PhysIK: Physically plausible and intuitive keyframing

Amir H. Rabbani* McGill University

Paul G. Kry† McGill University

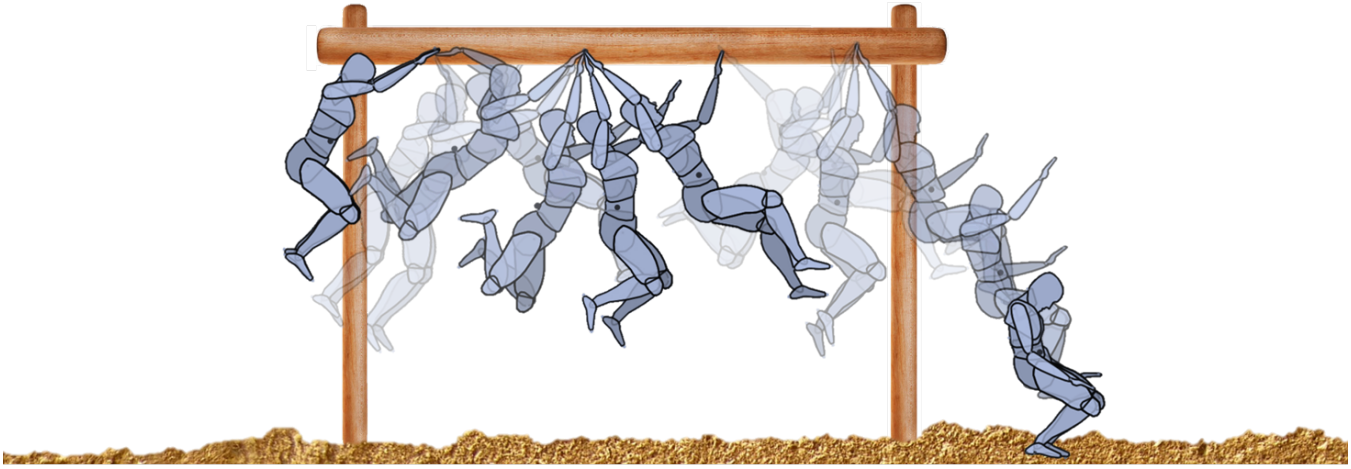

Figure 1: Creating successive swing motions followed by landing via constraining the interpolated pose keyframes with seven physcially plausible pendulum trajectory templates (swing) as well as a ballistic path at the end of the motion. The trajectory templates control the center of mass of the character. Hand and feet pinning constraints help the character to achieve natural grasps onto the monkey bar as well as a plausible landing posture.

Abstract

We present an approach for animating characters using inverse kinematics (IK) handles that allows for intuitive keyframing of physically plausible motion. Specifically, we extend traditional IK and keyframing to include center-of-mass (CM) and inertia handles along with physically based templates to help an animator produce trajectories that respect physics during dynamic activities, such as swinging, stepping, and jumping. Animators can easily control both posture and physics-based quantities (inertia shape, CM position, and linear momentum) when building motions from scratch, but also have complete freedom to create exaggerated or impossible motions. We present results for a variety of planar characters of different morphologies.

Index Terms: I.3.7 [Three-Dimensional Graphics and Realism]: Animation

1 Introduction

Inverse kinematics (IK) is an important tool used by animators to set a character's pose with intuitive handles, such as the positions of hands and feet, as opposed to creating animation by keyframing the angles of the skeletal joints directly.

The current state-of-the-art IK software solutions are mature and highly optimized. Additionally, to ease the creation of natural humanoid motion, IK techniques specific to humanoids (for instance, in Maya) can be used to produce very life-like virtual human animation. Furthermore, data-driven IK solutions are an excellent way to ensure that an animation exhibits natural postures and motion. However, data and heuristics used for humanoids will not generally apply to characters of different morphologies, such as quadrupeds or imaginary creatures. Thus, there is also a need for general tools and methods that can help animators produce plausible motion for all characters.

We address specifically the aspect of physical plausibility in developing a new tool that assists an animator in controlling momentum and inertia of a character while authoring new motions from scratch. The challenge is that most IK solvers use joint-space approaches to parameterize control in terms of individual joint actions. As a result, there are difficulties in formulating high-level tasks, like regulating momentum, as joint actions combine in a highly non-linear way. Hence, animators may struggle to produce desired results. Furthermore, any changes to the morphology of the character will require retuning of the keyframed trajectories. In our system, in contrast, adding a backpack, large boots, or a tail, need not have an adverse effect on the momentum control of a currently keyframed trajectory.

In this paper, we present a collection of techniques that allows animators to easily control the center-of-mass (CM) and inertia of a character, but also gives them complete freedom to create exaggerated or impossible motions. The CM and inertia become IK handles and can be animated in the same way that hands and feet are constrained in existing IK solvers, but we also couple this with a set of feature trajectory templates that help an animator create a variety of physically plausible trajectories.

For instance, Figure 1 highlights a simple application of our technique to keyframe a monkey bar example. In this case, the animator selects a pendulum motion template trajectory for the swing phase and a projectile motion template for the flight phase before landing. The pendulum template has the constraint that the CM trajectory should remain on a circular path whose center is set to be the

*e-mail: amir.rabbani@mail.mcgill.ca

†e-mail:kry@cs.mcgill.ca

pivot of the pendulum. Likewise the ballistic motion has the constraint that the linear momentum of the character is only affected by gravity, and thus the center of mass will undergo a constant vertical acceleration of -9.8 ms^{-2}. The animator can adjust the arc of the pendulum and the jump through a variety of controls such as the pivot position, maximum height, initial velocity, initial position, end position, and time of flight. Because these parameters are coupled, our tool automatically adjusts the parameters left unconstrained by the animator in order to satisfy the constraints of the template trajectories. Edits to the character's pose can be made during the swing and flight phases while preserving the desired CM trajectory.

To achieve natural grasping of the monkey bar as well as to set the stance at landing, we set appropriate pinning constraints for the hands and feet (heel and toes). We regulate the weights of these pins through keyframes to produce a graceful transition between swings, and similarly, into the final landing phase. Finally, the center of mass is smoothly animated with Hermite curves to prepare for transitions between the swing motions, and similarly to stop the character in landing. See Sections 5 and 6 for other examples that demonstrate the different template trajectories that we provide in our interface.

Creating physically plausible character motion through keyframing is a challenge for animators, and we see the results of our work being relevant to character animators everywhere. Four components make up the main contributions of our work:

- center-of-mass and rotational inertia IK handles;
- regulation of IK constraint compliance in a global solver;
- arbitrary re-rooting for simple sub-structure pose control;
- handle trajectory generation from physics-based templates.

