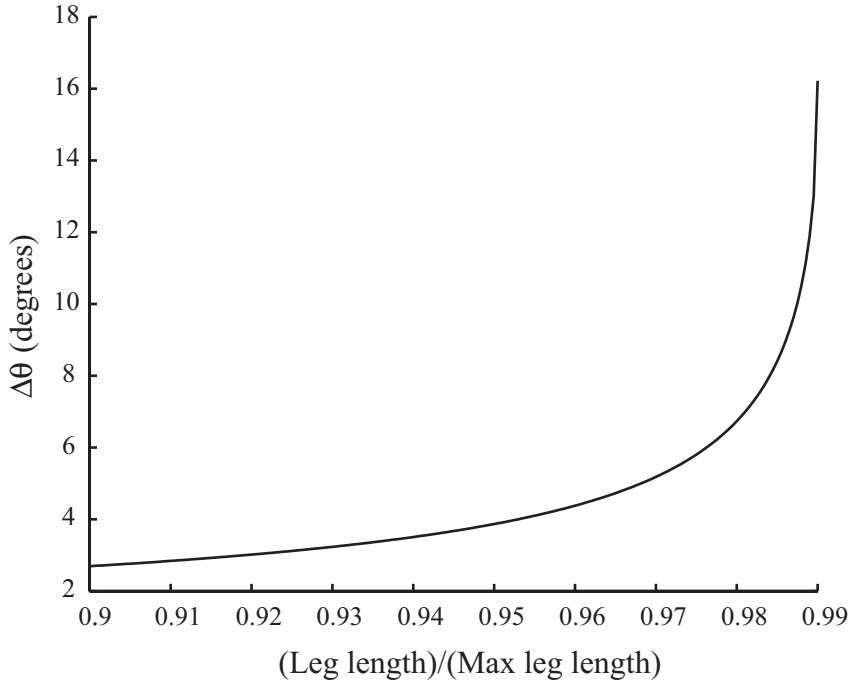

Figure 3.9: The adjustment in knee/elbow angle necessary to extend a limb by 1% of its maximum length versus the current length of the limb. Limb length is expressed as a percentage of its maximum length, and $l_1 = l_2$.

the knee/elbow and the distance from the target end effector position to the limb base is L', then both bone offsets are scaled by $\frac{L'}{L}$. This allows the end effector to be placed exactly at its target position. Note that as long as the limb is not perfectly straight, the distance from the limb base to the end effector is only affected by a fraction of the length changes applied to the bones. However, since bone lengths are adjusted only when the limb is nearly straight, almost all of the length adjustment goes directly into changing the limb length, and so usually only small amounts of scaling are necessary (no more than 3% in our experiments). Moreover, in our experience these length changes are considerably less distracting than knee/elbow popping.

3.2.6 Final Processing

The methods of Section 3.2.3–3.2.5 ensure that only smooth changes are made to a given DOF as long as it is affected by at least one constraint, but discontinuities can still occur when a DOF

switches from being constrained to being completely unconstrained or vice-versa. These discontinuities can be eliminated by blending off adjustments made to constrained DOFs into frames where they are unconstrained. We assume that all constraints have been enforced in the neighborhood $\mathbf{M_{i-L_4}}$ to $\mathbf{M_{i+L_4}}$, where L_4 is a user-defined constant. For each skeletal parameter on frame $\mathbf{M_i}$ that is not influenced by any constraints, the algorithm scans forwards and backwards L_4 frames until a frame is encountered where that parameter was adjusted to enforce a constraint. If there are no such frames in either direction, then that parameter is left unchanged. Otherwise, that parameter is adjusted as in Section 3.2.3. If the parameter is a root position or a limb scale factor, then $\mathbf{slerp}$ in Equations 3.6–3.9 is replaced by linear interpolation.

