Momentum-Mapped Inverted Pendulum Models for Controlling Dynamic Human Motions

TAESOO KWON

Hanyang University
and

JESSICA K. HODGINS

Carnegie Mellon University

Designing a unified framework for simulating a broad variety of human behaviors has proven to be challenging. In this article, we present an approach for control system design that can generate animations of a diverse set of behaviors including walking, running, and a variety of gymnastic behaviors. We achieve this generalization with a balancing strategy that relies on a new form of inverted pendulum model (IPM), which we call the momentum-mapped IPM (MMIPM). We analyze reference motion capture data in a pre-processing step to extract the motion of the MMIPM. To compute a new motion, the controller plans a desired motion, frame by frame, based on the current pendulum state and a predicted pendulum trajectory. By tracking this time-varying trajectory, the controller creates a character that dynamically balances, changes speed, makes turns, jumps, and performs gymnastic maneuvers.

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—*Animation*; I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—*Control theory*

General Terms: Experimentation, Human Factors

Additional Key Words and Phrases: Character animation, motion capture and reuse, rigid body simulation

ACM Reference Format:

Taesoo Kwon and Jessica K. Hodgins. 2017. Momentum-mapped inverted pendulum models for controlling dynamic human motions. *ACM Trans. Graph.* 36, 1, Article 10 (January 2017), 14 pages.
DOI: <http://dx.doi.org/10.1145/2983616>

This work was supported in part by NSF (NSF-IIS-0964581) and the NRF of Korea (2010-0029744, NRF-2014R1A1A1038386).

Authors' addresses: T. Kwon, Professor, Dept. of Computer Science & Engineering, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul Korea 133-791; email: taesoo@hanyang.ac.kr; J. K. Hodgins, Professor, Computer Science and Robotics, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213-3890; email: jkh@cs.cmu.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2017 ACM 0730-0301/2017/01-ART10 \$15.00

DOI: <http://dx.doi.org/10.1145/2983616>

1. INTRODUCTION

Simulated characters offer the promise of true interactivity by responding not only to the commands of the user, but also adapting to changes in the environment and the behavior of other characters. However, control systems that produce robust, natural-looking motion for a variety of behaviors have been hard to develop, particularly for characters with the complexity of a human figure.

In this article, we present an approach for control system design that can generate animations of a diverse set of human motions: running, jumping, and a variety of gymnastic behaviors as well as walking (Figure 1). Most of these behaviors are highly dynamic and are under-actuated because they contain extended flight phases. An appropriate angular momentum must be generated during ground contact to allow the character to perform the maneuver and regain balance after landing. We create a control system that can produce these behaviors by using a balancing strategy that relies on a new inverted pendulum model, which we call a momentum-mapped inverted pendulum model (MMIPM).

Conventional inverted pendulum models are often used in walking control systems to generate the desired footstep locations. These models track both the center of pressure and the center of mass (COM) of a human character. In contrast, our pendulum follows only the COM of the human character. Tracking just the COM produces a smooth pendulum trajectory (Figure 2) even when no foot is in contact with the ground and does not constrain the base position of the pendulum to be within the supporting polygon of the feet. Because the center of pressure (COP) is undefined during flight, a conventional inverted pendulum model is also undefined, creating discontinuities in the pendulum trajectory for behaviors such as running that includes a flight phase. The continuous motion of the MMIPM allows this model to be used to control balance for both fore/aft and side-to-side leaning throughout the behavior. For example, if the character is falling forward during the flight phase, future footstep locations can be adjusted continuously with the MMIPM model to allow corrections during the subsequent stance. This control is achieved via an intuitive geometric mapping between a full-body human pose and a pendulum pose. Because this mapping is smooth and is efficient to calculate, the full-body pose can be reconstructed from a pendulum pose and a set of constraints using an inverse-kinematics solver (Figure 3).

The contributions of this article are as follows:

- (1) The MMIPM model facilitates a momentum control scheme that allows the creation of control systems for behaviors that are beyond the capability of a standard inverted pendulum, such as a cartwheel or a flip.
- (2) The bi-directional mapping between the full-body pose and the pendulum provides an intuitive error-feedback mechanism

Fig. 1. Top: Input captured motion of walking, jumping, and gymnastics. Bottom: Simulated motion generated from controllers created using the MMIPM.

Fig. 2. MMIPM follows only the COM of the full human character unlike a conventional inverted pendulum that tracks both the center of pressure and the center of mass. Therefore, the MMIPM is defined throughout the entire motion and does not contain the discontinuities due to changes in contact that occur with a standard COM-COP inverted pendulum.

that allows the control of behaviors under disturbances and changing user input.

- (3) The mapping algorithm can be applied to a wide variety of behaviors because it is designed only using the generalized inertia and momentum.

To construct a control system for a new behavior, we first perform an offline analysis step (Figure 4), which computes the MMIPM trajectory for a reference motion. In an online planning step, the reference pendulum trajectory is then used to estimate the current state of the pendulum for a given state (position and velocity) of the human simulation (Figure 3(b)). The desired pendulum trajectory and the resulting desired human poses are regenerated at every timestep to form an error-feedback loop. The motion for the simulated human is synthesized by tracking the time-varying desired human poses from the motion planner with a full dynamic model of a human.

The initial controllers that result from this process require further improvement because of the errors that result from applying a tracking control system to an underactuated system. Although stable controllers can be obtained by carefully tuning the reference feet trajectories by hand, this process is tedious. Instead, we use an algorithm to optimize the foot trajectories using an additional offline process (Figure 4). This optimization process is performed only once for each reference motion and the optimized controller becomes stable and produces more natural motions. The optimized controller is also able to generate motions that are significantly different from those in the captured reference trajectory.

Fig. 3. (a) Reference alignment from the offline analysis step. (b) The position and orientation of the MMIPM can be directly obtained from a full-body human pose. The changes in any part of the body results in the changes in the pendulum configuration. (c) Using our inverse kinematics solver, a full-body pose can be reconstructed from a pendulum configuration and a set of constraints (such as foot position and orientation).

We demonstrate the functionality of our approach by developing a controller for human walking and running that is robust to changes in the environment and user commands. The controller is able to walk at speeds ranging from 1m/s to 2m/s, run at speeds ranging from 3m/s to 5m/s, and turn at speeds up to 1.5rad/s; run up a slope of 5 degrees and down a slope of 15 degrees; and withstand pushes up to 500N having a duration of 0.2s using a straight walking motion capture sequence, and up to 1,000N using a straight running motion capture sequence as the reference motions. The same algorithm can be used for generating a cartwheel, several jumps, skipping, and backflips, which have quite different patterns of motion. We evaluate the naturalness of the resulting motion by comparing to motion capture sequences of similar behaviors that were not used in the computation of the IPM.

This article is an extended version of a paper that was originally published in the *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* [Kwon and Hodgins 2010]. It is extended in the following ways. First, the control algorithm is reformulated based on a quadratic program instead of a hybrid dynamics solver. Second, the state-estimation algorithm based on point-cloud matching is replaced with the momentum-mapping algorithm, and two different inverted pendulum models are controlled using the same algorithm. Third, a multi-stage optimization framework is introduced, and the displacement maps for handhold locations, as well as footsteps, are optimized. As a result, the robustness of the running controller is improved by a factor of 2.5, measured by the magnitude of the maximum external force that can be applied. Several new controllers including stand, walk, and gymnastic controllers are also made possible by these enhancements to the algorithm.

2. RELATED WORK

Simulation of human motion has been addressed in both robotics and graphics. Graphics has more aggressively addressed dynamic balancing behaviors because robotics is constrained by the agility and power of the available hardware.

Fig. 4. In the analysis step, we obtain a pendulum trajectory that follows the trajectory of the COM extracted from the human reference trajectory. In the synthesis step, the motion is synthesized by tracking the time-varying human trajectory from the motion planner with a full dynamic model of a human. We improve the naturalness and robustness of the computed human motion by optimizing some of the input parameters to the motion synthesizer.

Balance Controllers for Robots. In robotics, many methods have been presented to control the motion of humanoid robots. Our work is based on the idea of preview control from an inverted pendulum proposed by Kajita et al. [2004] and Sugihara et al. [2008]. They deduce future COP positions to plan a trajectory across multiple steps, and control the COM to track the desired COP trajectory. Our approach differs from theirs in that we incorporate a reference motion capture sequence rather than a hand-designed pattern generator, and we use a geometric mapping instead of the COM and COP of the character when estimating the state of the inverted pendulum model.

While motions of human-scale biped robots have been limited to slow speed walking and running ($<2\text{m/s}$) due to technological constraints [Tajima et al. 2009; Kajita et al. 2004], simplified robots have performed dynamic motions such as gymnastics. Playter and Raibert [1992] developed biped robots that were capable of performing running, jumping, and a forward flip. Yamakita [2003] simulated a gymnastic movement involving a swing up and a somersault using a multi-link robot.