While all demonstrations in this paper are 2D, the techniques also apply to 3D with minor adjustments. The keyframing interface and the physic-based templates make no assumptions about the character dimension. However, a few modifications to the solver will be necessary for a transition to 3D. To open up avenue for further extensions of our work to 3D, we will first show how to compute control quantities in 3D and then explain how to adapt them to a 2D inverse kinematics solver.

Note that we do not treat the case of angular momentum conservation, but it is natural to extend our current framework to handle this case with adding a feature template that takes into account the coupling of the inertia and the angular velocity keyframes. We discuss this in more length at the end of the paper.

2 Related Work

Inverse kinematics is a problem of great importance in both computer graphics and robotics. For small systems, analytical solutions exist and can be very practical [9], but for complex articulated structures with large numbers of joints it is necessary to use iterative methods. An overview of methods to solve the inverse kinematics problem along with techniques to gracefully deal with singularities can be found in several good surveys [5, 7]. While a challenging problem, fast and robust solutions to the constrained inverse kinematics problem have been available for many years, with a vast amount of research addressing issues such as prioritized strategies for resolution of conflicting goals [2], motion editing applications [4], alternate mathematical formulations including non-linear programming [21], singularity-robust inversion [20], damped least squares [6], and fast solutions such as CCD and FABRIK [19, 1].

In most cases, the inverse kinematics problem is highly under constrained, and many solutions exist. While giving weight to a single natural rest pose can help, recent work has shown that motion data can also provide an effective way of selecting the position of redundant joints by appropriately weighting the least squares solution [10]. Latent variable models also prove useful for solving the inverse kinematics problem in a manner that preserves the style of example motion data [8]. More recently, latent variable models have been combined with prioritized motion constraints to achieve a broader range of goals [16].

While data-driven IK methods can produce very natural motion, it is still possible that poses will violate the condition necessary for stable balance. That is, a data driven IK solution does not explicitly control the center of mass to above the support polygon of the feet in a static pose. With the aim of producing physically plausible results that respect postural stability, inverse *kinetics* formulations include position control on the center-of-mass [3, 12]. This is straightforward to apply to any character provided that a mass can be assigned to the different geometries that make up the character.

This earlier work on inverse kinetics provides a critical tool for plausibly maintaining balance during standing and reaching, but does not directly address dynamic constraints such as momentum conservation during ballistic motion. In contrast, Shapiro and Lee [17] introduce an interactive system to let animators improve unrealistic motions in 3D animation by visualizing dynamic physical properties such as the center of mass trajectory and angular momentum during ballistic motion. These quantities are visualized to provide the animator with hint paths to let them modify the created motion such that it matches the physics-based path.

Other recent work presents a method to edit kinematic motion using momentum and force constraints [18], as well as a method that can directly deal with approximate secondary dynamics in combination with inverse kinematics constraints [11]. In comparison, our work is an interactive tool that helps the animator control important dynamic aspects of the motion through kinematic techniques. We not only provide the center of mass as a constraint within the inverse kinematics solve, but we also provide control of the magnitude of the character's inertia tensor, and more importantly, we provide a collection of handle template trajectories and tools that help an animator craft dynamic and physically plausible motions from scratch. Momentum and inertia are quantities that have been recognized by the robotics community as being important [13] for answering questions about balance, and Lee and Goswami have shown how to use the computation of the generalized mass inertia tensor of an articulated character for this purpose.

3 Handles for Physically Plausible Keyframing

In many IK problems, the objective is to control points at the end of limbs, such as a hand or foot. This can be solved by working with joints in isolated limbs. Hence, positioning of end points is often treated as a local IK problem. In contrast, there are cases where IK problem cannot be solved locally. For example, the CM is a function of the position of all limbs. In particular, changing the CM position becomes challenging in the presence of additional constraints at hands or feet. Thus, a solution that constrains the CM in addition to hands and feet is ideally best found with a global IK solver.

In order to provide the animator with tools to animate the rotational inertia and CM, we estimate the size and physical properties of the character's parts. We assume that this can be done automatically from the character's geometry. If the geometry is formed by set of closed meshes, then mass and rotational inertia of each segment can be computed from volume integrals with a reasonable material density. In the case of non-closed or more complex character geometries, we believe that it is not unreasonable to assume that the artist will also create a collection of simple collision proxies from which the mass and rotational inertia can be computed instead.

3.1 Derivation

In the following parts we identify the momentum related functions and show that we can use the gradient as a linear map in the control quantities. In order to derive equations for the inverse kinematics solver we need differential quantities of the desired control handles

with respect to the joints. That is, given a forward kinematics map $f : Q \to \mathbb{R}^n$ and a desired configuration $f_d \in \mathbb{R}^m$, we would like to solve the equation $f(\theta) = f_d$ for some $\theta \in Q$, where n and m vary based on the type of joints and the controlled features. This is a root finding problem, and it may have multiple solutions, a unique solution, or no solution at all, as discussed by Murray et al. [14]. Solutions to this mapping are obtained by iterative solvers that require differentiating the forward kinematic map with respect to the control quantities.

For an articulated character made up of many bodies, each body has a coordinate frame, and the position of the frame in world coordinates is a function of the position and orientation of the root along with the joint angles of the character. We assemble all parameters of a character's pose into a vector θ, including the root position and orientation, and thus we can think of orientation and position of each body as functions of θ.

3.1.1 Controlling the CM position

The CM position is the weighted average of the CM positions of all bodies. Suppose we would like to know this quantity in some world coordinate frame w. Given the position of the CM of a link, we can write this as a function of θ,

$$c(\theta) = \frac{1}{m_\sigma} \sum_b^N m_b {}^w r_b(\theta) \tag{1}$$

where r_b and m_b are the position and mass of body b respectively, N is the number of bodies, m_σ is the total mass and leading superscript w on r denotes that the quantity is expressed in world coordinates. We may want to control the CM of the character so that it stays in a given position, or follows a desired trajectory. For this we take partial derivative of Equation 1 with respect to θ

$$\frac{\partial c(\theta)}{\partial \theta} = \frac{1}{m_\sigma} \sum_b^N m_b \frac{\partial r_b(\theta)}{\partial \theta}. \tag{2}$$