So far we have only discussed plant constraints, but it is quite common to also require each end effector to stay above a ground plane. This can be ensured with two final post-processing steps. First, each end effector with no plant constraints on $\mathbf{M_i}$ is translated vertically by the smallest amount necessary to ensure that each heel, ball, wrist, and fingertips is above or on the floor. Each end effector with one plant constraint is rotated about the plant location by the smallest amount necessary to ensure that the second attached joint is above or on the floor. The IK algorithm of Section 3.2.5 is then used to adjust the limbs accordingly. Since the motion is continuous prior to this adjustment and the floor itself is a continuous surface, these adjustments are also continuous.

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 toward the top of the foot. Our algorithm applies the smallest non-negative rotation that places each toe above the floor. Let $\hat{\mathbf{n}}$ be the unit 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 $\hat{\mathbf{n}}$. Defining

$$\kappa = \sqrt{(\hat{n}_x^2 + \hat{n}_z^2)(\|\mathbf{v_T}\|^2 - c^2)}, \tag{3.16}$$

the solution to this system of equations is

$$\mathbf{v'_T} = (\frac{-c\hat{n}_y \mp \hat{n}_z\kappa}{\hat{n}_x^2 + \hat{n}_z^2}, c, \frac{-c\hat{n}_z \pm \hat{n}_z\kappa}{\hat{n}_x^2 + \hat{n}_z^2}), \tag{3.17}$$

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

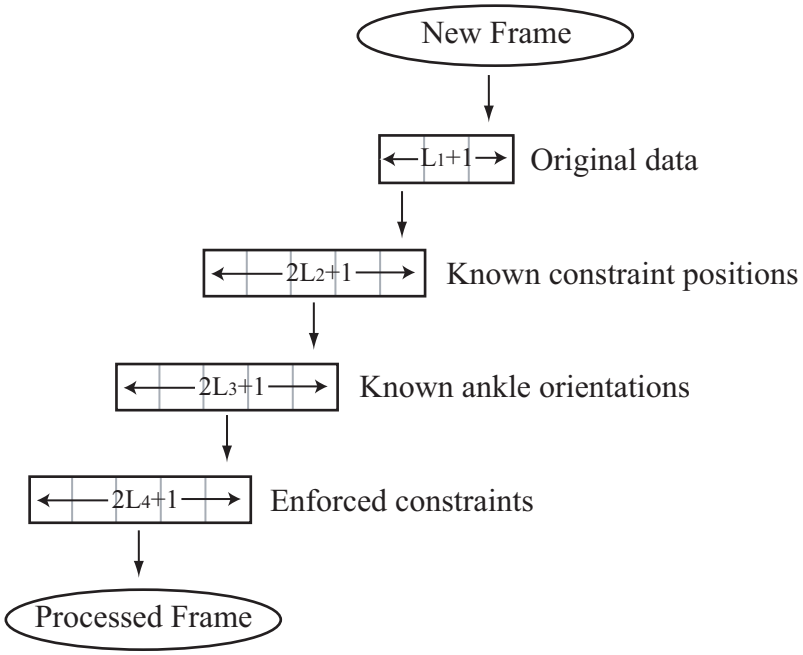

Figure 3.10: Diagram of an efficient online implementation. Four buffers are maintained to store original frame data, end effector configurations, frames with enforced constraints, and fully processed frames. Each buffer contains all the information necessary to create a single entry in the next buffer. Each time a new frame enters the top buffer, a processed frame is extracted from the bottom buffer.

3.2.7 Efficient Implementation

As shown in Figure 3.2, 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 3.10, 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 buffer and passed into a second buffer of length $2L_2 + 1$. In this buffer the ankle configuration for the central frame can be calculated using the methods of Section 3.2.3. 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 3.2.4 and then the leg parameters are adjusted according to Section 3.2.5. 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 3.2.6. 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.

