The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Control-Aware Mapping of Human Motion Data with Stepping for Humanoid Robots

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

Abstract— **This paper presents a method for mapping captured human motion with stepping to a humanoid model, considering the current state and the controller behavior. The mapping algorithm modifies the joint angle, trunk and center of mass (COM) trajectories so that the motion can be tracked and desired contact states can be achieved. The mapping is performed in two steps. The first step modifies the joint angle and trunk trajectories to adapt to the robot kinematics and actual contact foot positions. The second step uses a predicted center of pressure (COP) to determine if the balance controller can successfully maintain the robot's balance, and if not, modifies the COM trajectory. Unlike most humanoid control work that handles motion synthesis and control separately, our COM trajectory modification is performed based on the behavior of the robot controller. We verify the approach in simulation using a captured Tai-chi motion that involves unstructured contact state changes.**

Index Terms— **Humanoid Robots, Human Motion Data, Motion Synthesis, Control.**

I. Introduction

Using motion capture data is a potentially powerful way to realize a wide variety of expressive motions on humanoid robots. Even though motion capture data provide a fairly good reference, applying human motion data to robots is still not straightforward. For example, a human motion capture sequence is often physically infeasible for a robot due to the differences in the human and robot kinematics and dynamics. We therefore need some mapping algorithm that adapts human motion data to the different kinematics and dynamics of the robot.

Mapping human motion capture data to robots becomes difficult when the reference motion involves switching between different contact states because small control errors can result in a large discrepancy in the contact area and hence large differences in contact forces. It is therefore critical for the mapping algorithm to ensure that similar contact states can be achieved by the underlying controller with the current robot and contact states.

In this paper, we propose an online motion mapping technique that considers the current robot and contact states as well as the behavior of the robot controller. The method is online in the sense that it does not require the whole sequence of motion in advance and can map incoming motion data with a small constant delay (0.5 s in our implementation). More specifically, we extend our previous work [1] on simultaneous balancing and tracking by adding functionality to adjust the joint angle, trunk and center of mass (COM) trajectories so that the balance controller can successfully maintain balance while keeping the center of pressure (COP) inside the contact convex hull that varies during the motion due to stepping. We demonstrate that the proposed method can successfully realize motions with unstructured contact state changes through full-body dynamics simulation.

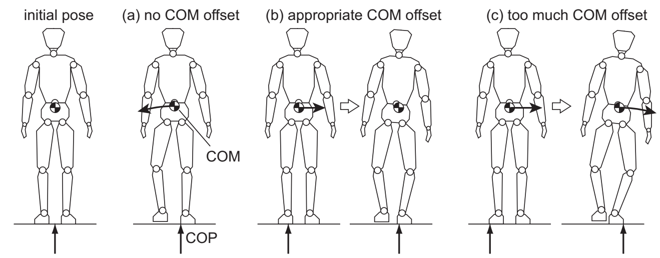

Fig. 1. An example of simple foot lifting: (a) no COM offset results in falling; (b) successful lifting requires appropriate amount of COM offset; (c) too much offset also results in falling after lifting the foot.

Our approach is different from most of the existing work in humanoid control where motion synthesis and robot control are handled separately. In the proposed controller framework, motion synthesis and control components are tightly connected by a mapping algorithm that is aware of the current state and the underlying controller.

II. Related Work

Using human motion capture data for controlling humanoid models have been actively investigated in the graphics and robotics fields.

In graphics, a number of algorithms have been proposed for synthesizing physically feasible motions of virtual characters either by dynamics simulation or optimization. If the character is sufficiently constrained or the control objective can be clearly defined, it is possible to synthesize plausible motions without human motion data. For example, Hodgins et al. [2] proposed a method to build task-specific controllers for simulated characters. Jain et al. [3] proposed an optimization scheme that directly outputs a joint angle sequence.

In order to synthesize more complex and stylized motions, however, using human motion capture data would be a more promising approach. Sok et al. [4] proposed a method for mapping human motion capture data for simulated virtual characters by calculating the optimal rectification of the original motion. They also developed a method for learning

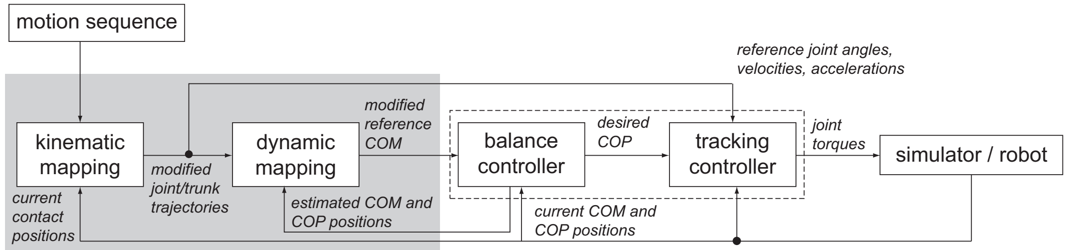

Fig. 2. Overview of the controller. The gray area is the contribution of this paper. The dashed box represents the balancing and tracking controller [1].

a controller for each observed behavior. Da Silva et al. [5] developed a method for designing a controller that realizes interactive locomotion of virtual characters. They also proposed to combine a model predictive controller with simple PD servos to obtain the optimal joint trajectories while responding to disturbances [6]. Muico et al. [7] proposed a controller that accounts for differences in the desired and actual contact states. They rely on other algorithms to map a motion capture sequence into a physically feasible motion for the character. Although these approaches can generate physically feasible motions similar to known motion capture sequences, they require either an extensive optimization process or a dedicated controller for each captured motion sequence.