Defining the terms for the CM and body Jacobians

$$\frac{\partial c(\theta)}{\partial \theta} = J_c \tag{3}$$

$$\frac{\partial r_b(\theta)}{\partial \theta} = J_b \tag{4}$$

we can rewrite Equations 2 as

$$J_c = \frac{1}{m_\sigma} \sum_b^N m_b J_b. \tag{5}$$

The jth block of J_b relates the change of joint angle j to the position of body b,

$$J_{bj} = \widehat{p_{bj}} R_{bj} \tag{6}$$

where p_{bj} and R_{bj} are the translational and rotational components of the transformation from joint coordinate frame j to body coordinate frame b, and $\wedge$ is a cross product operator in the form of a skew symmetric matrix for a 3×1 vector a:

$$\widehat{a} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \tag{7}$$

To adapt the center of mass handle to 2D we exclude the third dimension from the positional vectors, and the cross product operator will reduce to

$$\widehat{a} = \begin{bmatrix} a_y \\ -a_x \end{bmatrix}. \tag{8}$$

3.1.2 Controlling the rotational inertia

Total rotational inertia at the CM frame is the lower right 3×3 block of the generalized inertia tensor

$${}^c M = \begin{bmatrix} m_\sigma \mathbf{I} & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \tag{9}$$

where $\mathbf{I}$ is identity matrix. We obtain ${}^c M$ through summation of all body inertia matrices transformed to the CM coordinate frame

$${}^c M = {}^c_w Ad \sum_b^N {}^w_b Ad^T \, {}^b M \, {}^w_b Ad \tag{10}$$

where Ad is an *adjoint transformation* matrix [14]

$$Ad = \begin{bmatrix} R & \widehat{p} R \\ \mathbf{0} & R \end{bmatrix}. \tag{11}$$

Since there might be off-diagonal non-zero elements in I due to the correlation of the inertia axes, we perform singular value decomposition (SVD) on I to extract the rotational inertia elements along each axis. For a planar character the rotational inertia reduces to a single scalar quantity (as opposed to a 3×3 matrix in 3D), which corresponds to the element along the axis that is orthogonal to the plane.

In 2D we can think of the rotational inertia as a disk with a radius that is proportional to the square of off-plane scalar element. Hence the expression for computing the total inertia becomes

$$I(\theta) = \sum_b^N m_b (r_b(\theta) - c(\theta))^2 + I_b \tag{12}$$

where scalar I_b is the rotational inertia of body b. In order to compute the Jacobian we take partial derivative of Equation 12 with respect to θ

$$\frac{\partial I(\theta)}{\partial \theta} = 2 \sum_b^N m_b (r_b(\theta) - c(\theta))^T \left(\frac{\partial r_b(\theta)}{\partial \theta} - \frac{\partial c(\theta)}{\partial \theta} \right) \tag{13}$$

and replace the relevant terms from Equation 3 and Equation 4

$$J_r = 2 \sum_b^N m_b (r_b(\theta) - c(\theta))^T (J_b - J_c). \tag{14}$$

3.1.3 Linear and angular momentum

It is also useful to consider expressions for linear and angular momentum of the entire character. In the absence of contact, the angular momentum will be a conserved quantity, and linear momentum in the vertical direction will decrease at a constant rate due to gravity while linear momentum in the horizontal direction is conserved. Of note, the linear momentum is directly related to the CM velocity.

In 3-dimensional space, the velocity of a rigid body with respect to an inertial frame is written as a *twist*, and can be written as a 6 components vector in a given coordinate system. For instance, the twist of body b expressed in the coordinate frame attached to body b can be written as

$${}^b \zeta_b = \begin{bmatrix} v \\ \omega \end{bmatrix} \in \mathbb{R}^6. \tag{15}$$

Likewise, the momentum of body b in coordinate frame b is a 6 component vector that is linearly related to the body velocity

$${}^b \Phi_b = \begin{bmatrix} L \\ H \end{bmatrix} = {}^b M \, {}^b \zeta_b \tag{16}$$

where L and H are the linear and angular momentum vectors, and ${}^bM$ is the inertia tensor.

Just like twists, we use the adjoint transform (Equation 11) to change momentum from one coordinate to another, except that we use the inverse transpose. The total momentum of a multibody character, in coordinates at the CM of the character, is the sum of the momentum of all rigid bodies

$$ {}^c\Phi_\sigma = \sum_b^N {}^b_c Ad^{T\,b} M\, {}^b\zeta_b. \tag{17} $$

In two dimensions linear velocity v and linear momentum L become 2×1 vectors and angular velocity w and angular momentum H are scalar. We can likewise define the 2D adjoint transformation that transforms twists from coordinate frame a to coordinate frame b,

$$ {}^b_a Ad = \begin{bmatrix} R & \widehat{p} \\ \mathbf{0} & 1 \end{bmatrix} \tag{18} $$

where R becomes a 2×2 block and $\widehat{p}$ is the same as Equation 8. Let ${}^b\Gamma_b$ provide the twist of body b in coordinate frame b when multiplied by joint velocities $\dot{\theta}$. This is effectively the Jacobian of a rigid body's position, which is a function of the characters configuration θ. With this, we can then write an expression that relate the joint velocities to linear and angular momentum,

$$ {}^c\Phi_\sigma = \left(\sum_b^N {}^c_b Ad^{T\,b} M\, {}^b\Gamma_b \right) \dot{\theta} \tag{19} $$

from which the Jacobian is found as

$$ J_m = \sum_b^N {}^c_b Ad^{T\,b} M\, {}^b\Gamma_b. \tag{20} $$

The sum in Equation 19 interestingly contains the CM position Jacobian. That is, the upper two components of ${}^c\Phi_\sigma$ are the linear momentum of the entire character, which is same as the CM position Jacobian multiplied by the joint velocities. In future work we can use the expression for angular momentum in the bottom component to regulate the overall angular velocity of the character in a manner consistent with angular momentum conservation. Furthermore, while these quantities are derived for 2D planar characters, the derivations are very similar for 3D rigid motion.

4 Solving the Inverse Kinematics Problem

In this section we describe how to formulate and solve the inverse kinematics problem with a mixture of high and low priority constraints. While our solver is global in the sense that it uses full body configuration space to compute the solution, we also propose a method that allows for local edits to the character.

4.1 Physics-based solution

In order to solve for the joint angles that satisfy the physics-related control parameters we form a 6×1 vector with the desired handle quantities

$$ g_{des}(\theta) = \begin{bmatrix} I_{des} \\ c_{des} \\ \Phi_{des} \end{bmatrix}. \tag{21} $$

We also combine the three Jacobians that we computed so far into a weighted Jacobian matrix

$$ J_w = W \begin{bmatrix} J_r \\ J_c \\ J_m \end{bmatrix} \tag{22} $$

where W is a 6×6 diagonal user-specified weighting matrix in the form of

$$ W = \begin{bmatrix} [w_r]_{1\times 1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & [w_c]_{2\times 2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & [w_m]_{3\times 3} \end{bmatrix}. \tag{23} $$

The weighting matrix allows for additional control over the contribution of each momentum handle to the solution. Since the Jacobian matrix is not square, we use a pseudo-inverse method to compute the solution

$$ \Delta\theta = J_w^\dagger\, g_{des}(\theta) \tag{24} $$

where $J_w^\dagger$ is the pseudo-inverse of J_w and is computed by

$$ J_w^\dagger = {J_w}^T \left(J_w {J_w}^T \right)^{-1}. \tag{25} $$

Equation 24 provides a least square solution with minimal norm that remains robust as as long as singular configurations are avoided. We use an iterative method because our characters typically have many degrees of freedom (20 to 25) and the target configuration is defined by a few control quantities which makes it impossible to apply a closed-form IK solver.

