The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Simultaneous Tracking and Balancing of Humanoid Robots for Imitating Human Motion Capture Data

Katsu Yamane*† and Jessica Hodgins†*
*Disney Research, Pittsburgh †Carnegie Mellon University
Email: {kyamane|jkh}@disneyresearch.com

Abstract— **This paper presents a control framework for humanoid robots that uses all joints simultaneously to track motion capture data and maintain balance. The controller comprises two main components: a balance controller and a tracking controller. The balance controller uses a regulator designed for a simplified humanoid model to obtain the desired input to keep balance based on the current state of the robot. The simplified model is chosen so that a regulator can be designed systematically using, for example, optimal control. An example of such controller is a linear quadratic regulator designed for an inverted pendulum model. The desired inputs are typically the center of pressure and/or torques of some representative joints. The tracking controller then computes the joint torques that minimize the difference from desired inputs as well as the error from desired joint accelerations to track the motion capture data, considering exact full-body dynamics. We demonstrate that the proposed controller effectively reproduces different styles of storytelling motion using dynamics simulation considering limitations in hardware.**

Index Terms— **Humanoid Robots, Motion Capture Data, Balancing**

I. Introduction

Programming humanoid robots, especially to perform natural, human-like motions is a difficult task. They are usually programmed manually or by numerical optimization techniques to minimize, for example, energy consumption subject to dynamics and/or kinematics constraints. Although human motion capture data is potentially a good starting point, it is difficult to map captured data to humanoid robots because of differences in kinematics and dynamics parameters. In fact, most of the work in mapping human motion capture data to other humanoid models has been in the graphics field where full-body dynamics is not usually considered.

In this paper, we propose a control framework for humanoid robots that uses all joints simultaneously to track motion capture data and maintain balance (Fig. 1). We focus on tracking joint angle trajectories, although some tasks may require tracking other quantities such as end-effector trajectories which will be addressed in future work. Although the current controller only works in double support, the balancing task is distributed among all joints including those in the upper body. In addition, the controller does not require segmentation or intensive pre-processing of motion capture data, which makes it potentially applicable to realtime applications.

The controller comprises two components: a balance controller and a tracking controller. The balance controller attempts to keep the whole body balanced by using a simplified model for which robust balance controller can be easily designed. A typical example is an inverted pendulum with linear quadratic regulator (LQR), which we will use for our examples. The balance controller gives desired inputs to keep the simplified model balanced. Examples of such inputs include center of pressure (COP) and torques of representative joints.

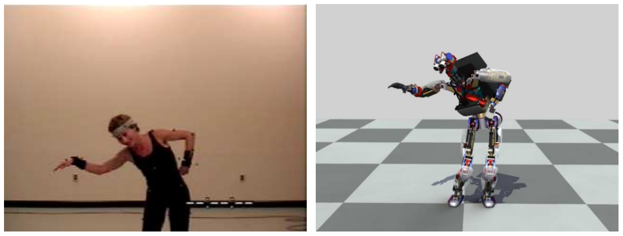


Fig. 1. An example of original human motion (left) and simulated humanoid motion (right).

The tracking controller tries to make the joints follow the reference trajectory specified by the motion capture data, while respecting the desired inputs given by the balance controller. Joint trajectory tracking is enabled by commanding desired joint accelerations based on joint angle and velocity errors as well as feedforward joint accelerations. The tracking controller then solves an optimization problem with a quadratic cost function including errors from desired inputs and joint accelerations.

We demonstrate the balancing and tracking ability of the proposed controller with a full-body dynamics simulation that takes into account joint velocity and torque limits. After showing basic balancing capability, we apply the controller to tracking motion capture clips of two subjects telling the same story. The resulting robot motions clearly preserves the original style of each subject. We also demonstrate the robustness by perturbing the inertial parameters of the simulation model.

This paper is organized as follows. In the next section, we review related prior work in the humanoid control field with focus on approaches using human motion data as reference. Section III gives the overview of the controller and basic equations for its derivation. Two main components, balance and tracking controller, are described in detail in Sections IV and V respectively. We present simulation examples in Section VI, followed by concluding remarks.

II. Related Work

Most of the current successful humanoid robots are programmed by first determining a center of pressure (COP, also known as zero moment point) trajectory based on the footprint and generating a physically consistent center of gravity (COG) trajectory using a simplified dynamics model such as inverted pendulum, and then computing the inverse kinematics to obtain the joint angles that satisfy the planned COG trajectory and footprints [1]–[6]. Some work [7], [8] also uses an inverted pendulum model for balancing. These frameworks have been successful because the reference motion is guaranteed to be physically consistent, but are not generally capable of tracking motion capture data because adding the reference joint trajectory on top of the planned trajectory might break the physical consistency.

Although applying human motion capture data to humanoid robots has been a growing area of research recently, few of these approaches successfully controlled real hardware with a floating base. Most work has focused on mapping human motion to humanoid robots with fewer degrees of freedom [9], [10] and categorizing human motion into different behaviors for humanoid motion synthesis [11], [12]. Safonova et al. [13] adapted captured upper body motions to humanoid robots considering the kinematic constraints such as joint angle and velocity limits. Converting motion capture data sequences to satisfy the full-body dynamics constraints of free-floating humanoids has been addressed in robotics [14], [15] as well as graphics [16], [17], but they are focused on planning and do not address the issue of recovering balance under disturbances.