Balance Controllers for Graphics. In computer graphics, research has focused on higher-level goals such as motion quality and interactive control. Hodgins et al. [1995] simulated a running human as well as other athletic behaviors such as diving and vaulting. Yin et al. [2007] introduced an effective balancing controller called SIMBICON for walking and running motions, and their controller was later generalized to more difficult tasks [Yin et al. 2008; Coros et al. 2009, 2010] and optimized to improve motion quality and robustness [Wang et al. 2009, 2010; Lee et al. 2010]. Wang et al. [2012] built and optimized walking and running controllers for 3D characters that are actuated using a set of musculotendon models with biologically motivated control laws.

Motion capture data has frequently been used to improve the quality of simulated motions. A balance controller coupled with quadratic programming was used by da Silva et al. [2008] to produce a control system for a character walking on a seesaw. Muico et al. [2009] simulated high-quality animation of agile movements such as sharp turns. The robustness of the controller was later enhanced by introducing algorithms that track multiple trajectories in parallel [Muico et al. 2011]. Ye and Liu [2010] proposed

an optimal feedback controller that enhances the capability of one single motion capture sequence under dynamically challenging conditions such as a walk with long steps, and a squat exercise. Liu et al. [2010] proposed a sampling-based method to create an open-loop controller given a motion capture trajectory. They demonstrated the reconstruction of a diverse set of captured motions, including walking, running, and contact-rich motions such as rolls and jumps. The results of open-loop sampling-based reconstruction was later extended to produce linear feedback policies that robustly track input motion capture clips [Liu et al. 2016]. Lee et al. [2014] presented a biped locomotion controller for models actuated by more than a hundred muscles. This paper used the same balance controller as the current article. Our approach is most closely related to approaches that use a simplified dynamics model such as an inverted pendulum [da Silva et al. 2008; Coros et al. 2010]. Our control algorithm differs in that it includes a term for the momentum of the system.

Momentum Control. Recently, several papers developed balance controllers that use linear and angular momentum, because the aggregate motion of a character can be intuitively described in terms of momentum for highly dynamic behaviors. de Lasa et al. [2010] introduced an approach to control of characters based on high-level features such as COM, angular momentum, and end-effectors. This approach was used to build controllers for human balancing, a standing jump, and walking that are robust to changes in body parameters. Mordatch et al. [2010] improved the robustness of the controller using a spring-loaded inverted pendulum (SLIP) model. Ha et al. [2012] introduced a method to generate natural human landing motions in real time via physical simulation without using any motion capture sequences. They later simplified the design process for complex dynamic controllers by introducing an intuitive framework to teach a character dynamic motor skills through progressive practice [Ha and Liu 2014]. Brown and others proposed a physics-based framework to generate rolling and handspring behaviors [Brown et al. 2013; Zordan et al. 2014]. They provided control for rotation by specifying desired rotation indices using the concept of “angular excursion”. The aforementioned papers used tracking objectives that try to follow fixed momentum profiles that are either user-specified or obtained from reference motions. Borno et al. [2014] introduced a phase variable to design state-dependent controllers for rotational movement. State-dependent control for rotation has the potential to robustly adapt to disturbances. We share the same goal but use a different approach.

There are other schemes that rely on the fundamental relations between the COP, the linear and angular momenta, and the ground reaction force (GRF). Macchietto et al. [2009] demonstrated a momentum control scheme capable of robustly balancing a standing character. In this work, a desired rate of change of angular momentum is derived to control the COP near the center of support. Wu and Zordan [2010] extended the control scheme for simulated characters to perform controlled steps. In this work, the desired rate of change of linear and angular momenta are set proportional to the current linear and angular momentum using negative damping coefficients. Lee and Goswami [2012] presented a momentum-based balance strategy for walking on uneven ground. This scheme used a desired rate of change of angular and linear momenta that is fixed given a user input and a task. This scheme guarantees that the desired foot GRF and COP during single and double support phases will be achieved. However, as mentioned in Lee and Goswami [2012], the issue of how to set the desired parameters has not been fully explored. Our scheme differs from theirs in that we calculate the desired rate of change of angular and linear

momenta in a state-dependent manner using a simplified model that is geometrically mapped to the full human model. The control parameters for the simplified model are directly obtained from the captured reference motion. Without explicit COP constraints, the control algorithm for the simplified model is contact-independent, and thus has the merit of easy manipulation of contact state and timing albeit the loss of precise control of the COP.

Motion Graphs. Although our controller is based on dynamic simulation, there have been a number of previous studies that generated realistic and reactive animations using kinematic approaches such as motion graphs [Lee et al. 2002; Arikan and Forsyth 2002; Kovar et al. 2002a; Treuille et al. 2007; McCann and Pollard 2007]. However, these methods are limited in that the resulting motions are restricted to a rearrangement of those in the graph. A few methods have been proposed to alleviate the limitations of pure graph-based controllers by allowing a continuous parameter space using multiple similar trajectories [Safonova et al. 2004; Shin and Oh 2006; Heck and Gleicher 2007; Lee et al. 2010]. A few physically based controllers have incorporated similar ideas to produce more robust and agile controllers [Wu and Popović 2010; Levine and Popović 2012; Muico et al. 2011]. However, combining the concepts of multi-trajectory continuous parameter space with the extrapolation capability of a physically based controller remains a challenge.

3. OVERVIEW

Given a captured reference motion and a small number of manually chosen parameters, our scheme automatically creates a controller that reproduces the captured motion and modifies the motion in the presence of disturbances and changes to the environment. The key insight behind our approach is an inverted pendulum model that is geometrically mapped to a full-body human pose. Our pendulum model is based on an inverted pendulum on a cart (IPC) model that is a rigid body system with two translation joints to move the cart horizontally and two rotational joints for the pendulum (Figure 7(a)). Because the rotational joint is unactuated, the inverted pendulum is inherently unstable and must be actively balanced by moving the cart horizontally. This property resembles the balancing actions of humans as they place their feet on the ground in locations that will result in balance.

Although the IPC model has been widely used for balancing tasks, our use of the model is different from existing approaches in that we use the angular and linear momentum of the human character to estimate the state of the pendulum rather than using the position of the COP. The state variables for the pendulum are its position and leaning angle, and their time-derivatives. To estimate the state of the pendulum that corresponds to a human pose generated during the simulation, our approach requires a smooth pendulum trajectory that mimics the trajectory of the COM extracted from the human reference motion (Figures 5 and 10).

To obtain the pendulum trajectory, we first perform an offline analysis step and create a pendulum trajectory generator (Figure 6(a), Section 4). The pendulum trajectory generator takes as input the initial pendulum state, the desired speed and turning rate, and produces a trajectory for an inverted pendulum. The input variables are optimized such that the resulting pendulum trajectory closely reproduces the COM extracted from the human reference motion.

The pendulum trajectory generator is also used as part of an online motion planner that takes as input the current pendulum state, the desired and the optimized variables for speed and turning rate, and produces a trajectory for an inverted pendulum that is then

Fig. 5. Reference pendulum trajectory for motion capture sequences is shown as a green bar.

converted into footstep locations and a desired full-body trajectory (Figure 6(b), Section 5.2). We use the pendulum trajectory generator instead of the optimized reference pendulum trajectory to allow deviation from the reference COM trajectory. For example, if the pendulum corresponding to the simulated human motion is leaning forward too much, the pendulum trajectory generator produces a trajectory that moves forward faster than the reference trajectory to prevent the character from falling forward. The modified pendulum trajectory is then converted to a desired full-body trajectory that is similarly modified from the reference full-body trajectory, resulting in a reduced velocity for the simulated character on the next step.

Finally, the motion for the simulated human is synthesized by tracking the time-varying full-body trajectory from the motion planner with a full dynamic model of a human (Figure 6(c), Section 5.3). The pendulum trajectory and the desired full-body trajectory are re-generated at every frame (120hz) to form an error-feedback loop.

This process produces initial controllers that are stable but we can improve the naturalness and robustness of the controller by optimizing displacement maps for correcting the desired full-body trajectory from the motion planner (Figure 6(d), Section 6). The objective function of the optimizer is the difference between the synthesized full-body motion and the desired full-body trajectory that is output from the motion planner. This optimization process is performed only once for each reference motion.