The desired physical quantities are set via an automatic system where the handle values are computed through processing the keyframes of each handle track in the interface panel. Equivalently, the desired parameters can be set by the animator through a set of interactive tools.

4.2 Natural motion with prioritized solver

Although solving Equation 24 ensures physical plausibility of our character motion, it does not guarantee for the character to maintain a natural posture. In our solver we include three types of constraints to achieve a natural motion:

- pinning constraint that control the velocity of links;
- desired joint angles;
- out of motion-range joint angle corrections.

Dealing with large collections of constraints is of interest in designing complex motions, but it can also become a problem when constraints are infeasible or conflict with one another. One solution is to give physics-based constraints higher priority than the postural constraints, as they are of greater importance and can not be violated.

We use a prioritized solver that takes advantage of the nullspace of $J_w^\dagger$ to allow for secondary constraints. For any arbitrary vector γ we can expand Equation 24

$$ \Delta\theta = J_w^\dagger g_{des}(\theta) + (\mathbf{I} - J_w^\dagger J_w)\gamma \tag{26} $$

where $(\mathbf{I} - J_w^\dagger J_w)$ performs a projection onto the nullspace of $J_w^\dagger$. By suitably choosing γ we can achieve secondary effects such as satisfying the postural (soft) constraints, while $J_w^\dagger g_{des}(\theta)$ provides us with minimum error solutions to the physics-based (hard) constraints [5]. We directly follow the work of Yamane and Nakamura [20] for the computation of γ. Our work, however, differs from their approach as we treat the pose tracker as lower priority constraint and compute it in γ, whereas they treat it as a primary constraint.

The problem of having additional control quantities is that the resulting equations may not have exact solution due to conflict. Even though solving the IK system with a pseudoinverse yields a least square solution, infeasible solutions might be produced when the configuration is singular. Therefore singularity robust methods, such as the one proposed by Deo and Walker [7] need to be applied.

Algorithm 1 PhysIK algorithm

while running **do** ▷ Processing the keyframes
  $\theta \leftarrow$ GET CURRENT JOINTS STATE
  $C \leftarrow$ COMPUTE CM(θ)
  $I \leftarrow$ COMPUTE INERTIA(θ)
  $T \leftarrow$ GET CURRENT TIME
  $(Ks_1, Ks_2) \leftarrow$ GET POSE KEYFRAMES(T)
  $(Kp_1, Kp_2) \leftarrow$ GET PIN KEYFRAMES(T)
  $(Kc_1, Kc_2) \leftarrow$ GET CM KEYFRAMES(T)
  $(Ki_1, Ki_2) \leftarrow$ GET INERTIA KEYFRAMES(T)
  $\theta_{\text{des}} \leftarrow$ INTERPOLATE(Ks_1, Ks_2)
  $P_{\text{des}} \leftarrow$ INTERPOLATE(Kp_1, Kp_2)
  $C_{\text{des}} \leftarrow$ TRAJECTORY TEMPLATE(Kc_1, Kc_2)
  $I_{\text{des}} \leftarrow$ INTERPOLATE(Ki_1, Ki_2)
  while not converged **do** ▷ IK solver
    $J_c \leftarrow$ CM JACOBIAN(θ, C) ▷ Eq. 5
    $J_r \leftarrow$ ROTATIONAL INERTIA JACOBIAN(θ, I) ▷ Eq. 14
    $J_{\text{hard}} \leftarrow$ COMBINE(J_c, J_r) ▷ Eq. 22
    $g_{des}(\theta) \leftarrow$ SET DESIRED HANDLES$(C_{\text{des}}, I_{\text{des}})$ ▷ Eq. 21
    $\Delta\theta_h \leftarrow$ SOLVE$(g_{des}(\theta), J_{\text{hard}})$ ▷ Eq. 24
    $J_{\text{soft}} \leftarrow$ SOFT CONSTRAINT JACOBIAN$(\theta, P_{\text{des}}, \theta_{\text{des}})$
    $\Delta\theta_s \leftarrow$ SOLVE SOFT CONSTRAINTS$(J_{\text{soft}}, J_{\text{hard}})$ ▷ Eq. 26
    $\Delta\theta \leftarrow \Delta\theta_h + \Delta\theta_s$
    $\theta \leftarrow$ UPDATE CHARACTER$(\Delta\theta)$
  end while
end while

A singularity robust method (SR), also known as damped pseudoinverse, uses a regularization parameter to relax the solution by letting the norm of the solution have some error

$$A^* = A^T (AA^T + k\mathbf{I})^{-1} \tag{27}$$

where A^* is the SR inverse of A, matrix $\mathbf{I}$ is the identity, and k is the damping parameter weighting between the error and the norm of the solution. For small values of k we get smaller errors, but we might as well encounter larger solutions around singular points. Using the SR inverse along with proper tuning of damping k helps us ease the singularity problem by allowing errors near singular points. We, however, want to use regularization only for the constraints that are secondary in terms of importance and use a typical pseudo-inverse method for the momentum related quantities. As a result, we use an undamped pseudo-inverse method for the hard constraints, and a damped pseudo-inverse technique for the soft constraints to achieve the best mixture in our prioritized solver. Algorithm 1 provides an overview of the steps involved in our IK solver.

4.3 Limited changes

Our system provides both the ability to adjust the character pose globally as well as making changes to a limited set of user specified joints. While making global edits helps fast sketching a pose, giving the animator the ability to apply limited changes to the body is also necessary. This is because an artist is typically interested in making subtle adjustments to a part of the character without affecting the whole body.

In order to isolate a group of joints, a local root R' is selected and the dragging of joint D determines the set of body limbs that are taken into account by the IK solver. Figure 2 shows how dragging a joint partitions the character into an active joint set (red nodes) and a passive set (blue nodes). Algorithm 2 shows how we mark all the joints as *included* or *excluded* depending on which subtree each joint belongs to.

