# Video-based 3D Motion Capture through Biped Control

Marek Vondrak* Brown University
Leonid Sigal† Disney Research Pittsburgh
Jessica Hodgins‡ Disney Research Pittsburgh
Odest Jenkins§ Brown University

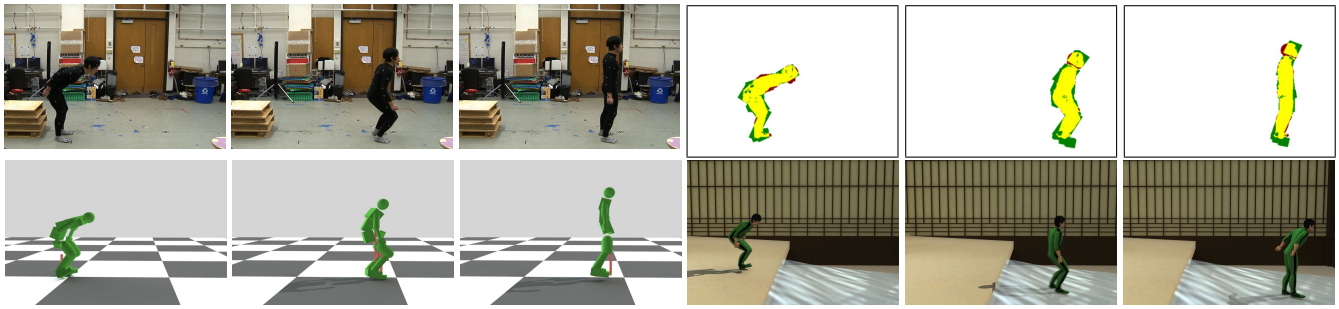


**Figure 1: Controller Reconstruction from Video:** *We estimate biped controllers from monocular video sequences (top) together with a physics-based responsive character (bottom left) that we can simulate in new environments (bottom right).*

## Abstract

Marker-less motion capture is a challenging problem, particularly when only monocular video is available. We estimate human motion from monocular video by recovering three-dimensional controllers capable of implicitly simulating the observed human behavior and replaying this behavior in other environments and under physical perturbations. Our approach employs a state-space biped controller with a balance feedback mechanism that encodes control as a sequence of simple control tasks. Transitions among these tasks are triggered on time and on proprioceptive events (*e.g.*, contact). Inference takes the form of optimal control where we optimize a high-dimensional vector of control parameters and the structure of the controller based on an objective function that compares the resulting simulated motion with input observations. We illustrate our approach by automatically estimating controllers for a variety of motions directly from monocular video. We show that the estimation of controller structure through incremental optimization and refinement leads to controllers that are more stable and that better approximate the reference motion. We demonstrate our approach by capturing sequences of walking, jumping, and gymnastics.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking;

**Keywords:** video-based motion capture, bipedal control

**Links:** DL PDF

*e-mail: marek@cs.brown.edu
†e-mail: lsigal@disneyresearch.com
‡e-mail: jkh@disneyresearch.com
§e-mail: cjenkins@cs.brown.edu

## 1 Introduction

Motion capture is a popular approach for creating natural-looking human characters. Traditionally, the data is captured using optical marker-based systems. However, such systems require instrumentation and calibration of the environment and the actor. One way to mitigate this limitation is to perform motion capture from video of an unmarked subject. This approach is particularly appealing for monocular video, which potentially allows motion capture from legacy footage. We address this problem by estimating controllers that are capable of simulating the observed human behavior.

The problems of video-based motion capture and biped control have been addressed independently by several research communities. Video-based pose tracking (or marker-less motion capture), has been a focus of much research in computer vision. Biped control, on the other hand, has evolved as a prominent area in computer graphics and robotics. In our formulation, we combine these two problems. Casting the video-based motion capture problem as one of direct estimation of underlying biped control leads to the following two benefits: (1) the physics-based biped controller can serve as an effective motion prior for estimating human motion with elements of physical realism from weak video-based observations, and (2) the recovered controller can be used to directly simulate responsive virtual characters.

We argue that by restricting kinematic motion to instances generated by a control structure (parameters of which we optimize), we can approximate general principles of motion, such as balance and contact, allowing response behaviors to emerge automatically. Because our controller structure is sparse, we are able to integrate information locally from multiple (tens of) image frames which induces smoothness in the resulting motion and resolves some of the ambiguities that arise in monocular video-based capture, without requiring a user-in-the-loop. The use of a controller also eliminates unrealistic behaviors such as ground or segment penetrations which are often found in results obtained by typical video-based motion capture approaches.

We recover both the structure and parameters for a (sparse) controller through incremental optimization and subsequent refinement. To help resolve camera projection ambiguities and to reduce the number of optimized parameters, we also make use of a weak motion capture-based PCA prior for the control pose targets encoded in each state. Our results are preliminary, but promis-

SIGGRAPH '12, August 05 - 09 2012, Los Angeles, CA, USA

ing. We are able to estimate controllers that capture the motion of the subject automatically from monocular video. However, when compared to most recent motion-specific controller architectures obtained by optimizing biomechanically inspired objectives [Wang et al. 2010] or using motion-capture as reference [Kwon and Hodgins 2010; Liu et al. 2010], the motions resulting from simulations of our controllers appear somewhat robotic and overpowered. We believe these issues are primarily due to approximations in the parameters of our model that make optimization from weaker and noisier monocular image inputs tractable. In particular, we allow joint torque limits that are higher than typical for a human. Other flaws result from properties that are difficult to detect in monocular video, such as the distinction between single and double support.

For many applications, particularly in graphics, the kinematic pose trajectories recovered using marker or marker-less motion capture technologies may not be sufficient as they are difficult to modify. Consider a video-based marker-less motion capture system as an input device for a video game. Recovering the skeletal motion may not be sufficient because the game requires a responsive virtual character that can adapt to the changing game environment (*e.g.*, sloped ground plane). Such features currently fall outside the scope of motion capture techniques and intermediate processing is typically necessary to produce a responsive character. The overarching goal of this research is to directly recover controller(s) from video capable of performing the desired motion(s). The proposed method and illustrated results are a step towards that ambitious goal; that, we hope, will serve as a stepping stone for future research. We view this form of capture of the control parameters, in addition to the joint angles, as a next step in the evolution of motion capture.

### 1.1 Related Work

The problems of video-based motion capture and bipedal control have been studied extensively in isolation in the computer vision, robotics, and computer graphics literature. Despite their research history, both remain open and challenging problems. Approaches that utilize physics-based constraints in estimation of human motion from video have been scarce to date. Given the significant literature in the area, we focus on the most relevant literature.

**Video-based Motion Capture:** The problem of 3D motion capture from a relatively large number (4-8) of camera views is well understood with effective existing approaches [Gall et al. 2009]. Three-dimensional motion-capture from monocular videos is considerably more challenging and is still an open problem. Most approaches in computer vision that deal with monocular video either rely on forms of Bayesian filtering, like particle filters, with data-driven statistical prior models of dynamics (*e.g.*, GPDM [Urtasun et al. 2006]) to resolve ambiguities, or alternatively, on *discriminative*, regression-based, predictions [Bo and Sminchisescu 2010] that estimate the pose directly based on image observations and (optionally) on a sequence of past poses [Sminchisescu et al. 2007]. Generative approaches tend to require knowledge of the activity *a priori*, and both often produce noticeable visual artifacts such as foot skate and inter-frame jitter. Below we review, in more detail, relevant methods that focus on physics-based modeling as a way of addressing these challenges.

The earliest work on integrating physical models with vision-based tracking can be attributed to [Metaxas and Terzopoulos 1993; Wren and Pentland 1998]. Both employed a Lagrangian formulation of dynamics, within a Kalman filter, for tracking of simple upper body motions using segmented 3D markers [Metaxas and Terzopoulos 1993] or stereo [Wren and Pentland 1998] as observations; both also assumed motions without contacts. More recently, Brubaker *et al.* [2008; 2007] introduced two 2D, biomechanically inspired models that account for human lower-body walking dynamics; their models were capable of estimating walking motions from monocular video using a particle filter as the inference mechanism. Similar to our approach in spirit, the inference was formulated over the control parameters of the proposed model. In contrast, our model is formulated directly in 3D and is not limited to walking.

Vondrak and colleagues [2008] proposed a full-body 3D physics-based prior for tracking. The proposed prior took the form of a control loop, where a model of Newtonian physics approximated the rigid-body motion dynamics of the human and the environment through the application and integration of forces. This prior was used as an implicit filter within a particle filtering framework, where frame-to-frame motion capture-based kinematic pose estimates were modified by the prior to be consistent with the underlying physical model. The model, however, was only able to bias results towards more physically plausible solutions and required Gaussian noise to be added to the kinematic pose at every frame to deal with noisy monocular image observations. In contrast, the results obtained in this paper are directly a product of a simulation.

Wei and Chai [2010] proposed a system, where a physics-based model was used in a batch optimization procedure to refine kinematic trajectories in order to obtain realistic 3D motion. The proposed framework relied on manual specification of pose keyframes, and intermittent 2D pose tracking in the image plane, to define the objective for the optimization. In contrast, we optimize control structure and parameters and do not rely on manual intervention in specifying the target poses for our controller; we also do not rely on manual specification of contacts or foot placement constraints as in [Wei and Chai 2010] or [Liu et al. 2005].

There has also been relevant work on identification of models and contacts. Bhat and colleagues [2002] proposed an approach for taking the video of a tumbling rigid body in free fall, of a known shape, and generating a physical simulation of the object observed in the video. Conceptually, their approach is similar to our goal, however, they only dealt with a simple rigid object. Brubaker *et al.* [2009] developed a method for factoring the 3D human motion (obtained using a marker-based or standard marker-less multi-view motion capture setup) into internal actuation and external contacts while optimizing the parameters of a parametric contact model.

**Biped Control:** Biped controllers have a long history in computer graphics and robotics. Existing approaches can be loosely categorized into two classes of methods: data-driven and model-based. Data-driven controllers rely on motion capture data and attempt to track it closely while typically adding some form of bipedal balance (e.g., through quadratic programming [Silva et al. 2008], a linear quadratic regulator [Silva et al. 2008], or non-linear quadratic regulator [Muico et al. 2009]). The benefit of such methods is the added naturalness in the motion that comes from motion capture data. With data-driven methods, however, it is often difficult to handle large perturbations and unanticipated impacts with the environment. A notable exception is the approach of [Liu et al. 2010] which allows physics-based reconstruction of a variety of motions by randomized sampling of controls within user-specified bounds. As a result, the original motion capture trajectories are effectively converted into open-loop control. Model-based approaches typically correspond to sparse controllers represented by finite state machines for phase transition control (with optional feedback balancing mechanisms). Our approach falls within this category, which we broadly refer to as state-space controllers. The benefit of this class of models is their robustness and versatility.