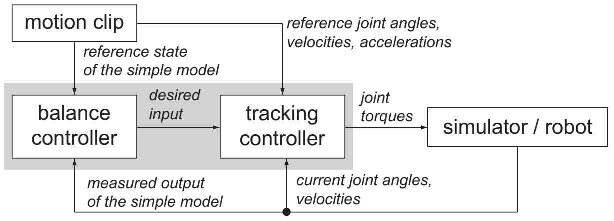
In fact, very few papers successfully controlled humanoid hardware based on human motion capture data. Nakaoka et al. [18] realized robot dancing motions by manually segmenting human motion data into different tasks and constructing a controller for each task. Sugihara et al. [19] proposed a method to generate physically consistent motion by optimizing COG trajectory based on an inverted pendulum model while respecting the reference joint trajectories during the inverse kinematics computation. However, both approaches require manual work for designing controllers or pre-processing captured data.

Some work has realized online tracking of upper-body motions during double support in full-body simulation [20] and hardware [21]. However, both approaches use the lower body specifically for balancing and therefore are not fully capable of tracking leg motions that may conflict with the balancing task.

III. Overview

A. Controller

Figure 2 shows the overview of the controller. The two main components are a balance controller and a tracking controller. The balance controller is responsible for keeping the whole body balanced, usually using a controller designed for a simplified dynamics model such as LQR for a linear inverted pendulum model. The output of the balance controller is the desired input to the simplified model such as center of pressure and/or torques of representative joints. The tracking controller is responsible for making every joint track the desired trajectory. It solves an optimization problem that respects both joint tracking and desired inputs to the simplified model and obtains the joint torques to be commanded to the robot.



motion clip
reference state of the simple model
reference joint angles, velocities, accelerations
balance controller
desired input
tracking controller
joint torques
simulator / robot
measured output of the simple model
current joint angles, velocities

Fig. 2. Overview of the controller.

B. Motion Capture Data Processing

We assume the use of a commercial optical motion capture system to obtain the reference motion data, which typically consist of a set of marker trajectories in the Cartesian space. The marker data are labeled and cleaned as in the normal motion capture pipeline. The data is then scaled to fit the robot's size and converted to joint angle data for the robot by an inverse kinematics algorithm [22] taking into account the joint motion range. Due to the joint limits and the difference between the kinematics of the subject and robot, the joint angle data usually have problems such as foot skating at points of contact. The original motion capture data are usually cleaned up offline to remove such problems.

In our pre-processing, we assume that both feet are flat on the floor at the initial frame and estimate the correct foot position and orientation by projecting those obtained using the raw marker data onto the floor. We then compute the inverse kinematics for new foot locations to obtain the cleaned joint angles and retain the difference from original joint angles. At each frame during control, we add the difference to the original data to obtain the cleaned reference joint angles. Although this correction is extremely simple, our controller does not require further cleanup.

C. Notations and Basic Equations

We denote the number of actuated joints of the robot by N_J. The total degrees of freedom (DOF) of the robot is then $N_G = N_J + 6$ including the 6 DOF of the translation and rotation of the root joint. The robot configuration is uniquely defined by the generalized coordinate $\boldsymbol{q} \in \Re^{N_G}$ whose first 6 components correspond to the root joint. We also denote the generalized force by $\boldsymbol{\tau}_G \in \Re^{N_G}$.

Humanoid robots usually move with some of their links in contact with the environment. Let N_C denote the number of links in contact. We represent the linear and angular velocities of the i-th contact link by a 6-dimensional vector $\dot{\boldsymbol{r}}_{ci}$. The relationship between the generalized velocity $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{r}}_{ci}$ is written as

$$\dot{\boldsymbol{r}}_{ci} = \boldsymbol{J}_{ci}\dot{\boldsymbol{q}} \tag{1}$$

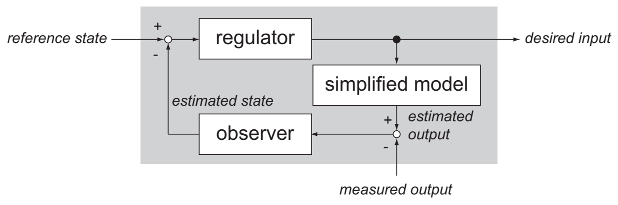


Fig. 3. Overview of the balance controller.

where $\boldsymbol{J}_{ci} \in \Re^{6\times N_G}$ is the Jacobian matrix of the i-th contact link's position and orientation with respect to the generalized coordinates. Differentiating Eq.(1), we obtain the relationship of the accelerations:

$$\ddot{\boldsymbol{r}}_{ci} = \boldsymbol{J}_{ci}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{ci}\dot{\boldsymbol{q}}. \tag{2}$$

We define the compound contact Jacobian matrix $\boldsymbol{J}_c$ by

$$\boldsymbol{J}_c = \begin{pmatrix} \boldsymbol{J}_{c1} \\ \boldsymbol{J}_{c2} \\ \vdots \\ \boldsymbol{J}_{cN_C} \end{pmatrix} \in \Re^{6N_C\times N_G}. \tag{3}$$

Because the root joint is not actuated, we can only control the joint torque vector $\boldsymbol{\tau}_J \in \Re^{N_J}$. In addition, each of the N_C links in contact with the environment receives contact force $\boldsymbol{f}_{ci}$ and moment around the link local frame $\boldsymbol{n}_{ci}$ $(i = 1, 2, \dots, N_C)$. We also define the compound contact force/moment vector by $\boldsymbol{f}_c = \left(\boldsymbol{f}_{c1}^T\ \boldsymbol{n}_{c1}^T\ \dots\ \boldsymbol{f}_{cN_C}^T\ \boldsymbol{n}_{cN_C}^T\right)^T \in \Re^{6N_C}$.