When solving for the soft constraints, we adjust the corresponding weight of those that are *included* to 1 and set the rest to 0. At the core of our algorithm we find the path P_m by marching from joint

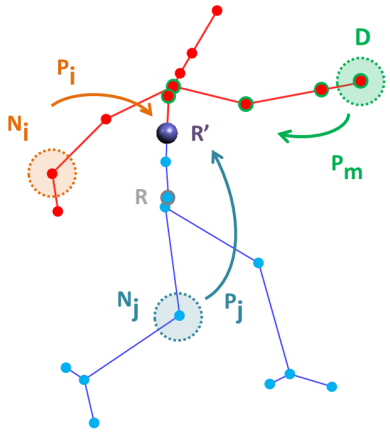

Figure 2: Dragging the joint D separates the body into active (red) and passive (blue) regions. Main path P_m to the local root R' shares at least one node with path P_i, which makes the joint N_i a member of the active set. Path P_j does not share any joints with P_m so its corresponding sample joint N_j is excluded from the IK solve.

Algorithm 2 Limited changes

while Local root is selected **do**
  $R \leftarrow$ GET THE ACTUAL ROOT
  $R' \leftarrow$ GET THE LOCAL ROOT
  $D \leftarrow$ GET DRAGGED NODE$(nodes)$
  while D is valid **do**
    MARK ALL NODES EXCLUDED$(nodes)$
    $P_m \leftarrow$ FIND PATH TO LOCAL ROOT(R', D)
    for all nodes **do**
      $N_i \leftarrow$ GET CURRENT NODE
      $P_i \leftarrow$ FIND PATH TO $R'(N_i, nodes, R')$
      if P_i and P_m share any node except R' **then**
        MARK INCLUDED(N_i)
      end if
    end for
    $N_s \leftarrow$ GET A SAMPLE INCLUDED NODE$(nodes)$
    $P_L \leftarrow$ FIND PATH TO $R'(N_s, nodes, R')$
    $P_R \leftarrow$ FIND PATH TO ACTUAL ROOT$(N_s, nodes, R)$
    if $P_L \subseteq P_R$ **then**
      MARK INCLUDED(R')
    end if
  end while
end while

D to R'. For any other joint we find the path P_i to R' and mark the joint *included* only if P_i and P_m share at least one joint other than R'. We do not allow for any translational update to the actual root when performing limited changes. Our method ensures no violation of the positional constraint when a pinned joint is *excluded* and is dragged by D.

5 Handle Trajectory Templates and Interface

Given the appropriate handles, such as the CM position and inertia, the key to our solution is to present the animator with a tool that allows them to intuitively edit these features of a motion while preserving physical plausibility. In this section, we describe a collection of tools and trajectory templates that provide this functionality.

Figure 3 shows part of the temporal controls interface presetned to the animator. Much like in any standard keyframing interface, there are different tracks for different quantities that have keyframes

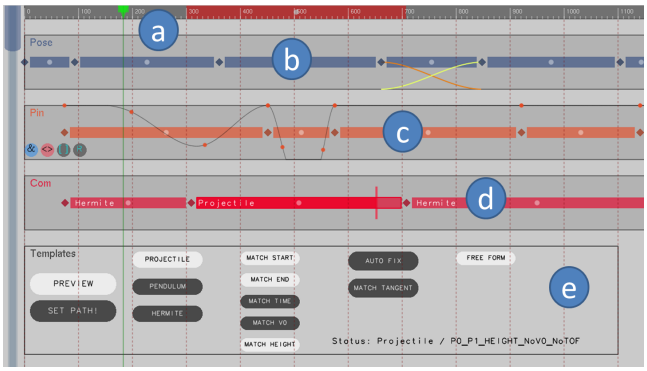

Figure 3: Screen capture of the workspace used to set keyframes and template trajectories. (a) Timing pane with a selected repeat area shown in red and a time scrubber shown in green. (b) Pose keyframing track where the type of interpolation is also set (note the ease-in-ease-out blue and orange curves). (c) Pin keyframes with a super imposed cubic spline to control pin gains. (d) CM keyframes with a hint for physically plausible timing requirement. (e) Trajectory template control panel.

set, such as pose tracker, hand or foot constraints, as well as the CM position. The animator can insert, delete, copy and paste, and edit keyframes, as well as dragging them to different times. Between a pair of keyframes, we allow the user to adjust different properties, such as setting the interpolation function in the pose track, the positional constraint gains in the pin track, and the template trajectories in the CM track.

In order to obtain the character configuration for the current time step, the system first uses the timing pane to select the active pair of keyframes in each track, then computes the desired quantities through processing the selected keyframes, and finally solves for the joint angle updates by inverse kinematics. The animator is provided with optional inclusion or exclusion of keyframes in the pin and the CM tracks. This is a direct feature of separating physics-based quantities and postural constraints in Equation 26.

Figure 4 shows a snapshot of the spatial controls available to the animator during the landing sequence of a jump. The CM position at the keyframes can be dragged, as can be the velocity. Pinning constraints are also an important part of designing the motion. In the figure, note how feet are naturally dragged towards the ground while preparing for landing. The desired touch down points are shown in red on the grass. While the character is still in the air, the pins are set with low gains that increase over time in preparation for the landing phase. The velocity of the CM at the time of landing is likewise used as the initial velocity of the CM Hermite curve in the landing phase, ensuring that there is a C1 continuous smooth transition between the projectial path and the landing motion.

We have investigated a few different templates for setting physically plausible character motion between keyframes.

Hermite. Cubic curves allow positoin and velocity to be matched connecting the segments between keyframes. In particular, we can ensure smooth trajectories for the CM leading into and exiting the other template trajectories. In order to generate a trajectory for a Hermite interval, CM positions are sampled along the path connecting two Hermite keyframes based on the time duration of the path, and the CM positions and velocities at each end. Although the cubic curves are computed for each Hermite interval separately, the resulting spline when connecting multiple segments will be C1 continuous.

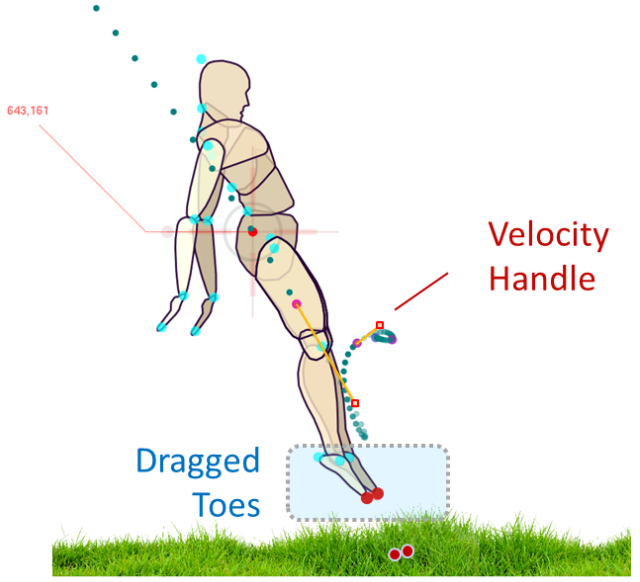

Figure 4: Screen capture of the interface showing handles available for controlling the landing motion of a jump.