4. MOTION ANALYSIS

We analyze a captured reference motion and use the COM trajectory that we extract to create a trajectory generator for an

Fig. 6. (a) In the analysis step, we create a pendulum trajectory generator from a reference motion trajectory. (b) The pendulum trajectory generator is used as part of an online motion planner that produces a trajectory for an inverted pendulum, which is then converted into footstep locations and a desired full-body trajectory. (c) In the synthesis step, the motion is synthesized by tracking the time-varying full-body trajectory from the motion planner with a full dynamic model of a human. (d) We improve the naturalness and robustness of the controller by optimizing displacement maps for correcting the desired full-body trajectory.

inverted pendulum. This pendulum trajectory generator is used for two purposes: state estimation and planning. The reference pendulum trajectory obtained from the trajectory generator defines a correspondence between each human state and pendulum state along a captured reference motion (Figure 5). Our pendulum state-estimation algorithm uses these correspondences. In planning, a new desired pendulum trajectory is generated at every timestep based on the current pendulum state. The difference between the reference pendulum trajectory and the desired pendulum trajectory allows us to generate a desired human motion that moves to correct errors in the current state of the simulation.

We first describe our human dynamic model and the inverted pendulum model, and then explain how the reference pendulum trajectory can be obtained from the reference human trajectory (motion capture data).

Human Dynamic Model. Our full-body character for simulation has 44 degrees of freedom (DOFs) including six unactuated DOFs at the pelvis. Each knee and elbow is modeled using a one DOF hinge

joint. Other joints have three DOFs. The mass and inertia matrix of each body part are calculated from a surface mesh based on a uniform density assumption and the mass of the human subject. The surface mesh is manually positioned such that it closely matches the motion capture marker positions.

Human Motion Annotation. We manually segment a given captured motion at every contact state change, and independently annotate the four end-effectors: left foot, right foot, left hand, and right hand. For each end-effector, we assign a Boolean value for the contact state and a piecewise linear curve that represents the time-varying weight $q(0 \leq q \leq 1)$ of the end-effector. The weight of an end-effector is one if it is in contact or expected to be in contact. Zero weights are assigned for limbs not involved in the stepping motions. By assigning a weight value to each end-effector, the discontinuities that occur when stepping pattern generation turns on and off are avoided. For example, during walking and running, the contact states for the hands are false, and the weights are zero. The weights for both feet are always one because the swing limbs need to be actively positioned for the upcoming contact. During a cartwheel, the weight of the hands increases continuously from 0 to 1 when the hand is moving toward the ground, while the weight of a foot moves from 1 to 0 during the same period (Figure 8). The time-varying weights from user annotation are necessary to decide whether to track the desired global configuration of each end-effector or not. The weights are also used in our inverse kinematics solver for reconstructing a full-body pose from a pendulum configuration. The continuously varying weights prevent discontinuities in the root position and orientation [Shin et al. 2001].

The forward facing direction of the character is annotated to define the reference coordinate system for representing the leaning angle, the desired velocity of the pendulum, the foot positions, and the root orientation of the character invariant to the world coordinates. When the reference motion involves a high-speed rotation during the flight phase, for example, a cartwheel or Popa jump, we define the forward facing direction by linearly interpolating the initial and final forward facing direction of the pelvis at the beginning and end of the jump so that the orientation of the reference coordinate is robustly defined.

Inverted Pendulum Models. We use two different inverted pendulum models depending on the balancing strategy required for the behavior: an IPC for stepping motions and a pivoted inverted pendulum (PIP) for standing motions. The former uses a foot placement strategy for balancing while the latter uses an ankle strategy. The IPC has two translation joints to move the cart and two rotational joints for the pendulum (Figure 7(a)). The rotational joint is unactuated and the pendulum is balanced by actively moving the cart. In contrast, a pivoted inverted pendulum has no translational joints and balances by actuating the rotational joint. We use two hinge joints to model the leaning of the pendulum where the axes of the joints are aligned with the instantaneous forward facing direction of the character. Although the pendulum is glued to the ground, the feet of the animated human are not.

In addition to the inverted pendulum models, a lumped mass (LM) model is used for independently controlling the height of the COM of the character during contact phases. The lumped mass is connected with the ground using a one-dimensional spring damper that allows only vertical movements.

Pendulum Trajectory Generator. We adopt an infinite-horizon linear quadratic regulator (LQR) for controlling these models

Fig. 7. An inverted pendulum and the corresponding pose of the human character. (a) An inverted pendulum on a cart can be actively balanced by applying force $\mathbf{F}$ to the base. (b) A running motion can be characterized by a smooth continuous trajectory of the IPC. (c) Sheared coordinate frames $\mathbf{S}_{t,i}$, $i \in \{1, 2\}$ are obtained by sampling the trajectory of the base of the pendulum. Actual end-effector positions and orientations are represented local to the sheared coordinate frames.

Fig. 8. A captured motion is segmented at every contact state change. For each end-effector, one Boolean value for the contact state and one continuous weight value of the end-effector are annotated.

because of its stability properties and computational efficiency [Dorato et al. 1994]. For each pendulum model, we use two independent, two-dimensional LQR controllers to regulate the motion of the pendulum along the forward direction and lateral direction of the character, respectively. The linear quadratic regulator controller is derived by writing the linearized dynamics of the simplified model about the upright pose (see Appendix A for details).

Using the LQR controllers and the annotated captured reference motion, we build pendulum trajectory generators that closely reproduce the COM trajectories extracted from the captured reference motions. For the PIP model and the LM model, the desired state offsets for the LQR controllers that produce trajectories that exactly match the captured reference motion can be found analytically by performing inverse dynamics. However, it is non-trivial to make a controller for the IPC model that closely follows the captured reference motions because the rotational joint is unactuated. To solve this problem, we formulate an optimization problem where the objective is to minimize the horizontal positional differences between the COM of the character and the COM of the pendulum. The unknown variables of the optimization are the key-frames of the time-varying desired velocities of the IPC model. For each motion segment s , we assign a key-frame of the desired velocity $\hat{\mathbf{x}}_s$, represented relative to the forward direction. We used a piecewise linear curve to produce a continuously varying desired velocity. The key-frames for the desired velocities are obtained by minimizing the following objective

Fig. 9. A multi-stage optimization scheme is used in both motion analysis and offline optimization. In the third stage of the optimization (Stage 2), only a subset of key-frames that belong to segments $s \in \{s_2, s_3, s_4\}$ are optimized, while the objective functions are evaluated for segment $s \in \{s_2, s_3, \dots, s_6\}$ to consider a small window of the future frames.

function:

$$\{\hat{\mathbf{x}}_s\} = \arg \min \sum_{i \leq N} \left\| \text{project}(\mathbf{x}_i^{cCOM} - \mathbf{x}_i^{pCOM}) \right\|^2,$$

where $\{\hat{\mathbf{x}}_s\}$ denotes the set of key-frames of the desired velocities, $\mathbf{x}_i^{cCOM}$ and $\mathbf{x}_i^{pCOM}$ denote the COM position of the character, and the pendulum at frame i . N denotes the number of frames in a motion clip, and $\text{project}(\dots)$ discards the vertical component of the 3D vector. Because we experimentally found that optimization in a high-dimensional space would likely lead to a local minima, we use a multi-stage optimization scheme (Figure 9) to reduce the number of dimensions of the search space. This scheme is used in both motion analysis (Section 4) and offline optimization (Section 6), for obtaining pendulum and full-body controllers, respectively.

Initially, all the key-frames are initialized to a reasonable value (the average COM velocity) obtained from the captured reference motion. This initial solution produces a reasonable trajectory for the pendulum, and it needs only slight adjustments via subsequent optimizations. In the f^{th} stage of the optimization, only the subset of key-frames that belong to segments $s \in \{s_f, s_{f+1}, s_{f+2}\}$ are optimized. The next stage of the optimization uses the best solution from the previous stage as the initial solution. This approach does not produce a global optimum but the optimization can be performed within a few seconds using a conjugate gradient algorithm and produced good results in our experiments.

The resulting desired velocities define a controller for the pendulum that reproduces the COM trajectory of the captured reference motion. The optimized desired velocity $\{\hat{\mathbf{x}}_s\}$ and the turning speed $\hat{\theta}_i^{pend}$ calculated from the forward-facing directions are stored for use at runtime. The resulting pendulum trajectory generator is used for the preview control performed online at every timestep of the simulation of the full-body model.

Graph Construction. To obtain greater generality from our motion capture data, we construct a motion graph for each reference motion [Kovar et al. 2002a]. Each edge of the graph corresponds to a motion segment, which corresponds to a set of consecutive frames having the same contact state. An edge contains both the reference human-motion segment and the corresponding reference pendulum trajectory. Small discontinuities along a graph path can be removed by blending two temporally overlapping motion segments. The same blending operation is applied to the motion capture data and the corresponding reference pendulum trajectory. This scheme is used to generate a controller that makes a transition from a backflip motion to an existing walking controller.

Fig. 10. A geometric alignment algorithm is used for pendulum-state estimation.

