Control of Instantaneously Coupled Systems Applied to Humanoid Walking

Eric C. Whitman and Christopher G. Atkeson

Abstract— **This paper presents an optimal controller for an Instantaneously Coupled System (ICS) which was designed by coordinating multiple lower-dimensional optimal controllers. We augmented subsystems of the ICS with coordination variables, and then used value functions to coordinate the augmented subsystems by managing tradeoffs of the coordination variables. We apply this method to humanoid walking and present a controller for a 3D simulation that uses multiple coordinated policies generated using Dynamic Programming. Additionally, we present simulated walking perturbation experiments as well as standing balance results from a force-controlled humanoid robot.**

I. INTRODUCTION

A humanoid robot should be able to operate in the presence of large disturbances. This paper proposes a method of control for bipedal walking that is capable of responding immediately to unexpected disturbances by modifying center of mass (CoM) motion, footstep location, and footstep timing. We use dynamic programming (DP) to design a nonlinear controller for a simple model of a biped. DP suffers from the "Curse of Dimensionality", with storage and computation costs proportional to R^d, where R is the grid resolution and d is the dimension of the state. However, breaking the control design problem into parts greatly reduces the storage and computation costs. For example:

$$R^{d/2} + R^{d/2} << R^d. \tag{1}$$

By breaking the model into multiple subsystems of lower dimensionality, we are able to work with a higher-dimensional model than would otherwise be computationally feasible. To capture the coupling between the subsystems while keeping them low-dimensional, we augment the subsystems with additional coordination variables. We use dynamic programming to produce optimal policies and value functions for each of the augmented subsystems. Then, by using the value functions to manage tradeoffs between the coordination variables, we coordinate the subsystem controllers such that the combined controller is optimal. Finally, we use the output of this high-level controller (CoM and swing foot accelerations) as the input to a low-level controller, which provides the joint torques necessary to produce those accelerations.

E. C. Whitman and C.G. Atkeson are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, USA. ewhitman@cmu.edu, cga@cmu.edu

An accompanying video can be found at http://www.contrib.andrew.cmu.edu/~ewhitman/humanoids2010.avi

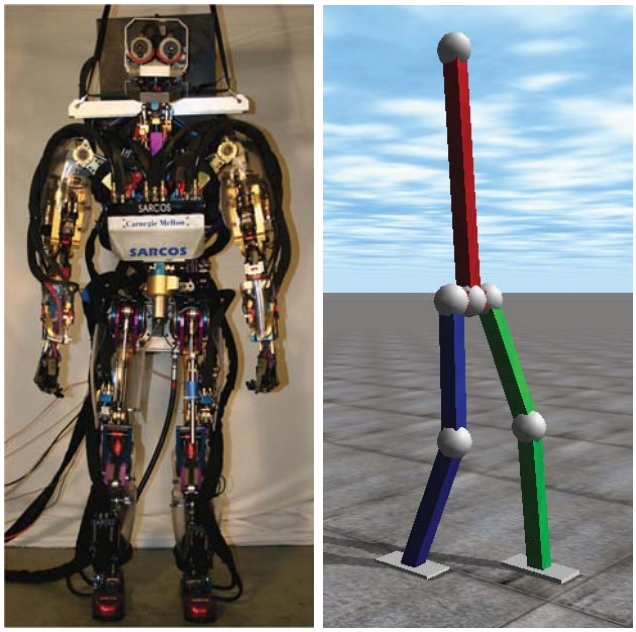

Fig. 1. The Sarcos Primus hydraulic humanoid robot (left) and the simulation based on it (right).

A. Related Work

The central problem faced by walking controllers is managing reaction forces, which are constrained by friction and the requirement that the center of pressure (CoP) be within the convex hull of the region of support. Many walking controllers focus on CoM motion. A standard method of control is to first generate a CoM trajectory and then track that trajectory with inverse kinematics [1]. Preview control of the CoP can be used to generate CoM trajectories [2]. By modifying the inverse kinematics for force control, it is possible to deal with small disturbances [3].

Unfortunately, even when tracking an optimal trajectory, the resulting controller is only optimal when near the desired trajectory, which is not the case following a significant unexpected disturbance. Due to constraints on reaction forces, linear independent joint controllers often can stabilize only a small region of state space. It is possible to frequently recalculate the CoM trajectory, taking into account the current robot state [4]. Model Predictive Control (MPC) and receding horizon control offer methods of generating trajectories online that continuously start from the current robot state [5].

For the system to recover from large disturbances, it is necessary to modify the reaction force constraints by adjust-

ing the footstep placement or timing. One possible approach to this is trajectory libraries, where multiple trajectories are generated in advance and an appropriate one is used depending on the current robot state. Examples of trajectory libraries are given for standing balance in [6] and for walking in [7]. It is also possible to modify MPC so that it determines foot placement online [8]. In [9], the footstep timing is modified online in response to manually changed footstep locations.

Because of many walkings systems' high-dimensionality, which makes control difficult, it is common to model parts of a walking system as decoupled so that the lower-dimensional subsystems can be controlled separately [10], [11]. PD servos on individual joints is a very basic form of such decoupling. Unless coordination is handled carefully, the combined controller will be sub-optimal because the subsystem controllers lack the information necessary to make optimal decisions. We present a method of coordination that produces an optimal combined controller.

In Section II, we propose the concept of Instantaneously Coupled Systems (ICS) and demonstrate that our method for coordinating multiple optimal subsystem controllers is equivalent to an optimal controller for the full system. We then model walking as an ICS in Section III and describe our walking controller in Section IV. We present simulated walking results in Section V and preliminary hardware results consisting of standing balance perturbation experiments on our Sarcos humanoid robot (pictured in Fig. 1) in Section VI. Sections VII and VIII offer a discussion and conclusion.