The most relevant work in robotics is the work by Nakaoka et al. [8] where they proposed a method to convert captured dancing motions to physically feasible humanoid motions by dividing a motion into a sequence of motion primitives and adjusting their parameters considering the physical constraints of the robot. Their method is also offline because they have to first identify the required primitives and then define a set of parameters for each primitive. In addition, they only consider the motion synthesis problem and rely on high-gain servo controllers to track the synthesized joint trajectories.

Adjusting the COM trajectory based on prediction is related to the zero-moment point (ZMP) preview control [9], where a simplified robot model is used to obtain the COM trajectory that realizes a predefined ZMP path. This work is also focused on pattern generation and does not deal with control.

In contrast to existing work, our method does not require special knowledge about the task, or extensive optimization processes based on known motion capture sequences. The only information required in addition to the joint angle trajectories is whether each link is in contact at each time step, which can be easily obtained by monitoring the foot height, velocity or contact force data. Our method can therefore work online with a constant delay.

III. Overview

This paper is an extension of the authors' earlier work on simultaneous balancing and tracking [1] that only allowed stationary contacts. We employ the same controller comprising a balancing controller and a tracking controller. The main contribution of this paper is the preprocessing component that performs several types of adjustment to the reference motion to realize a similar motion on the robot.

Figure 1 illustrates a motivating example for this work. Suppose that the robot is trying to lift one of its feet starting from the initial pose shown in the left figure. Simply changing the leg joint angles will cause the robot to fall down as shown in (a) because the COM is still in the middle of the feet while the COP moves under the supporting leg. In order to successfully lift a foot, the robot first has to shift the COM towards the supporting leg by moving the COP as shown in (b). However, the robot cannot maintain balance if the COM offset is too large to be handled by the balance controller as shown in (c). This example implies two important observations: 1) a mapping algorithm needs to start preparing for contact state changes before they actually happen, and 2) a mapping algorithm needs to know the controller capability in order to determine the appropriate amount of adjustments.

Figure 2 shows the controller overview. The dashed box represents the controller framework presented in [1]. In the new controller, the motion clip is first processed by the two mapping components to ensure that the robot can maintain balance with the balance controller, using the current measurements of the COM and COP positions. One of the mapping components also uses the balance controller to predict the behavior of the controller.

The mapping consist of kinematic and dynamic components. The *kinematic mapping* component uses the current contact locations to modify the original joint angle and trunk trajectories so that the kinematic relationship between the contact area and humanoid body becomes similar to the reference motion. The *dynamic mapping* component modifies the reference COM trajectory given to the balance controller in case the future COP position predicted by the simplified robot model leaves the contact convex hull.

We make the following assumptions about the contacts between the motion capture subject and the environment:

- The subject only makes physical interaction with a horizontal floor through the feet.
- Both feet of subject are in flat contact with the floor at the initial frame. As long as at least one of the feet is in contact, any contact state is allowed at other frames.
- The set of links in contact is known for each frame. The contacts do not have to be flat except for the initial frame, nor do we use detailed contact state information such as toe or heel contact.

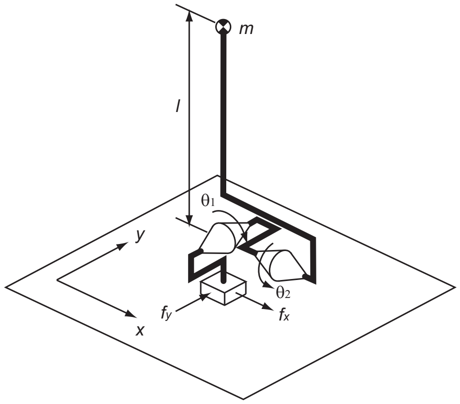

Fig. 3. Inverted pendulum model for the balance controller.

IV. Balance and Tracking Controllers

In this section, we briefly review the balance and tracking controllers used to obtain the joint torque command given to the robot. An earlier version of this controller has been presented in [1], but we have added several new elements to make it more robust and to handle contact state changes.

A. Balance Controller

The balance controller takes a reference COM position as input and calculates where the COP should be using an optimal controller designed for a simplified robot model. In this paper, we assume that a state-feedback controller is designed for the 3-dimensional 2-joint inverted pendulum with a mobile base shown in Fig. 3. In this model, the entire robot body is represented by a point mass located at the COM of the robot, and the position of the mobile base represents the COP. The state-feedback controller can be designed by, for example, linear quadratic regulator or pole placement.