5. MOTION SYNTHESIS

In this section, we describe how to simulate a variety of behaviors for a human character using the pendulum trajectory generator obtained from the reference human motion. As shown in Figure 6(c), the synthesis step consists of three components: state estimation, motion planning, and tracking.

5.1 State Estimation

At every simulation step, we first estimate the current state of the pendulum: position, leaning angle, velocity, and angular velocity. The position and leaning angle are calculated using a geometric mapping between a full-body human pose and a pendulum pose, and the velocity and angular velocity are calculated using the generalized momentum of the simulated character.

For the geometric alignment, we compute a constant-velocity motion that interpolates between the full-body reference pose and the current pose from the simulator (Figure 10). A generalized momentum can be calculated from that motion and converted to a positional and rotational displacement of the pendulum. The displacement is then used to obtain the current position and leaning angle of the pendulum by adding it to the state of the reference pendulum. The geometric alignment algorithm was inspired by the idea of whole-body angular excursion that has been used for estimating the overall orientation of simulated characters [Popovic et al. 2004; Brown et al. 2013]. The whole-body angular excursion is defined as:

$$\theta = \int_0^t \mathbf{w}(t) dt + \theta_0, \quad (1)$$

where the current orientation θ at time t is defined relative to the initial orientation θ_0 by integrating the overall rotational velocity $\mathbf{w}(t) = \mathbf{I}^{-1}(t)\mathbf{L}(t)$ from the inertia matrix $\mathbf{I}$ and the angular momentum $\mathbf{L}$. This term is the rotational analog of the COM, in that the COM position of the character at the current frame can be calculated in exactly the same manner by integrating the linear momentum.

However, the angular excursion inherently has orientation-drift because of the gradual accumulation of error in rotation over time. Even a small error in the estimated orientation and leaning angle of the pendulum can have a significant impact on the character's balance. Therefore, we eliminated the temporal integration of the angular velocity by using the reference trajectory:

$$\theta(t) = \int_0^1 \mathbf{w}(s) ds + \bar{\theta}(t), \quad (2)$$

where $\bar{\theta}(t)$ is the orientation and leaning angle of the reference pendulum corresponding to the current time t . The angular velocity

$\mathbf{w}$ can be obtained from a motion interpolating between the reference pose and the simulated pose, based on weighting parameter s in the closed unit interval $[0, 1]$. This modified definition is also analogous to the definition of the COM.

To handle the rotation and translation of the pendulum in a consistent manner, we obtain the generalized velocity $\mathbf{V}$ as follows:

$$\mathbf{V} = \mathbf{I}^{-1}\mathbf{H}_{\text{COM}}, \quad (3)$$

where $\mathbf{I}$ is the generalized inertia of the simulated character with respect to the global COM frame, and $\mathbf{H}_{\text{COM}}$ is the generalized momentum for pose alignment, which is calculated from the interpolated motion (see Appendix B). Then, the current configuration of the pendulum is obtained by integrating the constant velocity $\mathbf{V}$ over the closed unit interval $[0, 1]$ starting from the reference pendulum configuration. This integration is equivalent in meaning to Equation (2) and can be performed analytically using the Lie-group formulation [Park et al. 1995]:

$$\Theta_{\text{COM}}^{\text{pend}} = \exp(-\mathbf{V})\bar{\Theta}_{\text{COM}}^{\text{pend}}, \quad (4)$$

where $\Theta_{\text{COM}}^{\text{pend}} \in SE3$ is the configuration of the current pendulum measured at the COM of the pendulum. Finally, the resulting momentum $\mathbf{H}_{\text{COM}}$ for pose alignment is converted to the current estimated configuration of the pendulum $\Theta_{\text{COM}}^{\text{pend}}$, using Equations (3) and (4). The pendulum configuration is then corrected using a vertical translation to place the base of the pendulum on the ground.

The velocity and angular velocity of the pendulum can be calculated in a similar manner using the error between the generalized momentum of the simulated character, $\mathbf{M}_{\text{COM}}$, and that of the captured reference motion, $\bar{\mathbf{M}}_{\text{COM}}$, thus:

$$\mathbf{V}_{\text{COM}}^{\text{pend}} = \text{Ad}_{\mathbf{C}}(\bar{\mathbf{V}}_{\text{COM}}^{\text{pend}}) + \mathbf{I}^{-1}(\mathbf{M}_{\text{COM}} - \text{Ad}_{\mathbf{C}^{-1}}^*(\bar{\mathbf{M}}_{\text{COM}})), \quad (5)$$

where $\mathbf{V}_{\text{COM}}^{\text{pend}}$ is the global velocity of the pendulum measured at the COM of the pendulum, $\bar{\mathbf{V}}_{\text{COM}}^{\text{pend}}$ is that of the reference pendulum, and $\mathbf{C} \in SE3$ is the transformation matrix that has only the vertical rotation component of the pose alignment transformation $\exp(\mathbf{V})$. Equations (4) and (5) can be used for both types of pendulum models for balancing and the lumped mass model for the control of COM height.

5.2 Motion Planning

The pendulum trajectory generator computes a desired pendulum trajectory from the current state estimate of the pendulum (Figure 11(b)). The desired pendulum trajectory with a finite horizon is replanned each frame. Specifically, the vertical components of the pendulum orientations are first obtained by kinematic integration, and then the positions and leaning angles are calculated using a forward dynamics simulation of the inverted pendulum on a cart.

Given the planned pendulum trajectory, a desired human motion is generated on the fly by deforming the captured reference motion (Figure 11(a)) to match the pendulum trajectory. As shown in Figure 5, the contact positions in a captured reference motion are located near the trajectory of the cart of the pendulum, and the cart position at the middle of each support phase is close to the contact position of the support limb. Based on this observation, we generate a stepping behavior by sampling the reference position of the end-effector placement based on the trajectory of the cart (Figures 11 and 12) and extract a displacement map for each end-effector that stores the offsets between the reference positions and the actual positions from the captured reference motion. A new stepping pattern is generated by applying the displacement map to the predicted pendulum trajectory.

Fig. 11. (a) We generate reference coordinates $\bar{\mathbf{S}}_{\phi(t),k}$ at phase t for end-effector placements by sampling the reference pendulum trajectory. The actual end-effector positions are encoded locally to the reference coordinates. (b) At runtime, the pendulum trajectory is re-planned at every frame. The desired stepping locations are obtained by sampling the modified pendulum trajectory.

Fig. 12. Left: footsteps captured from the reference walking motion (x and z coordinates only). The green discs represent the right foot positions. We used the ball of the foot to represent the foot position. Right: a temporal plot of the reference z-position of the end-effector placement algorithm. The blue line represents the reference trajectory of the cart. The red line represents the origin of the reference coordinates sampled from the reference trajectory using the annotated contact timings. The green line represents the actual right foot locations obtained from the captured reference motion.

Let a full-body pose of the desired motion at frame i be defined by root transformation matrix $\mathbf{X}_i^G$, end-effector positions $\mathbf{y}_{i,k}$, $k \in \{1(\text{left foot}), 2(\text{right foot}), 3(\text{left hand}), 4(\text{right hand})\}$ in the global frame, weights $\mathbf{i}_{i,k}$, and local joint angles $\{\theta_i^j\}$, where the actual desired pose is constructed using an inverse kinematics solver.

Next, we will explain how we generate the end-effector positions of the desired pose based on the predicted pendulum trajectory. As shown in Figure 7(c), the end-effector positions $\mathbf{y}_{i,k}$, $k \in 1, 2, 3, 4$ are obtained using coordinate frames $\mathbf{S}_{i,k}$ located on the predicted pendulum trajectory:

$$\mathbf{y}_{i,k} = \mathbf{S}_{i,k} \bar{\mathbf{y}}_{j,k}, \quad (6)$$

where matrices $\mathbf{S}_{i,k}$ are provided by a stepping pattern generator that works independently for each end-effector k . Local end-effector positions $\bar{\mathbf{y}}_{j,k}$ are obtained from the reference human motion. Here, the frame number of the corresponding pose in the reference motion is denoted by j . As shown in Figure 7(c), an end-effector position $\bar{\mathbf{y}}_{j,k}$ is represented in a sheared coordinate frame so that the height of the end-effector is invariant to the pendulum leaning angle. We first describe the stepping pattern generator and how local end-effector

positions are obtained, and then describe the inverse kinematics solver that is based on the momentum-mapping algorithm.

