Control of a Walking Biped Using a Combination of Simple Policies

Eric C. Whitman and Christopher G. Atkeson
Carnegie Mellon University
Pittsburgh, PA 15213, USA
email: ewhitman@cmu.edu

***Abstract*— We present a decoupled controller for a simulated three-dimensional biped. To handle the high-dimensionality of the system, we break the dynamics down into multiple subsystems, which we control separately. For both the sagittal and coronal plane dynamics, we use dynamic programming to simultaneously optimize body motion, foot placement, and step timing for a two link inverted pendulum model. To use these simplified policies we map the full state to a simplified state, and then map the control action back onto the full system. The swing leg is controlled via continuously updated desired trajectories. These separate policies are then coordinated by the estimated time until touchdown, which is provided by the sagittal policy. By varying the lean angle or changing the sagittal policy we are able to contol the walking speed. We also evaluate the performance of our controller in terms of robustness to perturbations.**

I. INTRODUCTION

In this paper, we present a decoupled controller for a simulated bipedal walking system. We extend dynamic programming, a control technique that is typically limited to very low-dimensional systems, into higher dimensions by dividing the full system into multiple simpler systems, which can then be individually controlled more easily along the lines of [1]. Though the "separate" systems are, in reality, coupled, the coupling can be viewed as modeling error for the simple systems.

We coordinate the multiple policies through the estimated time until touch down. The estimated time until touch down behaves much like the phase in a central pattern generator [2] during steady-state walking, but handles transients that affect the step timing differently. Dynamic programming allows us to simultaneously and globally optimize foot placement, step timing, and body motion for a large volume of the state space, enabling an effective response to large perturbations.

A. Related Work

A common control paradigm for high degree of freedom (DoF) systems is to generate a nominal trajectory (possibly with associated feed-forward commands) and then stabilize the system around it with a feedback controller. Such trajectories are often generated by trajectory optimization [3]. This type of controller often only performs well in a narrow tube of state space around the nominal trajectory. Unfortunately, in an underactuated system, even high feedback gains typically cannot keep the system near the nominal trajectory in the presence of perturbations. Even in the absence of external perturbations, modeling error can push the system off of the desired trajectory. In many cases, it is extremely difficult to produce a sufficiently accurate model for the generation of feasible trajectories. This problem is made worse when feedback gains are reduced to make the system compliant.

One possible response to this problem is to produce policies that are valid for a large region of the state space, allowing them to handle large deviations form the nominal trajectory. The viable region of a trajectory-based controller can be increased by using a library of multiple trajectories, producing a controller that performs well in the union of the tubes around each trajectory in the library [4] [5]. Models and prediction can be avoided entirely by doing direct policy search [6] [7], typically with parametric policies. Intuition about the task can be used to develop sets of parametric policies that are particularly useful and easy to optimize [8]. Alternatively, some model-based control design strategies, such as dynamic programming [9] [10], can generate policies for aa large region of the state space.

Some researchers choose to base their controllers on simplified systems that capture important aspects of the full dynamics rather than attempting to accurately model the full system [10]. It may then be possible to decouple various aspects of control [11] [1]. The difference between the simplified system and the full system can be partially eliminated by learning the difference between the systems [12]. Alternately, the system can be controlled so as to behave like a simplified system [13] [14].

Many researchers design dynamically stable walking trajectories based on the zero moment point (ZMP). Based on the ZMP constraint, dynamically stable walking trajectories have been generated both numerically [15] and analytically [16]. Kajita et al. used Preview Control of the ZMP constraint to generate center of mass (CoM) trajectories [17]. A desired ZMP trajectory is chosen ahead of time based on specified footstep locations and timing. Then the CoM trajectory is calculated based on the desired ZMP trajectory. By randomly sampling the configuration space and constructing a tree of feasible paths, foot placement and whole-body motion can be simultaneously planned [18]. By treating the footstep location as an additional control in a linear model predictive control scheme, footstep location (but not timing) has been optimized along with the body motion [19].

B. Dynamic Programming

In the version of dynamic programming (DP) used in this paper, we divide the state space into a grid. We then iteratively solve for a steady state policy $u(x)$ and value

function $V(x)$ at each point in the grid according to

$$u(x) = \arg\min_u (L(x,u) + cV(f(x,u))) \quad (1)$$

$$V(x) = L(x,u) + cV(f(x,u)), \quad (2)$$

where x is the state, c is the discount factor, $L(x,u)$ is the one step cost function, and $x_{N+1} = f(x_N, u)$ 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 (1) to pick a new action, $u(x)$, from between only two choices: the current best action and a random action [20]. We then use (2) to update the value function accordingly. We iterate this procedure until the value and policy converge to a global optimum.