Projectile. This template has the constraint that the linear momentum of the character is only affected by gravity, and thus the center of mass will undergo a constant vertical acceleration of $-9.8\ \mathrm{ms}^{-2}$, while the horizontal momentum is conserved. The animator can adjust the arc of the jump through a variety of controls such as the maximum height, initial velocity, initial position, end position, and time of flight. Because these parameters are coupled, our tool automatically adjusts the parameters left unconstrained by the animator in order to satisfy the constraints of the template trajectory. In particular, we present three projectile templates, as shown in Table 1, each with a specific purpose that makes them intuitive to use by the animator. We label these templates as (I) , (II) and (III).

Projectile (I) presents a straightforward template where the animator will only need to set the start and end CM positions and specify the time of flight, where the height of the arc can be adjusted through changing the time interval of the keyframes. Note that this template does not use any velocity handles but updates its neighbours velocity handles to ensure smoothness.

Projectile (II) does not require a landing position and instead uses the take-off velocity handle to control the jump arc. In this case we allow the template to automatically compute the landing position and update the corresponding CM keyframe with the new value, $C_e{}'$. The free landing parameter helps the animator to explore different jump scenarios in the sketching phase only by regulating the initial velocity handle without being too much concerned with the landing position.

Projectile (III) uses an auxiliary interactive tool to set the desired maximum height. This is specially useful when we want to author jump motions where take-off and landing, as well as the height of the motion, are more important than the time of flight. For example, performing a slam dunk by a basketball player requires reaching a certain desired height while take-off and landing positions should remain on the floor and inside the playground. Because the time of flight might not match that of the keyframes we provide visual hints both on the ballistic path and in the keyframing panel to help the animator with the process, as shown in Figures 5 and 6. We also present an "auto-fix" feature to automatically compute a new time of flight $t_e{}'$ that satisfies the jump height and positional constraints.

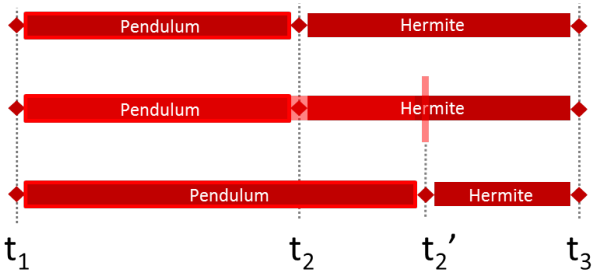

Figure 5: Projectile hint and auto-fix in the CM keyframing track. Top: Initial setup of the keyframes to be edited. Middle: Increasing the desired maximum height of the jump violates the timing of the end CM keyframe. Bottom: Corrected time of flight.

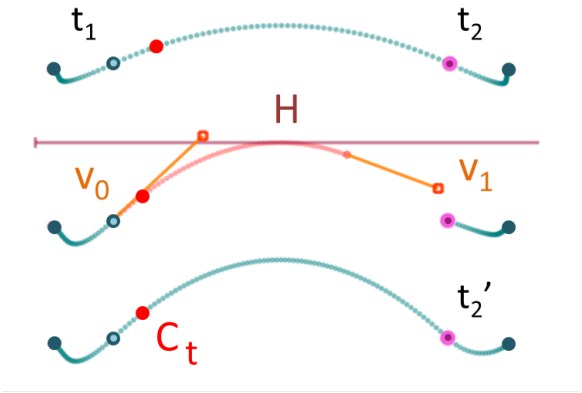

Figure 6: Projectile hint and auto-fix display related to the example shown in Figure 5 with the same order of steps. H is the desired maximum height, V_0 and V_1 are the initial and end velocity handles and C_t is the CM position at the current time. The rest of variables are the same as Figure 5.

Should the animator decides to use this feature, we move the end CM keyframe such that the new time interval matches $t_e{}'$, as shown in Figure 5.

Pendulum. The motion of the CM follows the motion of a pendulum, allowing animation of a character swinging between branches, on monkey bars or spiderman type of swing between buildings. This template assumes a frictionless pivot that results in an undamped motion trajectory of the CM. The only acting forces on the pendulum are the gravitational force and the force that keeps the mass point on the constrained path. The length, initial velocity and pivot position of the pendulum can be adjusted and keyframes can be set to transition at arbitrary times. Just like projectile (II) we do not use the end CM position in generating the pendulum path, rather we update the end keyframe with the position of the CM at the end of the time interval, as shown in Figure 7 as $C_3{}'$. Using the same template we can also generate path for an inverted pendulum by adjusting the initial velocity handle. An inverted pendulum allows animating a variety of interesting character motions such as walking locomotion or gymnastic front and back aerials. This is inspired by simplified models used to understand animal locomotion, and to control physics based character locomotion.

Table 1 summarizes the control variables available to the animator when using each feature trajectory template. We always update the neighbour CM trajectories when editing the path of a template in order to maintain the smoothness of the CM trajectory segments. For instance, in Figure 6 we see how the adjacent Hermite templates are updated as we edit the projectile path. Note that we can only update the velocity handles if they are part of the adjustable variables of the trajectory template (V_s and V_e). This means, for example, Hermite curves are always updated though both the initial and end velocity handles, while a pendulum template can only be updated through its initial velocity handle.

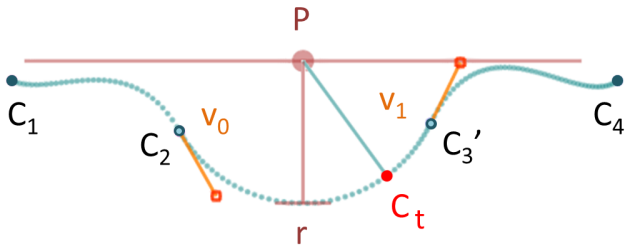

Figure 7: Generating a pendulum trajectory for a pair of CM positions given by C_1 and C_2. Initial velocity handle V_0, the pivot position P and the time interval between C_1 and C_2 are used to generate the trajectory. Hermite curves are also set for the keyframe pairs C_1 and C_2 as well $C_3{}'$ and C_4. C_t is the CM position at time t. Note that $C_3{}'$ is automatically updated by the template and the end velocity V_1 is used to update the cubic curve on the right-hand side.