Stepping Pattern Generation. We generate a stepping behavior by sampling the reference coordinates for the end-effector placement on the trajectory of the cart (Figure 11(a)). Each end-effector having non-zero importance is either on the ground, being positioned for placement on the ground, or has just lifted off. The stepping pattern generator is designed such that the end-effector stays at its desired position during each support phase and moves from the previous support position to the next support position along a shortest path during the swing phase. Our stepping pattern generator works independently for each limb, and the same algorithm is applied for all end-effectors. This independence allows us to model the flight phases of running and double stance phases of walking in a unified way because the start time of each swing phase does not have to be synchronized with the end of the support phase of the other limb. For notational simplicity, let us define an operator $\phi(\cdot)$ that converts contact phase t , $0 \leq t \leq 1$ to frame number: $\phi(t) = f_k + t(l_k - f_k)$, where f_k and l_k are the first and last frame of the current contact state at phase t . If end-effector k is in contact at phase t , then the reference coordinate for the support position at phase t is defined as $\mathbf{S}_{\phi(t),k} = \text{shear}(\mathbf{x}_{\phi(0.5)}^{\text{pend}}, \mathbf{q}_{\phi(0.5)}^{\text{pend}})$, where $\text{shear}(\cdot)$ denotes a transformation matrix that shears the vertical(y) axis to the pendulum axis defined by the second argument $\mathbf{q}_{\phi(0.5)}^{\text{pend}}$. The matrix is defined by a sequential multiplication of the horizontal translation matrix and x , z -shearing matrix, and a vertical rotation matrix. The center of the coordinate frame is fixed at $\mathbf{x}_{\phi(0.5)}^{\text{pend}}$ during the half stride using the pendulum configuration at the middle frame. The amount of shearing and the vertical orientation of the reference coordinate is also defined by the pendulum at the middle frame $\mathbf{q}_{\phi(0.5)}^{\text{pend}}$.

When the end-effector k is in the swing phase, then the position of the end-effector is encoded using a coordinate frame that linearly interpolates the nearby coordinate frames for the supporting phases. That is,

$$\mathbf{S}_{\phi(t),k} = \text{shear}(\mathbf{x}_{\phi(0.5)}^{\text{pend}}(1-t) + \mathbf{x}_{\phi(1.5)}^{\text{pend}}, \text{slerp}(t, \mathbf{q}_{\phi(0.5)}^{\text{pend}}, \mathbf{q}_{\phi(1.5)}^{\text{pend}})),$$

where $\mathbf{x}_{\phi(0.5)}^{\text{pend}}$ and $\mathbf{x}_{\phi(1.5)}^{\text{pend}}$ denote the position of the cart at the middle of the previous support phase and the next support phase, respectively.

The coordinates for end-effector positions on the reference pendulum trajectory $\bar{\mathbf{S}}_{j,k}$ can be defined in the same manner for all frames j . Then, the end-effector displacement map $\bar{\mathbf{y}}_j$ relative to the reference trajectory can be obtained as follows:

$$\bar{\mathbf{y}}_{j,k} = (\bar{\mathbf{S}}_{j,k})^{-1} \bar{\mathbf{y}}_{j,k}^G, \quad (7)$$

where $\bar{\mathbf{y}}_{j,k}^G$, $k \in 1, 2, 3, 4$ are the global end-effector positions in the reference human motion at frame j . Because the end-effector positions at the local frame $\bar{\mathbf{y}}_{j,k}$ are constant given j , they are computed only once in the offline analysis step. In a later optimization step, $\bar{\mathbf{y}}_{j,k}$ are further modified to produce a better controller (Section 6).

Inverse Kinematics (IK). An IK solver is used for computing the joint angles that position an end-effector on a planned location. At the same time, the resulting pose should match the planned pendulum configuration after momentum mapping. For algorithmic efficiency, we make use of a hybrid IK solver, where analytical

solutions are used for limbs and a numerical implementation is invoked for obtaining the optimal root configuration. This problem can be efficiently solved numerically in a low-dimensional search space ($=6$). The initial solution is obtained using the limb IK solver assuming that the root joint is rigidly attached to the pendulum. The limb IK solver attempts to solve the IK problem by rotating only the hip, knee, and ankle joints. Assuming the global orientation of the ankle joint is fixed, this problem can be solved using a closed-form solution in a fixed amount of time. We adopt a solution proposed by Kovar et al. [2002b] because of its knee-damping capability. The numerical implementation uses the following cost function E_{IK} :

$$E_{IK}(\mathbf{x}) = \sum_k \mathbf{i}_{i,k} \|\mathbf{y}_{i,k} - \mathbf{y}_k(\mathbf{x})\|^2 + w_m \text{dist}(\Theta_{i,COM}^{\text{pend}}, \Theta^{\text{pend}}(\mathbf{x})) + w_r \|\mathbf{x}\|^2, \quad (8)$$

where $\mathbf{x}$ is the positional and rotational displacement of the root joint from the initial configuration, $\mathbf{y}_k(\mathbf{x})$ is the k -th end-effector position computed using the limb IK solver, $\Theta_{i,COM}^{\text{pend}}$ is the planned pendulum configuration from the pendulum trajectory generator, $\Theta^{\text{pend}}(\mathbf{x})$ is the estimated pendulum configuration, and $\text{dist}(\cdot)$ measures the distance between two rigid transformation matrices. w_m and w_r are the scalar weights for the second and the third term, respectively. $0 \leq \mathbf{i}_{i,k} \leq 1$ is the weighting factor for limb k . The importance values for the IK solver are taken from the user annotation of limb weights (Section 4 and Figure 8).

Desired Velocity. The user can specify a motion that is different from the captured reference motion by modifying the desired speed $\hat{\mathbf{x}}_z$ and the turning speed $\hat{\theta}$. When the desired turning speed is modified, our controller automatically modifies the lateral desired speed $\hat{\mathbf{x}}_x$ so that centrifugal accelerations are generated:

$$\hat{\mathbf{x}}_z \leftarrow \hat{\mathbf{x}}_z + \alpha, \quad (9)$$

$$\hat{\theta} \leftarrow \hat{\theta} + \beta, \hat{\mathbf{x}}_x \leftarrow \hat{\mathbf{x}}_x + c\beta, \quad (10)$$

where α is the amount of modification in forward velocity, β is the amount of modification in turning speed, $\hat{\mathbf{x}}_x$ is the lateral desired speed, and c is an empirically chosen coefficient ($c = 3$ in our experiments).

Timing and Stride Adjustment. We adjust the duration of the single support phase based on the forward speed to prevent an excessively long stride. Let l be the distance between the previous support position and the next predicted support position, and $\bar{l}$ be the distance from the captured reference motion. Then, the scale factor for the duration $s_1 = \text{clamp}(\bar{l}/l, 0.5, 1.0)$. To prevent limping (an oscillation in stride length between left and right legs), the timing is further adjusted based on the ratio r between the previous stride and the prediction of the current stride. Finally, we set the total duration $t_i = \bar{t}_i \cdot \text{clamp}(s_1 \cdot r, 0.5, 1.0)$, where $\bar{t}_i$ is the original duration of segment i . Because our pendulum trajectory is smooth regardless of the foot-step locations, it does not need to be replanned.

5.3 Tracking

The motion planning step results in changes to the future pendulum trajectory and input to the IK solver such as desired footstep locations $\mathbf{y}_{i,k}$, which in turn leads to changes in the desired motion and all the quantities computed from the desired motion. Finally, the full-body animation is generated using a dynamics simulation that tracks the time-varying desired motion from the motion planner. We use a quadratic programming formulation to calculate joint

torques such that the desired motion is closely followed [Abe et al. 2007; da Silva et al. 2008]. Specifically, an objective function Q is minimized while satisfying an equality constraint that describes the equations of motion and a few inequality constraints for modeling linearized friction cones:

$$\min_{\ddot{\mathbf{q}}, \boldsymbol{\lambda}, \boldsymbol{\tau}} Q(\ddot{\mathbf{q}}, \boldsymbol{\lambda}, \boldsymbol{\tau}) \quad (11)$$

such that

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{b} = \boldsymbol{\tau} + \mathbf{J}_c^\top \mathbf{B}\boldsymbol{\lambda}, \quad (12)$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \quad (13)$$

$$\mathbf{a}_c = \mathbf{B}^\top \mathbf{J}_c \ddot{\mathbf{q}} + \mathbf{B}^\top \dot{\mathbf{J}}_c \dot{\mathbf{q}} + \ddot{\mathbf{B}}^\top \mathbf{J}_c \dot{\mathbf{q}} \geq \mathbf{o}_c, \quad (14)$$

where $\ddot{\mathbf{q}}, \boldsymbol{\lambda}, \boldsymbol{\tau}$ are the joint accelerations, contact forces, and joint torques, respectively. $\mathbf{J}_c$ contains the Jacobian matrices that relate joint velocities $\dot{\mathbf{q}}$ to global velocities at the contact points, $\mathbf{B}$ is the basis vector for the friction cones, $\mathbf{a}_c$ denotes the global accelerations at the contacts, and $\mathbf{o}_c$ denotes the velocity-dependent offsets at the contacts. In addition to these inequality constraints, we occasionally set bounds on joint accelerations when the knee and the elbow joints overextend.