State-space controller design has been one of the more successful approaches to closed-loop simulation for human figures, dating back to Hodgins *et al.* [1995]. State-space controllers encode de-

sired motion as a sequence of simple control tasks and this strategy has been effective in modeling variety of motions, including walking, running, vaulting and diving. Most methods augment such models with balance control laws [Hodgins et al. 1995], making them less reliant on environment and terrain. A particularly effective balance control law for locomotion tasks was introduced by Yin and colleagues [2007]. State-space controllers have also been shown to be composeable [Chu et al. 2007; Coros et al. 2009] and even capable of encoding stylistic variations [Wang et al. 2010]. Until recently, however, most models relied on manual parameter tuning. Several recent methods addressed automatic parameter estimation through optimization [Wang et al. 2009; Wang et al. 2010; Yin et al. 2008]. In our formulation, we leverage work from this literature (primarily [Wang et al. 2009]).

The simple control tasks within our state-space controller employ constraint-based inverse dynamics actuation to drive the character towards a target pose. This type of control is related to ideas of critically damped PD control dating back to [Ngo and Marks 1993]. However, unlike our controller, the constraints in their work are defined and solved individually for each joint degree of freedom. A very similar approach to ours, in terms of constraint-based control, has recently been proposed by Tsai and colleagues [2010]. However, their goal was to track dense motion capture trajectories.

**Contributions:** To our knowledge, this paper is the first to propose a method for recovery of full-body bipedal controllers directly from single-view video. Our method is capable of automatically recovering controllers for a variety of complex, three-dimensional, full-body human motions, including walking, jumping, kicking and gymnastics. The inference is cast as a problem of optimal control, where the goal is to maximize the reward function measuring consistency of the simulated 3D motion with image observations (or motion capture). The sparse nature of our controller allows estimation of parameters from noisy image observations without overfitting. Recovered controllers also contain a feedback balancing mechanism that allows replay in different environments and under physical perturbations. With respect to previous work in character bipedal control, our key contribution is the ability to optimize controller structure and parameters from monocular video.

## 2 Overview

The goal of our method is to estimate motion and controller from monocular video (or reference motion capture data). We optimize the controller structure and parameters such that the resulting motion produces a good match to image observations. Our approach has three important components:

**Body model and motion:** We start by introducing the basic formalism required to model the human body, its actuation and interactions with the environment. The 3D kinematic pose of an articulated human skeleton at time $t$ is represented by a state vector $\mathbf{x}_t = [\rho_t, \mathbf{q}_t^r, \mathbf{q}_t^k]$, comprising root position ($\rho_t$), root orientation ($\mathbf{q}_t^r$) and joint angles ($\mathbf{q}_t^k$). Articulated rigid-body dynamics and integration provide a function for mapping the *dynamic* pose, $[\mathbf{x}_t, \dot{\mathbf{x}}_t]$, at time $t$ to the pose at the next time instant, $[\mathbf{x}_{t+1}, \dot{\mathbf{x}}_{t+1}]$ (where $\dot{\mathbf{x}}_t$ is the time derivative of $\mathbf{x}_t$):

$$[\mathbf{x}_{t+1}, \dot{\mathbf{x}}_{t+1}] = f([\mathbf{x}_t, \dot{\mathbf{x}}_t], \tau_t). \quad (1)$$

This function is typically a numerical approximation of the continuous integration of internal joint torques, $\tau_t$, with respect to the current dynamic pose, $[\mathbf{x}_t, \dot{\mathbf{x}}_t]$.

**Actuation:** We propose a state-space controller for the actuation of the body that divides the control into a sequence of simple atomic actions, transitions among which occur on time or contact (see Section 4.1). We formulate constraint-based action controllers that are tasked with driving the simulated character towards a pre-defined posture. More formally, the joint torques, $\tau_t$, in Eq. (1) are produced by a *controller* $\pi$:

$$\tau_t = \pi([\mathbf{x}_t, \dot{\mathbf{x}}_t]; \mathcal{S}_M, \Theta) \quad (2)$$

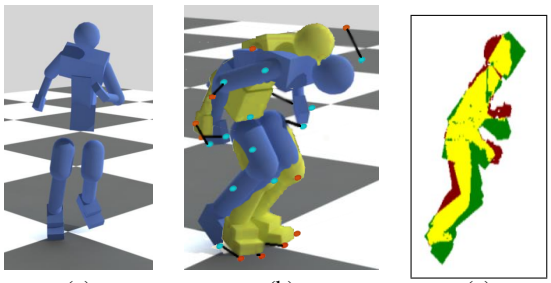
where $\mathcal{S}_M$ is the structure of the controller, $M$ is the number of states in the control structure (see Section 4.1) and $\Theta$ is a vector of controller parameters. A particular controller structure $\mathcal{S}_M$ induces a family of controllers, where the parameters $\Theta$ define the behavior of the controller. The simulation proceeds by iteratively applying Eq. (1) and (2), resulting in a sequence of kinematic poses, $\mathbf{x}_{1:T}$; because this formulation is recursive, initial kinematic pose $\mathbf{x}_0$ and velocities $\dot{\mathbf{x}}_0$ are also needed to bootstrap integration.

**Estimating controllers:** The key aspect of our method is optimization of the controller structure ($\mathcal{S}_M^*$), the parameters of the controller ($\Theta^*$), the initial pose ($\mathbf{x}_0^*$) and the velocities ($\dot{\mathbf{x}}_0^*$) in order to find a motion interpretation that best explains the observed behavior. This optimization can be written as a minimization of the energy function:

$$[\Theta^*, \mathcal{S}_M^*, \mathbf{x}_0^*, \dot{\mathbf{x}}_0^*] = \arg \min_{\Theta, \mathcal{S}_M, \mathbf{x}_0, \dot{\mathbf{x}}_0} E(\mathbf{z}_{1:T}), \quad (3)$$

that measures the inconsistency between the poses produced by the dynamic simulation (integration) with the controller and the image (or reference motion capture) observations, $\mathbf{z}_{1:T}$. In order to produce controllers that can generalize, the energy function also measures the quality of the controller itself in terms of its robustness and stability.

Because optimizing both controller structure and parameters is difficult, we first formulate a batch optimization over the controller parameters assuming a fixed controller structure and then introduce an incremental optimization procedure over the structure itself by utilizing the batch optimization as a sub-routine. We further refine the structure of the resulting controllers, by applying structural transformations and re-optimizing the parameters. The exact formulation of our objective function is given in Section 5.



(a) (b) (c)

**Figure 2: Body Model and Likelihoods:** *Illustration of the body model and the collision geometry used for simulation in (a) (see Section 3); (b) and (c) illustrate the motion capture and image-based likelihoods respectively (see Section 5).*

## 3 Body Model and Motion

We use a generic model of the human body that represents an average person. We encode the human body using 18 rigid body segments (with upper and lower segments for arms and legs, hands, two segment feet, three segment torso, and head). The center of mass and inertial properties of the segments are taken from population averages and biomechanical statistics [Dempster 1955]. For the

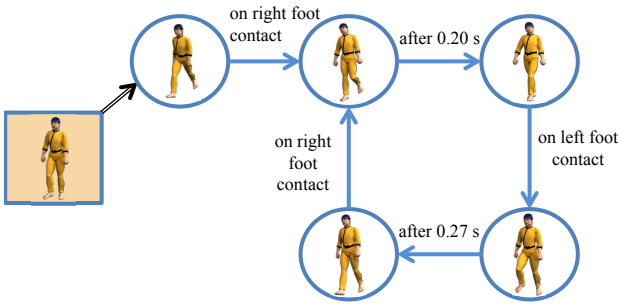



**Figure 3: State-space Controller:** *Walking motion as an iterative progression of simple control tasks (circles), transitions (arrows) and the initial pose (yellow box); structure and controller parameters were obtained by Algorithm 4 applied on the fast walk motion.*

purposes of simulation, each segment is parameterized by position and orientation in 3D space and has an associated collision geometry composed of geometric primitives (see Figure 2(a))[1]. Physical and environment restrictions are encoded using constraints. In particular, we impose (i) *joint constraints* that ensure that segments stay attached at the joints and have only certain relative degrees of freedom (*e.g.*, we model the knee and elbow as 1 DOF hinge joints), (ii) *joint limit constraints* that ensure that unconstrained degrees of freedom are only allowed to articulate within allowable ranges (e.g., to prevent hyperextension at elbows and knee joints), (iii) body segment *non-penetration* constraints and *environmental* constraints that model contact and friction. All three types of constraints are automatically generated by the simulator. Note that even though in simulation the state of our character is $\in \mathbb{R}^{108}$ (18 parts $\times$ 6 DOFs per part), for the purposes of actuation and control we only consider the state $\mathbf{x}_t = [\rho_t, \mathbf{q}_t^r, \mathbf{q}_t^k] \in \mathbb{R}^{40}$ comprising root position, root orientation and joint angles; where $\mathbf{x}_t$ effectively spans the null space of the joint constraints.

These constraints result in a large system of equations that can be expressed as a *mixed linear complementarity problem* (Mixed LCP), see [Pang and Facchinei 2003; Tsai et al. 2010]. Mixed LCP solvers can be used to solve this problem efficiently, yielding a set of forces (and torques) required to satisfy all of the above mentioned constraints; we use a publicly available implementation [Crisis 2006]. We provide more complete details of the Mixed LCP formulation, solution and contact model in the Appendix.

# 4 Actuation

We formulate our controller using constraints and directly integrate these constraints with the body and simulator-level constraints discussed above. As a result, the augmented Mixed LCP solution takes the actuation torques into account to ensure that the constraint forces anticipate the impact of actuation on the system.

We assume a SIMBICON [Yin et al. 2007] style controller, where a motion can be expressed as a progression of simple control tasks. Such paradigms have also been proposed in biomechanics (*e.g.*, [Schaal and Schweighofer 2005]) and amount to a hypothesis that the control consists of higher level primitives that compose to create the desired motion.

[1]We make the collision geometries for the upper legs shorter to avoid unintended collisions.

## 4.1 State-space Controller

An action-based (state-space) controller can be expressed as a finite state machine (see Figure 3) with $M$ states, where states correspond to the atomic control actions and transitions encode the rules under which the actions switch. Actions switch on timing or contact events (*e.g.*, foot-ground contact). For example, a walking controller that can be expressed in this way (and estimated by our method) is shown in Figure 3.

We denote the structure of the controller by $\mathcal{S}_M$. The structure describes which atomic actions need to be executed and how the actions switch in order to produce a desired motion for the character. We characterize the structure using a schematic representation that consists of (1) actions, denoted by $\odot_i$, (2) the types of transitions among those actions, which are denoted by arrows with associated discrete variables $\kappa_i$, $i \in [1, M]$, indicating when the transition can be taken (*e.g.*, $\kappa_i = \mathcal{T}$ for transition on time, $\kappa_i = \mathcal{L}$ for transition on left foot contact, and $\kappa_i = \mathcal{R}$ for transition on right foot contact), and (3) a symbol $\oplus \longrightarrow$ to indicate initial pose and velocity at simulation start. A chain controller for one single walk cycle in our representation may look something like this: $\mathcal{S}_4 = \{\oplus \longrightarrow \odot_1 \overset{\kappa_1=\mathcal{L}}{\longrightarrow} \odot_2 \overset{\kappa_2=\mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3=\mathcal{R}}{\longrightarrow} \odot_4\}$.