DP produces a controller that is valid for the entire volume considered. This makes it useful for optimizing transient responses to perturbations as well as optimizing steady state gait. It supports torque control approaches well, and produces a controller that is valid even far from the optimal trajectory. DP is also globally optimal (up to the grid resolution), avoiding potential problems with local minima, and does not require a terminal value function as is required by many trajectory optimization algorithms. A major advantage of DP is that it can easily handle discrete decisions such as whether or not to touch down now, allowing simultaneous optimization of trajectory, footstep timing, and footstep placement. Visualization of the policies and value functions produced by DP can increase physical intuition and lead to helpful insights.

C. Compass Gait

One popular simplified model of walking systems is the compass gait walker. It has a point mass body and two rigid, massless legs. During walking, the body travels in a series of arcs centered at the stance foot. Many researchers have studied various aspects of this system including passive stability [21], speed control [9], and rough terrain traversal [22]. Hardware has even been built specifically to mimic the dynamics of this simplified system as closely as possible [22].

In section 2, we describe the simulated system, and in section 3, we describe the controller. In sections 4 and 5, we describe the system's response to perturbations and control of the walking speed. In section 6, we discuss our results and future work, and we give a conclusion in section 7.

II. THE SYSTEM

The system we control (pictured in Fig. 1) is a simulated three-dimensional biped comprised of five rigid links: a torso, two thighs, and two calves. It is modeled on our Sarcos Primus System hydraulic humanoid robot [23] [24] of approximately human size and mass. It weighs 78 kg, with a 50 kg torso and 14 kg legs. The legs are 0.81 m long and the CoM is 1.00 m above the ground when it is standing straight. The joints can be controlled to have low impedance.

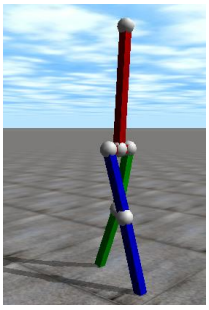

Fig. 1. The system walking.

Our simulation is a 12 DoF system: six to locate and orient the torso, two for each hip and one for each knee. Since the position and velocity of each of these degrees of freedom are required to fully describe the state of the system, it has a 24 dimensional state space. It is actuated by 12 torque-controlled joints. There is a roll and a pitch joint at each hip, a pitch joint at each knee, and three joints at each ankle. Though a foot is not explicitly modeled, while on the ground, the ankle may apply a torque between the calf and the ground in the roll or pitch directions as well as twisting around the axis of the calf. To keep the virtual foot flat on the ground, the roll and pitch ankle torques, τ_r and τ_p, are constrained to keep the center of pressure of the foot within the horizontal convex hull of the foot:

$$\begin{aligned} |\tau_r| &\leq w_f F_z \\ |\tau_p| &\leq l_f F_z \end{aligned} \quad (3)$$

where w_f and l_f are scaled to the width and length of a human foot, and F_z is the vertical force on the foot.

Contact between the feet and the ground is modeled as a spring and damper. The fricion cone constraint is given by

$$\frac{\sqrt{F_x^2 + F_y^2}}{F_z} < \mu \quad (4)$$

where F_x, F_y, and F_z are the components of the friction force and $\mu = 1.0$ is the coefficient of friction. If (4) is broken, slipping is modeled as resetting the rest position of the spring to the current location.

III. CONTROL ARCHITECTURE

The system has both a high-dimensional state space and a high-dimensional action space. Since control in such high-dimensional spaces is difficult, we divide the system into multiple lower-dimensional systems, which can be separately controlled much more easily [1]. With the exception of the ankle twist joints, each joint acts purely in either the sagittal or the coronal plane in the nominal standing pose. This allows us to approximate the sagittal and coronal dynamics as being decoupled and control them separately. The only coupling between the two planes that is acknowledged by our controller is that both sets of dynamics must switch between left and right stance at the same time. Though the system experiences brief periods of double support (approximately 1%-2% of the step), double support is ignored, and the

$$\ddot{\theta} = \frac{M(L_{1y}l\sin(\theta)\dot{\theta}^2 + lg\sin(\theta) - L_{1x}l\cos(\theta)\dot{\theta}^2 + L_{1x}g) + mL_2g\sin(\theta) + \tau}{I_2 + M(l^2 + L_{1y}l\cos(\theta) + L_{1x}l\sin(\theta)} \tag{5}$$

system is instead treated as being singly supported by the lead leg. Double support periods are so short because our gait is based on a compass gait, with relatively straight, noncompliant legs.

A. Sagittal Stance Leg Policy

The sagittal plane dynamics are seven DoF: three to locate and orient the torso and four more for the hips and knees. Its action space is five-dimensional and includes the torque at the pitch joints at both hips, both knees, and the stance ankle. The swing leg ankle torque is zero when not in contact with the ground. Rather than attempting to control this system directly, we simplify things further by developing a policy for a simplified system and then mapping this policy onto the full system. In addition to a policy for the simplified system, this approach requires mappings from full states to simplified states and from simplified actions to full actions. A schematic of the state simplification, policy, and action expansion process is shown in Fig. 2.