The equation of motion of the robot is written as

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{c} = \boldsymbol{N}^T\boldsymbol{\tau}_J + \boldsymbol{J}_c^T\boldsymbol{f}_c \tag{4}$$

where $\boldsymbol{M} \in \Re^{N_G\times N_G}$ is the joint-space inertia matrix and $\boldsymbol{c} \in \Re^{N_G}$ is the sum of Coriolis, centrifugal and gravity forces. Matrix $\boldsymbol{N} \in \Re^{N_J\times N_G}$ is used to map the joint torques into the generalized forces and has the form

$$\boldsymbol{N} = \begin{pmatrix} \boldsymbol{0}_{N_J\times 6} & \boldsymbol{1}_{N_J\times N_J} \end{pmatrix} \tag{5}$$

where $\boldsymbol{0}_*$ and $\boldsymbol{1}_*$ are zero and identity matrices of the sizes indicated by their subscripts respectively.

IV. Balance Controller

Figure 3 shows the structure of the balance controller. The controller consists of two main components: a regulator to compute the input to the simplified model to keep it balanced, and an observer to estimate the current state based on measurements. We can use any simplified model as long as it represents the dynamics of the humanoid robot and a balance controller can be designed. A typical example is a linear inverted pendulum, for which a regulator can be easily designed by pole placement or optimal control.

A. Details

Let us assume that the simplified model is linear and represented by the following state-space differential equation:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} & (6)\\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} & (7)\end{aligned}$$

where $\boldsymbol{x}$ is the state, $\boldsymbol{u}$ is the input, and $\boldsymbol{y}$ is the output of the simplified model. Also assume that we have designed a state feedback controller for balancing:

$$\boldsymbol{u} = \boldsymbol{K}(\boldsymbol{x}_{ref} - \boldsymbol{x}) \tag{8}$$

where $\boldsymbol{K}$ is a constant gain matrix and $\boldsymbol{x}_{ref}$ is a reference state, typically computed from the reference motion.

The observer compares the estimated and actual outputs to update the state estimate $\hat{\boldsymbol{x}}$ as

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{F}(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{9}$$

where $\boldsymbol{F}$ is the observer gain and $\hat{\boldsymbol{y}} = \boldsymbol{C}\hat{\boldsymbol{x}}$ is the estimated output. Because we do not have access to real state, we replace the state $\boldsymbol{x}$ with its estimate $\hat{\boldsymbol{x}}$ in Eq.(8):

$$\boldsymbol{u} = \boldsymbol{K}(\boldsymbol{x}_{ref} - \hat{\boldsymbol{x}}). \tag{10}$$

Using Eqs. (6), (7), (9) and (10), we obtain the following system of the estimated state and new input $\boldsymbol{u}_b = \left(\boldsymbol{x}_{ref}^T\ \boldsymbol{y}^T\right)^T$:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}_b\hat{\boldsymbol{x}} + \boldsymbol{B}_b\boldsymbol{u}_b \tag{11}$$

where

$$\begin{aligned} \boldsymbol{A}_b &= \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} - \boldsymbol{F}\boldsymbol{C} \\ \boldsymbol{B}_b &= \begin{pmatrix} \boldsymbol{B} & -\boldsymbol{F} \end{pmatrix}. \end{aligned}$$

Equation (11) describes how to estimate the current state of the simplified model based on a reference state and measured output. The estimated state and input to the simplified model computed by Eq.(10) will be used as the input to the tracking controller in Section V.

B. Inverted Pendulum Example

We present an example of a balance controller using a linear inverted pendulum as the simplified model.

Consider the 3-dimensional inverted pendulum model with two active linear joints, two unactuated joints and a point mass, shown in Fig. 4. The location of the linear joints and the point mass correspond to the COP and COM of the full-body model respectively. After linearization, the pendulum can be treated as two independent planar pendulums with joints (x, θ_1) and (y, θ_2). We will therefore use the pendulum with (x, θ_1) as an illustrating example.

We define the state, input and output vectors of the linear inverted pendulum as follows:

$$\begin{aligned} \boldsymbol{x} &= \begin{pmatrix} x & \theta_1 & \dot{x} & \dot{\theta}_1 \end{pmatrix}^T & (12)\\ \boldsymbol{u} &= f_x & (13)\\ \boldsymbol{y} &= \begin{pmatrix} x & l\theta_1 \end{pmatrix}^T. & (14)\end{aligned}$$

Note that we use the x coordinate of the point mass as output instead of θ_1. Although f_x is the actual input to the

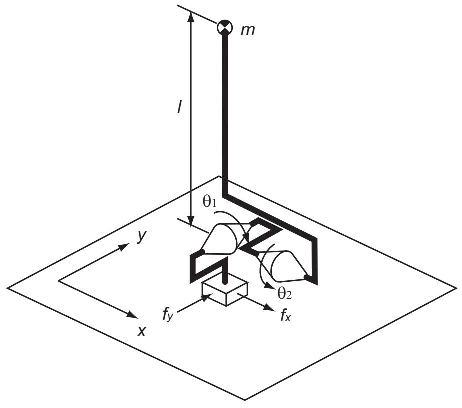


Fig. 4. Inverted pendulum model for the balance controller.

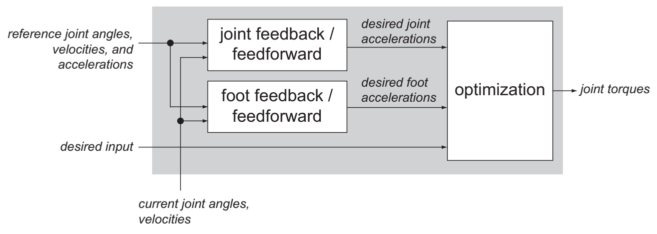

reference joint angles, velocities, and accelerations
joint feedback / feedforward
desired joint accelerations
foot feedback / feedforward
desired foot accelerations
optimization
joint torques
desired input
current joint angles, velocities

Fig. 5. Overview of the tracking controller.

inverted pendulum, there is no corresponding input in the whole-body model. Instead, we use x, which denotes where the COP of the whole-body model should be, as the desired input. Other possible forms of desired input include torques of representative joints if the simplified model contains active rotational joints such as in double inverted pendulum [8].