B. Dynamic Programming

The high-level controllers in this paper are generated using Dynamic Programming (DP) because in addition to providing policies that are valid for a large region of state space, DP also provides value functions, which we need for coordination. Value functions represent the future cost of starting in a particular state and following a given policy. Another major advantage of DP is that it can easily handle discrete decisions such as whether or not to touch down, allowing simultaneous optimization of trajectory, footstep timing, and footstep placement. Additionally, DP is useful for optimizing transient responses to perturbations as well as optimizing steady state gait. It is also globally optimal (up to the grid resolution), avoiding potential problems with local minima. In [12], DP on the Poincaré state was used to determine stride-level variables. DP was used for continuous control of some joints in [10] and of all joints in [13].

In the version of DP used in this paper, we divide the state space into a grid. We then iteratively solve for a steady state policy, $\mathbf{u}(\mathbf{x})$, and value function, $V(\mathbf{x})$, at each point in the grid in discrete time using

$$\mathbf{u}(\mathbf{x}) = \arg\min_{\mathbf{u}}(L(\mathbf{x};\mathbf{u}) + cV(f(\mathbf{x};\mathbf{u}))) \tag{2}$$

$$V(\mathbf{x}) = L(\mathbf{x},\mathbf{u}(\mathbf{x})) + cV(f(\mathbf{x},\mathbf{u}(\mathbf{x}))), \tag{3}$$

where $\mathbf{x}$ is the state, c is the discount factor, $L(\mathbf{x},\mathbf{u})$ is the one step cost function, and $\mathbf{x}_{N+1} = f(\mathbf{x}_N,\mathbf{u})$ is the dynamics. The discount factor is a constant slightly less than 1.0 necessary to make the value function converge for periodic systems that do not have a zero-cost limit cycle. In each iteration, for each grid point, we use (2) to pick a new action, $\mathbf{u}(\mathbf{x})$, from between only two choices: the current best action and a random action [14]. We then use (3) to update the value function accordingly. We iterate this procedure until the value and policy converge to a global optimum.

II. CONTROLLING INSTANTANEOUSLY COUPLED SYSTEMS

For a certain type of system, which we call Instantaneously Coupled Systems (ICS), it is possible to construct an optimal controller by coordinating multiple optimal lower-dimensional controllers. First, subsystems are augmented with coordination variables, which provide enough information to account for coupling to other systems. Then, value functions are used to trade off the coordination variables. This is useful because it reduces an optimal control problem to several lower-dimensional optimal control problems, which can be solved more easily.

A. Instantaneously Coupled Systems

We define an instantaneously coupled system (ICS) as a dynamic system made up of a set of N lower-dimensional systems. The state of, $\mathbf{x}_f$, and input to, $\mathbf{u}_f$, the full system are given by the composition of the states of and inputs to the lower-dimensional systems,

$$\mathbf{x}_f = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \tag{4}$$

and

$$\mathbf{u}_f = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}. \tag{5}$$

The dynamics of each system evolve independently,

$$\dot{\mathbf{x}}_i = f_i(\mathbf{x}_i, \mathbf{u}_i). \tag{6}$$

At M specific instants, however, the systems may be coupled such that the dynamics of the subsystems instantaneously depend on the full state,

$$\mathbf{x}_i^+ = f_i^c(\mathbf{x}_f^-, \mathbf{u}_i), \tag{7}$$

where the superscripts $-$ and $+$ indicate before and after the coupling event.

The time of the coupling, t_j, is determined by some condition on the full state:

$$\Phi(\mathbf{x}_f(t_j)) = 0 \tag{8}$$

There can be one or multiple coupled instants. We only consider systems with a finite number, M, of coupled instants.

B. Obtaining the Optimal Policy

For an ICS with a cost function of the form

$$C = \int \sum_{i=1}^{N} L_i(\mathbf{x}_i(t), \mathbf{u}_i(t))\, dt + \sum_{j=1}^{M} \left(g(t_j) + h(\mathbf{x}_f(t_j))\right), \tag{9}$$

we can construct the optimal policy by finding the optimal policies and value functions for augmented versions of the subsystems and then combining them. Costs of coupling event times and state (g and h) are optional and are not used by the controller presented in this paper.

First, we define a coordination state, $\mathbf{x}_c$, as some set of features of the full state, $\mathbf{x}_c = \Theta(\mathbf{x}_f)$. The features, $\mathbf{x}_c$, are a compact means of communicating the essential information about the full state between the subsystems, and must be selected such that it is possible to:

I. Rewrite the coupling dynamics (7) as

$$f_i^c(\mathbf{x}_f, \mathbf{u}_i) = \tilde{f}_i^c(\mathbf{x}_i, \mathbf{x}_c, \mathbf{u}_i). \tag{10}$$

II. Rewrite the last term in (9) as

$$h(\mathbf{x}_f(t_j)) = \tilde{h}(\mathbf{x}_c(t_j)). \tag{11}$$

III. Rewrite (8) as the intersection of conditions on the low-dimensional systems

$$\Phi(\mathbf{x}_f(t)) = \Phi_1(\mathbf{x}_1(t), \mathbf{x}_c(t)) \cap \ldots \cap \Phi_N(\mathbf{x}_N(t), \mathbf{x}_c(t)). \tag{12}$$

It is always possible to choose $\mathbf{x}_c = \mathbf{x}_f$, but this method will be more useful if an $\mathbf{x}_c$ that is lower-dimensional than $\mathbf{x}_f$ can be found.

Next, we construct the decision space, $\mathbf{x}_d$, by composing t_j and $\mathbf{x}_c(t_j)$ from each of the coupled instants.

$$\mathbf{x}_d = \{t_1, \mathbf{x}_c(t_1), t_2, \mathbf{x}_c(t_2), \ldots, t_M, \mathbf{x}_c(t_M)\} \tag{13}$$

If we hold $\mathbf{x}_d$ constant, the subsystems are completely decoupled and the conditions from (12) are constraints:

$$\Phi_i(\mathbf{x}_i(t_j), \mathbf{x}_c(t_j)) = 0. \tag{14}$$

With the systems decoupled, we can individually optimize each one with respect to

$$C_i = \int L_i(\mathbf{x}_i(t), \mathbf{u}_i(t))\, dt, \tag{15}$$

the only part of (9) that depends on the ith system. It then remains only to optimize over all possible choices of $\mathbf{x}_d$ and select the best one.