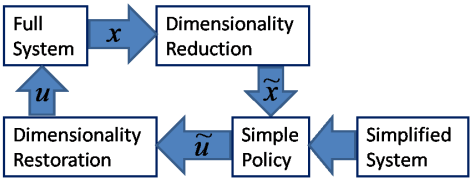

Fig. 2. A schematic of the process necessary to use a policy designed for a simplified system on a more complex system.

To simplify the system, we make several assumptions. We remove two DoF by assuming the stance foot is located at the origin. Another two DoF are avoided by ignoring the swing leg, which will be controlled separately, and is assumed to have low enough mass to have negligable effect on the dynamics of the torso and stance leg. This assumption is validated later when we show that this simplified system behaves similarly to the full system. We further assume that the stance knee be kept straight at all times and that the torso is locked at a constant angle with respect to vertical. This leaves us with a one DoF system: a two-link inverted pendulum with the upper link at a fixed angle as shown in Fig. 3.

The dynamics for this sytem are given by (5), where l=0.81 m, L_2=0.4 m, $L_{1x} = 0.4\sin(\phi)$, and $L_{1y} = 0.4\cos(\phi)$ are lengths as defined in Fig. 3 and ϕ is the torso lean angle (nominally 0.1 rad). The masses of the body and leg are given by $M = 50kg$ and $m = 14kg$ respectively, $I = ml^2/3$ is the moment of inertia of the leg measured around the foot, and τ is the ankle torque. The one degree of freedom, the angle between the leg and vertical is given by θ.

This system has a two-dimensional action space. In addition to the ankle torque, τ_1, the touchdown time is also

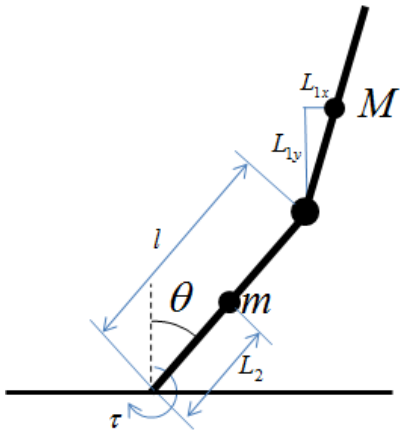

Fig. 3. Schematic of the simplified sagittal system.

controlled. Since the swing leg is neglected in this system, the control policy has a Boolean decision of whether or not to touch down at any time. At touchdown, both legs have the same length (straight knees), so they form an isosceles triangle with the ground. At touchdown, therefore, the leg angle flips signs ($\theta = -\theta_0$). Some energy is lost during impact with the ground, which we model as $\dot{\theta} = 0.65\dot{\theta}_0$.

We use dynamic programming to generate a policy for the simplified model. Our one step cost function is

$$L(x,u) = 6000(v - v_{des})^2 + \tau_1^2 + 0.02F_{x,grf}^2, \tag{6}$$

where v and v_{des} are the actual and desired (1.0 m/s) forward velocity of the hip, and $F_{x,grf}$ is the ground reaction force in the forward direction. The velocity and torque terms constitute the classic tradeoff between error (deviation from constant forward motion) and effort. The $F_{x,grf}$ term is included to help prevent large horizontal forces, which could result in slipping in the full system. A cost is used rather than a constraint because the full system GRF does not exactly match the simplified GRF, so any constraint on the simplified GRF would be insufficient. The ZMP constraint is enforced by limiting the ankle torque. The numeric constants are chosen empirically.

To map the full state to the simplified state, we take the three-dimensional vector from the stance foot to the stance hip and project it onto the sagittal plane. The leg angle, θ, is then the angle between the projected vector and the vertical. The leg angular velocity, $\dot{\theta}$, is obtained by taking the numerical derivative of θ.

To map the simplified action to the full system, which has more degrees of freedom, we must enforce the assumptions of the simplified model. To enforce the assumption of a constant length leg, we use a proportional-derivative (PD) servo to keep the stance leg knee straight ($K_p = 1500$ Nm; $K_d = 150$ Nm-s). We also use a PD servo ($K_p = 1000$ Nm;

$K_d = 150$ Nm-s) at the stance leg hip to keep the torso at a constant angle with respect to vertical. The stance leg ankle torque is controlled directly by the ankle policy computed by dynamic programming.

Fig 4 shows the trajectory that results from this controller in both the simplified and the full models. The resulting behavior is also compared visually in the attached video. Though the full model moves at a somewhat higher frequency (and higher speed), the shape of the plots are similar. Much of the difference in speed and shape comes from the approximation of the touch down model. The simplified system simply loses a fraction of its speed at touch down. In the full system, however, the torso also bobs forward at impact. The torque applied at the hip over the first portion of the step to return it to its nominal lean angle also propels the system forward, returning some of the lost speed to the system. You can also see that the first step of the full model starts out with a negative acceleration as a reaction to leaning the torso forward.