The objective function Q consists of five terms:

$$Q = Q_{\text{momentum}} + Q_{\text{tracking}} + Q_{\text{end-effector}} + Q_{\text{head}} + Q_{\text{contact force}}. \quad (15)$$

Q_{momentum} tries to reproduce the generalized momentum from the captured reference motion while applying error-feedback torques and forces from the pendulum models and the lumped mass model:

$$Q_{\text{momentum}} = (\dot{\mathbf{L}} - \dot{\mathbf{L}}_d)^\top \mathbf{W}_m (\dot{\mathbf{L}} - \dot{\mathbf{L}}_d), \quad (16)$$

where $\dot{\mathbf{L}}$ is the time-derivative of the momentum of the character, $\dot{\mathbf{L}}_d$ is the desired value of the time-derivative of the momentum of the character, and $\mathbf{W}_m$ is a diagonal weighting matrix that contains higher values for angular momentum ($=10^4$) than linear momentum ($=10^2$). Momentum Jacobian $\mathbf{J}_M$ relates joint velocities to momentum:

$$\dot{\mathbf{L}} - \dot{\mathbf{L}}_d = \mathbf{J}_M \ddot{\mathbf{q}} + \dot{\mathbf{J}}_M \dot{\mathbf{q}} - \dot{\mathbf{L}}_d. \quad (17)$$

The momentum profile of a human and that of a similarly moving inverted pendulum will never match exactly because of the reduced DOFs in the inverted pendulum model. To calculate the desired momentum $\dot{\mathbf{L}}_d$, we analyze the captured reference motion and extract the difference in the momentum profiles of the human and the pendulum. The difference is compensated through an error-feedback momentum obtained from the pendulum during simulation:

$$\dot{\mathbf{L}}_d = \bar{\dot{\mathbf{L}}} + \text{clampL}(\dot{\mathbf{L}}_{\text{pend}} - \bar{\dot{\mathbf{L}}}_{\text{pend}}), \quad (18)$$

where $\bar{\dot{\mathbf{L}}}$ is the time-derivative of the momentum calculated from the aligned reference motion, $\dot{\mathbf{L}}_{\text{pend}}$ and $\bar{\dot{\mathbf{L}}}_{\text{pend}}$ are the reference and the desired value for the time-derivative of momentum of the pendulum, respectively. The desired value $\bar{\dot{\mathbf{L}}}_{\text{pend}}$ is obtained by summing the control forces applied to the pendulum and the lumped mass model. $\text{clampL}(\cdot)$ clamps the magnitude of the error-feedback force because abrupt changes in momentum can cause jerk in the motion and simulation instability.

The second term in the objective function, Q_{tracking} , measures error in following the time-varying desired motion.

$$Q_{\text{tracking}} = (\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d)^T \mathbf{W}_t (\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d), \quad (19)$$

where $\ddot{\mathbf{q}}$ is the joint acceleration, and the desired acceleration $\ddot{\mathbf{q}}_d$ is defined as:

$$\ddot{\mathbf{q}}_d = a(\mathbf{q}_d - \mathbf{q}) + b(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \ddot{\mathbf{q}}_d, \quad (20)$$

where $\mathbf{q}_d$ and $\dot{\mathbf{q}}_d$ are the desired angles and angular velocities of all joints calculated from the desired human trajectory. The feed-forward acceleration $\ddot{\mathbf{q}}_d$ is calculated from a smoothed reference motion to avoid jerkiness. $\mathbf{W}_t$ is a diagonal weighting matrix that contains the same value ($=1$) for all DOFs except for the root joint, which has a much smaller value ($=10^{-6}$). We use the gains $a = 200$, $b = 30$, for all joints except for the big joints such as the hips and lower spine, which use $a = 800$, $b = 120$.

Directly controlling the end-effectors in Cartesian space is intuitive and often more effective than controlling them in joint-angle space. Therefore, we add the terms $Q_{\text{end-effector}}$ and Q_{head} in the objective function. These terms measure errors in the desired global configurations.

$$Q_k = (\ddot{\mathbf{y}} - \ddot{\mathbf{y}}_d)^T \mathbf{W}_k (\ddot{\mathbf{y}} - \ddot{\mathbf{y}}_d), \quad (21)$$

$$\ddot{\mathbf{y}} - \ddot{\mathbf{y}}_d = \mathbf{J}_k \ddot{\mathbf{q}} + \dot{\mathbf{J}}_k \dot{\mathbf{q}} - \ddot{\mathbf{y}}_d, \quad (22)$$

where $\ddot{\mathbf{y}}$ and $\ddot{\mathbf{y}}_d$ are the simulated and desired values for the time-derivatives of the global generalized velocity of end-effector or head k . $\mathbf{W}_k$ is a diagonal weighting matrix that contains larger values ($=6 \cdot 10^4$) for positional velocities than the weight for angular velocities ($=6 \cdot 10^3$). Desired acceleration $\ddot{\mathbf{y}}_d$ is defined as:

$$\ddot{\mathbf{y}}_d = k_p \left(\text{Ad}_{\mathbf{R}_{\mathbf{y}_d} \mathbf{Y}^{-1}} (\log(\mathbf{Y}_d \mathbf{Y}^{-1})) \right) + k_v (\dot{\mathbf{y}}_d - \dot{\mathbf{y}}), \quad (23)$$

where $\mathbf{Y}_d$ and $\mathbf{Y}$ are the desired and current configuration of the end-effector k , and $\dot{\mathbf{y}}_d$ is the desired generalized velocity of the end-effector calculated from the desired motion from the motion planner:

$$\dot{\mathbf{y}}_d = \text{Ad}_{\mathbf{R}_{\mathbf{y}_d} \mathbf{Y}_i^{-1}} (\log((\mathbf{y}_{i+1} \mathbf{y}_i^{-1})/dt)). \quad (24)$$

The last term, $Q_{\text{contact force}}$, minimizes the joint torque and the contact force:

$$Q_{\text{contact force}} = (\boldsymbol{\tau}^T, \boldsymbol{\lambda}^T) \mathbf{W}_f (\boldsymbol{\tau}^T, \boldsymbol{\lambda}^T)^T, \quad (25)$$

where $\mathbf{W}_f$ is the diagonal weighting matrix. Term $Q_{\text{contact force}}$ is necessary to make the Hessian matrix of the objective function positive definite. Also, this term reduces impacts during the support phase, and thus is helpful for generating smooth motions.

The resulting accelerations $\ddot{\mathbf{q}}$ from the quadratic problem are integrated to achieve a forward dynamics simulation. By looping through the above three steps of state estimation, motion planning, and tracking, the controller can generate various motions. All the steps are executed at 120hz, which is the same frequency as the reference motions.

6. OPTIMIZATION

We improve the quality of the simulated motion using optimization as an additional offline step. Although our approach produces a working controller without optimization, the discrepancy between the simple model, inverted pendulum, and the full-body character can lead to degraded motion quality. We compensate for such errors

by computing corrections to the output of the motion planner and the tracking controller.

To measure the motion quality of the simulated motion, we first generate a reference human trajectory using the motion planner. The objective function is the difference between the reference motion from the motion planner and the simulated motion. Specifically, we measure the sum of squared pose differences and the COM-trajectory difference between the reference trajectory and the simulated motion. The pose difference is measured using the distance between two sample point clouds matched by vertically rotating and horizontally translating the second cloud to best match the first [Kovar et al. 2002a]. The sample points are evenly distributed over the entire body. The trajectory difference is also measured using the same metric. The two terms are weighted such that they have similar variance.

We optimize corrections for desired end-effector positions. Specifically, we optimize control points for piecewise linear displacement maps. The i th control points for correcting the position of the end-effector k $\Delta \mathbf{y}_{i,k}$, $k \in \{1, 2, 3, 4\}$ are optimized only when they have non-zero weights, and the end-effector either is contacting the ground or is near the ground. These corrections manipulate the contact forces so that the output trajectory becomes more similar to the desired trajectory. A piecewise linear curve with three key-frames is used for each motion segment, at the first frame, at the middle of the segment, and at the final frame of the segment. When a motion segment is short (e.g., double stance phases in walking and single support phase in running), only two key-frames are used to reduce the dimensionality of the optimization space. The first key-frame of a segment is constrained to be the same as the last key-frame of the previous segment along all possible graph paths to ensure continuity.

This setup corresponds to a 36-dimensional search space when optimizing cyclic walking and running motions but reduces to 18 dimensions if we assume symmetry between the left and right leg. In our experiments, optimization was performed first for 10 strides and then refined for an additional 20 strides.