The reference state of this inverted pendulum model is $\boldsymbol{x}_{ref}^T = (x_{rc}\ 0\ 0\ 0)^T$ where x_{rc} is the x coordinate of the COM position computed from the reference joint angles. The measured output $\boldsymbol{y}$ consists of the x coordinates of the actual COP and COM positions.

We then design a regulator for the inverted pendulum. Here we apply LQR, which determines the state feedback gain $\boldsymbol{K}$ such that the following cost function is minimized:

$$J = \int_0^\infty \left(\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}\right) dt \tag{15}$$

where $\boldsymbol{Q} \geq 0$ and $\boldsymbol{R} > 0$ are symmetric weight matrices. The weight matrices can be easily determined by observing the response to disturbances. For example, we can determine the weights so that the desired COP does not go out of the contact area when the maximum possible external force is applied.

V. Tracking Controller

Figure 5 shows the structure of the tracking controller. The controller consists of two local controllers and a joint torque optimization.

A. Local Controllers

The local controllers compute the desired accelerations of joint and contact links based on the reference and current position and velocity as well as the reference accelerations. In the joint controller, the desired acceleration $\hat{\ddot{q}}$ is computed as follows at each joint:

$$\hat{\ddot{q}} = \ddot{q}_{ref} + k_d(\dot{q}_{ref} - \dot{q}) + k_p(q_{ref} - q) \tag{16}$$

where q is the current joint position, q_{ref} is the reference joint position in the captured data, and k_p and k_d are constant position and velocity gains that may be different for each joint.

We assume that the position and orientation of the root joint is available either by accelerometer and gyro sensors or by computing the kinematics assuming that at least one of the feet is flat on the ground. We can therefore compute the desired linear and angular accelerations of the root joint, and combine them with all desired joint accelerations to form the desired acceleration vector $\hat{\ddot{\boldsymbol{q}}} \in \Re^{N_G}$. Control law (16) is the same as the one used in resolved acceleration control [23] except that the root joint is not actuated and the desired acceleration may be altered by the optimization part described later. In order to keep the feet from slipping, and potentially to realize the desired contact state, we also compute the desired feet accelerations $\hat{\ddot{\boldsymbol{r}}}_c \in \Re^{6N_C}$ using the same control law.

B. Optimizer

The task of the optimizer is to compute the control inputs based on the information obtained so far: $\hat{\ddot{\boldsymbol{q}}}$, $\hat{\ddot{\boldsymbol{r}}}_c$ and the desired input to the simplified model obtained by the balance controller. In most cases, however, these conditions conflict with each other. We therefore perform an optimization to compute a set of joint torques that respects all of these quantities.

The unknowns of the optimization are the joint torques $\boldsymbol{\tau}_J$ and contact forces $\boldsymbol{f}_c$. The cost function to be minimized is

$$Z = Z_s + Z_q + Z_c + Z_\tau + Z_f \tag{17}$$

and each of the five terms will be described in detail in the following paragraphs.

The term Z_s addresses the error from the desired input to the simplified model. Because the mapping from the simplified model to the full-body model can be in any form, here we consider two examples of such a mapping: center of pressure (COP) and torque of a representative joint. Cost function Z_s then becomes the sum of the errors associated with these quantities, i.e.,

$$Z_s = e_{COP} + e_\tau. \tag{18}$$

First consider the case where the desired input includes the desired location of the COP $\boldsymbol{r}_p = (r_{px}\ r_{py}\ 0)^T$. The COP error is represented as

$$e_{COP} = \frac{1}{2}\boldsymbol{f}_c^T\boldsymbol{P}^T\boldsymbol{W}_P\boldsymbol{P}\boldsymbol{f}_c \tag{19}$$

where $\boldsymbol{P}$ is the matrix that maps $\boldsymbol{f}_c$ to the resultant moment around the desired COP and can be computed as follows: we first obtain matrix $\boldsymbol{T} \in \Re^{6\times 6N_C}$ that converts the individual contact forces to total contact force and moment around the world origin by

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_1 & \boldsymbol{T}_2 & \dots & \boldsymbol{T}_{N_C} \end{pmatrix} \tag{20}$$

and

$$\boldsymbol{T}_i = \begin{pmatrix} \boldsymbol{1}_{3\times3} & \boldsymbol{0}_{3\times3} \\ [\boldsymbol{p}_{ci}\times] & \boldsymbol{1}_{3\times3} \end{pmatrix} \tag{21}$$

where $\boldsymbol{p}_{ci}$ is the position of the i-th contact link and $[\boldsymbol{a}\times]$ is the cross product matrix of a 3-dimensional vector $\boldsymbol{a}$. The total force/moment is then converted to resultant moment around COP by multiplying the following matrix:

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & r_{py} & 1 & 0 & 0 \\ 0 & 0 & -r_{px} & 0 & 1 & 0 \end{pmatrix} \tag{22}$$

which leads to $\boldsymbol{P} = \boldsymbol{C}\boldsymbol{T}$.