To accomplish this, we augment the state of each of the subsystems with $\hat{\mathbf{x}}_d$,

$$\hat{\mathbf{x}}_d = \{\hat{t}_1, \mathbf{x}_c(t_1), \hat{t}_2, \mathbf{x}_c(t_2), \ldots, \hat{t}_M, \mathbf{x}_c(t_M)\} \tag{16}$$

$$\hat{t}_j = t_j - t \tag{17}$$

which has the trivial dynamics $\dot{\hat{t}}_j = -1$ and $\dot{\mathbf{x}}_c = 0$. This allows us to generate subsystem controllers that can apply the coupling dynamics (10) and know when to do so. We switch from the time of coupling in $\mathbf{x}_d$ to the time until coupling in $\hat{\mathbf{x}}_d$ to eliminate the dependence on time in our subsystem controllers. We then produce optimal (with respect to (15)) policies and value functions for each of the augmented systems subject to (14). Any method that produces both policies and value functions can be used, but in this paper, we use dynamic programming.

Now, if we have an $\mathbf{x}_f$, we can hold each of the $\mathbf{x}_i$'s constant and get the value as only a function of $\mathbf{x}_d$. This allows us to rewrite (9) as only a function of $\mathbf{x}_d$, t, and $\mathbf{x}_f$:

$$C = \sum_{i=1}^{N} V_i(\mathbf{x}_d, t | \mathbf{x}_i) + k(\mathbf{x}_d) \tag{18}$$

where $k(\mathbf{x}_d) = \sum_{j=1}^{M} g(t_j) + \tilde{h}(\mathbf{x}_c(t_j))$. We then select the best decision state,

$$\mathbf{x}_d^* = \arg\min_{\mathbf{x}_d} C(\mathbf{x}_d, t, \mathbf{x}_f). \tag{19}$$

Having selected $\mathbf{x}_d$, we can look up each of the $\mathbf{u}_i$'s from the individual optimal policies and compose them to form $\mathbf{u}_f$ according to (5).

III. WALKING AS AN ICS

To generate a walking controller, we first approximate walking as an ICS. Summing the forces and torques on the system gives us dynamics equations for the CoM

$$\mathbf{f}_L + \mathbf{f}_R + \mathbf{f}_g = m\ddot{\mathbf{c}} \tag{20}$$

$$(\mathbf{p}_L - \mathbf{c}) \times \mathbf{f}_L + \tau_L + (\mathbf{p}_R - \mathbf{c}) \times \mathbf{f}_R + \tau_R = \dot{\mathbf{l}} \tag{21}$$

where $\mathbf{c}$, $\mathbf{p}_L$, and $\mathbf{p}_R$ are the positions of the CoM, left and right feet, $\mathbf{f}_L$, $\mathbf{f}_R$, τ_L, and τ_R are the reaction forces and torques generated at the feet, $\mathbf{f}_g = [0, 0, -g]^\mathsf{T}$ is the force of gravity, m is the mass, and $\mathbf{l}$ is the angular momentum. Since the absolute position is rarely relevant, it is useful to place the origin of the coordinate system at the stance foot so that the CoM location, $\mathbf{c}$, and the swing foot location, $\mathbf{p}_w$, are defined relative to the stance foot. During double support, the foot that will be in stance next is considered the stance foot. It is also useful to define the total reaction force and torque as follows:

$$\begin{aligned} \mathbf{f} &= \mathbf{f}_L + \mathbf{f}_R \\ \tau &= \tau_L + \tau_R. \end{aligned} \tag{22}$$

During single support, the swing foot cannot generate reaction force, so one of the pairs of force and torque must be zero. If we then constrain our policy such that $\dot{\mathbf{l}} = 0$ and $\ddot{\mathbf{c}}_z = 0$, (20) and (21) simplify to the well-known Linear Inverted Pendulum Model (LIPM) [15] [16]. We further constrain the dynamics with $\dot{\mathbf{c}}_z = 0$ and $\mathbf{c}_z = h$ and write the LIPM dynamics as

$$\ddot{\mathbf{c}}_x = \mathbf{c}_x \frac{g}{h} + \frac{\tau_y}{mh} \tag{23}$$

$$\ddot{\mathbf{c}}_y = \mathbf{c}_y \frac{g}{h} + \frac{\tau_x}{mh}. \tag{24}$$

We model the swing leg as fully controllable and treat the acceleration of the swing foot, $\ddot{\mathbf{p}}_w$, as a control variable.

During double support, there is no swing foot to accelerate, but the horizontal CoM acceleration depends on how the weight is distributed between the two feet, which we define as

$$w = \frac{\mathbf{f}_{L,z}}{\mathbf{f}_{L,z} + \mathbf{f}_{R,z}}. \tag{25}$$

We assume that we can select w during double support such

that

$$\ddot{\mathbf{c}}_x = \frac{\tau_y}{mh} \tag{26}$$

$$\ddot{\mathbf{c}}_y = \frac{\tau_x}{mh}. \tag{27}$$

Equations (26) and (27) are approximations because they require that both

$$w = \frac{\mathbf{c}_x - \mathbf{p}_{L,x}}{\mathbf{p}_{R,x} - \mathbf{p}_{L,x}} \tag{28}$$

and

$$w = \frac{\mathbf{c}_y - \mathbf{p}_{L,y}}{\mathbf{p}_{R,y} - \mathbf{p}_{L,y}}. \tag{29}$$

It is only possible to simultaneously satisfy (28) and (29) if the CoM is directly above the line between the two feet. However, this approximation is small because the CoM is usually near this line during double support, double support is brief, and the low-level controller can often fix some of the discrepancy by adjusting τ. This approximation is necessary because it allows us to decouple the sagittal and coronal dynamics, and it is useful because it allows us to calculate the CoM acceleration without knowing the position of both feet.