Template	C_s	C_e	V_s	V_e	ΔT	H/P	$C_e{}'$	$t_e{}'$
Hermite	x	x	x	x	x			
Projectile I	x	x			x			
Projectile II	x		x		x		x	
Projectile III	x	x			x	x		x
Pendulum	x		x		x	x	x	

Table 1: Template control variables. C_s and C_e are start and end CM positions, V_s and V_e are the start and end velocities, ΔT is the time duration of the trajectory, H/P means either use of desired height (H) or pivot position (P), $C_e{}'$ is the computed end CM position and $t_e{}'$ is the corrected end time.

6 Results

We present results for a variety of examples, which are best seen in the supplementary video. Our keyframing system allows for quickly authoring a motion with visual hints and assist control to the animator whenever a physical constraint is violated. The humanoid example has size and mass distributions which come from a study by Nasa [15]. We also show that our method does not make any assumption regarding the morphology of the character. All computations were run on a dual-core 2.4 GHz machine with 12 GB RAM. Each IK step with both hard and soft constraints takes about 250 μs. This helped us achieve real time preview display and user interactions in the keyframing interface, which is a crucial factor in animation creation process.

Limited changes. Figure 8 shows reproducing results similar to those demonstrated by Yamane and Nakamura [20], as well as how our limited change algorithm provides more control over editing body posture. In this figure, we compare postural changes of the character with and without a user selected local root, where edits are only applied to a user selected region. The limbs coloured in green are temporarily excluded from the edit process. Our algorithm ensures dragging an active limb results in a new posture that

also meets the positional constraints of the passive pins. This can be seen from parts (d) and (f) in the figure where the pinned feet remain in their positions while the the right hand is dragged.

Monkey bar. As shown in Figure 1 as well as in the supplementary video, all three types of trajectory templates are put together to create an animation where the character performs sportive actions in a highly dynamics manner. Successive swings consist of seven pendulum templates that are connected by Hermite cubic curves to achieve smooth CM trajectories in transitions. A projectile template is used for a plausible landing motion and a Hermite template steers the character to its rest pose. Pin gains are also scheduled to achieve natural arms and feet motion in making and breaking contacts with the monkey bar, as well as in landing.

Jump motion. In the supplementary video we present a long jump example that well demonstrates the application of IK handle in our keyframing system. The character center of mass follows a projectile motion created by constraining a desired height, take off and landing positions. Not only Hermite template handles at take off and landing help creating a smooth curve for the center of mass, but also based on the created ballistic path they are automatically updated to match velocity when interpolating keyframes. Plausible take off and landing feet postures are achieved through pinning the toes and heels to the ground and regulate their gain using the interface. Figure 9 shows an example of changing the pose of the character without any necessity to regulate the IK handles. In the video we show how to use the interface to author an example jump motion using the techniques described here.

Dive motion. Figure 10 shows a fast way of creating the poses that are common in a dive motion simply by regulating the inertia of the character. In this figure we maximize the body inertia to achieve full stretch in the beginning and at the end, and decrease it at the top of the ballistic curve. We use the inertia handle along with the projectile template to quickly generate results shown in this figure.

Work flow. In the video we present our work flow and how an artist is provided with visual hints to correct the physics of a jump motion. We also show an easy autofix button to automatically correct the timing of the keyframes when, for example, the timing difference between two keyframes does not allow the character reach its target landing position.

Face hugger. This example shows how the IK handles are not dependant on the morphology of the character. Figure 11 illustrates an alien character with many body limbs. While creating a pose could be challenging due to many joints of the face hugger, we show changing the inertia can provide a fast way of moving the limbs towards a desired pose. The video shows symmetrical postures are achieved by merely using the inertia handle. We also show that a predator pose emerges naturally by pinning two legs on the ground and moving the tail sideways while lowering the inertia quantity.

Performance. Figure 12 shows an example convergence plot for a typical case of editing the character with varying number of pins. For the sake of performance test we set the desired pose of the character to a fixed rest pose before each call to the IK solver. Note that while this could make it more challenging for the solver to converge, in practice we typically set the desired pose from the solution at the previous step of animation to get faster convergence rates. In our experiments we found that iteration numbers between 30 to 50 generally produce convincing motions while allowing for real time interactions with the character.

7 Conclusion

In this paper we derive expressions for the center of mass, momentum, and the magnitude of the inertia tensor, and we use these as constraints within an inverse kinematics solver to allow animators to easily animate physically plausible motion. Furthermore, physically based template trajectories help an animator produce trajectories that respect physics during dynamic activities. The solver includes soft constraints on the pose of all joints to encourage natural poses, while producing postures that meet reaching objectives and foot placements with constraints of adjustable weight. Re-rooting likewise allows an animator to easily specify local pose changes. Creating physically plausible character motion through keyframing is challenging, and we expect our results to be relevant to character animators of all levels of experience.

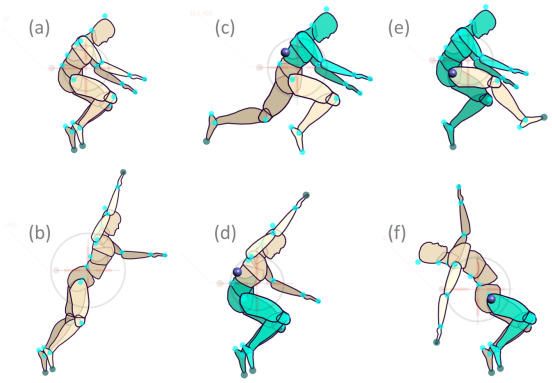

Figure 8: Limited changes. (a) Default pose with feet pinned. (b) Dragging right hand results in full body change. (c) Using local root to only adjust the lower part of the body. (d) Using the same local root to adjust the upper part. (e) Isolating the right leg through a local root at the right hip. (f) Dragging the right hand changes the body posture but has no effect on the excluded limbs. Note that the positional constraint of the right foot is also satisfied.

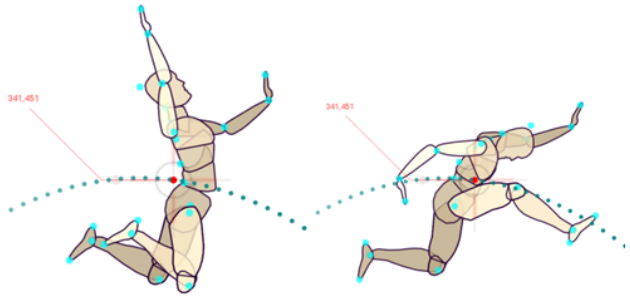

Figure 9: Editing a jumping character. Left: Initial pose. Right: New pose. Note in both cases the center of mass position on the ballistic curve remains fixed.