For non-cyclic motions, the set of variables becomes too large to be optimized (≥ 100 for backflips). In our experiments, optimization in more than 25 dimensions was likely to lead to a local minima. Therefore, we use the multi-stage optimization scheme shown in Figure 9 that was used to obtain pendulum controllers. In the f^{th} stage of the optimization, only a subset of the key-frames that belong to segments $s \in \{s_f, s_{f+1}, \dots, s_e\}$ are optimized, where the last segment e is the largest number such that the number of variables does not exceed 25. The objective function is evaluated for synthesized motion that corresponds to segments $s_f, s_{f+1}, \dots, s_{e+2}$ so that a small window of the future is considered (two segments long in our experiments). Whenever non-recoverable errors, such as a fall, are detected, the optimization restarts from the first stage using the current best solution as the initial solution. The multi-stage optimization finishes once all variables are optimized.

Because the search space contains many local minima, we used a randomized optimization algorithm called the covariance matrix adaption evolution scheme (CMAes) [Hansen and Ostermeier 1996]. For each subset, 100 iterations were performed. This optimization took a maximum of 9 hours on a cluster using 8 nodes (64 cores). Timing statistics are summarized in Table 1.

7. RESULTS

In order to test our control algorithms, we simulate a variety of motions: standing, walking, running, skipping, spinning, backflips, a Popa jump, a straddle jump, and a roundoff. The roundoff is

Table I. Statistics on the Computation Times for Each Step of Our Approach Applied to the Walking and the Roundoff Motions

	computation time		note
	walk	roundoff	
Analysis (preprocessing)	150s	137s	200 iterations
Optimization (preprocessing)	312m	397m	100 iterations
Synthesis (online)	21s	20s	10s of motion

Fig. 13. The controllers for human walking and running are robust to changes in the environment and user commands.

similar to a cartwheel except that the gymnast lands with two feet placed together on the ground, facing backwards. The Popa jump is a straddle pike jump with full turn on the floor (Figure 5), and is named after Celestina Popa [Wikipedia 2014]. Each motion is generated from a single captured reference motion of the desired behavior. We also show a controller that makes a transition from a backflip motion to an existing walking controller. Examples of each type of motion are shown in the accompanying video.

Overall, the simulated gymnastic motions are quite similar to the captured motions. However, there are also some artifacts. For example, the simulated character performs the roundoff without using her left hand, and the character bounces off the ground for a short duration after landing from the backflip. There were other undesirable short flight phases that look like foot-skate artifacts, though they are technically not. Increasing the weight for the contact force minimization term and decreasing gains removes some of those artifacts.

A controller can produce significant variation without re-optimization (Figure 13(a)). In the accompanying video, the character runs and makes several turns. The controllers can also recover from external disturbances. Approximately 500N to 800N forces of duration 0.2s are applied at chest height. Our running controller can recover from multiple 1,000N, 0.2s pushes from the side, as shown in the video. The controller is the most robust when pushed from the side, but can withstand about 70% of the maximum force regardless of the pushing direction and timing. The same controller can be used to generate running motions on a 5-degree uphill or a 15-degree downhill slope.

The walking controller has a similar robustness to disturbances. As shown in Figures 14 and 1, the motions have similar speed profiles and appearance. As shown in Figure 13(b), three 300N forces of duration 0.2s are applied at chest height. The robustness varies depending on the pushing direction and the phase of the gait cycle when the disturbance occurs. Our walking controller can recover from 500N, 0.2s pushes when pushed from the side. Figure 13(c) and (d) show that the same controller can be used

Fig. 14. Comparison between the captured walking motion and simulated motion.

to generate walking motions on a 10-degree uphill or a 25-degree downhill slope.

8. DISCUSSION

In this article, we present an approach to constructing a unified framework for simulating multiple behaviors. Our approach analyzes a reference motion data sequence to extract information on each behavior, which is then used to adjust an LQR controller for an inverted pendulum. When the motion is synthesized, a motion planner is used to compute where the hands and feet should be placed and the details of the trajectory that the dynamic simulation should track. An optimization loop adjusts a set of control points to make the motion more robust and more natural looking.

We manually tune the weighting factors for the LQR controller and the QP objective function. Although a wide range of constants

work in practice, we noticed that increasing the gains improved robustness at the expense of motion quality. A more thorough analysis of this tradeoff would be an interesting area for future work.

Unlike the walking and running controllers, none of the gymnastic controllers, except the handstand, were robust enough to withstand 50N, 0.2s external pushes. We believe that the controllers failed because they are constrained to follow a fixed path from the motion graph. For example, the standing balance controller is always chosen after landing. However, even a very slight push of a human performing such a dynamic motion would result in rapid foot steps to avoid a fall. To create a more robust gymnastic controller, multiple balancing strategies (e.g., the ankle strategy, hip strategy, or stepping strategy) should be chosen based on the state of the simulation.

Because we do not take into account collisions between the limbs, the limbs may intersect when the character is pushed from the side, or is making sharp turns. We believe that this problem could be fixed by adding a term that penalizes intersections to the objective function of the optimization (Section 6), and by allowing the baseline (footstep spacing) of the running motion to be a parameter in the optimization.

Our control system is based on an inverted pendulum, and therefore, it mainly uses the placement of the foot or the torque around the ankle joint for balance recovery. Humans often use arm motions or upper-body motions to assist in recovery [Shiratori et al. 2009]. The use of a more complex model such as a double inverted pendulum or reaction wheel pendulum might further improve the robustness of our scheme and allow additional motion to be mapped onto the upper body. We believe that the idea of momentum mapping can be generalized to other types of simplified models. However, the cost and difficulty of the analysis and optimization would increase if the models had more DOFs.

The current implementation of our scheme is about two times slower than real time. However, a speedup could be achieved via code optimization. The main bottlenecks are the prediction of the future pendulum trajectory performed at 120hz, the calculation of the system matrix for the full human model, the quadratic programming, and the use of a scripting language for easier debugging. The control might be equally robust at a lower frequency of motion planning, which would reduce the required computation time significantly.

APPENDIX A

The linear quadratic regulator controller is derived by writing linearized dynamics of the simplified model about the upright pose in a state-space form: $\dot{\mathbf{s}} = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{u}$, where $\mathbf{s}$ is a state-vector containing the joint states and their time-derivatives. The actuation force applied to the pendulum (the PIP model), to the cart (the IPC model), or to the mass (the LM model) is $\mathbf{u}$. For the IPC model, we use the LQR matrix $\mathbf{Q} = \text{diag}(0, 30000, 30000, 0)$ and $\mathbf{R} = 1$ to minimize deviations from the desired speed of the cart while minimizing the leaning angle of the pendulum and using minimal control force. For the PIP model, we use the LQR matrix $\mathbf{Q} = \text{diag}(30000, 30000)$ to minimize deviations from the desired leaning angle of the pendulum while minimizing the angular velocity of the pendulum and using minimal control torque. For the LM model, we use the LQR matrix $\mathbf{Q} = \text{diag}(30000, 0.01)$ to minimize deviations from the desired height of the COM using minimal control force. Although we linearize the pendulum dynamics as if the height and the inertia matrix of the pendulum is constant, we update those values at every frame using the COM height and the composite rigid body inertia (CRB inertia) matrix of the human character at the corresponding

frame in the captured reference motion. This technique is sometimes called the state-dependent Riccati equation (SDRE) control, and has proven to be effective in practice despite its sub-optimality [Cloutier 1997].

Our pendulum model is similar to the reaction mass pendulum (RMP) model [Lee 2007] in that the inertia of the full-body character is considered. However, our model uses different balancing strategies. The RMP model uses an “inertia-shaping” strategy, which can be effective for motions involving rapid rotations, such as diving and figure skating. Although this strategy can be reformulated based on our momentum mapping, we choose to use the foot-placement balancing strategy of the IPC model, and the ankle strategy of the PIP model because we focus on foot-step driven gymnastic actions in this work.