These dynamics constitute a 5 degree of freedom (DoF) ICS with a 10-dimensional state space (position and velocity for each DoF),

$$\begin{aligned}\mathbf{x}_f = \{&\mathbf{c}_x, \dot{\mathbf{c}}_x, \mathbf{c}_y, \dot{\mathbf{c}}_y, \mathbf{p}_{w,x}, \\ &\dot{\mathbf{p}}_{w,x}, \mathbf{p}_{w,y}, \dot{\mathbf{p}}_{w,y}, \mathbf{p}_{w,z}, \dot{\mathbf{p}}_{w,z}\}\end{aligned} \tag{30}$$

and a 5-dimensional action space (one for each DoF),

$$\mathbf{u}_f = \{\tau_y, \tau_x, \ddot{\mathbf{p}}_{w,x}, \ddot{\mathbf{p}}_{w,y}, \ddot{\mathbf{p}}_{w,z}\}. \tag{31}$$

We can then partition the state and action space into 5 subsystems, one for each DoF:

$$\begin{array}{ll} \mathbf{x}_s = \{\mathbf{c}_x, \dot{\mathbf{c}}_x\} & \mathbf{u}_s = \{\tau_y\} \\ \mathbf{x}_r = \{\mathbf{c}_y, \dot{\mathbf{c}}_y\} & \mathbf{u}_r = \{\tau_x\} \\ \mathbf{x}_x = \{\mathbf{p}_{w,x}, \dot{\mathbf{p}}_{w,x}\} & \mathbf{u}_x = \{\ddot{\mathbf{p}}_{w,x}\} \\ \mathbf{x}_y = \{\mathbf{p}_{w,y}, \dot{\mathbf{p}}_{w,y}\} & \mathbf{u}_y = \{\ddot{\mathbf{p}}_{w,y}\} \\ \mathbf{x}_z = \{\mathbf{p}_{w,z}, \dot{\mathbf{p}}_{w,z}\} & \mathbf{u}_z = \{\ddot{\mathbf{p}}_{w,z}\} \end{array} \tag{32}$$

where the subscripts, s, r, x, y, and z, refer to the sagittal stance, coronal stance, swing-x, swing-y, and swing-z subsystems.

The systems are only coupled during stance transitions (touch down and lift off). We choose a common state that describes the horizontal location of the swing foot, $\mathbf{x}_c = \{\mathbf{p}_x, \mathbf{p}_x\}$. In order to keep the decision state, $\hat{\mathbf{x}}_d$, low-dimensional, we consider only the next transition ($M = 1$) and make assumptions about all future transitions. This gives us a decision state of

$$\hat{\mathbf{x}}_d = \{t_t, x_{td}, y_{td}\} \tag{33}$$

where t_t is the time until transition, and $\{x_{td}, y_{td}\}$ is the location where the swing foot will touch down. For lift off transitions, x_{td} and y_{td} can be omitted. The stance subsystems assume that subsequent transitions will have the nominal timing (0.1 second double support and 0.4 second single support), but that they will be able to select future touchdown locations. The swing subsystems assume that subsequent transitions will have nominal values from steady state walking. Fig. 2 shows t_t as a function of time for the walking simulation starting from rest and accelerating to steady state walking at 0.56 m/s. During single support, it is convenient to refer to t_t as time until touchdown, t_{td}.

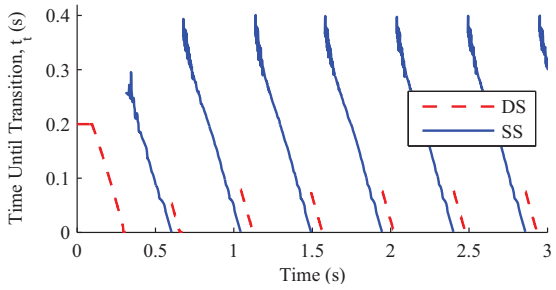

Fig. 2. Time until transition, t_t versus time. To reduce computation, policies are only computed for $t_t < 0.2$ during double support.

This selection of $\mathbf{x}_c$ and the resulting $\hat{\mathbf{x}}_d$ allows our subsystem controllers to determine the optimal action for all possible choices of footstep timings and locations. The value functions can then be used to determine which choice of these variables is optimal for the full ICS.

We minimize the cost function

$$\begin{aligned}C = \int (&w_1 \tau_x^2 + w_2 \tau_y^2 + w_3 (\dot{\mathbf{c}}_x - v_{des})^2 + w_4 (\mathbf{c}_y - w_h)^2 + \\ &w_5 (\mathbf{p}_{w,z} - h_{fc})^2 + \ddot{\mathbf{p}}_{\mathbf{w}}^{\mathsf{T}} \mathbf{W}_6 \ddot{\mathbf{p}}_{\mathbf{w}})\, dt\end{aligned} \tag{34}$$

subject to the constraint that

$$\mathbf{p}_w(t_{td} = 0) = \{x_{td}, y_{td}, 0\}. \tag{35}$$

The values w_1 through w_5 are weighting constants, $\mathbf{W}_6$ is a diagonal weighting matrix, w_h is half the width of the hips, and h_{fc} is the nominal foot clearance height.

Note that (34) has the form of (9) and that (35) can be decoupled as in (12). This model of walking thus meets all the criteria of an ICS. If we omit the dimensions of $\hat{\mathbf{x}}_d$ that do not affect the dynamics, the original 10-dimensional system is equivalent to a coordinated set of one 3-dimensional system and four 4-dimensional systems.