The behavior of the controller is determined by parameters describing the actions and transitions within the given controller structure. The parameters of a state-space controller can be expressed as $\Theta = [(\mathbf{s}_1, \theta_1, \vartheta_1), (\mathbf{s}_2, \theta_2, \vartheta_2), \ldots, (\mathbf{s}_M, \theta_M, \vartheta_M), \nu_1, \nu_2, \ldots]$, where $\mathbf{s}_i$ is the representation of the *target pose*, for the angular configuration $[\mathbf{q}^r, \mathbf{q}^k]$ of the body that the controller is trying to achieve while executing $\odot_i$ and $\theta_i$ and $\vartheta_i$ are parameters of the corresponding control and balance laws, respectively, used to do so; $\nu_i$ are transition timings for those states where transitions happen based on time ($\kappa_i = \mathcal{T}$). We also define a function $g(\mathbf{s}_i)$ that maps $\mathbf{s}_i$ into the angle representation[2] of $[\mathbf{q}^r, \mathbf{q}^k]$.

State-space controllers allow concise representation of motion dynamics through a sparse set of target poses and parameters; in essence allowing a key-frame-like representation [Wei and Chai 2010] of the original motion. The use of this sparse representation allows more robust inference that is not as heavily biased by the noise in the individual video frames. Despite the sparsity, however, the representation still allows the expressiveness necessary to model variations in style and speed [Wang et al. 2010] that we observe in video.

### 4.1.1 Encoding Atomic Control Actions

The function of the *atomic* controller is to produce torques necessary to drive the pose of the character towards a target pose $g(\mathbf{s}_i)$ while keeping the body balanced. We extend the method of Yin and colleagues [2007] for control and balance to set up the control laws. As in [Yin et al. 2007], we maintain balance of the body by (1) using an active feedback mechanism to adjust the target pose and (2) applying torques on the body in order to follow the prescribed target orientation for the root segment. To make the method more robust, we replace Yin's Proportional Derivative (PD)-servo with control based on inverse dynamics. The control behavior is determined by the values of the control parameters $\theta_i$ and balance parameters $\vartheta_i$. Intuitively, these parameters encode how the target pose should be adjusted by the balance feedback mechanism and define the speed with which the adjusted pose should be reached.

[2]The simplest case is where the encoding of the target pose is the same as $[\mathbf{q}^r, \mathbf{q}^k]$, $g(\mathbf{s}_i) = \mathbf{s}_i$; a more effective representation is discussed in Sec. 4.1.2.

**Algorithm 1** : Application of control forces

1: Solve for $\tau_t$ using inverse dynamics to satisfy Eq. (4) and Eq. (5) and apply $\tau_t$
2: Determine by how much Eq. (6) is violated:
$$\tau^r_{error} = \tau^r_t + \sum_{k \in \text{Neighbor}(r)} \tau^k_t$$
3: Apply $-\tau^r_{error}$ to the swing upper leg

**Control:** We formulate our atomic controllers based on constraints (similar to [Tsai et al. 2010]). Assuming that the body is currently in a pose $\mathbf{x}_t$, and the state-machine is executing an action from the finite state $\odot_i$, we define an atomic controller using *pose difference stabilization*. A stabilization mechanism imposes constraints on the angular velocity of the root, $\dot{\mathbf{q}}^r_t$, and joint angles, $\dot{\mathbf{q}}^k_t$, [Cline 2002] in order to reduce the difference between the current pose and the adjusted target pose $[\mathbf{q}^r_d, \mathbf{q}^k_d]$, where $[\mathbf{q}^r_d, \mathbf{q}^k_d]$ is obtained by modifying the target pose $g(\mathbf{s}_i)$ from the machine state by the balance feedback. As a result, the atomic controller defines these constraints:

$$\{\dot{\mathbf{q}}^r_t = -\alpha^r_i(\mathbf{q}^r_t - \mathbf{q}^r_d)\} \quad (4)$$
$$\{\dot{\mathbf{q}}^k_t = -\alpha^k_i(\mathbf{q}^k_t - \mathbf{q}^k_d)\}, \quad (5)$$

parameterized by $\alpha_i$. The set of control parameters for our model is then $\theta_i = \{\alpha_i\}$. To reduce the number of control parameters that are optimized, left side and the right side of the body share the same parameter values.

**Underactuation:** The formulation above directly controls the global orientation of the root, which is essential for balance. As in Yin and colleagues [2007], it does so by directly applying forces on the root segment. However, actuation forces should only be produced by joints in the body model and the root should be unactuated. Consequently, this control is only valid if the net torque on the root segment is equal and opposite to the sum of torques of all connected segments:

$$\{\tau^r_t = -\sum_{k \in \text{Neighbor}(r)} \tau^k_t\}. \quad (6)$$

Because we cannot express Eq. (6) directly as a constraint with Eq. (4) and Eq. (5), we employ a prioritized approach. This approach satisfies Eq. (6) exactly and uses a "best effort" strategy to satisfy the other two constraints (Eq. (4) and Eq. (5)). Towards that end, we use the method from Yin and colleagues [2007] to decompose extra torques applied at the root to additional reaction torques applied at the thighs (see Algorithm 1).

We first solve for the torques $\tau_t$ that satisfy Eq. (4) and Eq. (5) using inverse dynamics (ignoring Eq. (6)) and apply the torques to the character. We then compute $\tau^r_{error}$ — the amount of violation in Eq. (6). The value of $\tau^r_{error}$ equals exactly the "extra" torque received by the root segment that is not "explained" by any of the joints attached to the root. To alleviate this error, we apply the corresponding reaction torque $-\tau^r_{error}$ to the swing upper leg segment (to ensure that Eq. (6) holds). This approach is similar to Yin and colleagues [2007]. As a result of this correction, Eq. (6) and Eq. (4) hold exactly and Eq. (5) may slightly be violated due to the $-\tau^r_{error}$ torque applied at the swing upper leg.

**Balance feedback:** The constraints on the root segment attempt to maintain the world orientation of the character. Active feedback mechanisms, however, may be necessary to prevent falling when the environment changes or perturbations are applied. We generalize the balance feedback mechanism of Yin and colleagues [2007] to dynamically adjust the target orientation of the swing upper leg from $g(\mathbf{s}_i)$. The feedback mechanism uses three additional control parameters encoded into a vector $\vartheta_i \in \mathbb{R}^3$.

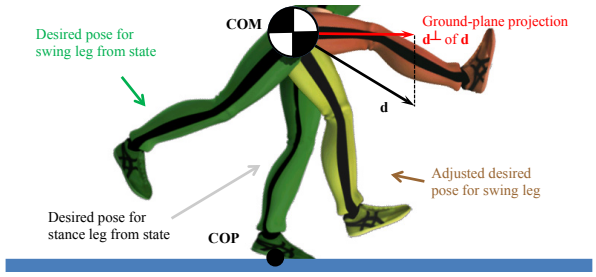



**Figure 4: Balance Feedback:** *Target orientation of the swing upper leg (yellow) is the result of blending between the target orientation stored in the state (green) and an orientation where the leg points along the ground-plane projection* $\mathbf{d}^\perp$ *of* $\mathbf{d}$ *(red), produced by the feedback mechanism.*

Intuitively, the purpose of the balance control law is to affect the motion of the swing foot in order to move the character toward a balanced pose where the center of mass ($\mathbf{COM}$) is above the center of pressure ($\mathbf{COP}$). The feedback is implemented by synthesizing a new target orientation for the swing upper leg and using it as a target (Figure 4). The controller will then track an adjusted target pose $[\mathbf{q}^r_d, \mathbf{q}^k_d]$, where the target orientation $\mathbf{q}^{k,SUL}_d$ for the swing upper leg is set as $\mathbf{q}^{k,SUL}_d = \text{slerp}(g(\mathbf{s}_i)^{k,SUL}, \text{orient}(\mathbf{d}^\perp), \vartheta_{i,3})$, where slerp is a spherical linear interpolation function, $SUL$ refers to the DOFs representing the orientation of the swing upper leg in the pose, $\mathbf{d} = \vartheta_{i,1}(\mathbf{COM} - \mathbf{COP}) + \vartheta_{i,2}(\dot{\mathbf{COM}} - \dot{\mathbf{COP}})$, $\mathbf{d}^\perp$ is $\mathbf{d}$ projected to the ground plane and $\text{orient}(\mathbf{d}^\perp)$ produces a leg orientation such that the leg's medial axis points along $\mathbf{d}^\perp$. The remaining DOFs within $[\mathbf{q}^r_d, \mathbf{q}^k_d]$ are copied from $g(\mathbf{s}_i)$.

**Force Limits:** To ensure that the controller does not apply superhuman forces, we limit the joint torques via limits on the solution to the Mixed LCP problem. We limit the torques generated about the individual joint axes, where a joint is defined between parent and child segments with masses $m_{\text{parent}}$ and $m_{\text{child}}$, using a single parameter $\gamma = 120$: $-\gamma\frac{m_{\text{parent}}+m_{\text{child}}}{2} \leq \tau^{axis}_t \leq \gamma\frac{m_{\text{parent}}+m_{\text{child}}}{2}$. Intuitively, these bounds implement the notion that heavier body segments contain larger muscle volume and hence can apply larger torques about the joint.

When we solve for control torques, control constraints are automatically combined with all other constraints in our simulator to anticipate effects of body articulation and contact. Our constraint-based control is different from a PD-servo. A PD-servo assumes that the control torques are linear functions of the differences between the current and the target poses of the body. It also assumes that each degree of freedom can be controlled independently and relies on a feedback mechanism to resolve the resulting interactions within the system. In contrast, our control mechanism solves for the torques necessary to approach the target pose, by explicitly taking into consideration constraints present among rigid body segments and the environment. Because we may not want to reach the target pose immediately, parameters $\alpha_i$ modulate the fraction of the pose difference to be reduced in one simulation step. More exact control and less reliance on feedback allow us to simulate at much lower frequencies (*e.g.*, 120 *Hz*).

#### 4.1.2 Parameterization for $\mathbf{s}_i$

The vector $\mathbf{s}_i$ encodes the representation of the target pose $g(\mathbf{s}_i)$ for the machine state $\odot_i$, where the function $g(\mathbf{s}_i)$ maps $\mathbf{s}_i$ into the angle representation of $[\mathbf{q}^r, \mathbf{q}^k]$. For a given class of motions, it is reasonable to assume that $g(\mathbf{s}_i)$ should lie within a manifold of poses likely for that motion. To this end, we propose a PCA prior

for $g(\mathbf{s}_i)$. We learn a linear basis $\mathbf{U}$ from training motion capture data[3] using SVD and let,

$$g(\mathbf{s}_i) = \mathbf{U}\mathbf{s}_i + \mathbf{b}. \quad (7)$$

We then only need to store PCA coefficients $\mathbf{s}_i$ in the parameter vector $\Theta$. We keep enough principal components to account for 95% of the variance in the data. The PCA representation of poses significantly reduces the dimensionality of our pose search space from $40M$ (where $M$ is the number of states in the state-space controller) to roughly $8M$ for most motions. We also found this method to lead to faster and more robust convergence. We add a uniform prior on $\mathbf{s}_i$ such that the coefficients are within $\pm 4\sigma$. Similarly, we assume that the initial kinematic pose can be encoded in the same PCA space, so that $\mathbf{x}_0 = \mathbf{U}\mathbf{s}_0 + \mathbf{b}$, and optimize $\mathbf{s}_0$ instead of $\mathbf{x}_0$.