APPENDIX B

The generalized momentum of a human character $\mathbf{H}_{\text{COM}}$ for pose alignment can be calculated as follows:

$$\mathbf{H}_{\text{COM}} = \sum_b \text{Ad}^*_{(\mathbf{T}_{\text{Global}}^{\text{COM}} \mathbf{T}_b)^{-1}} (\mathbf{I}_b \mathbf{V}_b), \quad (26)$$

where $\text{Ad}^*(\cdot)$ is the wrench transformation operator of the Lie-group formulation [Park et al. 1995], $\text{Ad}^*_{(\mathbf{T}_{\text{Global}}^{\text{COM}} \mathbf{T}_b)^{-1}}(\cdot)$ maps a body-local momentum to a global momentum about the COM of the human character, $\mathbf{T}_{\text{Global}}^{\text{COM}} \in SE3$ is the transformation between the COM of the full-body human and the global frame, $\mathbf{T}_b$ is the translational transformation between the global frame and the b th body frame of the current simulated pose, $\mathbf{I}_b$ is the generalized inertia, and $\mathbf{V}_b \in se3$ is the generalized velocity of the b th body part with respect to the b -th body frame. We calculate each body velocity $\mathbf{V}_b$ assuming that the body moves steadily from the current simulated configuration to the reference configuration using a shortest path screw motion $\exp(\mathbf{V}_b s)$, $s \in [0, 1]$. Such a motion is unique, and the velocity $\mathbf{V}_b$ can be calculated analytically by:

$$\mathbf{V}_b = \text{Ad}_{\bar{\mathbf{T}}_b^{-1}} (\log(\bar{\mathbf{T}}_b \mathbf{T}_b^{-1})), \quad (27)$$

where $\text{Ad}_{\bar{\mathbf{T}}_b^{-1}}$ is the twist transformation that maps a spatial velocity to a body-local velocity, and $\bar{\mathbf{T}} \in SE3_b$ is the transformation of the b th body frame of the corresponding pose in the reference motion. We use overlined letters to denote properties obtained from the reference trajectories to distinguish them from those computed from the predicted/desired trajectories. We use reference trajectories that are approximately aligned to the current simulated pose, using horizontal translation and vertical rotation based on the local coordinate system of the character. Our momentum-based state estimation scheme is not sensitive to this initial alignment, and any other reasonable initial alignment would work equally well.

ACKNOWLEDGMENTS

We thank Junggon Kim for providing his dynamics simulator, Spencer Diaz and Moshe Mahler for character modeling and video editing, and Justin Macey for motion capture sessions. We thank the anonymous reviewers, Jongmin Kim, Yoonsang Lee, Young J. Kim, and Yun-hyeong Kim for their careful reading of our manuscript and their many insightful comments and suggestions. We also thank Autodesk for their donation of the 3D animation and rendering package, Maya.

REFERENCES

- Yeuhi Abe, Marco da Silva, and Jovan Popović. 2007. Multiobjective control with frictional contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 249–258.
- Okan Arikan and David A. Forsyth. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics* 21, 3 (2002), 483–490.
- Mazen Al Borno, Eugene Fiume, A. Hertzmann, and M. de Lasa. 2014. Feedback control for rotational movements in feature space. *Computer Graphics Forum* (2014).
- David Brown, Adriano Macchietto, Kang Kang Yin, and Victor Zordan. 2013. Control of rotational dynamics for ground behaviors. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 55–61.
- James R. Cloutier. 1997. State-dependent Riccati equation techniques: An overview. In *Proceedings of the 1997 American Control Conference*. 932–936.
- Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2009. Robust task-based control policies for physics-based characters. *ACM Transactions on Graphics* 28, 5 (2009), 1–9.
- Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2010. Generalized biped walking control. *ACM Transactions on Graphics* 29, 4 (2010), 130.
- Marco da Silva, Yeuhi Abe, and Jovan Popović. 2008. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics* 27, 3 (2008), 1–10.
- Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-based locomotion controllers. *ACM Transactions on Graphics* 29, 4 (2010), 131.
- Peter Dorato, Vito Cerone, and Chaouki Abdallah. 1994. *Linear-Quadratic Control: An Introduction*. Simon & Schuster.
- Sehoon Ha and C. Karen Liu. 2014. Iterative training of dynamic skills inspired by human coaching techniques. *ACM Transactions on Graphics* 34, 1 (2014), 1:1–1:11.
- Sehoon Ha, Yuting Ye, and C. Karen Liu. 2012. Falling and landing motion control for character animation. *ACM Transactions on Graphics* 31, 6 (2012), 155.
- Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *International Conference on Evolutionary Computation*. 312–317.
- Rachel Heck and Michael Gleicher. 2007. Parametric motion graphs. In *Proceedings of the 2007 ACM Symposium on Interactive 3D Graphics and Games*. 129–136.
- Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating human athletics. In *ACM SIGGRAPH*. 71–78.
- Shuuji Kajita, Takashi Nagasaki, Kenji Kaneko, Kazuhito Yokoi, and Kazuo Tanie. 2004. A hop towards running humanoid biped. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*. 629–635.
- Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002a. Motion graphs. *ACM Transactions on Graphics* 21, 3 (2002), 473–482.
- Lucas Kovar, John Schreiner, and Michael Gleicher. 2002b. Footskate cleanup for motion capture editing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 97–104.
- Taesoo Kwon and Jessica Hodgins. 2010. Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 129–138.
- Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics* 21, 3 (2002), 491–500.
- Sung-hee Lee. 2007. Reaction mass pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*.
- Sung-Hee Lee and Ambarish Goswami. 2012. A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Autonomous Robots* 33, 4 (2012), 399–414.
- Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010. Data-driven biped control. *ACM Transactions on Graphics* 29, 4 (2010), 129.
- Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics* 33, 6 (2014), 218:1–218:11.
- Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010. Motion fields for interactive character locomotion. *ACM Transactions on Graphics* 29, 6 (2010), 138.
- Sergey Levine and Jovan Popović. 2012. Physically plausible simulation for character animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 221–230.
- Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics* 35, 3 (2016).
- Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. *ACM Transactions on Graphics* 29, 4 (2010), 128.
- Adriano Macchietto, Victor Zordan, and Christian R. Shelton. 2009. Momentum control for balance. *ACM Transactions on Graphics* 28, 3 (2009), 1–8.
- James McCann and Nancy Pollard. 2007. Responsive characters from motion fragments. *ACM Transactions on Graphics* 26, 3 (2007), 6.
- Igor Mordatch, Martin de Lasa, and Aaron Hertzmann. 2010. Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics* 29, 4 (2010), 71.
- Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics* 28, 3 (2009), 1–9.
- Uldarico Muico, Jovan Popović, and Zoran Popović. 2011. Composite control of physically simulated characters. *ACM Transactions on Graphics* 30, 3 (2011), 16.
- Frank C. Park, James E. Bobrow, and Scott R. Ploen. 1995. A lie group formulation of robot dynamics. *The International Journal of Robotics Research* 14, 6 (1995), 609–618.
- Robert R. Playter and Marc Raibert. 1992. Control of a biped somersault in 3D. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1. 582–589.
- Marko B. Popovic, Andreas G. Hofmann, and Hugh M. Herr. 2004. Angular momentum regulation during human walking: Biomechanics and control. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*. 2405–2411.
- Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics* 23, 3 (2004), 514–521.
- Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. 2001. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics* 20, 2 (2001), 67–94.
- Hyun Joon Shin and Hyun Seok Oh. 2006. Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 291–298.
- Takaaki Shiratori, Brooke Coley Raki Cham, and Jessica K. Hodgins. 2009. Simulating balance recovery responses to trips based on biomechanical

- principles. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 37–46.
- Tomomichi Sugihara. 2008. Simulated regulator to synthesize ZMP manipulation and foot location for autonomous control of biped robots. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*. 1264–1269.
- Ryosuke Tajima, Daisaku Honda, and Keisuke Suga. 2009. Fast running experiments involving a humanoid robot. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*. 1571–1576.
- Adrien Treuille, Yongjoon Lee, and Zoran Popović. 2007. Near-optimal character animation with continuous control. *ACM Transactions on Graphics* 26, 3 (2007), 7.
- Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2009. Optimizing walking controllers. *ACM Transactions on Graphics* 28, 5 (2009), 1–8.
- Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2010. Optimizing walking controllers for uncertain inputs and environments. *ACM Transactions on Graphics* 29, 4 (2010), 73.
- Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics* 31, 4 (2012), 25.
- Wikipedia. 2014. Celestina Popa—Wikipedia, The Free Encyclopedia. (2014). [Online; accessed 3-April-2014]. https://en.wikipedia.org/wiki/Celestina_Popa.
- Chun-Chih Wu and Victor Zordan. 2010. Goal-directed stepping with momentum control. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 113–118.
- Jia-chi Wu and Zoran Popović. 2010. Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics* 29, 4 (2010), 72.
- M. Yamakita, M. Kishikawa, and T. Sadahiro. 2003. Motion control for robust landing of acrobat robot (SMB). In *Proceedings of the International Conference on Intelligent Robots and Systems*. IEEE, 1141–1146.
- Yuting Ye and C. Karen Liu. 2010. Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics* 29, 4 (2010), 74.
- KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2008. Continuation methods for adapting simulated skills. *ACM Transactions on Graphics* 27, 3 (2008), 1–7.
- KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple biped locomotion control. *ACM Transactions on Graphics* 26, 3 (2007), 105.
- Victor B. Zordan, David F. Brown, Adriano Macchietto, and KangKang Yin. 2014. Control of rotational dynamics for ground and aerial behavior. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1356–1366.

Received December 2014; revised September 2016; accepted September 2016