IV. WALKING CONTROLLER

We use the principle of an ICS to generate a walking controller for a simulated biped based on our Sarcos Primus System [17] [18] hydraulic humanoid robot with force-controlled joints. The simulation is of approximately human size (CoM is 1.0 m high when standing straight) and mass (78 kg). It is a 3D 5-link (torso and two 2-link legs) rigid body simulation with 16 degrees of freedom: 6 to locate and orient the torso as well as 3 at each hip and 1 at each knee. It is controlled by 12 torque controlled joints: 3 at each hip, 1 at each knee, and roll and pitch actuation between each point foot and the ground. The CoP constraint of a finite-size foot is simulated by enforcing

$$\begin{aligned}|\tau_x| &\le w_f \mathbf{f}_z \\ |\tau_y| &\le l_f \mathbf{f}_z\end{aligned} \tag{36}$$

on each foot where $w_f = 0.05m$ and $l_f = 0.1m$ are approximately half the width and length of a human foot. Friction (coefficient of friction is $\mu = 0.5$) is modeled as a spring and damper between the foot and the ground. When the friction cone,

$$\frac{\sqrt{\mathbf{f}_x^2 + \mathbf{f}_y^2}}{\mathbf{f}_z} < \mu, \tag{37}$$

or yaw torque constraint,

$$\frac{\tau_z}{\mathbf{f}_z} < \mu_r, \tag{38}$$

is violated, slipping is modeled by resetting the rest position of the spring.

We use coordinated DP polices to produce an optimal controller for the ICS described in Section III. This functions as our high-level controller, providing input CoM and swing foot accelerations to a low-level controller, which outputs joint torques.

A. High-Level Control

1) Pre-Computation: Policies and value functions are generated for each of the five subsystems using dynamic programming. A discount factor of 0.9995 is used, which corresponds to costs fading to half importance after 1.4 seconds (nearly 3 steps).

For the swing-z system, the dynamics are not affected by x_{td} or y_{td}, so it is sufficient to generate a policy on the 3-dimensional state space of $\{\mathbf{p}_{w,z}, \dot{\mathbf{p}}_{w,z}, t_{td}\}$. The touch down constraint is enforced by switching to a special controller for $t_{td} < 0.03$ s, which applies whatever acceleration is necessary to touch down at the right time and returns a value determined by the magnitude of the necessary acceleration. An analytic controller is necessary because the magnitude of the gradient of the optimal controller approaches infinity as t_{td} approaches zero. As a result, the policy can not be well represented by a grid for small t_{td}.

The swing-x system is affected by x_{td}. However, by shifting the reference frame and redefining the state as $\{\mathbf{p}_{w,x} - x_{td}, \dot{\mathbf{p}}_{w,x}, t_{td}\}$, we have a system with the same dynamics and cost function as the original, but only a 3-dimensional state space. The constraint, $\mathbf{p}_{w,x} - x_{td} = 0$ when $t_{td} = 0$, is enforced similarly to the swing-z system. The swing-y system can be made 3-dimensional by a similar shift in reference frame.

These five DP policies (three 3-dimensional and two 4-dimensional policies) are equivalent to a single 10-dimensional DP policy for the entire ICS. If we use a resolution of 100 states per dimension, the coordinated version uses 2.3×10^8 states as opposed to 1.0×10^{20} states for the equivalent single policy. Computing the DP policies is computationally intensive and can take on the order of a day for our 4-dimensional policies. They are computed before use, and this computation does not affect the run-time performance of the controller.

2) Run-Time Computation: At run time, we combine the value functions to obtain $\mathbf{x}_d^*$ as in (19). During double support, the arg min operation is only a 1-dimensional search, so it can be performed by a fine resolution brute force search. During single support, however, the search space is 3-dimensional, and brute force search is computationally expensive. To speed up the search, we note that all five value functions depend on t_{td}, but that only 2 each depend on x_{td} and y_{td}, and that none of the value functions depend on both x_{td} and y_{td}. We wish to first find $x_{td}^*(t_{td})$ and $y_{td}^*(t_{td})$, so that we can then perform a 1-dimensional search over $V(t_{td}|x_{td}^*(t_{td}), y_{td}^*(t_{td}), \mathbf{x}_f)$.

To do this, we approximate the value functions (during pre-computation) in such a way that they can be added quickly and that $x_{td}^*(t_{td})$ and $y_{td}^*(t_{td})$ of the sums can be found analytically. For the coronal and swing-y value functions, we approximate the value function, $V(t_{td}, y_{td}|\mathbf{x}_i)$, with a series of parabolic approximations to $V(y_{td}|t_{td}, \mathbf{x}_i)$ for evenly spaced values of t_{td}. Each parabola is created by placing the vertex at the minimum of $V(y_{td}|t_{td}, \mathbf{x}_i)$ and using a point to either side to estimate the second derivative. Two surfaces can then be added quickly by adding the parabolas, and $x_{td}^*(t_{td})$ is simply the location of the vertex of the sum. Fig. 3 shows an example surface approximation.

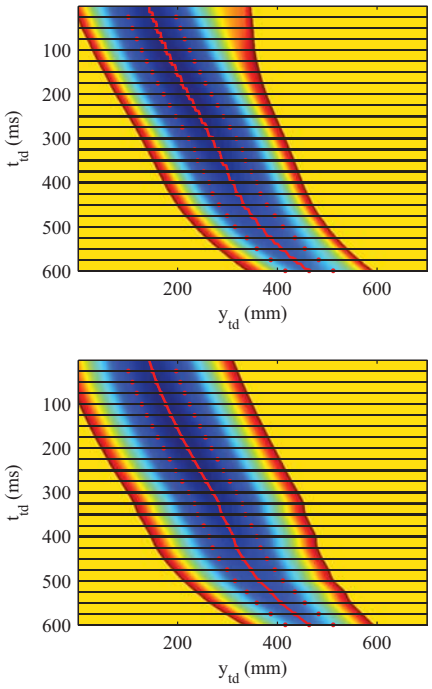

Fig. 3. The coronal stance value function, $V(t_{td}, y_{td}|\mathbf{c}_y = 0.08, \dot{\mathbf{c}}_y = 0)$, from the DP tables (top) and from the parabolic approximation (bottom). The red line shows $y_{td}^*(t_{td})$. The dots show the points used to generate the parabolic approximation, and the horizontal black lines show the location of the parabolas.