Combining the state-feedback controller with an observer to estimate the current state, the whole balance controller is described by the following equation:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}_b \hat{\boldsymbol{x}} + \boldsymbol{B}_b \boldsymbol{u}_b \tag{1}$$

where $\hat{\boldsymbol{x}}$ is the estimated state of the inverted pendulum and $\boldsymbol{u}_b$ is the input to the balance controller, which comprises the reference COM position and measured COM and COP positions. The matrices $\boldsymbol{A}_b$ and $\boldsymbol{B}_b$ are computed from the state equation of the inverted pendulum model, the state feedback gain, and the observer gain. In the tracking controller, we use the mobile base position element of $\hat{\boldsymbol{x}}$ as the desired COP position $\boldsymbol{r}_b = (r_{bx}\ r_{by}\ 0)^T$.

The balance controller essentially handles the joint movements as one of disturbances and obtains a COP position that can recover and maintain balance under those disturbances, although it cannot guarantee global stability due to the difference between the linear inverted pendulum model and the whole-body robot model, as well as the limitation of the contact area. Note that, however, none of the existing balance controllers can ensure global stability of free-standing humanoid robots for the same reason.

B. Tracking Controller

The tracking controller first calculates the desired joint and trunk accelerations, $\hat{\ddot{q}}$, using the same feedback and feedforward scheme as in the resolved acceleration control [10]:

$$\hat{\ddot{q}} = \ddot{q}_{ref} + k_d(\dot{q}_{ref} - \dot{q}) + k_p(q_{ref} - q) \tag{2}$$

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. The desired acceleration is also calculated for the six degrees of freedom (DOF) of the trunk and foot links using the same control scheme. We denote the vector comprising the desired accelerations of all DOF by $\hat{\ddot{\boldsymbol{q}}}$.

We also compute the desired accelerations of the foot links, $\hat{\ddot{\boldsymbol{r}}}$, using the same feedforward and feedback control as Eq.(2). The foot link velocities $\dot{\boldsymbol{r}}$ are related to the generalized coordinates $\dot{\boldsymbol{q}}$ by

$$\dot{\boldsymbol{r}} = \boldsymbol{J}_c \dot{\boldsymbol{q}} \tag{3}$$

where $\boldsymbol{J}_c = \partial \boldsymbol{r} / \partial \boldsymbol{q}$ is the Jacobian matrix of foot link position and orientation with respect to the generalized coordinates. Taking the time derivative of both sides yields the relationship of the accelerations:

$$\ddot{\boldsymbol{r}} = \boldsymbol{J}_c \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_c \dot{\boldsymbol{q}}. \tag{4}$$

Using this relationship, we modify the desired generalized accelerations $\hat{\ddot{\boldsymbol{q}}}$ so that they match the desired foot link accelerations.

$$\hat{\ddot{\boldsymbol{q}}}' = \hat{\ddot{\boldsymbol{q}}} + \boldsymbol{J}_c^+ \left(\hat{\ddot{\boldsymbol{r}}} - \boldsymbol{J}_c \hat{\ddot{\boldsymbol{q}}} - \dot{\boldsymbol{J}}_c \dot{\boldsymbol{q}} \right) \tag{5}$$

where $\boldsymbol{J}_c^+$ is a generalized inverse of $\boldsymbol{J}_c$. In our implementation, we use the pseudo inverse assuming that the legs are not in a singular configuration.

The tracking controller then solves an optimization problem to obtain the joint torques that minimizes a cost function comprising the COP, joint acceleration and foot acceleration errors.

The unknowns of the optimization are the joint torques $\boldsymbol{\tau}_J$ and contact forces $\boldsymbol{f}_c$, subject to the whole-body equation of motion of the robot

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}\boldsymbol{\tau}_J + \boldsymbol{J}_c^T \boldsymbol{f}_c \tag{6}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{c}$ denotes the gravity, Coriolis and centrifugal forces and $\boldsymbol{S}$ is the matrix that maps the joint torques to the generalized forces.

The cost function to be minimized is

$$Z = Z_b + Z_q + Z_c + Z_\tau + Z_f + Z_p + Z_m \tag{7}$$

where the terms represent the error in COP (Z_b and Z_p), the the desired accelerations (Z_q and Z_c), the joint torques and contact forces (Z_τ and Z_f), and the moment around COM (Z_m). The last two terms have been added to the original version [1] to improve the robustness of the controller. We now describe those terms in more detail. In the following description, $\boldsymbol{W}_*$ denotes the positive-definite weight matrices for each term.

The term Z_b addresses the error between the desired and actual COP positions. Using the desired COP position $\boldsymbol{r}_b = (r_{bx}\ r_{by}\ 0)^T$ determined by the balance controller as described in the previous subsection, Z_b is computed by

$$Z_b = \frac{1}{2}\boldsymbol{f}_c^T \boldsymbol{P}^T \boldsymbol{W}_b \boldsymbol{P} \boldsymbol{f}_c \tag{8}$$

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 $\boldsymbol{T}$ that converts the individual contact forces to total contact force and moment around the world origin by

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_1 & \boldsymbol{T}_2 & \dots & \boldsymbol{T}_{N_C} \end{pmatrix} \tag{9}$$

and

$$\boldsymbol{T}_i = \begin{pmatrix} \boldsymbol{1}_{3\times 3} & \boldsymbol{0}_{3\times 3} \\ [\boldsymbol{r}_i \times] & \boldsymbol{1}_{3\times 3} \end{pmatrix} \tag{10}$$

where $\boldsymbol{r}_i$ 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:

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & r_{by} & 1 & 0 & 0 \\ 0 & 0 & -r_{bx} & 0 & 1 & 0 \end{pmatrix} \tag{11}$$

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

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

$$Z_q = \frac{1}{2}(\hat{\ddot{\boldsymbol{q}}} - \ddot{\boldsymbol{q}})^T \boldsymbol{W}_q (\hat{\ddot{\boldsymbol{q}}} - \ddot{\boldsymbol{q}}). \tag{12}$$

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

$$Z_c = \frac{1}{2}(\hat{\ddot{\boldsymbol{r}}}_c - \ddot{\boldsymbol{r}}_c)^T \boldsymbol{W}_c (\hat{\ddot{\boldsymbol{r}}}_c - \ddot{\boldsymbol{r}}_c). \tag{13}$$

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

$$Z_\tau = \frac{1}{2}(\hat{\boldsymbol{\tau}}_J - \boldsymbol{\tau}_J)^T \boldsymbol{W}_\tau (\hat{\boldsymbol{\tau}}_J - \boldsymbol{\tau}_J) \tag{14}$$

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.,

$$Z_f = \frac{1}{2}(\hat{\boldsymbol{f}}_c - \boldsymbol{f}_c)^T \boldsymbol{W}_f (\hat{\boldsymbol{f}}_c - \boldsymbol{f}_c) \tag{15}$$

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

Z_p is a new term representing the difference from the desired and actual COP of individual feet. This term has been introduced in addition to the total COP term Z_b to prevent foot rotation. The desired COP of each foot is set to the center of its contact area. Z_p is computed using a formulation similar to Z_b:

$$Z_p = \frac{1}{2}\boldsymbol{f}_c^T \boldsymbol{P}_p^T \boldsymbol{W}_b \boldsymbol{P}_p \boldsymbol{f}_c \tag{16}$$

where $\boldsymbol{P}_p = \boldsymbol{C}_p \boldsymbol{T}_p$,

$$\begin{aligned} \boldsymbol{C}_p &= diag\{\boldsymbol{C}_i\} \\ \boldsymbol{T}_p &= diag\{\boldsymbol{T}_i\} \\ \boldsymbol{C}_i &= \begin{pmatrix} 0 & 0 & r_{piy} & 1 & 0 & 0 \\ 0 & 0 & -r_{pix} & 0 & 1 & 0 \end{pmatrix} \end{aligned}$$

and $\boldsymbol{r}_{pi} = (r_{pix}\ r_{piy}\ 0)$ is the desired COP of foot i.

Another new term Z_m represents the moment around the COM and has been included to keep the angular momentum constant. Z_m is computed by

$$Z_m = \frac{1}{2}\boldsymbol{f}_c^T \boldsymbol{V}^T \boldsymbol{W}_m \boldsymbol{V} \boldsymbol{f}_c \tag{17}$$

where $\boldsymbol{V}$ is written as

$$\begin{aligned} \boldsymbol{V} &= \begin{pmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 & \dots & \boldsymbol{V}_{N_C} \end{pmatrix} \\ \boldsymbol{V}_i &= \begin{pmatrix} [\boldsymbol{s}_i \times] & \boldsymbol{1}_{3\times 3} \end{pmatrix} \\ \boldsymbol{s}_i &= \boldsymbol{r}_i - \boldsymbol{r}_m \end{aligned}$$

using the COM position $\boldsymbol{r}_m$.

Using Eqs. (4) and (6), the cost function can be converted to the following quadratic form:

$$Z = \frac{1}{2}\boldsymbol{y}^T \boldsymbol{A} \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{b} + c \tag{18}$$

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

The optimization problem has an analytical solution

$$\boldsymbol{y} = -\boldsymbol{A}^{-1}\boldsymbol{b}. \tag{19}$$

Our previous paper [1] provides more details on how to choose the weights for the cost function and deal with joint angle and torque limits.

V. Kinematic Mapping

A. Joint Angle Mapping

Due to the difference in the kinematics of the human subject and the robot model, the joint angles obtained by an inverse kinematics calculation using the robot model may not result in the same contact state as in the original motion. The first mapping is applied to the joint angles to fix this problem, relying on the assumption that the feet are in flat contact at the initial frame.

We first obtain the contact link positions and orientations using the joint angles at the initial frame and project them onto the floor. We then obtain the joint angles that achieve the new contact link positions and orientations by running a numerical inverse kinematics algorithm using the original joint angles as the initial guess. Finally, we store the differences between the original and new joint angles. The joint angle mapping is performed by simply adding those differences to the joint angles at each frame of the original motion capture data.

Note that this simple mapping algorithm does not always realize exactly the same contact state in the robot model. We could perform an additional inverse kinematics calculation at each frame to realize the same contact state. When we are using human motion capture data, however, it is often difficult to determine the desired foot position and orientation because the precise human model is unknown. Also, the inverse kinematics can be computationally expensive for online processes. We therefore limit our joint angle mapping to the simple method described above and let the controller and the robot dynamics determine the best contact state to track the reference motion.