We train activity-specific PCA priors from marker-based motion capture obtained off-line. In this way, we obtain a family of models $\{\mathbf{U}_i, \mathbf{b}_i\}$ where $i$ is over the set of activities considered (five PCA models for walk, jump, spin kick, back handspring and cartwheel) as well as an activity-independent model $\{\mathbf{U}_0 = \mathbf{I}_{40\times 40}, \mathbf{b}_0 = \mathbf{0}_{40}\}$. Because, in practice, we do not know the type of activity being performed at test time, we run our framework with each prior and choose the resulting controller that best fits observations according to the objective function $E(\mathbf{z}_{1:T})$.

Unlike traditional kinematic prior models, we use PCA priors in a very limited way — to parameterize target poses in the controller so that we can capture important correlations between joint angles that otherwise may not be observable, or well constrained, by the image observations. In doing so, we do not make use of any temporal information present in the data and allow targets to significantly exaggerate poses. We found PCA priors unnecessary when optimizing from less ambiguous data (*e.g.*, reference motion capture or favorable videos).

#### 4.1.3 Transitions

In our framework, state transitions within a state-space controller can happen on timing ($\kappa_i = \mathcal{T}$) or contact events ($\kappa_i \in \{\mathcal{L}, \mathcal{R}\}$). Transitions on timing happen once the simulation time spent in the state is $\geq \nu_i$ for the current state $i$. Transitions on contact events happen when the associated body segment is supported by a contact force, as determined by the simulator.

## 5 Estimating Controllers

In order to obtain a controller that approximates the motion in the video (or in the reference motion capture), we need to estimate both the structure of the controller appropriate for the given motion, $\mathcal{S}_M$ (including number of states $M$ and the types of transitions $\kappa_i$) and parameters, $\Theta$, of the controller optimized to fit the observations and the priors; as well as the initial pose, $\mathbf{x}_0$, and velocities, $\dot{\mathbf{x}}_0$. This optimization amounts to minimizing the objective function, in Eq. (3) with respect to $\mathcal{S}_M$, $\Theta$, $\mathbf{x}_0$ and $\dot{\mathbf{x}}_0$. Our objective function,

$$E(\mathbf{z}_{1:T}) = \lambda_l E_{\text{like}} + \lambda_s E_{\text{stability}} + \lambda_p E_{\text{prior}} + \lambda_c E_{\text{contact}}, \quad (8)$$

contains four terms measuring the inconsistency of the simulated motion produced by the controller with the image-based (or reference motion capture) observations and the quality of the controller itself; where $\lambda_i$ are weighting factors designating the overall importance of the different terms. We now describe the four error terms.

**Image-based Likelihood:** The key component of the objective function is the likelihood term, $E_{\text{like}}$. In the image case, it measures the inconsistency of the simulated motion, $\mathbf{x}_{1:T}$, with the foreground silhouettes, $\mathbf{z}_{1:T}$. Assuming that the likelihood is independent at each frame, given the character's motion, we can measure the inconsistency by adding up contributions from each frame. We determine this contribution by projecting a simulated character into the image (assuming a known camera projection matrix) and computing the difference from image features at each pixel. We adopt a simple silhouette likelihood [Sigal et al. 2010] and define a symmetric distance between the estimated binary silhouette mask, $S_s^e$ (see Figure 2 (c), green blob), at time $s$, and the image binary silhouette mask, $S_t^i$ (see Figure 2 (c), red blob), at time $t$. This approach results in the following formulation for the energy term that we optimize as part of the overall objective:

$$E_{\text{like}} = \sum_{t=1}^{T} \frac{B_{t,t}}{B_{t,t} + Y_{t,t}} + \frac{R_{t,t}}{R_{t,t} + Y_{t,t}}, \quad (9)$$

where the $Y_{t,s}$, $R_{t,s}$ and $B_{t,s}$ terms count pixels in the $S_t^i \wedge S_s^e$, $S_t^i \wedge \neg S_s^e$ and $\neg S_t^i \wedge S_s^e$ masks, that is, $Y_{t,s} = \sum_{(x,y)} S_t^i(x,y) S_s^e(x,y)$, $R_{t,s} = \sum_{(x,y)} S_t^i(x,y)\left[1 - S_s^e(x,y)\right]$ and $B_{t,s} = \sum_{(x,y)} \left[1 - S_t^i(x,y)\right] S_s^e(x,y)$. We estimate silhouettes in video using a standard background subtraction algorithm [Elgammal et al. 2000]. The background model comprised the mean color image and intensity gradient, along with a single 5D covariance matrix (estimated over the entire image).

**Motion Capture Likelihood:** To use reference motion capture data instead of video data to estimate controllers, we define a motion capture likelihood as a sum of squared differences between the markers attached to the observed skeleton and the simulated motion. In particular, we attach three markers to every segment and let

$$E_{\text{like}} = \sum_{t=1}^{T} \sum_{j=1}^{18} \sum_{k=1}^{3} ||m_{t,j,k}^i - m_{t,j,k}^e||_2^2 \quad (10)$$

where $m_{t,j,k}^i \in \mathbb{R}^3$ is the location of the $k$-th marker attached to the $j$-th segment at time $t$ (computed using forward kinematics) from the reference motion capture and $m_{t,j,k}^e \in \mathbb{R}^3$ is the location of $k$-th marker attached to the $j$-th segment of the simulated character.

**Stability:** The likelihood defined above will result in controllers that may fail near the end of the sequence because the optimization has a short horizon. We make an assumption that subjects in our video sequence end up in a stable posture. To ensure that the controller ends up in a similar, statically stable pose, we add an additional term that measures inconsistency of the simulation for time $\Delta T$ past time $T$, where $T$ is the time of the last observation. We reuse the formulation of $E_{\text{like}}$ in Eq. (9) and define this term as

$$E_{\text{stability}} = \sum_{t=T+1}^{T+\Delta T} \frac{B_{T,t}}{B_{T,t} + Y_{T,t}} + \frac{R_{T,t}}{R_{T,t} + Y_{T,t}}. \quad (11)$$

**Prior:** To bias the solution towards more likely interpretations, we propose a prior over the state-space control parameters. Generally, there are four types of parameters that we optimize: representation of the target poses for atomic controllers, $\mathbf{s}_i$, parameters of atomic controllers, $\alpha_i$, transition times, $\nu_i$, and balance feedback parameters, $\vartheta_i$; in addition, we optimize the initial pose $\mathbf{x}_0$ and velocities $\dot{\mathbf{x}}_0$. For $\alpha_i, \nu_i, \vartheta_i$ we use uninformative uniform priors over the range of possible values: $\alpha_i \sim \mathcal{U}(0.001, 0.2)$, $\nu_i \sim \mathcal{U}(0.1, 0.5)$, $\vartheta_{i,1} \sim \mathcal{U}(-1, 1)$, $\vartheta_{i,2} \sim \mathcal{U}(-1, 1)$ and $\vartheta_{i,3} \sim \mathcal{U}(0, 1)$. We also impose a uniform prior on $\mathbf{s}_i$, $\mathbf{s}_i \sim \mathcal{U}(-4\sigma, 4\sigma)$. The uniform priors over parameters are encoded using a linear penalty on the values of the parameters that are outside the valid range. In particular, for every variable, $v$, with a uniform prior $v \sim \mathcal{U}(a, b)$, we add the following term to the prior, $E_{\text{prior}}(\Theta)$: $|\max(0, v - b)| + |\min(0, v - a)|$.

[3] We encode each angle as $[sin(\cdot), cos(\cdot)]$ to avoid wrap around.

**Contact:** In our experience, controllers optimized with these objectives often result in frequent contact changes. This issue is particularly problematic for low clearance motions like walking. For example, a walking controller may stumble slightly if that helps to produce a motion that is more similar to what is observed in video or reference motion capture. This behavior hinders the ability of the controller to be robust to perturbations in the environment. To address this issue, we assume that there is no more than one contact state change between any two consecutive atomic actions in the state-space controller. This policy is motivated by the observation that contact state change yields discontinuity in the dynamics and hence must be accompanied by a state transition. State transitions, however, may happen for other reasons (e.g., performance style). To encode this knowledge in our objective, we define a contact state change penalty term: $E_{\text{contact}} = \sum_{i=i}^{M-1} c(i)$, where

$$c(i) = \begin{cases} 0 & \text{0 or 1 contact change between } \odot_i \text{ and } \odot_{i+1} \\ 10,000 & > \text{1 contact change between } \odot_i \text{ and } \odot_{i+1} \end{cases}$$

We define a contact state change as one or more body segments changing their contact state with respect to the environment.

## 5.1 Optimization

Now that we have defined our objective function, we need to optimize it with respect to $\mathcal{S}_M$, $\Theta$, $\mathbf{x}_0$, $\dot{\mathbf{x}}_0$ to obtain the controller of interest. The parameters $\Theta$ are in general closely tied to the structure of the controller $\mathcal{S}_M$. We first formulate a batch optimization over the controller parameters assuming a known and fixed controller structure $\mathcal{S}_M$ and then introduce an iterative optimization over the structure itself by utilizing the batch optimization as a sub-routine.

For all our optimizations we use gradient-free Covariance Matrix Adaptation (CMA) [Hansen 2006]. CMA is an iterative genetic optimization algorithm that maintains a Gaussian distribution over parameter vectors. The Gaussian is initialized with a mean, $\mu$, encoding initial parameters, and diagonal spherical covariance matrix with variance along each dimension equal to $\sigma^2$. CMA proceeds by: (1) sampling a set of random samples from this Gaussian, (2) evaluating the objective $E(\mathbf{x}_{1:T})$ for each of those samples (this step involves simulation), and (3) producing a new Gaussian based on the most promising samples and the mean. The number of samples to be evaluated and to be used for the new mean is chosen automatically by the algorithm based on the problem dimensionality.

### 5.1.1 Batch Optimization of Controller Parameters

Given a state-space controller structure, $\mathcal{S}_M$, batch optimization infers the controller parameters $\Theta = [\,(\mathbf{s}_1, \alpha_1, \vartheta_1), (\mathbf{s}_2, \alpha_2, \vartheta_2), \ldots, (\mathbf{s}_M, \alpha_M, \vartheta_M),\ \nu_1, \nu_2, \ldots\,]$ and the initial pose $\mathbf{x}_0$ and velocity $\dot{\mathbf{x}}_0$ of the character by minimizing the objective function $E(\mathbf{z}_{1:T})$. The optimization procedure (integrated with CMA) is illustrated in Algorithm 2. Batch optimization is useful when we have a reasonable guess for the controller structure. Batch optimization of cyclic controllers is particularly beneficial because weak observations for one motion cycle can be reinforced by evidence from other cycles, making the optimizations less sensitive to observation noise and less prone to overfitting to local observations.