The same is done with the sagittal stance and swing-x value functions, using x_{td} instead of y_{td}. With the value functions reduced to only a function of t_{td} as shown in Fig. 4, they can be quickly added and searched for t_{td}^*. Then, we can look up $x_{td}^*(t_{td}^*)$ and $y_{td}^*(t_{td}^*)$. Having determined $\hat{\mathbf{x}}_d$, we can now look up the appropriate controls from the individual policies.

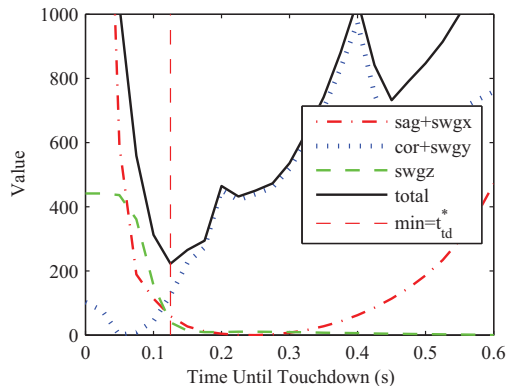

Fig. 4. Value as a function of t_{td} following a push to the side. For ease of view, values are normalized so that the minimum is 0. Due to the push, the coronal policy wants to touch down soon, but it must compromise with the other policies to pick the best time, t^*_{td} for the full ICS.

B. Low-Level Control

The output of the high-level controller is the desired horizontal CoM acceleration, $\ddot{\mathbf{c}}_{x,des}$ and $\ddot{\mathbf{c}}_{y,des}$, as well as the desired swing foot acceleration $\ddot{\mathbf{p}}_{w,des}$. The objective of the low-level controller is to generate joint torques which will achieve these accelerations as well as enforce the constraints assumed by the high-level control, $\mathbf{c}_z = h$ and $\dot{\mathbf{l}} = 0$.

It is important to note that the high-level controller does not generate trajectories. Instead, it maps directly from system state to desired accelerations without maintaining any controller state. This allows it to react to perturbations and accumulated modeling error in real time, but it also means that we do not have desired positions or velocities, which precludes the use of traditional trajectory tracking techniques. In the place of trajectory tracking, we use a form of inverse dynamics to generate joint torques. The feedback gains of our controller are embedded in the gradients of the high-level DP policies.

We use PD controllers to enforce the $\mathbf{c}_z = h$ constraint and maintain a desired torso orientation, giving us $\ddot{\mathbf{c}}_{z,des}$ and the desired total moment. It is then straightforward to compute the desired total reaction force, $\mathbf{f}$, and torque, τ. During double support, we divide the total reaction force between the two feet while enforcing the CoP (36) and friction (37), (38) constraints by minimizing

$$C = \frac{\mathbf{f}_{L,x}^2}{\mathbf{f}_{L,z}} + \frac{\mathbf{f}_{R,x}^2}{\mathbf{f}_{R,z}} + \frac{\mathbf{f}_{L,y}^2}{\mathbf{f}_{L,z}} \frac{\mathbf{f}_{R,y}^2}{\mathbf{f}_{R,z}} + a\left(\frac{\tau_{L,x}^2}{\mathbf{f}_{L,z}} + \frac{\tau_{R,x}^2}{\mathbf{f}_{R,z}} + \frac{\tau_{L,y}^2}{\mathbf{f}_{L,z}} \frac{\tau_{R,y}^2}{\mathbf{f}_{R,z}}\right). \quad (39)$$

This cost function has the useful property that it produces the same CoP offset for both feet, ensuring that there is as much margin as possible between the CoP and the edge of the foot.

We then use Dynamic Balance Force Control (DBFC) as presented in [19] to generate joint torques, τ_j. DBFC uses a weighted pseudo-inverse with regularization to solve

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & -\mathbf{S} \\ \mathbf{J}(\mathbf{q}) & 0 \end{bmatrix} \begin{pmatrix} \ddot{\mathbf{q}} \\ \tau_j \end{pmatrix} = \begin{pmatrix} N(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{J}(\mathbf{q})\hat{\dot{\mathbf{f}}} \\ -\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \hat{\ddot{\mathbf{p}}} \end{pmatrix} \quad (40)$$

where $\mathbf{q}$ is a vector of base coordinates and joint angles, $\mathbf{M}(\mathbf{q})$ is the mass matrix, $\mathbf{J}(\mathbf{q})$ is the Jacobian of both feet, $\mathbf{S} = [0, \mathbf{I}]$ selects the actuated elements of $\mathbf{q}$, $\hat{\dot{\mathbf{f}}} = [\mathbf{f}_L{}^{\mathsf{T}}, \tau_L{}^{\mathsf{T}}, \mathbf{f}_R{}^{\mathsf{T}}, \tau_R{}^{\mathsf{T}}]^{\mathsf{T}}$, and $\hat{\ddot{\mathbf{p}}} = [\ddot{\mathbf{p}}_L^{\mathsf{T}}, \ddot{\mathbf{p}}_R^{\mathsf{T}}]^{\mathsf{T}}$.

Since we do not use PD joint torques in addition to the DBFC output, even small errors in the foot acceleration produced can accumulate over time. In order to more accurately match desired foot accelerations, we add an integrator on foot acceleration,

$$\ddot{\mathbf{p}}_{w,int} = \int (\ddot{\mathbf{p}}_{w,des} - \ddot{\mathbf{p}}_w)\, dt, \quad (41)$$

which we add to the desired foot acceleration, $\ddot{\mathbf{p}}_{w,des}$, used by DBFC. The integrator is not necessary for stable walking, but it significantly improves the robustness to perturbations.

V. ROBUSTNESS AND SPEED CONTROL

An important characteristic of any controller is its ability to reject perturbations. In particular, the size of the largest perturbation that does not cause the system to fail is a useful metric for systems where failure is well defined. One practical difficulty with using this as a metric of performance for walking is that the result of a perturbation depends on the timing, location, and direction of the perturbation.