3.3 Results

This section present results of some experiments run on an implementation of our constraint enforcement algorithm. We set $L_1 = L_2 = L_3 = L_4 = \frac{1}{4}s$ and $\rho = 135°$. Each experiment was run on a machine with a 1.3GHz Athlon processor. Without optimizing the code beyond what was described in Section 3.2.7, the average processing time for a frame with constraints on each end effector was 1.03ms, for a frame rate of about 1000fps. In general, the average time needed to process a frame was proportional to the average number of constrained end effectors. The maximum amount that a limb was stretched or shrunk in our experiments was 3% of its original length at full extension, and the average amount of length change was less than 1%. For comparison, in a series of perceptual experiments involving two-link chains, Harrison et al. [37] found that length changes below 2.7% are typically undetectable even to viewers who are focused on finding them, and considerably larger changes can go unnoticed if the viewer's attention is focused elsewhere.

We first compared our algorithm with one in Kaydara's FILMBOX 3, a popular software package for processing motion capture data. Data from an optical motion capture system was fit to a skeleton using both FILMBOX's default settings and one with the "reach-feet" option activated, which tracks foot markers to better preserve footplants. We then annotated the feet of the default motion with plant constraints and applied our end effector cleanup algorithm. While our algorithm exactly satisfied plant constraints, some footskate remained in FILMBOX's motion, possibly due to errors in marker tracking and skeleton fitting. Figure 3.11 illustrates this by showing the height of the right ball as a function of time for each version of the motion. Also, FILMBOX's motion