B. Trunk Trajectory Mapping

Another type of adjustment is to globally transform the reference trunk position and orientation in the motion capture data to account for the difference in the reference and actual contact positions. The feet may not land at the reference position due to control errors. In such cases, using the same reference position for the trunk may result in falling due to the difference in its position relative to the contact area.

To solve this problem, we apply translation, rotation and scaling to the trunk trajectory every time a new contact is established. Figure 4 summarizes the parameters used to determine each transformation. We first calculate the center of reference and actual contact convex hulls by simply taking the average of all contact points:

$$\hat{\boldsymbol{c}} = \frac{1}{M}\sum \hat{\boldsymbol{p}}_i \tag{20}$$

$$\boldsymbol{c} = \frac{1}{M}\sum \boldsymbol{p}_i \tag{21}$$

where $\hat{\boldsymbol{c}}$ and $\boldsymbol{c}$ are the center of contact convex hull in reference and actual motions respectively, M is the number of contact points, and $\hat{\boldsymbol{p}}_i$ and $\boldsymbol{p}_i$ are the positions of the i-th contact point in reference and actual motions.

We then perform a principal component analysis of the contact point positions with respect to their center. Let $\hat{\theta}_1$ and θ_1 denote the angle of the first principal component axis with respect to the x axis of the inertial frame. Also let $\hat{s}_i$ and s_i $(i = 1, 2)$ denote the singular values of the reference and actual contact points respectively.

For a pair of reference trunk position $\hat{\boldsymbol{p}}$ and orientation $\hat{\boldsymbol{R}}$, the transformed position $\boldsymbol{p}$ and orientation $\boldsymbol{R}$ are obtained as follows:

$$\boldsymbol{p} = \boldsymbol{\Omega}\boldsymbol{S}(\hat{\boldsymbol{p}} - \hat{\boldsymbol{c}}) + \boldsymbol{c} \tag{22}$$

$$\boldsymbol{R} = \boldsymbol{\Omega}\hat{\boldsymbol{R}} \tag{23}$$

where

$$\boldsymbol{S} = \begin{pmatrix} s_1/\hat{s}_1 & 0 & 0 \\ 0 & s_2/\hat{s}_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{24}$$

and $\boldsymbol{\Omega}$ is the rotation matrix representing the rotation of $\theta_1 - \hat{\theta}_1$ around the vertical axis.

Note that we update the transformation only when a contact is established. It would provide positive feedback if we update the transformation when it is still possible to move the foot in contact, because the tracking controller would move the contact position in the same direction as the transformation update. We therefore update the transformation only when the vertical contact force at every link in contact exceeds a large threshold (100 N in our implementation) after the contact state has changed in the reference motion.

VI. Dynamic Mapping

This section describes a mapping algorithm that modifies the COM trajectory of the reference motion so that the balance controller can keep the robot balanced with the provided contact area. The mapping algorithm predicts the future COP positions based on the current robot state and original reference COM trajectory. If the COP is leaving the contact convex hull, it calculates a new COM trajectory so that the COP stays within the contact convex hull and sends the next COM position to the balance controller as the reference COM. If a foot is making a new contact, on the other hand, it is preferable to bring the COP close to the edge of the current contact area so that the weight of the robot helps the foot come down.

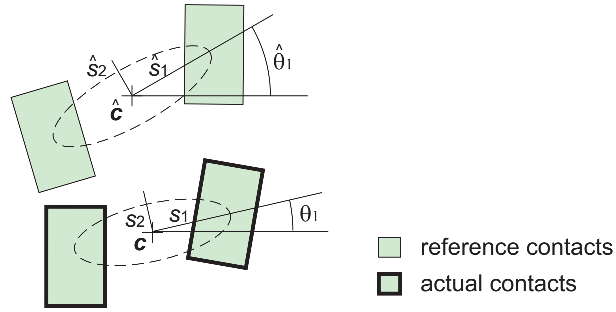

Fig. 4. Parameters associated with the translation, rotation and scaling of trunk position and orientation.

We first describe how to modify the COM trajectory so that the COP moves to a given desire position n frames in the future. We use a discretized state equation model of the balance controller Eq.(1):

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{25}$$

where $\boldsymbol{x}_k$ is the state and $\boldsymbol{u}_k$ is the input at sampling time k. In contrast to the balance controller (1), the controller model (25) does not include an observer because we are not using real measurements here. Therefore, $\boldsymbol{u}_k$ only includes the COM trajectory in the reference motion. We choose the COP position as the output and define the output equation accordingly as

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k. \tag{26}$$

For a given initial state $\boldsymbol{x}_0$ and the COM trajectory for the next n frames, $\hat{\boldsymbol{u}}_k$ $(k = 0, 1, \ldots, n-1)$, we can predict the location of the COP n frames later by

$$\hat{\boldsymbol{y}}_n = \boldsymbol{C}\left(\boldsymbol{A}^n\boldsymbol{x}_0 + \boldsymbol{M}_n\hat{\boldsymbol{U}}_n\right) \tag{27}$$

where

$$\boldsymbol{M}_n = \begin{pmatrix} \boldsymbol{A}^{n-1}\boldsymbol{B} & \boldsymbol{A}^{n-2}\boldsymbol{B} & \ldots & \boldsymbol{B} \end{pmatrix} \tag{28}$$

$$\hat{\boldsymbol{U}}_n = \begin{pmatrix} \hat{\boldsymbol{u}}_0 \\ \hat{\boldsymbol{u}}_1 \\ \vdots \\ \hat{\boldsymbol{u}}_{n-1} \end{pmatrix}. \tag{29}$$