Fig. 5 shows the walking pattern produced by a push of 30 Newton-seconds (N-s) to the left (0.38 m/s change in the CoM velocity), and Fig. 6 shows the effect of push angle and timing on the maximum survivable perturbation. Force perturbations lasting 0.1 seconds are administered to the torso CoM at various angles while the system is walking with an average speed of 0.56 m/s. Data is shown in Fig. 6 for perturbations beginning at increments of 0.1 seconds after the left foot lifts off.

Walking speed can be changed by switching the sagittal stance policy to one computed with a different v_{des}. The policies are global, so no transition is necessary, and the policies can be switched at any point during the step. Similarly, the system can start from rest and achieve steady state walking without switching policies. Fig. 5 shows the walking pattern produced by switching speeds, and Fig. 7 shows how the velocity varies after changing policies.

VI. PRELIMINARY HARDWARE RESULTS

We implemented a version of our controller for standing balance on the Sarcos Primus System humanoid robot with force-controlled hydraulic joints. Results of standing perturbation experiments in which we pushed the robot are shown in Fig. 8. Pushes were administered to the back of the torso and measured by a force sensor in the pushing implement.

We model the CoM dynamics of standing using the LIPM. We place the origin of our coordinate frame halfway between the feet and use the dynamics in (23) and (24) to model the

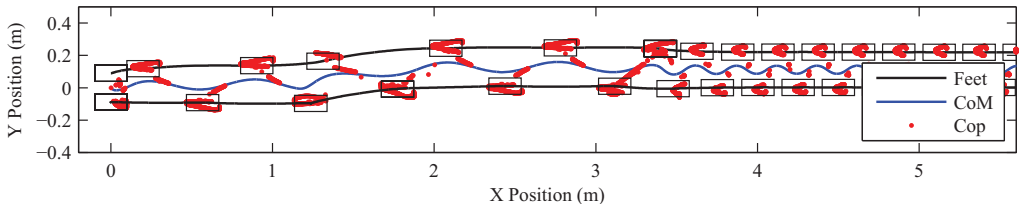

Fig. 5. Walking data, starting from rest. At the 5th step it is pushed to the left with a 30 Ns impulse. After 5 seconds, it switches from $v_{des} = 0.63$ to $v_{des} = 0.25$.

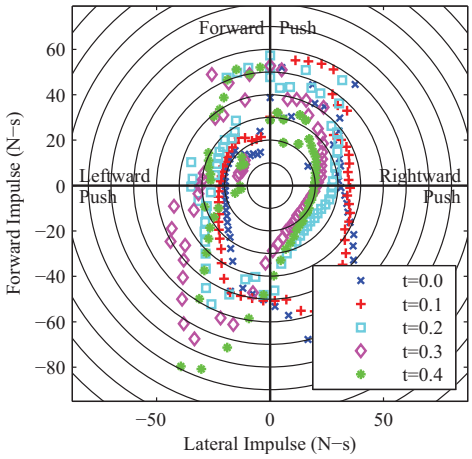

Fig. 6. Polar plot of the maximum survivable perturbation of our walking simulation as a function of angle and time. Data is shown for perturbations occuring at various times after left foot lift off. A point represents the maximum survivable perturbation in a given direction. Concentric circles are in increments of 10 Newton-seconds.

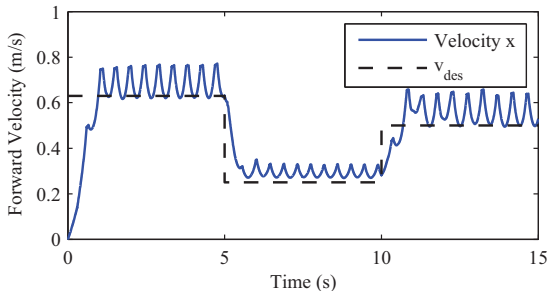

Fig. 7. Forward speed as the sagittal stance policy is changed. Starts from rest with $v_{des} = 0.63$, switches to $v_{des} = 0.25$ after 5.0 seconds, and to $v_{des} = 0.5$ after 10.0 seconds.

horizontal motion of the CoM. Our controller attempts to minimize the one step cost function

$$L = \tau_x^2 + \tau_y^2 + x^2 + y^2. \tag{42}$$

Unlike walking, standing has no stance transitions, so the sagittal and coronal motion are completely decoupled. Additionally, we need not consider a swing leg. We can therefore produce a high-level controller by coordinating two 2-dimensional policies (a policy for each horizontal axis defined over CoM position-velocity space). Since they are completely decoupled, no decision state is required. The desired CoM accelerations produced by this controller are then used by the low-level controller described in Section IV-B to generate low-level joint torques.

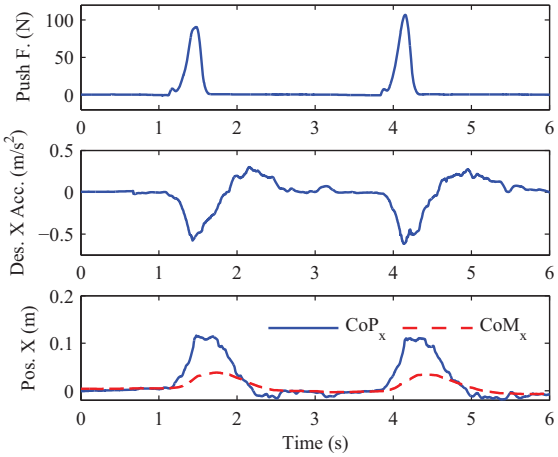

Fig. 8. Standing balance perturbation experiments on the Sarcos humanoid robot for two forward pushes.

VII. DISCUSSION AND FUTURE WORK

The controller's current version strives to maintain a desired torso orientation, but it does not otherwise address angular momentum. Manipulation of angular momentum, however, can be useful for recovering from large perturbations, as rotational accelerations cause a temporary shift in the CoM without moving the center of pressure [20].