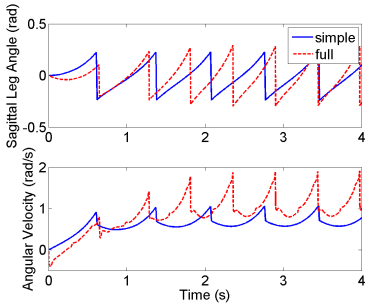

Fig. 4. Comparison of walking in the sagittal plane for the simplified and full models starting from rest. We plot only the stance leg angle, which switches between the left and right physical legs at the vertical discontinuities.

B. Sagittal Swing Leg Policy

The swing leg must be controlled to touch down at the appropriate angle and angular velocity when the stance leg policy commands it to. The knee must also be straight at touchdown, but bent enough to not contact the ground during swing.

Fortunately, both the dynamics and the controller are predictable, so we can know ahead of time at what time and angle touchdown will be commanded. This is done by simulating forward on the simplified sagittal model until touchdown and recording the duration of the swing as well as the final state. The touchdown time and state can be pre-computed starting from each point on the dynamic programming grid. This takes only as long as a few dynamic programming iterations, so adds little to the total pre-processing time. These are only estimates because the full system dynamics differ from the simplified dynamics on which they are calculated. This inaccuracy is partially mitigated by continuously recalculating the estimate for the majority of the step. For the final 0.1 seconds of each step, however, the estimated touch down angle and angular velocity are fixed because the cost of continued adjustment (smooth motion is essential to touching down without slipping) outweighs the increased accuracy. From this point, the commanded time until touchdown, t_{td}, counts down in real time, as shown in Fig. 5. For a typical step, the estimates of touchdown angle, touchdown angular velocity, and step duration at the beginning of the step are off by about 30%, 45%, and 15% respectively. This error is due entirely to the difference between the full and simplified dynamics. At touchdown, the estimation error for t_{td} is typically less than a few milliseconds.

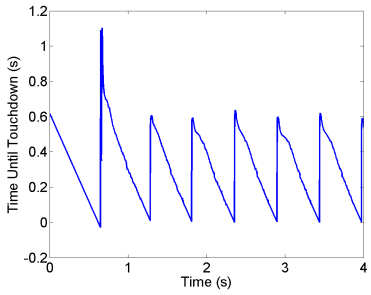

Fig. 5. Commanded time until touchdown, t_{td}. It is usually allowed to vary with the updating estimate of when the simplified policy will command touchdown. At the end of the step, it is forced to count down in real time.

To simplify control of the leg, we treat it as if it were telescoping, using the knee to control the length, l, and the hip to control the orientation. As with the stance leg, the swing leg angle is measured by projecting the vector between the foot and hip into the sagittal plane and measuring the angle between the projected vector and vertical. While the hip must sweep the leg from its liftoff to its touchdown angle over the course of a step, the knee must bend to lift the foot above the ground while swinging forward. We therefore command a foot height, d, of 5 cm for the majority of the step.

We use inverse kinematics to compute the knee angle, θ_k, and angular velocity, $\dot{\theta}_k$, necessary to maintain the foot at the desired hight given the current hip height and leg orientation. Once the desired θ_k and $\dot{\theta}_k$ are calculated, we can use a PD servo ($K_p = 1500$ Nm; $K_d = 150$ Nm-s) to control the knee in order to obtain the desired foot height.

To ensure touch down with a straight knee at desired time, we override the foot height controller with direct knee commands in order to meet the constraint

$$\theta_k \leq 2t_{td} \tag{7}$$

at the end of a step. Commanded and actual knee angles as well as the resulting foot height are shown in Fig. 6.

The swing leg hip torque is controlled by a PD ($K_p = 1000$ Nm; $K_d = 150$ Nm-s) servo on the swing leg angle (angle between vertical and the hip-ankle vector). When stance changes at touch down, a desired trajectory represented as a cubic spline is generated starting at the current angle and angular velocity of the leg and ending at the anticipated touchdown angle and angular velocity at the commanded touchdown time. This trajectory is constantly updated as the

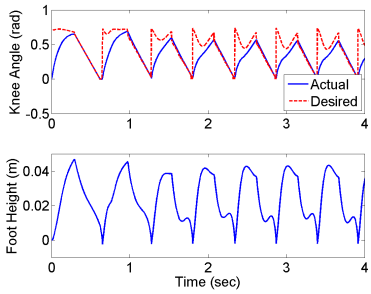

Fig. 6. Knee angles and foot height for the swing leg.

target changes with a new spline generated from the current angle and angular velocity of the hold trajectory to the new target. Once the target is fixed towards the end of the step, the trajectory is also fixed and no longer regenerated. When the trajectory ends ($t_{td} < 0$) or when the foot is very near the ground ($d < 0.005$), the swing leg angle is velocity controlled to match the angular velocity of the stance leg. This is done to prevent both the foot contacting the ground at a high velocity and large internal forces during the brief periods of double support, either of which could result in a slip. The swing leg trajectory is shown in Fig. 7.