Reliance of $\Theta$ on the controller structure $\mathcal{S}_M$ suggests an approach where we enumerate the controller structures, estimate parameters in a batch, as above, and choose the best structure based on the overall objective value. Unfortunately, such an enumeration strategy is impractical because the number of possible controllers is exponential in $M$. For example, see Figure 8 for an illustration of a complicated gymnastics motion that we estimate to require $M = 20$ states. In addition, without good initialization, in our experience, optimization of the high-dimensional parameter vector often gets stuck in local optima. To alleviate these problems, we propose an approach for estimating the controller structure incrementally that also gives us an ability to have better initializations for optimization of control parameters.

**Algorithm 2** : Controller batch optimization using CMA
$[\Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = \textbf{BatchOp}\ (\mathcal{S}_M, \mathbf{x}_0, \mathbf{Z}, \mathbf{U}, \mathbf{b}, \Theta, \dot{\mathbf{x}}_0)$

**Input:** State-space controller structure ($\mathcal{S}_M$); initial pose ($\mathbf{x}_0$); PCA prior ($\mathbf{U}$, $\mathbf{b}$); observations / image features ($\mathbf{Z} = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T\}$)
**Optional Input:** Controller parameters ($\Theta$); initial velocity ($\dot{\mathbf{x}}_0$);
**Output:** Controller parameters ($\Theta$); initial pose ($\mathbf{x}_0$); initial velocity ($\dot{\mathbf{x}}_0$); objective value ($E$)

1: **if** $\dot{\mathbf{x}}_0 = \emptyset, \Theta = \emptyset$ **then**
2: Initialize initial velocity: $\dot{\mathbf{x}}_0 = \mathbf{0}$
3: Initialize controller parameters ($\Theta$):
$\mathbf{s}_i = \mathbf{s}_0, \alpha_i = 0.1, \vartheta_i = [0, 0, 0], \nu_i = 0.25\ \forall i \in [1, M]$
4: **end if**
5: Project initial pose onto PCA space: $\mathbf{s}_0 = \mathbf{U}^{-1}(\mathbf{x}_0 - \mathbf{b})$
6: Initialize variance: $\Sigma = \mathbf{I}\sigma$
7: Initialize mean: $\mu = [\Theta, \mathbf{s}_0, \dot{\mathbf{x}}_0]^T$
8: **for** $i = 1 \rightarrow N_{ITER}$ **do**
9: **for** $j = 1 \rightarrow N_{POPULATION}$ **do**
10: Sample controller parameters and initial pose:
$[\Theta^{(j)}, \mathbf{s}_0^{(j)}, \dot{\mathbf{x}}_0^{(j)}] \sim \mathcal{N}(\mu, \Sigma)$
11: Reconstruct initial pose:
$\mathbf{x}_0^{(j)} = \mathbf{U}\mathbf{s}_0^{(j)} + \mathbf{b}$
12: **for** $t = 1 \rightarrow T + \Delta T$ **do**
13: Control and simulation:
$\begin{bmatrix} \mathbf{x}_t^{(j)} \\ \dot{\mathbf{x}}_t^{(j)} \end{bmatrix} = f\left(\begin{bmatrix} \mathbf{x}_{t-1}^{(j)} \\ \dot{\mathbf{x}}_{t-1}^{(j)} \end{bmatrix}, \pi\left(\begin{bmatrix} \mathbf{x}_{t-1}^{(j)} \\ \dot{\mathbf{x}}_{t-1}^{(j)} \end{bmatrix}, \Theta^{(j)}\right)\right)$
14: **end for**
15: Compute objective:
$E^{(j)} = \lambda_l E_{\text{like}} + \lambda_s E_{\text{stability}} + \lambda_p E_{\text{prior}} + \lambda_c E_{\text{contact}}$
16: **end for**
17: $[\mu, \Sigma] = \text{CMA\_update}\ (\mu, \Sigma, \{\Theta^{(j)}, \mathbf{s}_0^{(j)}, \dot{\mathbf{x}}_0^{(j)}, E^{(j)}\})$
18: **end for**
19: Let $j^* = \arg\min_j E^{(j)}$
20: **return** $\Theta^{(j^*)}$, $\mathbf{x}_0^{(j^*)}$, $\dot{\mathbf{x}}_0^{(j^*)}$, $E^{(j^*)}$

### 5.1.2 Incremental Optimization of Controller Structure

Our key observation is that we can optimize controllers incrementally, such that the controller structure can be estimated locally and simultaneously with estimation of control parameters. Our approach greedily selects the structure and the parameters of the controller as new observations are added (see Figure 5). The basic idea is to divide the hard high-dimensional batch optimization over the entire sequence into a number of easier lower-dimensional optimization problems over an expanding motion window. To this end, we start with a simple generic initial controller with one state and incrementally construct a state chain in stages that eventually explains the entire motion observed in the video. At each stage, we expand the current window by $T_s$ frames and re-optimize the current controller to fit the frames in the window, adding new states to the chain as is necessary (see Algorithm 3).

The incremental optimization proceeds in stages. At the first stage, the first $T_s$ frames of the video sequence (the initial motion window) are optimized using batch optimization (see Algorithm 3, lines 2-3), assuming a fixed initial controller structure with only one state, $\mathcal{S}_1 = \{\oplus \longrightarrow \odot_1\}$. At all later stages, the current motion window is expanded by the subsequent $T_s$ frames from the video sequence (or reference motion capture) and the new window is re-optimized using a *number of local optimizations* until the whole motion is processed. The purpose of the local optimizations is to propose possible updates to the current controller (mainly the addition of a state to the current chain). At each stage, the addition of a state to the current chain is proposed and tested by re-optimizing the controller with and without this addition. The controller for

**Algorithm 3** : Incremental optimization of chain controller $[\mathcal{S}_M, \Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = \textbf{IncremOp}\ (\mathbf{x}_0, \mathbf{Z}, \mathbf{U}, \mathbf{b})$

**Input:** Initial pose ($\mathbf{x}_0$); PCA prior ($\mathbf{U}$, $\mathbf{b}$); observations / image features ($\mathbf{Z} = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T\}$)
**Output:** State-space controller structure ($\mathcal{S}_M$); controller parameters ($\Theta$); initial pose ($\mathbf{x}_0$); initial velocity ($\dot{\mathbf{x}}_0$); objective value ($E$)

Number of observations to add per stage:
  $T_s = T/N_{STAGES}$
Initialize controller structure:
  $M = 1 \qquad \mathcal{S}_1 = \{\oplus \longrightarrow \odot_1\}$
Optimize parameters:
  $[\Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = \textbf{BatchOp}\ (\mathcal{S}_1, \mathbf{x}_0, \mathbf{z}_{1:T_s}, \mathbf{U}, \mathbf{b})$
**for** $i = 2 \rightarrow N_{STAGES}$ **do**
  Re-optimize parameters:
    $[\Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = \textbf{BatchOp}\ (\mathcal{S}_M, \mathbf{x}_0, \mathbf{z}_{1:iT_s}, \mathbf{U}, \mathbf{b}, \Theta, \dot{\mathbf{x}}_0)$
  Try to add a state:
    $\mathcal{S}_M^+ = \{\mathcal{S}_M \overset{\kappa_M = \mathcal{T}}{\longrightarrow} \odot_{M+1}\}$
    $[\Theta^+, \mathbf{x}_0^+, \dot{\mathbf{x}}_0^+, E^+] = \textbf{BatchOp}\ (\mathcal{S}_M^+, \mathbf{x}_0, \mathbf{z}_{1:iT_s}, \mathbf{U}, \mathbf{b}, \Theta, \dot{\mathbf{x}}_0)$
  **if** $E^+ < E$ **then**
    $\mathcal{S}_{M+1} = \mathcal{S}_M^+ \qquad M = M + 1$
    $[\Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = [\Theta^+, \mathbf{x}_0^+, \dot{\mathbf{x}}_0^+, E^+]$
  **end if**
**end for**

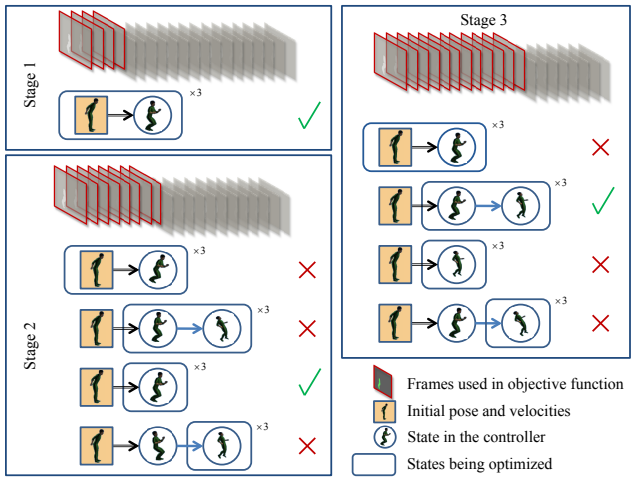



**Figure 5: Incremental Optimization:** *Illustration of the incremental construction of the state chain. At each stage, a number of local changes are proposed to the current controller. The new controllers are reoptimized and the controller with the lowest objective value is chosen for the next stage.*

the next stage is chosen based on the best objective value after re-optimization. See Figure 5 for an illustration of the process.

**Implementation Details:** To avoid the optimization of longer and longer chains and to make the optimizations efficient, we only optimize the last one or two states in the controller (this approximation gives us a fixed compute time regardless of the number of states in the controller structure). Note this approximation is not illustrated in Algorithm 3 due to the complexity of notation it would entail (but it is illustrated in Figure 5). In addition, we found it effective to run multiple **BatchOp** optimizations when executing Algorithm 3. In particular, for every instance of **BatchOp** in Algorithm 3 we run six **BatchOp** optimizations: optimizing the last one or two states, each with three different seeds, and choose the best.

### 5.1.3 Controller Structure Refinement

The incremental optimization described in the previous section allows us to fit control to video or reference motion capture. The chain structure of the controller and transitions on timing make the controller well behaved and easier to optimize. However, this control structure may not be *optimal* in terms of stability or compactness. Therefore, we propose an additional refinement on the controller structure.