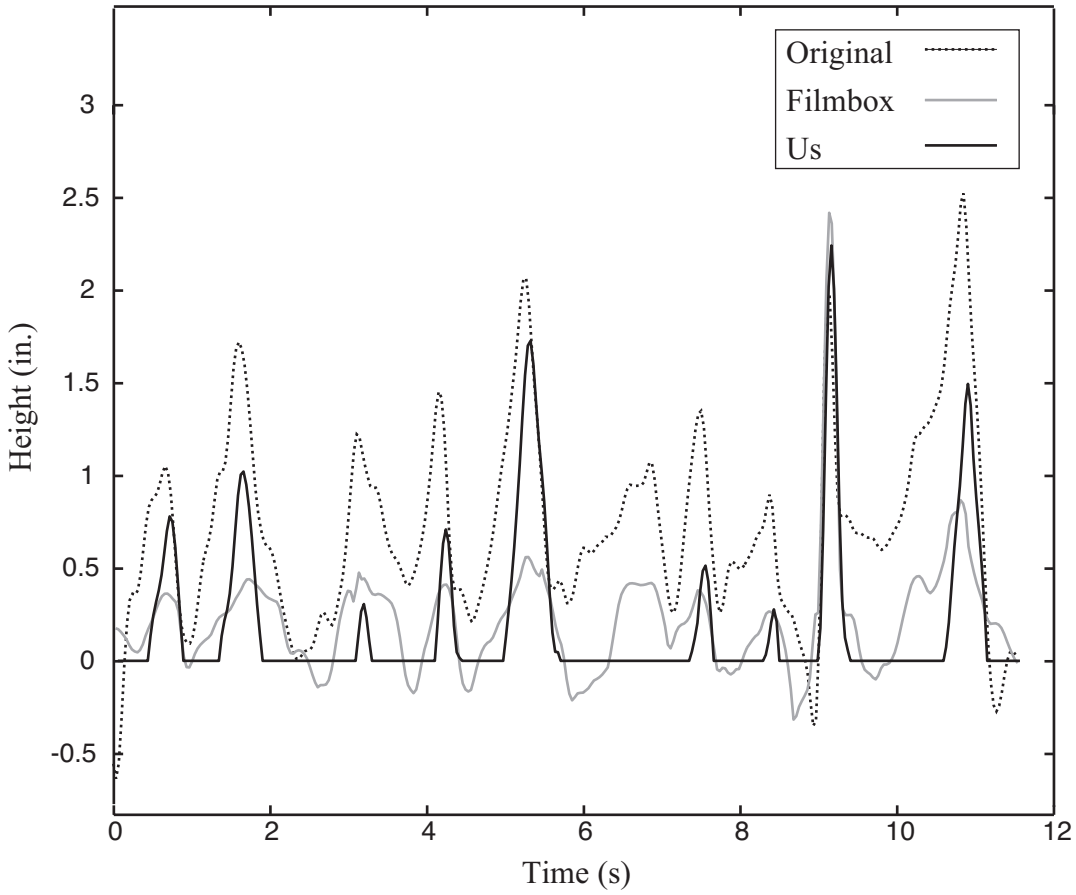

Figure 3.11: The height of the right ball in a segment of motion where the actor turned around in place several times. The dotted plot results from skeletal data generated using FILMBOX 3.0's default settings for fitting marker data to a skeleton. The grey plot is based on FILMBOX's "reach-feet" mode, which executes their method for eliminating artifacts like footskate. The black plot shows the results of applying our algorithm to the default FILMBOX fit.

exhibited some knee popping, whereas our motions was free of such artifacts. We note that our comparative success is in part due to the fact that our algorithm was supplied with the exact time intervals when footplants were to occur, whereas FILMBOX relied on measured marker positions.

We next tested our algorithm in the context of several motion editing applications. Videos are available at http://www.cs.wisc.edu/graphics/Gallery/Kovar/Cleanup/.

1. **Blend Postprocessing.** Existing methods for creating blends (typically, transitions and interpolations) [69, 96, 76] can fail to preserve important kinematic constraints, and the same

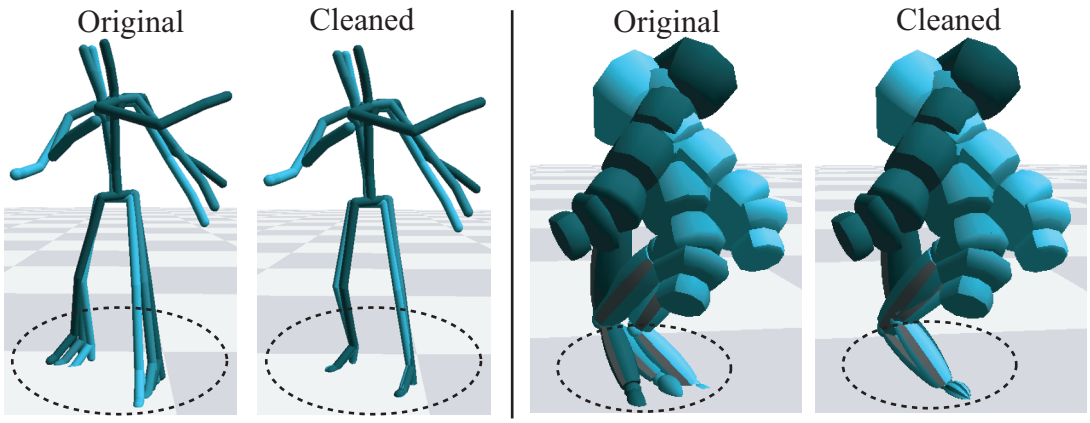

Figure 3.12: Left: A transition from walking to snatching causes the character's feet to splay apart when they should be fixed on the ground. **Right:** An interpolation of two one-handed handsprings yields a motion where the character's hand unnaturally slides across the floor. For clarity, only the torso and support arm are shown.

is true of the methods developed in this dissertation. We used our algorithm to enforce plant constraints in several transitions and interpolations; Figure 3.12 shows results for a transition from walking to snatching an object off a table and for an interpolation of two one-handed handsprings. This application of end effector cleanup will be used extensively in later chapters.

2. **Motion Salvage.** We received a motion from a professional 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 salvaged this motion by using our algorithm to eliminate the considerable footskate without adding any new visible artifacts. Figure 3.13 shows the height of the right ball on the damaged motion before and after applying our algorithm. We also applied our algorithm to a cartwheel motion where the hands penetrated the ground and failed to remain stationary when they were bearing the character's weight (Figure 3.13).