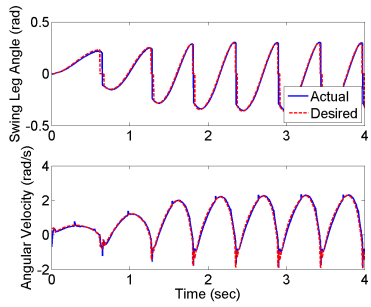

Fig. 7. Swing leg angle and angular velocity. The commanded leg angle is plotted as 0 when the leg angle is under velocity control. The small wiggle in angular velocity that occurs in the middle of each step is a result of the knee switching from height to angular control.

C. Coronal Policy

The full coronal plane dynamics have five DoF and a four dimensional action space consisting of the roll joints in both hips and ankles. The stance leg is controlled in a similar manner to how it is controlled in the sagittal plane; the full state is again mapped onto a simpler set of dynamics for which a policy can be easily developed as in Fig. 2.

The simplified dynamics used are nearly identical to those used for the sagittal case, except that a third state is added, estimated time until touchdown, t_{td}. The dynamics of this state are modeled as counting down in real time until it reaches zero, at which point touch down occurs and t_{td} is reset to the average step duration. Unlike the sagittal plane controller, the coronal plane controller does not decide when to touch down; instead, it can select at what angle to touch down. Following touchdown, the new leg angle, θ, will be the selected angle and $\dot{\theta}$ will be a constant fraction of the angular velocity before touch down ($\dot{\theta} = 0.65\dot{\theta}_0$ as in the simple sagittal system). Whereas the torso always leans forwards in the sagittal plane, in the coronal plane, the torso mass is to the left of the right hip and to the right of the left hip. Accordingly L_{1x} must change signs at touchdown (magnitude of 0.09 m).

Again, we construct a policy for the system using dynamic programming, this time using a cost function of

$$L = 100000y^2 + 10\dot{y}^2 + \tau_1^2 + 0.1F_{y,grf}^2 \qquad (8)$$

where y is the horizontal location in the coronal plane of the hip This is equivalent to (6), but with a desired velocity of zero and the addition of the y^2 term, which tends to keep the legs close to vertical. The leg angle and its derivative are extracted from the full state by projecting the leg vector into the coronal plane, and t_{td} is obtained from the sagittal controller. The stance ankle is controlled directly by the dynamic programming policy, and the stance hip is used to servo the torso to vertical. A similar look-ahead technique to that used to calculate the time and angle of touchdown in the sagittal controller is used to pre-compute what the commanded angle of touch down will be in the coronal controller. The swing hip is then servoed ($K_p = 4000$ Nm; $K_d = 250$ Nm-s) in the coronal plane to the appropriate angle.

Fig. 8 shows the trajectory for the simplified and full models following this policy. The behavior of the two systems is also compared visually in the attached video. It is unremarkable that the periods match relatively well because in this case the period is specified as a design parameter. The impact of touchdown causes the torso to roll slightly away from the new stance leg. The torque necessary to return the torso to vertical is responsible for the large angular acceleration of the coronal leg angle seen at the beginning of each step.

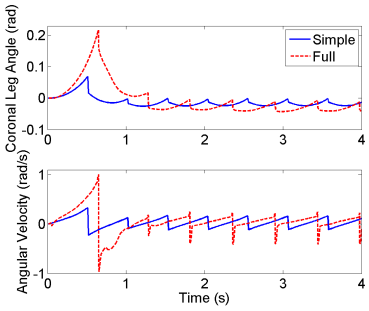

Fig. 8. Comparison of walking in the coronal plane for the simplified and full models. We plot only the stance leg angle, which switches between the left and right physical legs.

D. Yaw Control

The simplest way to control yaw is to use the ankle twist joints. They are aligned along the axis of the calf, which is on average aligned with the yaw axis. Servoing ($K_p = 500$

Nm; $K_d = 30$ Nm-s) the joints to zero is enough to produce walking that is neutrally stable in yaw. Adding a controlling term that is proportional to the difference between the actual and desired yaw angle stabilizes the system about the desired yaw angle. Unfortunately, only very slight yaw stability can be obtained in this manner because large gains make the system fail due to significant coupling with the coronal plane dynamics. The coupling is large because the normal motion of the leg brings the shin axis partially in line with coronal plane torques.