The case where desired input includes torques of N_r representative joints, $\hat{\boldsymbol{\tau}}_r \in \Re^{N_r}$, is trivial. Let $\boldsymbol{R} \in \Re^{N_r \times N_C}$ be the matrix to extract the torques of representative joints from $\boldsymbol{\tau}_J$. The error can be written as

$$e_\tau = \frac{1}{2}(\hat{\boldsymbol{\tau}}_r - \boldsymbol{R}\boldsymbol{\tau}_J)^T \boldsymbol{W}_r (\hat{\boldsymbol{\tau}}_r - \boldsymbol{R}\boldsymbol{\tau}_J). \tag{23}$$

The term Z_q denotes the error from the desired joint accelerations, i.e.,

$$Z_q = \frac{1}{2}(\hat{\ddot{\boldsymbol{q}}} - \ddot{\boldsymbol{q}})^T \boldsymbol{W}_q (\hat{\ddot{\boldsymbol{q}}} - \ddot{\boldsymbol{q}}). \tag{24}$$

The term Z_c denotes the error from the desired contact link accelerations, i.e.,

$$Z_c = \frac{1}{2}(\hat{\ddot{\boldsymbol{r}}}_c - \ddot{\boldsymbol{r}}_c)^T \boldsymbol{W}_c (\hat{\ddot{\boldsymbol{r}}}_c - \ddot{\boldsymbol{r}}_c). \tag{25}$$

The term $\boldsymbol{Z}_\tau$ is written as

$$Z_\tau = \frac{1}{2}(\hat{\boldsymbol{\tau}}_J - \boldsymbol{\tau}_J)^T \boldsymbol{W}_\tau (\hat{\boldsymbol{\tau}}_J - \boldsymbol{\tau}_J) \tag{26}$$

where $\hat{\boldsymbol{\tau}}_J$ is a reference joint torque, which is typically set to a zero vector and hence Z_τ acts as a damping term for the joint torque.

The term Z_f has a similar role for the contact force, i.e.,

$$Z_f = \frac{1}{2}(\hat{\boldsymbol{f}}_c - \boldsymbol{f}_c)^T \boldsymbol{W}_f (\hat{\boldsymbol{f}}_c - \boldsymbol{f}_c) \tag{27}$$

where $\hat{\boldsymbol{f}}_c$ is a reference contact force, which is also typically set to the zero vector.

Using Eqs. (2) and (4), the cost function can be converted to the following quadratic form:

$$Z = \frac{1}{2}\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{b} + c \tag{28}$$

where $\boldsymbol{y} = (\boldsymbol{\tau}_J^T \ \boldsymbol{f}_c^T)^T$ is the unknown vector.

The optimization problem has an analytical solution

$$\boldsymbol{y} = -\boldsymbol{A}^{-1}\boldsymbol{b}. \tag{29}$$

C. Considering Contact Force and Hardware Limits

We have so far assumed that any contact force is available. In reality, however, frictions and moments around x and y axes have limitations. Real hardware also has limitations in joint angles, velocities and torques. We could add inequality constraints to enforce these constraints, but solving the optimization problem would take significantly longer than simply using Eq.(29).

We deal with these limitations by adjusting the parameters in the optimization instead of adding constraints, hence without changing the solution (29). The drawback is that the limitations are not always met, but the expectation is that the balance controller can compensate for the difference between approximate and exact solutions.

For the contact force limitations, we set larger values for elements of $\boldsymbol{W}_f$ corresponding to the frictions and moments. To address the joint torque limit, we utilize the reference joint torque used in Eq.(26). If any of the joint torques exceeds its limit at a sampling time, we set the corresponding reference torque to the limit in the next sampling time and increase the weight. We can therefore expect that the excess torque would be relatively small and thus having little effect even if the torque is saturated by the limit.

VI. Simulation Results

A. Simulation Setup

We use a dynamics simulator with rigid-body contact model developed at University of Tokyo [24], whose precision has been demonstrated in some simulation settings [25]. We use the model of the humanoid robot developed by Sarcos and owned by Carnegie Mellon University (Fig. 6). The robot has 34 joints in total (excluding hands and eye pan/tilt) and we use 25 of them (fix neck and wrist DOFs) for the experiments.

The joint kinematics and inertial parameters are derived from the CAD model. We used experimentally-verified joint motion range and joint torque limit information as well as the design specification for the joint velocity limit. The joint motion range constraint is enforced during the inverse kinematics computation, but we did not consider the joint motion range in simulation assuming that the joints track the reference trajectory well enough. If a joint velocity comes close to the limit, we add a strong damping torque to reduce the speed. If the optimized joint torque exceeds the limit, it is reset to the maximum value before the simulator computes the joint acceleration.

The weights for LQR cost function (15) are

$$\begin{aligned} \boldsymbol{Q} &= diag\{1.0 \times 10^7 \ 1.0 \times 10^8 \ 1.0 \times 10^2 \ 1.0 \times 10^3\} \\ \boldsymbol{R} &= 1.0 \end{aligned}$$

which were chosen so that COP does not go out of the contact area for a large impact. The observer gains are chosen so that the estimated state converges sufficiently fast compared to the poles of the closed loop. The feedback gains for the joint and contact link tracking are $k_p = 4.0$ and $k_d = 4.0$ except where otherwise noted. All weights for the cost function were chosen to be diagonal with all elements

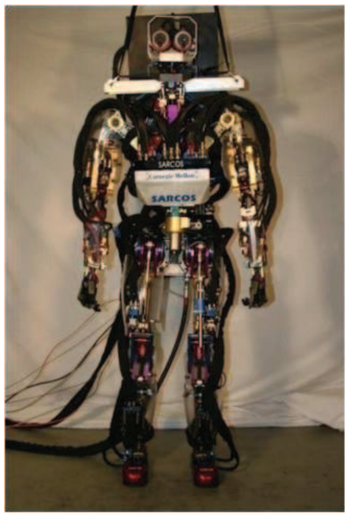


Fig. 6. Humanoid robot for the simulation model.