7.1 Future work

While the results shown in this paper are 2D, the derivations of center of mass an inertia handles are the same in 3D. Our derivation of momentum, both angular and linear opens up the possibility of providing a template trajectory that preserves angular momentum during ballistic motion. Combined with a control of the rotational inertia, this can provide an animator the means to produce physically valid animation that would be very hard to produce otherwise. While it is straightforward to change the angular velocity of a character to ensure that angular momentum is preserved when the angular inertia changes, it is more challenging to do this with time varying end point constraints with varying weights. We believe that this can be addressed by combining a temporally local solves with a more general optimization over a larger temporal window. Other

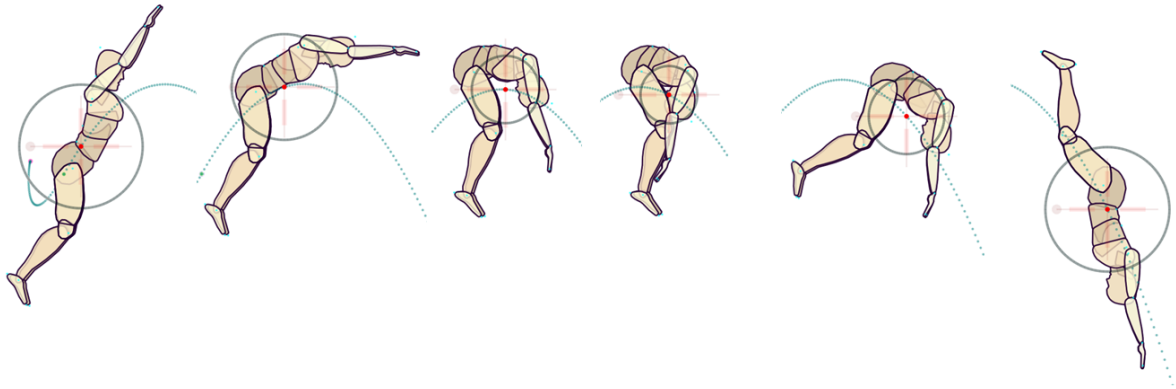

Figure 10: Dive motion using the projectile template and interpolation of rotational inertia. The inertia handle is shown as a grey circle. Note the change of inertia from left to right.

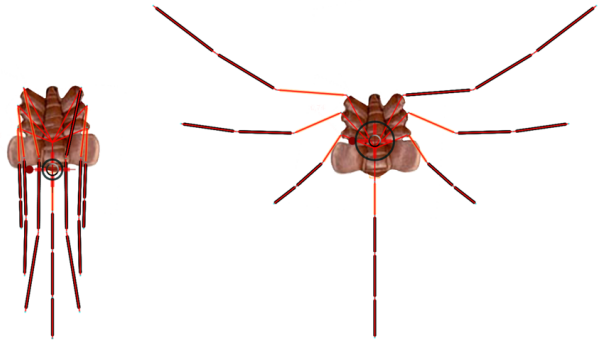

Figure 11: Face hugger: Only using the inertia handle to change the legs configuration. Increasing the inertia results in a natural transition from left to right.

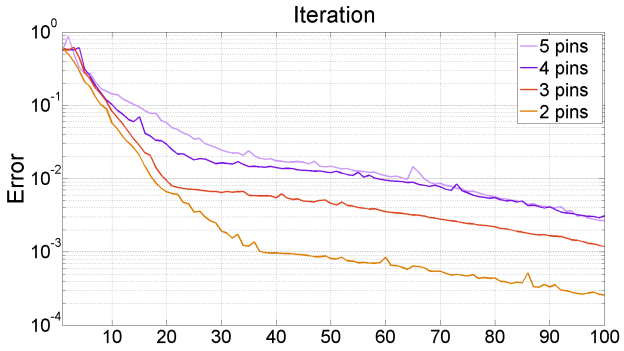

Figure 12: Example convergence plot of an IK solve with different pin numbers. All soft and hard constraints are included in this example.

interesting extensions to our work include the estimation of forces through inverse dynamics to allow animators to understand when and how the motions they designed become less plausible.

References

[1] A. Aristidou and J. Lasenby. Fabrik: a fast, iterative solver for the inverse kinematics problem. *Graphical Models*, 73(5):243–260, 2011.

[2] P. Baerlocher and R. Boulic. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6):402–417, 2004.

[3] R. Boulic, R. Mas, and D. Thalmann. A robust approach for the control of the center of mass with inverse kinetics. *Computers & Graphics*, 20(5):693–701, 1996.

[4] R. Boulic and D. Thalmann. Combined direct and inverse kinematic control for articulated figure motion editing. *Computer Graphics Forum*, 11(4):189–202, 1992.

[5] S. R. Buss. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods, 2004. unpublished survey.

[6] S. R. Buss and J.-S. Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics, GPU, and Game Tools*, 10(3):37–49, 2005.

[7] A. Deo and I. Walker. Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems*, 14(1):43–68, 1995.

[8] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Transactions on Graphics (TOG)*, 23(3):522–531, 2004.

[9] M. Kallmann. Analytical inverse kinematics with body posture control. *Computer Animation and Virtual Worlds*, 19(2):79–91, 2008.

[10] T. Komura, A. Kuroda, S. Kudoh, T. C. Lan, and Y. Shinagawa. An inverse kinematics method for 3D figures with motion data. In *Computer Graphics International 2003*, pages 266–271, July 2003.

[11] P. G. Kry, C. Rahgoshay, A. Rabbani, and K. Singh. Inverse kinodynamics: Editing and constraining kinematic approximations of dynamic motion. *Computers & Graphics*, 36(8):904–915, 2012.

[12] R. Kulpa and F. Multon. Fast inverse kinematics and kinetics solver for human-like figures. In *Proceedings of IEEE Humanoids*, pages 38–43, 2005.

[13] S.-H. Lee and A. Goswami. Reaction mass pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. In *ICRA*, pages 4667–4672, 2007.

[14] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC, March 1994.

[15] H. G. A. A. M. R. L. W.-P. A. OH. Anthropometry and mass distribution for human analogues. volume 1. military male aviators. Final report, Anthropology Research Project, 1988.

[16] D. Raunhardt and R. Boulic. Motion constraint. *The Visual Computer*, 25(5):509–518, 2009.

[17] A. Shapiro and S.-H. Lee. Practical character physics for animators. *IEEE Computer Graphics and Applications*, 31(4):45–55, 2011.

[18] K. W. Sok, L. Yamane, K., and J. Hodgins. Editing dynamic human motions via momentum and force. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 11–20, 2010.

[19] L.-C. T. Wang and C. C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *Robotics and Automation, IEEE Transactions on*, 7(4):489–499, 1991.

[20] K. Yamane and Y. Nakamura. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics*, 9:352–360, 2003.

[21] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics (TOG)*, 13:313–336, 1994.