**Algorithm 4** : Complete optimization with structure refinement $[\mathcal{S}_M, \Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = \textbf{IncremPlusRefinement}\ (\mathbf{x}_0, \mathbf{Z}, \mathbf{U}, \mathbf{b})$

**Input:** Initial pose ($\mathbf{x}_0$); PCA prior ($\mathbf{U}$, $\mathbf{b}$); observations / image features ($\mathbf{Z} = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T\}$)
**Output:** State-space controller structure ($\mathcal{S}_M$); controller parameters ($\Theta$); initial pose ($\mathbf{x}_0$); initial velocity ($\dot{\mathbf{x}}_0$); objective value ($E$)

Incremental optimization:
  $[\mathcal{S}_M, \Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = \textbf{IncremOp}\ (\mathbf{x}_0, \mathbf{Z}, \mathbf{U}, \mathbf{b})$
Structure transformation (for contact transitions):
  $\mathcal{S}_M^{'} = \mathbb{T}_{\perp}(\mathcal{S}_M)$
  $[\Theta^{'}, \mathbf{x}_0^{'}, \dot{\mathbf{x}}_0^{'}, E^{'}] = \textbf{BatchOp}\ (\mathcal{S}_M^{'}, \mathbf{x}_0, \mathbf{Z}, \mathbf{U}, \mathbf{b}, \Theta, \dot{\mathbf{x}}_0)$
**if** $E^{'} < E(1+\delta)$ **then**
  $[\mathcal{S}_M, \Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = [\mathcal{S}_M^{'}, \Theta^{'}, \mathbf{x}_0^{'}, \dot{\mathbf{x}}_0^{'}, E^{'}]$
**end if**
Structure transformation (for cyclicity):
  $\mathcal{S}_M^{'} = \mathbb{T}_{\infty}(\mathcal{S}_M)$
  $[\Theta^{'}, \mathbf{x}_0^{'}, \dot{\mathbf{x}}_0^{'}, E^{'}] = \textbf{BatchOp}\ (\mathcal{S}_M^{'}, \mathbf{x}_0, \mathbf{Z}, \mathbf{U}, \mathbf{b}, \Theta, \dot{\mathbf{x}}_0)$
**if** $E^{'} < E(1+\delta)$ **then**
  $[\mathcal{S}_M, \Theta, \mathbf{x}_0, \dot{\mathbf{x}}_0, E] = [\mathcal{S}_M^{'}, \Theta^{'}, \mathbf{x}_0^{'}, \dot{\mathbf{x}}_0^{'}, E^{'}]$
**end if**

We make the following observation: there exists an equivalence class of controllers all of which can simulate the same motion. For example, a one-and-a-half cycle walking controller can be represented in at least three different ways:
(1) using a chain controller with transitions on timing:
$\mathcal{S}_6 = \{\oplus \longrightarrow \odot_1 \overset{\kappa_1 = \mathcal{T}}{\longrightarrow} \odot_2 \overset{\kappa_2 = \mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3 = \mathcal{T}}{\longrightarrow} \odot_4 \overset{\kappa_4 = \mathcal{T}}{\longrightarrow} \odot_5 \overset{\kappa_5 = \mathcal{T}}{\longrightarrow} \odot_6\}$
(2) using a chain controller with some transitions on contact:
$\mathcal{S}_6^{'} = \{\oplus \longrightarrow \odot_1 \overset{\kappa_1 = \mathcal{L}}{\longrightarrow} \odot_2 \overset{\kappa_2 = \mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3 = \mathcal{R}}{\longrightarrow} \odot_4 \overset{\kappa_4 = \mathcal{T}}{\longrightarrow} \odot_5 \overset{\kappa_5 = \mathcal{L}}{\longrightarrow} \odot_6\}$, or
(3) using a cyclic controller:
$\mathcal{S}_4 = \{\oplus \longrightarrow \odot_1 \overset{\kappa_1 = \mathcal{L}}{\longrightarrow} \odot_2 \overset{\kappa_2 = \mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3 = \mathcal{R}}{\longrightarrow} \odot_4 \overset{\kappa_4 = \mathcal{T}}{\longrightarrow} \odot_1\}$.
All of these controllers produce exactly the same simulation results with the same atomic action controllers (assuming stable gait and that transitions on time in $\mathcal{S}_6$ are chosen coincident with contact events in $\mathcal{S}_6^{'}$ and $\mathcal{S}_4$). However, $\mathcal{S}_6^{'}$ and $\mathcal{S}_4$ are more robust and $\mathcal{S}_4$ is more compact in terms of representation; so in practice we would likely prefer $\mathcal{S}_4$ over the alternatives.

In the chain controllers obtained in the previous section, we often see that transitions on time encoded in the control structure coincide with contact events (within a few frames). In fact, the contact term in our objective function implicitly helps to enforce such transitions. We take advantage of this observation and propose two transformation functions that can take the chain controllers and transform their structure to make them more robust and compact as above (Algorithm 4). We define a transformation $\mathcal{S}_M^{'} = \mathbb{T}_{\perp}(\mathcal{S}_M)$ that replaces transitions on timing with appropriate transitions on contact in $\mathcal{S}_M$ if the timing transition is within $0.2\ s$ of the contact (and there is only one contact change within the $0.2\ s$ window). Because timing events often do not happen exactly on contact, but close to it (hence the time threshold), we also re-optimize parameters with the $\Theta$ obtained in Section 5.1.2 as the initial guess. Similarly, we define a transformation that greedily looks for cyclicity $\mathcal{S}_M^{'} = \mathbb{T}_{\infty}(\mathcal{S}_M)$ by comparing the type of transition, target pose and control parameters to previous states. Again, after the transformation was applied, the parameters are re-optimized with a good initial guess to account for minor misalignment. Transformed controllers are chosen instead of the simple chain controller if the resulting objective value is within $\delta = 15\%$ of the original. This use of $\delta$ approximates the minimum description length criterion that allows us to trade off fit to data for compactness of representation.

| Motion (mocap) | Frames | Batch | Inc | Refined | Structure |
|---|---|---|---|---|---|
| Fast walk | 207 | 3.9 *cm* | 2.3 *cm* | 2.4 *cm* | $\oplus \longrightarrow \odot_1 \overset{\kappa_1=\mathcal{R}}{\longrightarrow} \odot_2 \overset{\kappa_2=\mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3=\mathcal{L}}{\longrightarrow} \odot_4 \overset{\kappa_4=\mathcal{T}}{\longrightarrow} \odot_5 \overset{\kappa_5=\mathcal{R}}{\longrightarrow} \odot_2$ |
| Jump | 241 | 7.4 *cm* | 4.6 *cm* | 4.3 *cm* | $\oplus \longrightarrow \odot_1 \overset{\kappa_1=\mathcal{T}}{\longrightarrow} \odot_2 \overset{\kappa_2=\mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3=\mathcal{T}}{\longrightarrow} \odot_4 \overset{\kappa_4=\mathcal{T}}{\longrightarrow} \odot_5$ |
| Twist jump | 226 | 7.0 *cm* | 4.4 *cm* | 4.4 *cm* | $\oplus \longrightarrow \odot_1 \overset{\kappa_1=\mathcal{T}}{\longrightarrow} \odot_2 \overset{\kappa_2=\mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3=\mathcal{T}}{\longrightarrow} \odot_4 \overset{\kappa_4=\mathcal{L}}{\longrightarrow} \odot_5 \overset{\kappa_5=\mathcal{T}}{\longrightarrow} \odot_6$ |
| Spin kick | 131 | 10.6 *cm* | 9.4 *cm* | 9.2 *cm* | $\oplus \longrightarrow \odot_1 \overset{\kappa_1=\mathcal{T}}{\longrightarrow} \odot_2 \overset{\kappa_2=\mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3=\mathcal{T}}{\longrightarrow} \odot_4$ |
| Cartwheel | 175 | 21.5 *cm* | 11.8 *cm* | 11.8 *cm* | $\oplus \longrightarrow \odot_1 \overset{\kappa_1=\mathcal{T}}{\longrightarrow} \odot_2 \overset{\kappa_2=\mathcal{T}}{\longrightarrow} \odot_3 \overset{\kappa_3=\mathcal{R}}{\longrightarrow} \odot_4 \overset{\kappa_4=\mathcal{T}}{\longrightarrow} \odot_5 \overset{\kappa_5=\mathcal{L}}{\longrightarrow} \odot_6 \overset{\kappa_6=\mathcal{T}}{\longrightarrow} \odot_7$ |
| Back handspring | 775 | — | 6.3 *cm* | 6.3 *cm* | $\oplus \longrightarrow \odot_1 \overset{\kappa_1=\mathcal{T}}{\longrightarrow} \odot_2 \overset{\kappa_2=\mathcal{T}}{\longrightarrow} \ldots \overset{\kappa_{18}=\mathcal{T}}{\longrightarrow} \odot_{19} \overset{\kappa_{19}=\mathcal{T}}{\longrightarrow} \odot_{20}$ |

**Figure 6: Controller optimization from reference motion capture:** *Incremental optimization performs better than batch optimization, and the refinement step further improves the fit resulting in a more robust controller structure.*

| Motion (video) | Frames | Batch | Inc | Refined |
|---|---|---|---|---|
| Fast walk | 207 | 6.7 *cm* | 4.2 *cm* | 4.2 *cm* |
| Jump | 241 | 9.5 *cm* | 6.8 *cm* | 6.8 *cm* |
| Twist jump | 226 | 11.9 *cm* | 9.6 *cm* | 9.6 *cm* |
| Back handspring | 154 | 33.3 *cm* | 17.6 *cm* | 17.5 *cm* |
| VideoMocap jump | 91 | N/A | N/A | N/A |

**Figure 7: Controller optimization from monocular video:** *Incremental optimization performs considerably better than batch.*

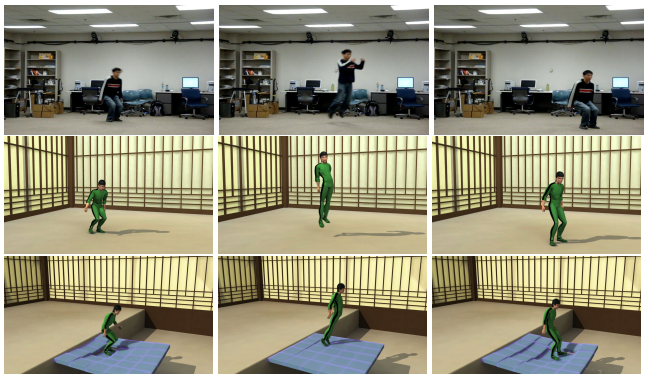


**Figure 9: Video Result:** *Input jump sequence from VideoMocap (top), reconstructed motion simulated by the controller estimated from the video (middle), new motion simulated in a modified environment (bottom).*

## 6 Experiments

We collected a dataset consisting of motions with synchronized video and motion capture data. Motion capture data was recorded at 120 *Hz* and video at 60 *Hz* using a progressive scan 1080P camera. To speed up the likelihood computations, we sub-sample observed silhouette images to $160 \times 90$. Motions in this dataset include: jump, twist jump, spin kick, back handspring, cartwheel and fast walk. To enable comparison with previous work, we also make use of the 30 *Hz* jump sequence from VideoMocap [Wei and Chai 2010] in our experiments. For sequences where calibration is unavailable, we solve for calibration using an annotation of joint locations in the first frame, assuming a fixed skeleton for the character. While the appearance of the subject in all these sequences is relatively simple, we want to stress that we make no use of this aspect in our likelihood or optimization and only use generic foreground silhouettes as observations. All PCA models were learned from a single adult male subject[4], but are applied to other subjects in our test set (*e.g.*, jump from [Wei and Chai 2010]).

[4]We took motion capture from the subject performing fast walk, jump, twist jump and spin kick to train our PCA models. For those sequences, we used data from disjoint trials for training and testing.