3. **Retargeting.** *Motion retargeting* is the transfer of an existing motion to a character with a body different than the original performer. In general, motion retargeting can be quite challenging, particularly if one wants to adapt a human motion to an object with different articulations [31]. In simpler cases, such as when the performer and character have the same

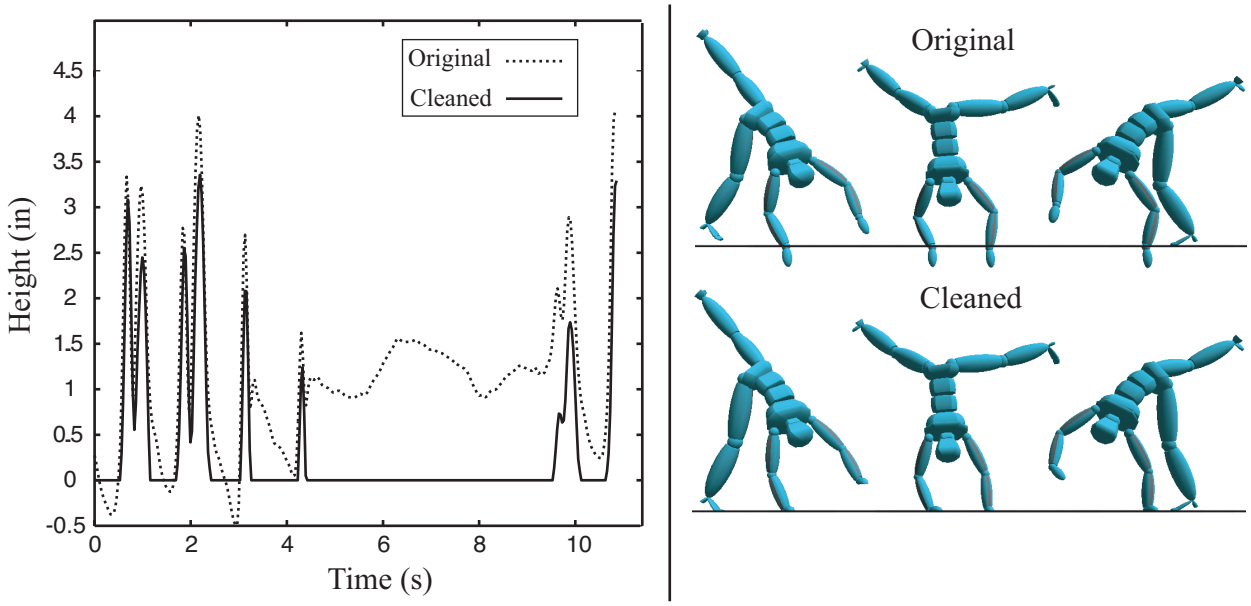

Figure 3.13: Left: The height of the right ball before and after applying our end effector cleanup algorithm. The character walked a few steps, stopped for a few seconds, and then continued walking. Note that the original motion has no clear footplants, even when the character stands still (4.2s-8.3s). **Right:** Before end effector cleanup, the character's hands penetrate the ground; afterward, they remain properly planted.