IV. ROBUSTNESS TO PERTURBATIONS

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. 9 shows the effect of push angle and timing on the maximum survivable perturbation. Impulsive force perturbations are administered to the torso CoM at various angles. Data is shown for perturbations at increments of 0.1 seconds after the left foot touches down. An unperturbed step takes about 0.58 seconds. For perturbations to the front or right (upper-right quadrant of Fig. 9), the timing does not matter very much; the maximum survivable perturbation is roughly constant regardless of when the perturbation occurs. For perturbations to the left or rear (lower-left quadrant of Fig. 9), however, the system is significantly more stable mid step than at the beginning or end of the step.

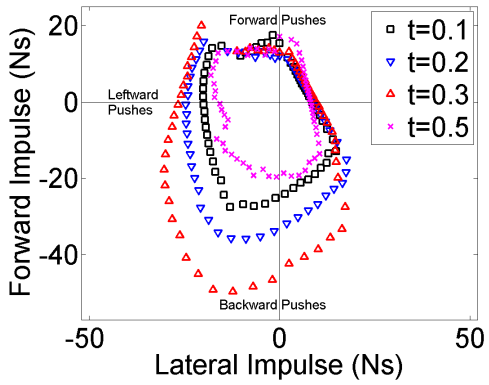

Fig. 9. Polar plot of the maximum survivable perturbation as a function of angle and time. Data is shown for perturbations occuring at various times after left foot touch down. A point represents the maximum survivable perturbation in a given direction.

There are two distinct failure modes: tipping and slipping. Slipping is very sensitive to the coefficient of friction, but tipping is not. Fig. 10 shows that increasing the coefficient of friction has a large effect on how large a perturbation is required to induce failure in some directions (forward), but almost no effect in other directions. In fact, doubling the coefficient of friction very nearly doubles the resistance to perturbations in some directions. Changing the height of the perturbation has little effect on the lateral ground reaction force; however, it does have a significant effect on the perturbing torque around the stance foot, which is what causes tipping. Fig. 10 also shows the effect of locating perturbations at the torso CoM versus 20 cm higher. This has a noticeable effect in some directions (right) and a smaller effect in other directions. It has a particularly small effect in the direction where increasing the coefficient of friction has a large effect. For a given direction, the relative effect of changing the friction and perturbation height helps to distinguish between tipping and slipping failures.

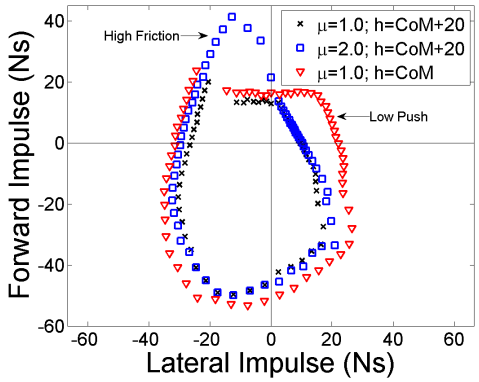

Fig. 10. Effect of changing the coefficient of friction, μ, and height, h, of the perturbation. Perturbations occur 0.3 seconds after left foot touch down. Perturbations are administered either at the torso CoM or 20 cm above the torso CoM.

V. SPEED CONTROL

There are two readily available methods for controlling speed within this framework. Either the torso lean angle or the desired velocity of the sagittal policy can be adjusted.

Increasing the torso lean angle pushes the mass of the torso farther forward relative to the footstep location. This causes an increased torque around the stance foot, which propells the system forward. Policies are computed assuming a constant lean angle, and deviations from this angle function as continuous perturbations, speeding or slowing the system.

Fig. 11 shows the effect of lean angle on speed for a sagittal policy computed with a desired velocity of 1.0 m/s and lean angle of 0.10 radians. The system exhibits stable walking with actual lean angles between -0.02 (slightly backwards) and 0.11 radians. Even at the nominal lean angle, the system walks slower than the desired velocity of 1.0 m/s because given the cost function, (6), the policy must tradeoff the cost of deviation from the desired velocity and the cost of the additional torque necessary to go faster. The full systems travels faster than the simplified system because it loses less energy at touchdown.

For a greater change in speed, it is necessary to change the sagittal policy. The speed can be varied continuously by interpolating between a few policies, each computed for a different speed. This method has the advantage that the system will still do what the policy expects, so the

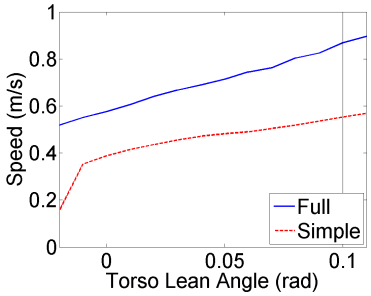

Fig. 11. Effect of changing the lean angle on walking speed. The vertical black lines represents the nominal lean angle for which the sagittal policy was designed.

estimates of touchdown angle and time will be more accurate. Additionally, the step lengths and torques used will be closer to optimal. Performance is best when changing policies and lean angles in tandem.

