Robust Dynamic Walking Using Online Foot Step Optimization

Siyuan Feng†, X Xinjilefu∗, Christopher G. Atkeson† and Joohyung Kim‡

Abstract—To enable robust dynamic walking on the Atlas robot, we extend our previous work [1] by adding a receding-horizon component. The new controller consists of three hierarchies: a center of mass (CoM) trajectory planner that follows a sequence of desired foot steps, a receding-horizon controller that optimizes the next foot placement to minimize future CoM tracking errors, and an inverse dynamics based full body controller that generates instantaneous joint commands to track these motions while obeying physical constraints. The proposed controller is implemented and tested on the Atlas robot. It is capable of walking with strong external perturbations such as recovering from large pushes and traversing unstructured terrain.

I. INTRODUCTION

Our walking controller [1] for the DARPA Robotics Challenge (DRC) consisted of two main modules: a long term center of mass (CoM) trajectory planner using a simple model, and a full body inverse dynamics based controller for generating instantaneous joint commands that are consistent with the physical constraints. This approach was suitable for our “slow and steady” strategy for the DRC, but it relied on accurate models and state estimation and precious control especially for the CoM states. It could not handle large CoM tracking errors because the original plan can become infeasible due to limited CoM acceleration. Replanning is an intuitive and practical solution for this issue, but for unstable systems like walking robots, the time budget for replanning is very limited. In this paper, we present an extension to our hierarchical approach by introducing a receding-horizon component between the existing CoM planner and the full body controller. This new module can rapidly replan for a short horizon based on the long term information from the CoM planner.

Many successful model based approaches for humanoid walking compute reference CoM / Zero-Moment Point (ZMP) trajectories, which are then open-loop tracked using a full body controller. Preview Control [2] is a very popular walking pattern generation method using Linear Inverted Pendulum Model (LIPM). Capture point [3] is another useful conceptual tool for balancing, and it can also be generalized to walking [4], [5]. Similarly, divergent component of motion [6], [7] is introduced to encode the unstable part of the LIPM dynamics and used for walking pattern generation. To improve robustness, Receding Horizon Control [8] (also known as Model Predictive Control) can be used for push recovery and improve walking pattern generation. An online approach proposed by Nishiwaki et al. rapidly adjusts the reference ZMP trajectory based on the measured robot state to account for external perturbations [9]. It is further extended to vary foot placement or step timing using characteristics of Preview Control [10], in which timing can be resolved by a line search. Another foot placement strategy based on Preview Control is proposed by Urata et. al [11], [12]. A special form of Preview Control’s cost function is used to enable fast computation of the CoM / ZMP trajectory, which is used as an evaluation function to optimize for the next foot step.

Foot placement and CoM trajectory can also be generated simultaneously by solving a linear trajectory optimization problem using LIPM dynamics [13]–[16]. These approaches are typically formulated as quadratic programs that optimize for foot placement and the time derivatives of the ZMP. Piecewise linear acceleration is assumed to reduce the number of necessary samples (optimization variables). Instead of following the desired foot steps, these approaches can track overall behavioral goals such as a desired average speed or reaching some long term position [17]. It can also be extended to combine the ankle, hip and stepping strategies [18] by using a linear model with angular momentum. Linear receding-horizon controllers are also extensively used in [19] for push recovery with strong disturbances. Our receding-horizon component is formulated similarly to this group, but the objective is to follow the nominal high level plan.

Before going into details about the receding-horizon component, we want to briefly summarize our previous work on controlling the Atlas robot, which consists of a CoM trajectory planner and a inverse dynamics based full body controller. These components remain largely the same in the new controller.

A. Center of Mass Trajectory Optimization

A nominal CoM trajectory is optimized by Differential Dynamic Programming (DDP) [20] given a sequence of desired foot steps. DDP is a local iterative trajectory optimization technique that can be applied to nonlinear dynamics. In addition to the optimized trajectory, DDP also produces a linear policy and a local second order approximation for the value function along the trajectory, which are used to guide the receding-horizon foot placement controller and the full body controller. Let $X^t$ and $u^t$ be the state and control on the optimized trajectory at time step $t$. $X$ be the estimated state, and $X_e = X - X^t$. We can compute the control and
approximated the value function with
\[
V(X) \approx V_t^0 + V_t^{XX} e^{X X} + \frac{1}{2} X_T^{XX} X_e \\
u(X) = u^t - K_t X_e. \tag{1}
\]
We focus on level walking in this work, so LIPM is used instead of the nonlinear 3D point mass model for CoM trajectory planning. The biggest advantage for using a linear model is fast computation, which only requires one DDP iteration to converge. Consequently, desired CoM trajectory can be recomputed within a few milliseconds after touchdown using the estimated states, which simplifies software implementation as well.

### B. Full Body Control

We use the same full body controller developed for the DRC Finals [1] to generate instantaneous joint level commands. On every control step, it solves inverse dynamics formulated as a quadratic program. Like many others, our formulation originates from [21]. A set of desired accelerations (e.g. CoM, swing foot, etc.) are specified by the higher level modules as inputs, and it optimizes for a combination of generalized acceleration, joint torques and contact wrenches that best track those desired motions while obeying dynamics constraints. For controlling a high degree of freedom system such as a humanoid robot, there are two popular approaches for resolving redundancies: ranking the objectives using different weights [1], [5], [19], [22], [23]; and imposing a strict hierarchy on the objectives [24]–[28]. We prefer the simpler and faster weighted approach.