Suppose that a desired COP position $\boldsymbol{y}_{ref}$ at frame n is given. We find a new reference COM trajectory $\boldsymbol{U}'$ such that $\boldsymbol{y}_n$ becomes as close as possible to $\boldsymbol{y}_{ref}$, by solving the following optimization problem:

$$\boldsymbol{U}' = \underset{\boldsymbol{U}}{\operatorname{argmin}}\, Z_d \tag{30}$$

where the cost function Z_d is

$$Z_d = \frac{1}{2}\left(\boldsymbol{y}_n - \boldsymbol{y}_{ref}\right)^T\left(\boldsymbol{y}_n - \boldsymbol{y}_{ref}\right) + \frac{1}{2}\left(\boldsymbol{U} - \hat{\boldsymbol{U}}_n\right)^T \boldsymbol{W}\left(\boldsymbol{U} - \hat{\boldsymbol{U}}_n\right), \quad (31)$$

$\boldsymbol{y}_{ref}$ is the desired COP inside the contact convex hull, $\boldsymbol{W}$ is a constant positive-definite weight matrix, and

$$\boldsymbol{y}_n = \boldsymbol{C}\left(\boldsymbol{A}^n \boldsymbol{x}_0 + \boldsymbol{M}_n \boldsymbol{U}\right). \quad (32)$$

The first term of Eq.(31) tries to bring the COP as close as possible to a chosen point $\boldsymbol{y}_{ref}$ while the second term penalizes the deviation from the original COM trajectory. This optimization problem also has an analytical solution.

At every control cycle, we predict the COP locations up to N frames in the future using Eq.(25) and the COM trajectory computed from the reference motion after the kinematic mapping. We activate the COM trajectory modification process in the following two situations:

1) The predicted COP leaves the contact convex hull.
2) A link is making a new contact but the COP is still inside the previous contact convex hull.

We describe how to determine $\boldsymbol{y}_{ref}$ and $\boldsymbol{W}$ in the following paragraphs.

In case 1), we set $\boldsymbol{y}_{ref}$ inside the contact convex hull considering the following two issues. First, it would be better to keep a safety margin from the boundary because the resulting COP may not be exactly at $\boldsymbol{y}_{ref}$. We also want to minimize the COM trajectory change and maintain the original reference motion as much as possible. To consider both issues, we first obtain the closest point on the contact convex hull boundary from the predicted COP, $\boldsymbol{b}_1$, and the center of the contact convex hull, $\boldsymbol{c}_1$. We then obtain $\boldsymbol{y}_{ref}$ by

$$\boldsymbol{y}_{ref} = h\boldsymbol{b}_1 + (1-h)\boldsymbol{c}_1 \quad (33)$$

where $h = h_{max}(N-n)/N$ and h_{max} is a user-specified constant. By this interpolation between $\boldsymbol{b}_1$ and $\boldsymbol{c}_1$, $\boldsymbol{y}_{ref}$ becomes closer to $\boldsymbol{c}$ as n becomes larger, which is based on the observation that it is more difficult to change the COM in a shorter period.

In case 2), the robot can still maintain balance but may not be able to initiate a new contact. We therefore try to bring the COP close to the edge of the current contact area so that the weight of the robot can help achieving the new contact. In this case, we take $\boldsymbol{c}_2$ at the center of the contact area that will be added by the new contact, and $\boldsymbol{b}_2$ becomes the intersection of the line connecting the predicted COP and $\boldsymbol{c}_2$ and the border of the current contact convex hull. We then obtain $\boldsymbol{y}_{ref}$ by

$$\boldsymbol{y}_{ref} = h\boldsymbol{b}_2 + (1-h)\boldsymbol{c}_2. \quad (34)$$

$\boldsymbol{W}$ represents how much we allow the COM trajectory to deviate from the original. Because we hope to return to the original COM position at the last frame, it seems reasonable to give smaller values to the earlier frames and larger values to the later frames. In our implementation, we calculate the i-th diagonal component of $\boldsymbol{W}$ by wi^2 where w is a user-specified constant.

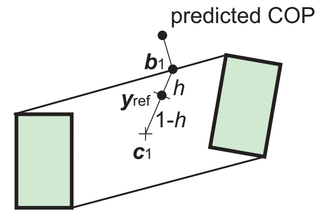

Fig. 5. Obtaining modified COP position when the predicted COP leaves the contact convex hull.

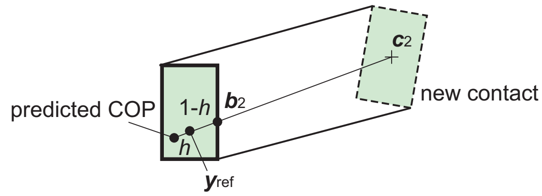

Fig. 6. Obtaining modified COP position when the predicted COP is inside the previous contact convex hull.