Fig. 12 shows the effect of changing policies on the forward speed. It walks for eight steps with one policy, then switches to a second sagittal policy while holding everything else constant. The first policy has a desired velocity of 1.0 m/s; the second policy has a desired velocity of 0.25 m/s, but ends up walking significantly faster because little energy is lost in the full system at touchdown while taking such short steps at low speeds. It takes about two steps to get to approximately the steady state speed, but several more steps for the speed to completely level off.

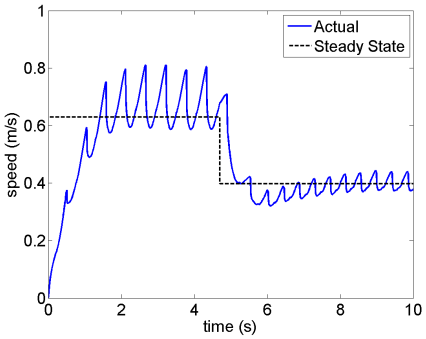

Fig. 12. Effect of changing the policy on walking speed. Steady state reffers to the steady state velocity achieved using the current policy.

VI. DISCUSSION AND FUTURE WORK

In this paper, we presented a controller for a walking biped based on decoupling it into multiple separate systems. These subsystems were each simple enough that robust controllers for them could be designed. Though the couplings between the subsystems were ignored in the controller design, the individual controllers were sufficiently robust to function within the simulated full system.

Though this study was done entirely in simulation, we believe that this control scheme is well-suited to use on real hardware. Since we use only simplified models of portions of the dynamics, it may reduce the need to construct a detailed model of the system, one of the more difficult problems in standard control schemes. One advantage of this control architecture is that it is modular. By this, we mean that any one of the sub-controllers can be modified or replaced without requiring any changes to the other controllers. This is helpful from a control design standpoint because it allows specific problems to be addressed directly. This modularity also means that the same general framework can be used to produce other types of walking. For example, instead of basing the sagittal controller on the compass gait, we could have based it on the Linear Inverted Pendulum Model (LIPM) [25], which keeps the hips and CoM at a constant height. Both the compass gait and the LIPM model can be viewed as constraints in leg angle - hip height space, and any such constraint could have been used [14]. The modular structure also leaves room for several improvements to the current controller. The policies in this controller are developed on simplified systems, and are therefore only optimal for the simplified systems. The closer the full dynamics are to the simplified dynamics, the more likely the policies are to be reasonable. One way to match the full dynamics more closely is to relax the constraint of a fixed lean angle for the torso in the sagittal plane by adding the angle and angular velocity of the torso as additional states and the torque at the hip as an additional action. This would allow torso dynamics to be taken into account when formulating the sagittal policy. It would also allow for a more accurate touch down model. Additionally, it would allow the controller to take advantage of torso dynamics, which would be particularly helpful when changing speed. We leave this refinement for future work.

Dynamid Programming requires very little run-time computation, merely a constant time table lookup (and interpolation). On the other hand, it requires extensive offline computation. The offline computation is exponential in the number of state dimensions, which is a major reason why coordinating multiple simpler policies is worthwhile. Operation under other conditions such as varied walking speed or sloped terrain can be handled by computation of additional policies suitable to those conditions as in [1] [9] [22].

In this paper, we used a master-slave architecture of policy coordination where the sagittal policy determined when touch down would occur and the coronal policy accepted this value. One can easily imagine the reverse scenario, where the coronal policy determined the timing and the sagittal policy accepted it. Future work will develop a collaborative approach to policy coordination. Dynamic programming (used to develop the policies) produces a value function, V, in addition to the policy. We can assume that the value function for the combined policy is equal to the sum of the value of the individual policies as in [26]:

$$V = V_s + V_c \tag{9}$$

If both the sagittal and coronal policies were developed with time until touch down as a state, and we then split that state into the physical state, $\mathbf{x}$, and t_{td} (the same for both policies),

we can rewrite (9) as

$$V(\mathbf{x_s}, \mathbf{x_c}, t_{td}) = V_s(\mathbf{x_s}, t_{td}) + V_c(\mathbf{x_c}, t_{td}). \quad (10)$$

Written in this way, selecting t_{td} is a matter of minimizing value:

$$t^*_{td} = \arg\min_{t_{td}} V_s(\mathbf{x_s}, t_{td}) + V_c(\mathbf{x_c}, t_{td}), \quad (11)$$

which is a simple lookup operation on the value function from dynamic programming. Such a system would provide a good mechanism for timing to be dictated by whichever policy it was most important to. This can be generalized by including the swing leg or any number of additional controllers in the selection of t_{td} so long as $V(\mathbf{x}, t_{td})$ can be calculated for those controllers.