We have estimated controllers from motion capture and video. We test performance of the recovered controllers in their ability to reproduce observed motions as well as to synthesize new motions in novel environments and under the effect of external disturbances. In all our experiments, we only rely on manual specification of a rough estimate of the initial pose for the character and, for the video sequences, corresponding camera calibration information. We use the same generic model of the human body with the same mass properties in all cases. We employ a marker-based error metric from [Sigal et al. 2010] to evaluate accuracy of our pose reconstructions against the ground truth, and report Euclidean joint position distance averaged across joints and frames in *cm*. We first show results using reference motion capture as an input and then illustrate our algorithm on monocular videos.

### 6.1 Reference Motion Capture

We compare performance of the batch (**Batch**) method, that optimizes parameters given a known controller structure $\mathcal{S}_M$ (but no initialization except for the initial pose), to the incremental (**Inc**) method and the incremental method with refinement (**Refined**) that recover the structure automatically (Figure 6). Despite the fact that the batch method contains more information (controller structure), our incremental method is substantially more accurate and the refinement step further improves the performance in all but one case. Notice that for the back handspring, we were unable to run the batch method, as it required optimization over a prohibitive number of states and never converged. Furthermore, for a complicated motion, knowing or guessing the controller structure *a priori* is difficult. Our incremental method can estimate this structure automatically. The structure of the resulting controllers, after refinement, are illustrated in Figure 6 (last column). As a result of our optimization, we automatically build controllers with varying numbers of states (from 4 to 20) and the controllers can be both linear or cyclic (as in fast walk).

We obtain controllers that represent the motion well, even in the case of a long and complex gymnastic sequence in Figure 8 (top and middle). In some cases, the quantitative errors are higher. We attribute these errors to the sparse nature of our controller (that inadvertently approximates the motion) and the inability of the optimizer to find the global optimum. Despite quantitative differences, the results are dynamically and visually similar (see video). We are able to improve performance by reducing the length of our stages, where the solution in the limit approximates inverse dynamics. However, controllers obtained this way are less responsive and are unusable for video because they severely overfit.

**Robustness to changes in environment:** We test robustness of the recovered controllers by altering the environment, as shown in Figure 8 (bottom). Despite the dynamic environment and different geometry, the character is able to adapt and perform the desired motion. Further environment alteration tests are illustrated in the next section, where we recover controllers from video.

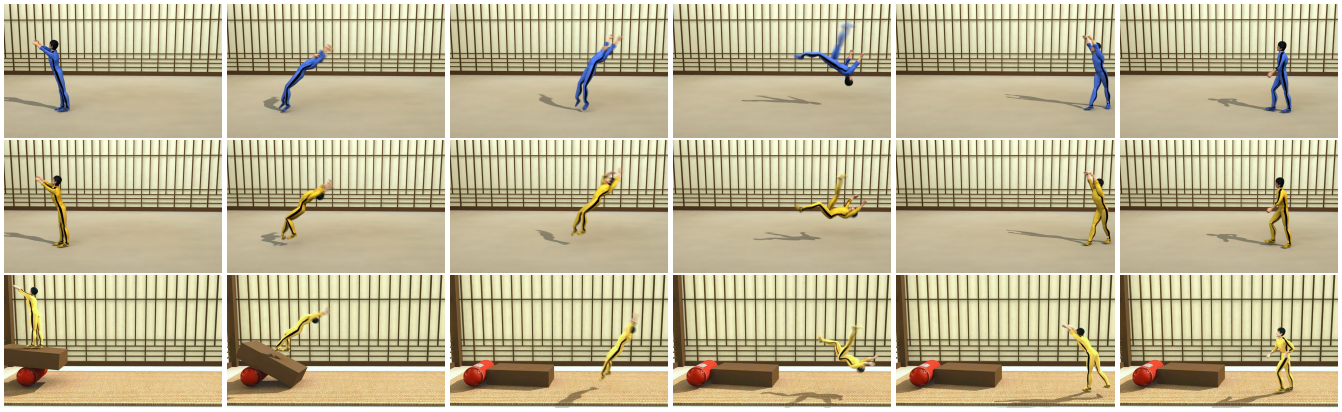


**Figure 8: Motion Capture Result:** *Input reference motion capture (top), reconstructed motion simulated with our controller (middle), simulated motion in a new environment (bottom).*

**Robustness to pushes:** We quantify robustness of our controllers to external disturbances with push experiments; after [Wang et al. 2009; Lee et al. 2010]. Because we have a different model for actuation and dynamics, it is difficult to match conditions to [Wang et al. 2009; Lee et al. 2010] exactly. For this experiment, we choose a cyclic walking controller that results from optimization with reference motion capture as an input. Despite the fact that we only optimize the controller to match roughly 1.5 cycles of the motion, the controller is able to produce a mostly stable gait (it sometimes stumbles after 7 to 10 walk cycles, but always recovers afterwards). We perform a test where we apply a push force to the center of mass of the torso for 0.4 seconds once every 4 seconds. The controller passes the test if the character still walks after 40 seconds. Our controller can resists pushes of up to 210 $N$, 294 $N$, 273 $N$ and 294 $N$ from front, rear, right and left sides respectively when we use relatively large joint torque bounds. When the controller has 75% of the joint torque limits (just enough for the controller to walk without substantially altering the gait), the character survives pushes of up to 42 $N$, 67 $N$, 33 $N$ and 42 $N$ respectively.

### 6.2 Single-view Video

We report quantitative performance for controller optimization from single-view video in Figure 7. The reconstruction error is on average only 80% higher than that of optimizations from motion capture, despite a substantially more complex problem and optimization. For the batch optimizations, we report the best results out of 15 independent optimization runs. In many cases, other runs produced much inferior results. Our incremental method is much more repeatable. Unlike optimizations from reference motion capture, we see less of a benefit for structure refinement in reducing quantitative error when optimizing from video. The objective function is multimodal in this case, with the modes far apart (a well known issue in the marker-less motion capture literature). While the incremental optimization is very effective in finding good local optima, escaping those optima is difficult through refinement.

**Robustness to changes in environment:** We made a number of alterations to the environment. *Jump:* For a jump, we changed the geometry of the ground plane, attached 3.5 $kg$ skis to the feet and dialed down the coefficient of friction by a factor of 2,000; these alterations emulate a very slippery skiing slope. The results of such alterations can be seen in Figure 1 and the accompanying video. Notice that the character is able to perform the desired motion while dynamically adjusting to the changed environment and maintaining balance at the end. *Twist jump:* In this case, we give the character a 13 $kg$ mass to carry with its right arm as it performs the jump. This modification alters the upper body motion throughout the sequence. At the end of the simulation the character leans forward to counterbalance the weight in the hand, which needs to be held behind the torso according to the last controller state. These results can be seen in the video.

**Comparison to Wei and Chai [2010]:** We also compare the performance of our method to the one described in [Wei and Chai 2010]. The two methods have a similar goal, but estimate the motion in very different ways. In [Wei and Chai 2010] six keyframe annotations, specification of the contact states and four additional 2D refinement constraints have to be manually specified for the tested jump sequence. In contrast, our approach is fully automatic and requires only specification of a rough initial pose at the first frame; the contact states are recovered implicitly through optimization. While we are not able to recover the hand motion (in part because the jumps in our PCA model are quite different from the one in video), we are able to generate a controller capable of replaying the motion with all the intricacies of the dynamics. Furthermore, because we are recovering a controller, we are able to put the character in a different environment (on a trampoline) and see the motion under physical and, in this case, dynamic perturbations (Figure 9). Our fully automatic method achieves average reconstruction errors as low as 4.2 $cm$ which is comparable to the average error of 4.5 $cm$ reported in [Wei and Chai 2010].

**Computation time:** Incremental optimization for 200 frames of motion capture data takes about 2.75 hours; the refinement procedure takes 15 minutes (single eight-core machine with a 2.8 $GHz$ Core i7 CPU and 4 $GB$ of RAM) — approximately 3 hours total. Video optimizations take 40% longer due to use of more expensive likelihoods.

## 7 Discussion

We presented a novel approach for capturing bipedal controllers directly from single-view video and in the process reconstructing human motion from a single video stream. In doing so we simultaneously tackled two very challenging problems: bipedal control from noisy data and marker-less motion capture. We believe that by formulating the problem in such a way as to solve the two problems simultaneously, we are better leveraging the information present in the video of human motion and thereby simplifying the marker-less tracking problem. We are able to estimate controllers for a variety of complex and highly dynamic motions. Controllers from both sources of data are robust and can be used in new environments with novel terrain and dynamic objects.

While we are able to estimate controllers that are capable of generating motion, for a variety of human behaviors, some of the subtleties in the motion are not accurately reconstructed; the motions tend to look robotic and overpowered. We believe there are three main sources of errors that are contributing to these artifacts. First, the end effectors (feet and hands) are hard to observe in video and are not well constrained by the image likelihood model. From a sagittal view, it is also difficult to disambiguate single stance from double stance. Second, parameters found by CMA are not guaranteed to be optimal, because we are solving a very difficult high-dimensional optimization problem. Third, we believe the overpowered balancing behavior observed is due to relatively high joint torque bounds — our character by design has very strong joints, including ankles and knees. Higher bounds make the optimization easier. Improvements could probably be achieved for locomotion and other highly dynamic behaviors by having more sophisticated objectives that include energetics, as in [Wang et al. 2010]; we plan to explore these objectives in the future.

While we took a puristic approach of only using single-view video, in practice multiple views (or Xbox's Kinect data) will undoubtedly simplify the problem and lead to an improved ability to capture 3D motion. While we do not utilize user-in-the-loop constraints, as we opted for a completely automatic method, those can easily be added to further refine the motions as was done by Wei and Chai [2010]. Preliminary experiments show that we can indeed achieve better performance by having the user click on the joint locations in the image. Our experiments with motion capture data (perfect 3D data) further demonstrate this point.

The PCA priors that we introduced for representing the target poses for the state proved to be quite useful. However, they are limited to the motions we trained them on and not all stylistic variations can be captured by a discrete set of such learned models. In the future, we hope to explore refinement schemes where we can estimate controllers with the motion prior and then perhaps refine them by removing the prior term from the objective.

**Acknowledgments:** This work was supported in part by ONR PECASE Award N000140810910 and ONR YIP Award N000140710141. We would like to thank Moshe Mahler and Valeria Reznitskaya for model creation, skinning and rendering; Xiaolin Wei and Jinxiang Chai for data from [Wei and Chai 2010]; Katsu Yamane, David J. Fleet and Jack M. Wang for useful discussions.

## References

BARAFF, D. 1996. Linear-time dynamics using Lagrange multipliers. In *ACM SIGGRAPH*.

BHAT, K. S., SEITZ, S. M., POPOVIC, J., AND KHOSLA, P. 2002. Computing the physical parameters of rigid-body motion from video. In *ECCV*, 551–566.

BO, L., AND SMINCHISESCU, C. 2010. Twin Gaussian processes for structured prediction. *IJCV 87*, 1–2.

BRUBAKER, M. A., AND FLEET, D. J. 2008. The Kneed Walker for human pose tracking. In *IEEE CVPR*.