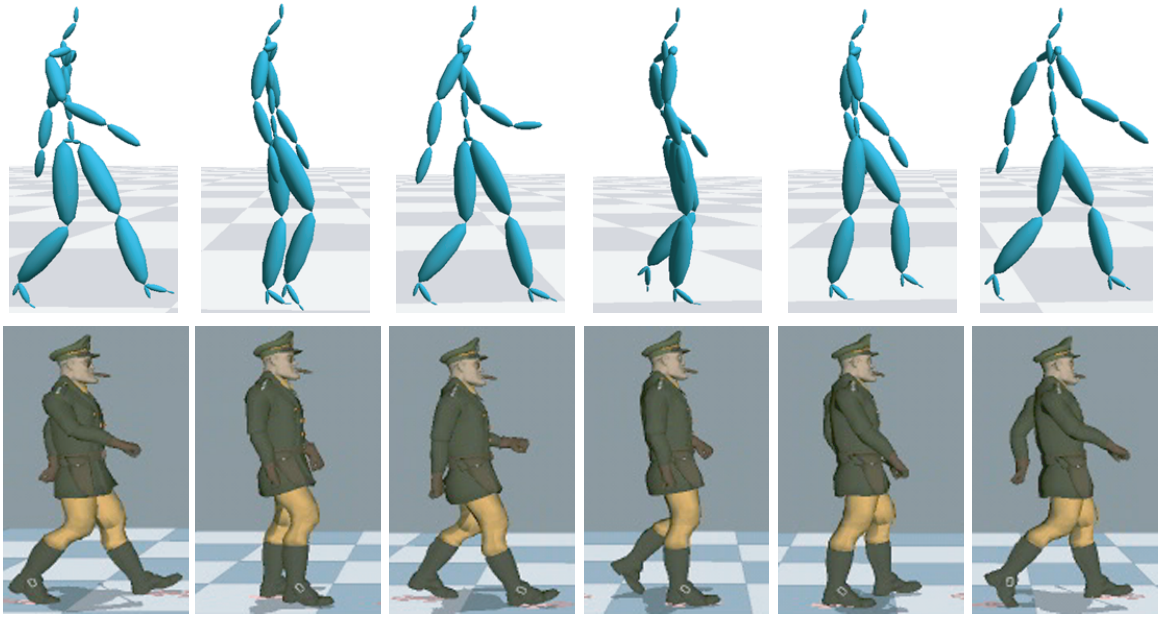

Figure 3.14: The skeleton for a walking motion and character mesh that it was retargeted to.

body topology but different proportions, our end effector cleanup algorithm can suffice. We retargeted a motion captured walk onto a cartoon character by scaling the root translation data according to differences in bone lengths and then using our end effector cleanup algorithm to satisfy footplant constraints. Figure 3.14 shows the original skeleton and the character mesh.

4. **Path Editing.** We have used our end effector cleanup algorithm to enforce footplant constraints after applying the path editing operation developed by Gleicher [32]. Our algorithm is a simple, efficient alternative to the spacetime optimization used in the original paper.

3.4 Discussion

This chapter has introduced a simple and efficient algorithm for adjusting a motion to satisfy kinematic constraints on end effectors, while at the same time ensuring that only smooth changes are made. This algorithm is not only useful on its own, but also can be employed as a postprocessing step for other algorithms that might fail to preserve kinematic constraints. In particular, this algorithm is a crucial component of the motion models developed in Chapters 4–6: by automatically enforcing end effector constraints, the range of realistic motion that can be produced by graph-based and blending-based models is greatly increased.

The methods in this chapter only guarantee C_0 continuity; it is possible that limb velocities will change discontinuously. We believe this is reasonable since in some cases (e.g., sudden impacts) velocities truly are effectively discontinuous. We have experimented with using C_1 displacement maps [45] for the interpolation methods of Sections 3.2.3 and 3.2.6, based on finite-difference estimates of the derivatives of parameter adjustments, but we did not notice a qualitative difference in the resulting motion.

The primary disadvantage of our algorithm is that it is *not* a general IK solver; we limit the types of adjustments that can be made in order to allow an analytic solution (for example, spine DOFs are not changed, and we do not consider joint limits or self-intersection). This was a conscious decision made to accommodate automatic (i.e., unsupervised) constraint enforcement, which is needed in

subsequent chapters. Specifically, since our motion models are limited to synthesizing motions that are reasonably similar to the original examples, constraint violations in synthesized motion are likely to be small. It is therefore more desirable to have a robust algorithm that will always enforce constraints smoothly than a semi-stable algorithm that exercises every available degree of freedom. For more general IK tasks, however, algorithms employing constrained nonlinear optimization methods are more appropriate [101].

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 previous work has used strictly rigid skeletons. More generally, it would be useful to understand how different artifacts — foot-skate, over-stretched limbs, sudden changes in joint orientation, and so on — may be balanced so as to produce a desired change while minimizing visual disturbance. Although recent perceptual studies [74, 37] have shed some light on this issue, considerably more work is needed in order to have a principled understanding of how different kinds of adjustments affect a viewer's evaluation of a motion.