VII. CONCLUSION

We have produced a controller for a walking biped based on decoupling the system into multiple simpler systems, and coordinating the simpler controllers designed using Dynamic Programming. We are able to generate policies that are valid for a large region of the high-dimensional state of the full system. This allows the system to react to large perturbations. Unlike ZMP preview control, we simultaneously globally optimize body motion, step location, and step timing. This allows for an optimized adjustment of footstep timing and location in response to unexpected perturbations.

VIII. ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation under grants DGE-0333420, ECCS-0325383, EEC-0540865, and ECCS-0824077.

REFERENCES

[1] Michiel van de Panne, Eugene Fiume, and Zvonko G. Vranesic, "A controller for the dynamic walk of a biped over variable terrain", in *Proceedings of the 31st Conference on Decision and Control*.

[2] Jun Morimoto, Gen Endo, Jun Nakanishi, Sang-Ho Hyon, Gordon Cheng, Darrin Bentivegna, and Christopher G. Atkeson, "Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking", *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 1579–1584, May 2006.

[3] M.W. Hardt, *Multibody dynamical Algorithms, Numerical Optimal Control, with Detailed Studies in the Control of Jet Engine Compressors and Biped Walking*, PhD thesis, University of California, San Diego, 1999.

[4] Pierre-Brice Wieber and Christine Chevallereau, "Online adaptation of reference trajectories for the control of walking systems", *Robotics and Autonomous Systems*, vol. 54, pp. 559–566, July 2006.

[5] Chenggang Liu and Christopher G. Atkeson, "Standing balance control using a trajectory library", in *Proc. of the IEEE/RSJ IROS*, 2009.

[6] M. van de Panne and E. Fiume, "Sensor-actuator networks", in *Proceedings of ACM SIGGRAPH*, 1993, pp. 335–342.

[7] J. Andrew (Drew) Bagnell, Sham Kakade, Andrew Ng, and Jeff Schneider, "Policy search by dynamic programming", in *Neural Information Processing Systems*. December 2003, vol. 16, MIT Press.

[8] Chee-Meng Chew and Gill A. Pratt, "A general control architecture for dynamic bipedal walking", in *Proceedings of the IEEE International Conference on Robotics & Automation*, 2000, pp. 3989–3995.

[9] 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, pp. 124–130.

[10] M. Stilman, CG Atkeson, JJ Kuffner, and G. Zeglin, "Dynamic programming in reduced dimensional spaces: Dynamic planning for robust biped locomotion", *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 2399–2404, 2005.

[11] Marc H. Raibert, *Legged robots that balance*, Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[12] John R. Rebula, Fabian Canas, Jerry E Pratt, and Ambarish Goswami, "Learning capture points for bipedal push recovery", in *Proceedings of the IEEE International Conference on Robotics & Automation*, 2008.

[13] Jerry Pratt, Peter Dilworth, and Gill Pratt, "Virtual model control of a bipedal walking robot", in *IEEE Conference on Robotics and Automation*, 1997, pp. 193–198.

[14] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*, CRC Press, 2007.

[15] Satoshi Kagami, Koichi Nishiwaki, Tomonobu Kitagawa, Tomomichi Sugihara, Masayuki Inaba, and Hirochika Inoue, "A fast generation method of a dynamically stable humanoid robot trajectory with enhanced zmp constraint", in *In Proceedings of the IEEE International Conference on Humanoid Robotics*, 2000.

[16] Kensuke Harada, Shuuji Kajita, Kenji Kaneko, and Hirohisa Hirukawa, "An analytical method on real-time gait planning for a humanoid robot", *International Journal of Humanoid Robotics*, 2004.

[17] 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 *In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, September 2003, pp. 1620–1626.

[18] Kensuke Harada, Mitsuharu Morisawa, Shin-Ichiro Nakaoka, Kenji Kaneko, and Shuuji Kajita, "Kinodynamic planning for humanoid robots walking on uneven terrain", *Journal of Robotics and Mechatronics*, vol. 21, no. 3, pp. 311–316, 2009.

[19] H. Diedam, D. Dimitrov, P.B. Wieber, K. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control", in *Proc. of the IEEE/RSJ IROS*, 2008, pp. 1121–1126.

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

[21] Ambarish Goswami, Bernard Espiau, and Ahmed Keramane, "Limit cycles in a passive compass gait biped and passivity mimicking control laws", *Journal of Autonomous Robots*, vol. 4, no. 3, pp. 273–286, 1997.

[22] Katie Byl and Russ Tedrake, "Approximate optimal control of the compass gait on rough terrain", in *Proceedings of the IEEE International Conference on Robotics & Automation*, 2008, pp. 1258–1263.

[23] 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, pp. 211–218.

[24] 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.

[25] 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 & Automation*, April 1991, vol. 2, pp. 1405–1511.

[26] Daphne Koller and Ronald Parr, "Computing factored value functions for policies in structured mdps", in *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. 1999, pp. 1332–1339, Morgan Kaufmann.