VII. Simulation Results

We verify the effectiveness of the proposed controller with a simulator based on a rigid-body contact model and a forward dynamics algorithm described in [11]. Note that the contact forces in the simulation are computed independently of the controller and may be different from the ones expected by solving the optimization problem described in Section IV-B, allowing us to emulate one of the disturbances in hardware experiments. We use a humanoid robot model with 25 joints (31 DOF including the translation and rotation of the root joint).

We use linear quadratic regulator for the balance controller with the same parameters as in [1]. The feedback gains for the tracking controller are $k_p = 36$ and $k_d = 12$. The parameters for the adjustment components are $N = 100$, $h_{max} = 0.8$ and $w = 0.01$. The control system (25) is discretized at the sampling time of 5 ms, meaning that the dynamic mapping component looks 0.5 s future in order to determine if COM modification is required. Note that the sampling time for discretization does not necessarily have to be the same as the control cycle. In fact, using larger sampling time for Eq.(25) than the control cycle can reduce the computation time for dynamic mapping.

A. Dynamic Mapping Example

We first show a simple example where the dynamic mapping is required to realize a stepping motion. The reference is a manually-generated simple motion where the right foot is lifted during the period between $t = 1.0$ and 2.0 s while the left foot is in flat contact with the floor all the time. Figure 7 shows the COM and COP trajectories along the sideways of the robot, the positive axis pointing to its left. The green area in the bottom figure depicts the contact area of the left foot.

The bottom graph shows that the COP never reaches the left foot without dynamic mapping, and therefore the robot cannot lift its right foot. If the mapping is turned on, on the

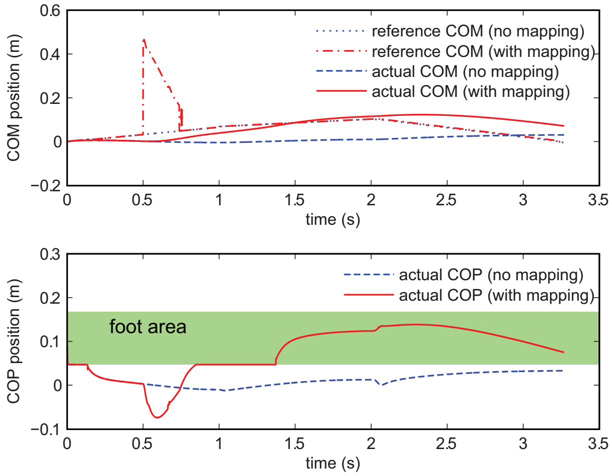

Fig. 7. COM and COP trajectories for the simple right foot lifting example (best viewed in color).

other hand, the reference COM is modified at $t = 0.5$ s when the lifting time comes into the COP prediction window and the dynamic mapping component realizes that the COP will not be under the left foot in time. The COP shifts towards the right foot first in response to the new reference COM, which pushes the COM farther towards the left foot. The COP eventually reaches the left foot at around $t = 0.8$ s so the robot is able to lift the right foot.

B. Application to a Complex Motion

We use a motion capture sequence of Tai-chi from the Carnegie Mellon University Motion Capture Data Library [12]. The marker position data are converted to joint angle data by an inverse kinematics algorithm with joint motion range constraints [13]. Tai-chi motion involves many unstructured contact state changes such as transition to toe or heel contact and slight repositioning of a foot while in contact.

Figure 8 shows how the dynamic mapping works in complex motions. The images are taken at approximately 0.2 s interval. The red arrow indicates the total contact force, the small ball around the trunk is the reference COM, the large ball around the trunk is the actual COM, and the small ball on the floor is the optimized COP. The COM trajectory modification is activated in (b), which shows a jump in the reference COM compared to (a). The optimized COP initially moves towards the left (from the reader's view) to bring the COM towards the reference, but eventually enters the left foot sole in (d). Finally, the robot is able to rotate the right foot as shown in (f).

Figure 9 shows snapshots taken every 4 s from the simulated motion. The faded model represents the corresponding reference pose after applying the kinematic mapping. The supplemental movie includes the clips corresponding to Figures 8 and 9.

VIII. CONCLUSION

In this paper, we presented a method for mapping human motion capture data to humanoid robots. The most notable feature of the method is that it not only deals with the different kinematics and dynamics of the robot, but also determines the amount of adjustments based on the current robot and contact states as well as the capability of the balance controller. We demonstrated in simulation that the proposed method can successfully make a humanoid robot imitate a human Tai-chi motion.

We are currently conducting hardware experiments of the balance and tracking controllers. Although most of the components can run in real time at 2 ms control cycle, it takes much longer when the COM trajectory modification takes place. We can potentially run the whole controller in real time by running this process in parallel.

There are also some possible theoretical extensions. Although the current method allows non-flat contacts in the reference motion, it does not guarantee that a particular contact state is realized on the robot. We could command those contact states by choosing the desired COP along the edge of a foot, but it would require very fine COP manipulation. The current method can only handle motions with relatively slow stepping due to the long lookahead time. Improving the dynamic mapping algorithm to realize faster contact state switching is another interesting extension. Furthermore, allowing free-flight phase would extend the method to more agile motions such as running and jumping.