By avoiding pre-planned trajectories, our controller can react immediately to unexpected perturbations. It selects a new time and location of touchdown at every control cycle (currently 400 Hz). In the sense that it avoids tracking a planned trajectory, our method resembles MPC. However, to compute the control in real time, MPC approaches typically use linear dynamics and fix the step timing. We, on the other hand, avoid any requirement of linearity by performing our optimization iteratively offline. However, offline computation forces us to consider all possible states, which subjects us to the "Curse of Dimensionality" and constrains the dimensionality of our dynamics. Our coordination scheme does let us handle higher-dimensional systems by breaking them into lower-dimensional subsystems, but it requires that

the full system be an ICS. This, in turn, requires that both the dynamics and cost function be capable of being decoupled (except at transitions).

One major advantage of this control framework is its flexibility: Coordination is not dependent on any particular dynamic model, cost function, or constraints. While we must take care to maintain low dimensionality and the ability to decouple our system, we remain free from any further restrictions - for instance, linearity - on the dynamics or cost function. In fact, many of the motion's characteristics can be controlled by adjusting the cost functions, and the variables considered can be changed by combining or adding additional subsystems.

This flexibility opens the door to several possible extensions. We plan to produce a more human-like walking motion by discarding the LIPM dynamics and the constant $\mathbf{c}_z = h$ constraint in favor of a more natural (and efficient) height constraint. The inclusion of an additional policy that controls arm swing and yaw torque may also improve the gait's efficiency and aesthetics.

Furthermore, we plan to achieve walking on steps or otherwise non-flat ground by setting the $\tilde{h}(x_{td}, y_{td})$ term in the cost function, (18), to a terrain cost and allowing the touchdown constraint to depend on x_{td} and y_{td}. The former action would invalidate our parabolic approximations of the value functions, but a more general method of performing the minimization in (19) - such as gradient descent starting from multiple locations - can be used, instead. Implementing walking on the Sarcos humanoid force-controlled robot will be the focus of future work.

VIII. CONCLUSION

This paper defined an Instantaneously Coupled System and demonstrated the equivalence of coordinated policies that are optimal for the subsystems to a single controller that is optimal for the full ICS. We apply this theory to walking and present a walking controller for a simulated biped. Our controller optimizes center of mass motion as well as footstep timing and location, and it can react in real time to perturbations and accumulated modeling error. We also present standing balance experiments on a force-controlled humanoid robot.

IX. ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation through a Graduate Research Fellowship and under grants DGE-0333420, ECCS-0325383, EEC-0540865, ECCS-0824077, and IIS-0964581.

REFERENCES

[1] A. Takanishi, T. Takeya, H. Karaki, and I. Kato, "A control method for dynamic biped walking under unknown external force", in *IEEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*. 1990, vol. 29, pp. 795–801, IEEE.

[2] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point", in *Proceedings of the IEEE International Conference on Robotics and Automation*, September 2003, pp. 1620–1626.

[3] Y. Fujimoto and A. Kawamura, "Proposal of biped walking control based on robust hybrid position/force control", in *Proceedings of IEEE International Conference on Robotics and Automation*. 1996, vol. 3, pp. 2724–2730, IEEE.

[4] Koichi Nishiwaki and Satoshi Kagami, "High frequency walking pattern generation based on preview control of ZMP", in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2006, pp. 2667–2672.

[5] Pierre-Brice Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2006.

[6] Chenggang Liu and Christopher G. Atkeson, "Standing balance control using a trajectory library", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[7] Pierre-Brice Wieber and Christine Chevallereau, "Online adaptation of reference trajectories for the control of walking systems", *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 559–566, 2006.

[8] Holger Diedam, Dimitar Dimitrov, Pierre-Brice Wieber, Katja Mombaur, and Moritz Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[9] Mitsuharu Morisawa, Kensuke Harada, Shuuji Kajita, Shinichiro Nakaoka, Kiyoshi Fujiwara, Fumio Kanehiro, Kenji Kaneko, and Hirohisa Hirukawa, "Experimentation of humanoid walking allowing immediate modification of foot place based on analytical solution", in *Proceedings of IEEE International Conference on Robotics and Automation*, October 2007, pp. 3989–3994.

[10] Michiel van de Panne, Eugene Fiume, and Zvonko G. Vranesic, "A controller for the dynamic walk of a biped across variable terrain", in *Proceedings of the 31st Conference on Decision and Control*, December 1992, pp. pp. 2668–2673.

[11] KangKang Yin, Kevin Loken, and Michiel van de Panne, "Simbicon: simple biped locomotion control", in *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, New York, NY, USA, 2007, p. 105, ACM.

[12] Thijs Mandersloot, Martijn Wisse, and Christopher G. Atkeson, "Controlling velocity in bipedal walking: A dynamic programming approach", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2006.

[13] Mike Stilman, Christopher G. Atkeson, James J. Kuffner, and Garth Zeglin, "Dynamic programming in reduced dimensional spaces: Dynamic planning for robust biped locomotion", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, pp. 2399–2404.

[14] Christopher G. Atkeson, "Randomly sampling actions in dynamic programming", in *Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007.

[15] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode", in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1991, vol. 2, pp. 1405–1411.

[16] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa, "The 3d Linear Inverted Pendulum Model: A simple modeling for biped walking pattern generation", in *Proceedings of the IEEE International Conference on Robotics and Automation*, November 2001, pp. 240–246.

[17] Darrin C. Bentivegna, Christopher G. Atkeson, and Jung-Yup Kim, "Compliant control of a compliant humanoid joint", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007.

[18] Jung-Yup Kim, Christopher Atkeson, Jessica Hodgins, Darrin Bentivegna, and Sung Ju Cho, "Online gain switching algorithm for joint position control of a hydraulic humanoid robot", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2007.

[19] Benjamin J. Stephens, "Dynamic balance force control for compliant humanoid robots", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, Available online at www.cs.cmu.edu/~bstephe1/papers/iros10.pdf.

[20] Benjamin Stephens, "Humanoid push recovery", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, November 2007.