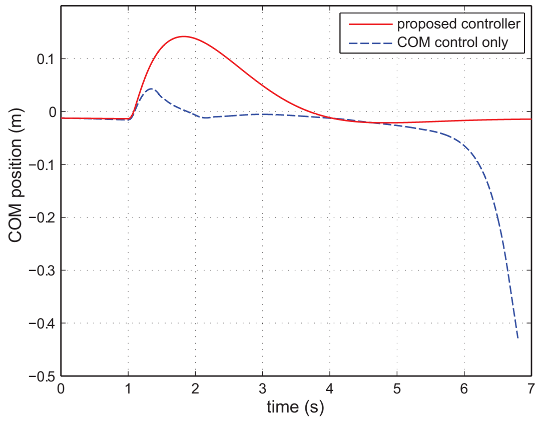


Fig. 7. COM position in the front direction during simple balancing with the COM control only (blue dashed) and proposed controller (red solid).

being 1 except for $\boldsymbol{W}_\tau$ and $\boldsymbol{W}_c$. The diagonal elements of $\boldsymbol{W}_\tau$ were set to 0 by default and, if a joint torque exceeded the limit, the corresponding value was changed to 1. The diagonal elements of $\boldsymbol{W}_c$ were set to 1×10^{-9} and 1×10^{-7} for vertical forces and other forces, respectively.

B. Simple Balancing with Inverted Pendulum Model

We first demonstrate the basic function of the balance controller by using a fixed posture as reference. The robot is hit by a 250 N force at the neck joint from t=1 to 1.1 s while trying to keep the initial posture. As a reference, we used a slightly different version of the controller where the balance controller is replaced by a simple COM position controller that computes the desired COM acceleration to bring it back to the original position with the same feedback/feedforward controller (16) and the same gains. The optimizer then tries to realize the desired COM acceleration instead of desired COP position. The two controllers share the same joint and contact link tracking controllers.

Figure 7 shows the COM position in the forward direction of the robot resulting from two controllers. The proposed

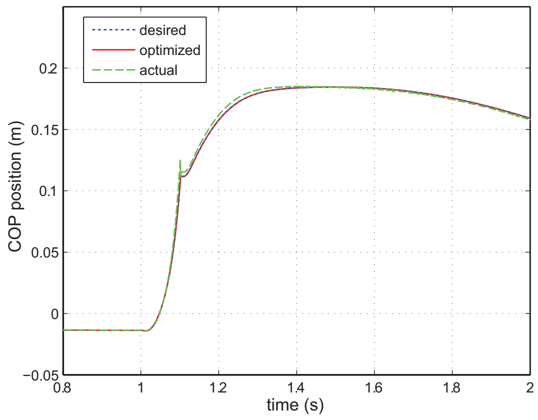


Fig. 8. COP position in the front direction during simple balancing with the proposed controller; blue dotted: desired position by balance controller; red solid: optimized position by optimizer, and green dashed: actual position in simulation.

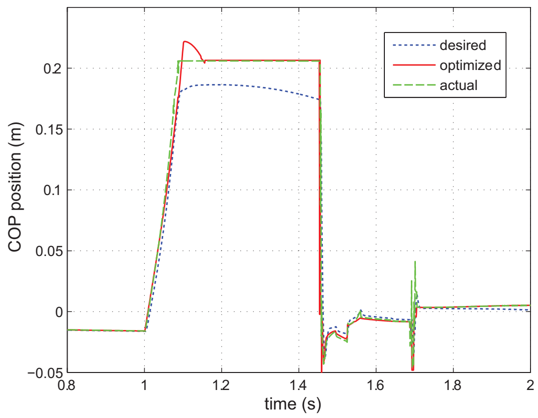


Fig. 9. COP position in the front direction during simple balancing with COM control only.

controller successfully brings the robot back to the original posture. The COM controller version, on the other hand, can stop the COM even earlier than the proposed controller, but the robot eventually falls backward. The reason is that, although the COM controller version can stop the COM motion that action comes at the cost of moving the upper body forward rapidly, and it cannot compensate for the upper body recovery motion. This error can be potentially fixed by tuning the joint and COM feedback gains.

The COP positions under the proposed controller and its COM control version are shown in Figs. 8 and 9 respectively (note that we are using different time scale from Fig. 7 to highlight the most important part). In Fig. 8, the desired and optimized COP are almost identical, and the actual COP is also very close. In Fig. 9 where the desired COP is not considered, the COP also moves forward to give the COM negative acceleration, but the movement is so fast that it eventually has to use more control effort to bring the whole system to equilibrium.