ACKNOWLEDGEMENTS

The authors would like to thank Justin Macey for capturing and cleaning motion capture data.

REFERENCES

[1] K. Yamane and J. Hodgins, "Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robot Systems*, 2009, pp. 2510–2517.

[2] J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien, "Animating Human Athletics," in *Proceedings of ACM SIGGRAPH '95*, Los Angeles, CA, 1995, pp. 71–78.

[3] S. Jain, Y. Ye, and C. Liu, "Optimization-based interactive motion synthesis," *ACM Transactions on Graphics*, vol. 28, no. 1, p. 10, 2009.

[4] K. Sok, M. Kim, and J. Lee, "Simulating biped behaviors from human motion data," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[5] M. Da Silva, Y. Abe, and J. Popović, "Interactive simulation of stylized human locomotion," *ACM Transactions on Graphics*, vol. 27, no. 3, p. 82, 2008.

[6] ——, "Simulation of human motion data using short-horizon model-predictive control," in *Eurographics*, 2008.

[7] U. Muico, Y. Lee, J. Popović, and Z. Popović, "Contact-aware nonlinear control of dynamic characters," *ACM Transactions on Graphics*, vol. 28, no. 3, 2009.

[8] 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.

[9] 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.

[10] 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.

[11] K. Yamane and Y. Nakamura, "Dynamics simulation of humanoid robots: Forward dynamics, contact, and experiments," in *The 17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, 2008.

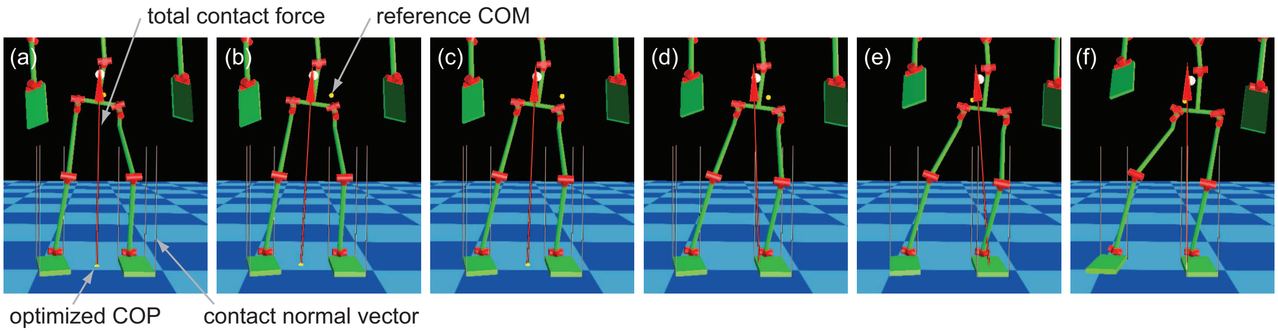

Fig. 8. Example of dynamic mapping. Although the soles and floor are modeled as completely flat surfaces, there may be only three contact points between a foot and the floor because of small penetrations.

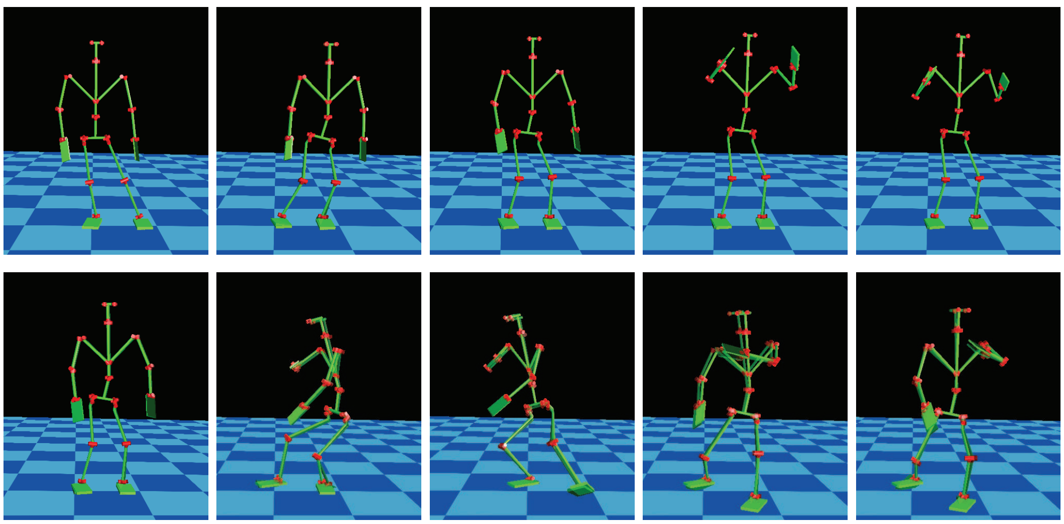

Fig. 9. Simulation result of a Tai-chi motion. The faded model shows the reference pose after applying the kinematic mapping.

[12] "Carnegie Mellon University Graphics Lab Motion Capture Database," http://mocap.cs.cmu.edu/.

[13] 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.