BRUBAKER, M. A., FLEET, D. J., AND HERTZMANN, A. 2007. Physics-based person tracking using simplified lower body dynamics. In *IEEE CVPR*.

BRUBAKER, M. A., SIGAL, L., AND FLEET, D. J. 2009. Estimating contact dynamics. In *ICCV*.

CHU, D., SHAPIRO, A., ALLEN, B., AND FALOUTSOS, P. 2007. A dynamic controller toolkit. In *ACM SIGGRAPH Video Game Symposium (Sandbox)*, 21–26.

CLINE, M. 2002. *Rigid Body Simulation with Contact and Constraints*. Master's thesis, The University of British Columbia.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2009. Robust task-based control policies for physics-based characters. In *ACM Transactions on Graphics*, vol. 28.

CRISIS. 2006. *http://crisis.sourceforge.net/*.

DEMPSTER, W. T. 1955. Space requirements of the seated operator: Geometrical, kinematic, and mechanical aspects of the body with special reference to the limbs. Tech. rep., Wright-Patterson Air Force Base 55-159.

ELGAMMAL, A., HARWOOD, D., AND DAVIS, L. 2000. Nonparametric model for background subtraction. In *ECCV*.

GALL, J., STOLL, C., DE AGUIAR, E., THEOBALT, C., ROSENHAHN, B., AND SEIDEL, H.-P. 2009. Motion capture using joint skeleton tracking and surface estimation. In *IEEE CVPR*, 1746–1753.

HANSEN, N. 2006. The CMA evolution strategy: A comparing review. *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms.*, 75–102.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *ACM SIGGRAPH*, 71–78.

KWON, T., AND HODGINS, J. 2010. Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Transactions on Graphics 29*, 4.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Transactions on Graphics 29*, 4.

METAXAS, D., AND TERZOPOULOS, D. 1993. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on PAMI 15*, 6, 580–591.

MUICO, U., LEE, Y., POPOVIC, J., AND POPOVIC, Z. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics 28*, 3.

NGO, T., AND MARKS, J. 1993. Spacetime constraints revisited. In *ACM SIGGRAPH*.

ODE. 2006. *http://www.ode.org/*.

PANG, J. S., AND FACCHINEI, F. 2003. Finite-dimensional variational inequalities and complementarity problems (i). Springer.

SCHAAL, S., AND SCHWEIGHOFER, N. 2005. Computational motor control in humans and robots. *Cur. Op. in Neurobio.*, 6.

SIGAL, L., BALAN, A., AND BLACK, M. J. 2010. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision 87*, 1–2, 4–27.

SILVA, M. D., ABE, Y., AND POPOVIC, J. 2008. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics 27*, 3.

SMINCHISESCU, C., KANAUJIA, A., AND METAXAS, D. 2007. Bm3e: Discriminative density propagation for visual tracking. *IEEE Transactions on PAMI 29*, 11.

TSAI, Y.-Y., CHENG, K. B., LIN, W.-C., LEE, J., AND LEE, T.-Y. 2010. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE Transactions on Visualization and Computer Graphics 16*, 2, 325–337.

URTASUN, R., J.FLEET, D., AND FUA, P. 2006. Gaussian process dynamical models for 3d people tracking. In *IEEE CVPR*.

VONDRAK, M., SIGAL, L., AND JENKINS, O. C. 2008. Physical simulation for probabilistic motion tracking. In *IEEE CVPR*.

WANG, J., FLEET, D. J., AND HERTZMANN, A. 2009. Optimizing walking controllers. *ACM Transactions on Graphics 28*, 5.

WANG, J., FLEET, D. J., AND HERTZMANN, A. 2010. Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics 29*, 4.

WEI, X., AND CHAI, J. 2010. Videomocap: Modeling physically realistic human motion from monocular video sequences. *ACM Transactions on Graphics 29*, 4.

WREN, C. R., AND PENTLAND, A. 1998. Dynamic models of human motion. In *Automatic Face and Gesture Recognition*.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple biped locomotion control. *ACM Transactions on Graphics 26*, 3.

YIN, K., COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2008. Continuation methods for adapting simulated skills. *ACM Transactions on Graphics 27*, 3.

## Appendix: Simulation

Our controller is integrated directly with the simulation engine [Crisis 2006] that represents system dynamics in maximal coordinates. In doing so, we use constraints to update body pose over time and to achieve articulation and perform control. [Crisis 2006] employs simulation, constraint and contact model formulations that are identical to those in [ODE 2006]; we briefly outline them below.

**Constraint Model:** Identically to ODE, we formulate constraints as linear equations over velocities of body pairs that should be maintained subject to constraint force bounds, and use the Lagrange multiplier method to express (impulsive, *i.e.*, we compute forces that realize the target velocities at the next time step) constraint forces as a solution to a Mixed LCP problem built from the constraint specifications. Note, velocities and forces include both linear and angular components. We use a generic constraint model capable of modeling both bilateral as well as unilateral constraints, soft constraints and constraints with finite force bounds. For the purposes of exposition, we assume each constraint is one dimensional such that it removes one DOF from the system.

A constraint is described by (1) two indices $A$ and $B$ that identify the bodies in the rigid body system that the constraint acts on, (2) two $1 \times 6$ Jacobian matrices $\mathbf{J}_A$ and $\mathbf{J}_B$ defining scaling factors for constrained body velocities in the constraint equation, (3) a value $c$ for the right-hand side of the constraint equation, (4) bounds $\lambda^{lo} \leq 0$ and $\lambda^{hi} \geq 0$ on the unknown multiplier $\lambda$ (to be solved for) that scales the constraint force due to the constraint and (5) constraint softness parameter $s \geq 0$ that defines the tradeoff between maintaining the constraint exactly (as when $s = 0$) and minimizing the magnitude of the applied constraint force (proportional to $|\lambda|$). The constraint holds if the generated constraint force, determined by $\lambda$, is within the bounds, $\lambda^{lo} \leq \lambda \leq \lambda^{hi}$ and the following complementarity conditions defining coupling between the resulting velocity of the rigid body system and the produced constraint forces hold,

$$\lambda = \lambda^{lo} \quad \Rightarrow \quad (\mathbf{J}_A \cdot \mathbf{v}_A + \mathbf{J}_B \cdot \mathbf{v}_B) + s\lambda \geq c \quad (12)$$

$$\lambda = \lambda^{hi} \quad \Rightarrow \quad (\mathbf{J}_A \cdot \mathbf{v}_A + \mathbf{J}_B \cdot \mathbf{v}_B) + s\lambda \leq c \quad (13)$$

$$\lambda^{lo} < \lambda < \lambda^{hi} \quad \Rightarrow \quad (\mathbf{J}_A \cdot \mathbf{v}_A + \mathbf{J}_B \cdot \mathbf{v}_B) + s\lambda = c, \quad (14)$$

where $\mathbf{v}_A$ (or $\mathbf{v}_B$) is the linear and angular velocity of the body $A$ (or $B$) after all constraint forces due to all $\lambda$ multipliers have been applied on the system. Intuitively, the constraint attempts to maintain $\mathbf{J}_A \cdot \mathbf{v}_A + \mathbf{J}_B \cdot \mathbf{v}_B = c$ subject to constraint force bounds $\lambda^{lo} \leq 0 \leq \lambda^{hi}$, meaning the equation is allowed to be violated if one of the constraint force bounds is hit. The bounds determine whether the generated constraint force can "pull" ($\lambda^{lo} < 0$) or "push" ($\lambda^{hi} > 0$) or do both and how strongly.

For example, to implement the control law from Eq. (5) for an angular DOF $j$ between the segments $\mathrm{parent}$ and $\mathrm{child}$ with a common axis $\mathbf{a}$, we define the following 1-DOF constraint

$$\begin{aligned} \mathbf{a} \cdot (\omega_{\mathrm{parent}} - \omega_{\mathrm{child}}) \quad &= \quad -\alpha_i^{k,j}(\mathbf{q}_t^{k,j} - \mathbf{q}_d^{k,j}) \text{ subject to} \\ -\gamma \frac{m_{\mathrm{parent}} + m_{\mathrm{child}}}{2} \leq \quad &\lambda \quad \leq \gamma \frac{m_{\mathrm{parent}} + m_{\mathrm{child}}}{2}, \end{aligned} \quad (15)$$

where $\omega_{\mathrm{parent}}$ and $\omega_{\mathrm{child}}$ are the angular velocities of the of two body segments.

**Contact Model:** We use standard "friction pyramid" [ODE 2006] approximation for modeling impulsive contact forces. At each contact point that involves bodies $A$ (*e.g.*, ground surface geometry) and $B$ (*e.g.*, foot), we impose three constraints to constrain relative body velocity along the three orthogonal directions: the contact normal $\mathbf{n}$ that points away from $A$ towards $B$, the first tangential direction $\mathbf{x}$ and the second tangential direction $\mathbf{y}$. The first constraint acts along the normal direction and prevents body penetration, the remaining two implement friction along the tangential plane:

$$\mathbf{v}_n = 0 \quad \text{subject to bounds} \quad 0 \leq \lambda_n \leq +\infty \quad (16)$$

$$\mathbf{v}_x = 0 \quad \text{subject to bounds} \quad -\mu\lambda_n \leq \lambda_x \leq \mu\lambda_n \quad (17)$$

$$\mathbf{v}_y = 0 \quad \text{subject to bounds} \quad -\mu\lambda_n \leq \lambda_y \leq \mu\lambda_n, \quad (18)$$

where $\mu \geq 0$ is a friction coefficient, $\mathbf{v}_n$ (or $\mathbf{v}_x$, $\mathbf{v}_y$) is velocity of body $B$ at the contact relative to the velocity of the other body $A$ along $\mathbf{n}$ (or $\mathbf{x}$, $\mathbf{y}$), $\lambda_n$ (or $\lambda_x$, $\lambda_y$) is the Lagrange multiplier associated with the constraint along $\mathbf{n}$ (or $\mathbf{x}$, $\mathbf{y}$).

**Constraint Solver:** The constraint solver solves for the $\lambda$ multipliers for the individual constraints such that the resulting forces are all within the bounds and all complementarity conditions hold. We use the method of [Baraff 1996] and a variant of the Dantzig pivoting solver from [ODE 2006] to obtain valid $\lambda$ multipliers for constraints with fixed force bounds. To handle friction constraints Eq. (17) and Eq. (18) with bounds depending on $\lambda_n$, we alternate between fixing and re-estimating the bounds for the LCP. We solve the LCP with fixed bounds in a loop, using the values of $\lambda_n$ from the current iteration to update and fix bounds for the $\lambda_x$ and $\lambda_y$ values in the next iteration. To minimize a risk of over-constraining the LCP, we use soft constraints with $s > 0$. We also use constraints with finite force bounds (*e.g.*, for control) which can always be satisfied by applying forces with maximum possible magnitudes.

**Simulation:** We use Newton-Euler equations of motion to describe the motion of the simulated rigid body system under effects of external forces (*e.g.*, gravity), actuation forces and constraint forces produced by the constraint solver. We integrate the equations of motion using Euler method with a fixed step size of $1/120$ s.