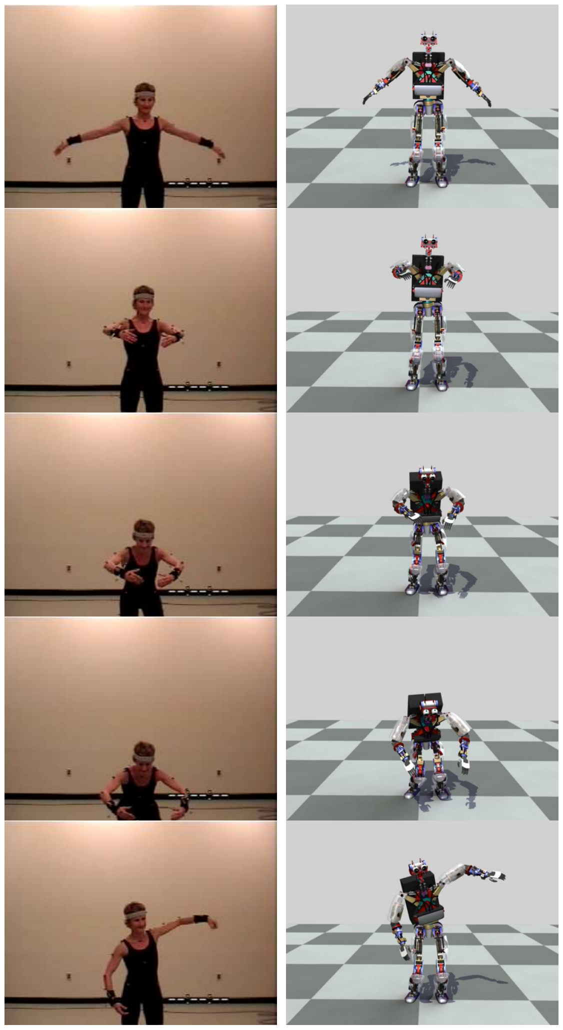


Fig. 10. Original (left) and simulated (right) motions of “I’m a little teapot,” subject 1.

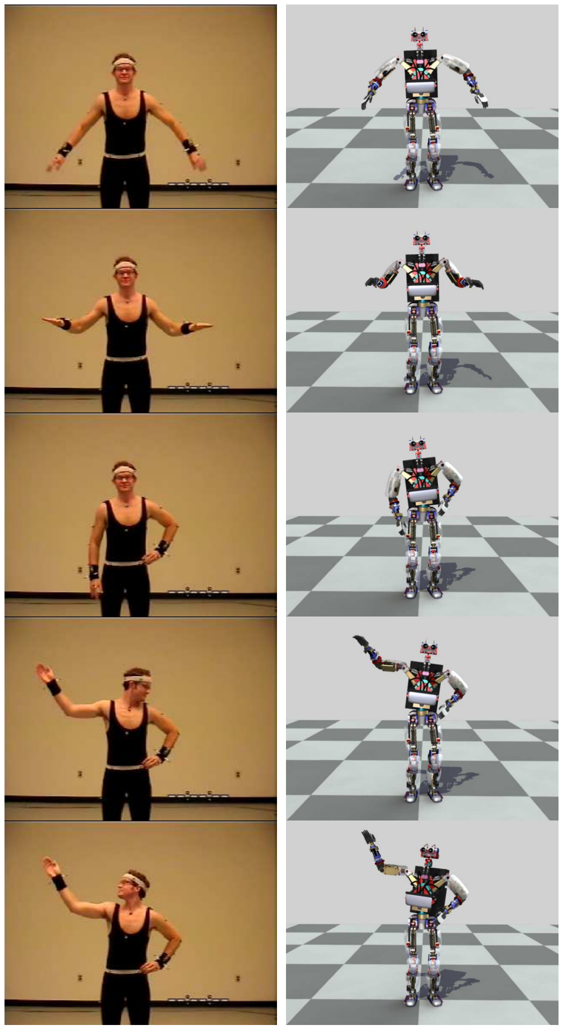


Fig. 11. Original (left) and simulated (right) motions of “I’m a little teapot,” subject 2.

C. Tracking Motion Capture Reference

We use a storytelling motion as an example of motion where joint tracking ability is important. We randomly chose motion capture clips of two actors performing the nursery theme “I’m a little teapot” from CMU Motion Capture Data Library [26]. The comparison of motion capture and simulated robot motions are shown in Figs. 10 and 11 as well as in the supplemental movie. The proposed controller was able to reproduce robot motion that preserves the styles of the original motions.

D. Disturbance Example: Error in Mass Parameters

Finally, we perturbed the inertial parameters of the simulation model to emulate modeling errors due to using CAD models that typically ignore small parts such as wires and, in our hydraulic robot, the significant mass of the oil in the cylinders and tubes. We increased the mass and inertia of each link in the simulation model by a random ratio between 5 and 15%. The reference model for control was kept the same. Because the estimated contact force is always smaller than the actual force, the robot cannot keep standing with the original gain ($k_p = 4.0$, $k_d = 4.0$). However, we could successfully generate similar motion by increasing the gains to $k_p = 16.0$, $k_d = 8.0$ as shown in Fig. 12.

VII. Conclusion

In this paper, we presented a new framework for allowing floating-base humanoid robots to simultaneously keep balance and track motion capture data. The controller combines a balance controller designed for a simplified dynamics model of the robot and a tracking controller for individual joints. The optimizer obtains the joint torques that respect the outputs of both balance and tracking controllers so that the robot can maintain balance while tracking the reference joint trajectory.

As shown in the simulation results, the balance controller can deal with various types of disturbances including differences between simplified and full-body dynamics, inertial parameter errors, joint motions unknown to the balance controller, and external forces. In general, the inverted pendulum model and simple COM feedback control result in

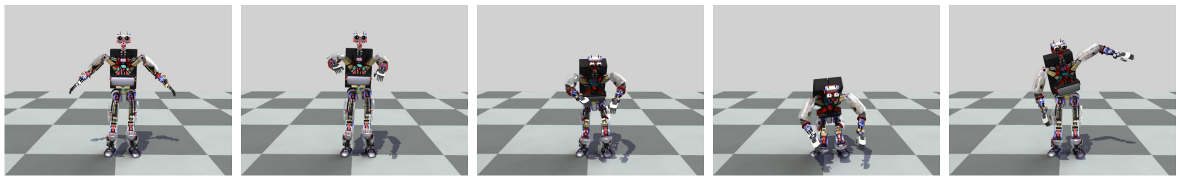


Fig. 12. “I’m a little teapot,” subject 1 with modified inertial parameters for the simulation (shows the same set of frames as Fig. 10).

qualitatively similar COP positions. However, it is very difficult to determine optimal feedback gains and a wrong gain choice would result in undesirable behaviors. We can design a state-feedback controller for the inverted pendulum model more intuitively.

Another advantage of using a simplified model for balancing is that we know where the COP would be. This feature would potentially allow us to plan the COP ahead of time to generate stepping motions by extending the methods proposed in [4], [6].

Motion capture data is prone to error due to the inevitable mismatch between the subject and humanoid model, and usually requires pre-processing to make sure that desired contact state is met throughout the motion. In contrast, our controller required little pre-processing probably because of the small feedback gains that allowed the joints to adjust to minor misalignment of the feet. In addition, the preprocessing is simple enough to be done in real time.

Acknowledgements

The authors would like to thank Justin Macey for capturing and cleaning motion capture data, Stuart Anderson for his help in building the simulation model, and Moshe Mahler for creating the images and movies.

References

[1] J. Yamaguchi, A. Takanishi, and I. Kato, “Development of a Biped Walking Robot Compensating for Three-axis Moment by Trunk Motion,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems*, 1993, pp. 561–566.

[2] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, “The Development of Honda Humanoid Robot,” in *Proceedings of International Conference on Robotics and Automation*, 1998, pp. 1321–1326.

[3] K. Nagasaka, I. Masayuki, and H. Inoue, “Walking Pattern Generation for a Humanoid Robot Based on Optimal Gradient Method,” in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Tokyo, Japan, October 1999, pp. 908–913.

[4] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, “Biped walking pattern generation by using preview control of zero-moment point,” in *Proceedings of IEEE International Conference on Robotics and Automation*, 2003, pp. 1620–1626.

[5] K. Löffler, M. Gienger, and F. Pfeiffer, “Sensor and control design of a dynamically stable biped robot,” in *Proceedings of IEEE International Conference on Robotics and Automtation*, 2003, pp. 484–490.

[6] T. Sugihara, “Simualted regulator to synthesize zmp manipulation and foot location for autonomous control of biped robots,” in *Proceedings of IEEE International Conference on Robotics and Automation*, 2008, pp. 1264–1269.

[7] S. Kudoh, T. Kumura, and K. Ikeuchi, “Stepping motion for a human-like character to maintain balance against large peturbations,” in *Proceedings of IEEE International Conference on Robotics and Automtation*, 2006, pp. 2661–2666.

[8] B. Stephens, “Integral control of humanoid balance,” in *Proceedings of IEEE/RSJ International Conference on intelligent Robots and Systems*, 2007, pp. 4020–4027.

[9] A. Ude, C. Man, M. Riley, and C. Atkeson, “Automatic generation of kinematic models for the conversion of human motion capture data into humanoid robot motion,” in *Proceedings of International Conference on Humanoid Robots*, 2000.

[10] A. Shon, K. Grochow, and R. Rao, “Robotic imitation from human motion capture using gaussian processes,” in *Proceedings of International Conference on Humanoid Robots*, 2005, pp. 129–134.

[11] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, “Embodied symbol emergence based on mimesis theory,” *International Journal of Robotics Research*, vol. 24, no. 4/5, pp. 363–378, 2004.

[12] O. Jenkins and M. Mataric, “Deriving action and behavior primitives from human motion data,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 2551–2556.

[13] A. Safonova, N. Pollard, and J. Hodgins, “Optimizing Human Motion for the Control of a Humanoid Robot,” in *2nd International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[14] A. DasGupta and Y. Nakamura, “Making Feasible Walking Motion of Humanoid Robots from Human Motion Captured Data,” in *Proceedings of International Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 1044–1049.

[15] K. Yamane and Y. Nakamura, “Dynamics Filter—Concept and Implementation of On-Line Motion Generator for Human Figures,” *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 421–432, June 2003.

[16] S. Tak, O. Song, and H. Ko, “Motion balance filtering,” *Eurographics 2000, Computer Graphics Forum*, vol. 19, no. 3, pp. 437–446, 2000.

[17] A. Safonova, J. Hodgins, and N. Pollard, “Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces,” *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 514–521, 2004.

[18] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa, and K. Ikeuchi, “Generating whole body motions for a biped robot from captured human dances,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[19] T. Sugihara, W. Takano, K. Yamane, K. Yamamoto, and Y. Nakamura, “Online dynamical retouch of motion patterns towards animatronic humanoid robots,” in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 2005, pp. 117–122.

[20] V. Zordan and J. Hodgins, “Motion Capture-Driven Simulations that Hit and React,” in *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, San Antonio, TX, July 2002, pp. 89–96.

[21] C. Ott, D. Lee, and Y. Nakamura, “Motion capture based human motion recognition and imitation by direct marker control,” in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 2008, pp. 399–405.

[22] K. Yamane and Y. Nakamura, “Natural Motion Animation through Constraining and Deconstraining at Will,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 352–360, July-September 2003.

[23] J. Luh, M. Walker, and R. Paul, “Resolved Acceleration Control of Mechanical Manipulators,” *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 468–474, 1980.

[24] K. Yamane and Y. Nakamura, “A Numerically Robust LCP Solver for Simulating Articulated Rigid Bodies in Contact,” in *Robotics: Science and Systems*, 2008.

[25] ——, “Dynamics simulation of humanoid robots: Forward dynamics, contact, and experiments,” in *The 17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, 2008.

[26] “CMU graphics lab motion capture database,” http://mocap.cs.cmu.edu/.