To produce accurate motions on the robot, tracking just the inverse dynamics torque is not sufficient due to modeling errors [1], [5], [22]. Therefore, an additional velocity tracking simplification is made: Linear Inverted Pendulum Model (LIPM); fixed foot orientation; fixed foot timing; point foot; and short double support phase and zero CoM acceleration during double support. Linear dynamics and fixed timing are necessary to make the system dynamics linear with respect to the optimization variable, so the problem becomes convex and fast to solve. The point foot assumption forces the CoP to coincide with foot placement, so that we do not need to sample in time to take into account variable CoP. With these assumptions, the time evolution of the CoM state can be expressed as a linear function of the initial CoM state and the CoP (foot placement) based on LIPM dynamics.

#### A. Foot Step Optimization with Quadratic Programming

The CoM state \( X \) is defined as \([x \: \dot{x}]^T\), and the foot placement is denoted by \( p \). Given known timing \( t \), future CoM state can be expressed as:

\[
X = A(t)X_0 + B(t)p
\]

\[
A(t) = \begin{bmatrix}
\omega e^{\omega t} + e^{-\omega t} & e^{\omega t} - e^{-\omega t} \\
\omega (e^{\omega t} - e^{-\omega t}) & e^{\omega t} + e^{-\omega t}
\end{bmatrix}
\]

\[
B(t) = \begin{bmatrix}
1 - 0.5(e^{\omega t} + e^{-\omega t}) \\
0.5\omega (e^{-\omega t} - e^{\omega t})
\end{bmatrix},
\]

where \( \omega = \sqrt{g/\ell}, \) and \( X_0 \) is the initial state.

During swing, let \( p_{TD} \) be the remaining duration of the current swing phase, \( p_{cur} \) be the current stance foot position, and \( X \) be the estimated current CoM state. Given by Eq. 2, the CoM state at planned touchdown can be computed as \( X_p = A(t_{TD})X + B(t_{TD})p_{cur} \). Assuming zero CoM acceleration during double support, the CoM state at liftoff is \( X_{LO} = X_{TD} + [\dot{x}_{TD} T_{DS} 0]^T \), where \( T_{DS} \) is the duration of double support. For foot placement \( p \) and any time \( t \) during the next swing phase, the CoM state can then be computed as

\[
X_t = A(t)X_{LO} + B(t)p. \tag{3}
\]

The cost function consists of two terms: one for foot placement deviation from the planned location \( p^* \), and another for CoM state tracking error during the next swing phase.

\[
\min_p \sum_t (X_t - X_t^*)^T V_t (X_t - X_t^*) + w(p - p^*)^2, \tag{4}
\]

where \( w \) is a weight, and \( X_t^* \) and \( V_t \) are the nominal CoM state and the second order derivative of the value function computed by DDP (Eq. 1) sampled at time \( t \) after liftoff. In the current implementation, five equally timed \( X^* \) and \( V \) are sampled in Eq. 4.

A set of linear inequality constraints are used to approximate the allowed stepping region. The simplest box constraint relative to the current stance foot is used in this implementation. The swing foot has to be placed within \( \pm 0.5m \) in the \( X \) direction and between \([0.17, 0.6]\)m away from the current stance foot in the \( Y \) direction. Ideally, the foot step planner will generate these constraints in addition to the nominal foot step taking sensor inputs into account. The foot step planner can also produce a cost map which can substitute the first term in Eq. 4. The orientation of the foot step is not optimized, and the desired orientation is used for the swing foot.

#### B. Simple Example

In this simple example, we use the same LIPM for both planning and simulation. There is no double support phase, and the leg swings perfectly. CoM height is set to \( 0.88m \). The overall task is to walk in place, and the desired foot steps...
During simulation, we allow some control over CoP, which is used as the end knot points of a spline for generating the nominal swing foot pose $x_d^*$, velocity $\dot{x}_d^*$ and acceleration $\ddot{x}_d^*$ during swing, which are then used to compute the input target acceleration $\ddot{x}^*$ for the full body controller. We have experimented with updating the knot point directly with $p$, but the resulting $\ddot{x}^*$ changes too drastically between time steps, and causes wild swing foot motions that can sometimes destabilize the walking cycle. Instead, we keep the spline interpolation the same, and compute $\ddot{x}^*$ with the following heuristic:

$$\ddot{x}^* = K_p(x_d^* + \alpha(p - p^*) - x) + K_d(\dot{x}_d^* - \dot{x}) + \ddot{x}_d$$

$$\alpha = \begin{cases} \frac{t - t_{LO}}{T_{SS}}, & \text{if } t < t_{LO} + T_{SS} \\ 1, & \text{otherwise} \end{cases}$$

where $T_{SS}$ is the duration of the swing phase, $t_{LO}$ is the liftoff time, and $t$ is the current time. This scheme produces a much smoother $\ddot{x}^*$ since it only incorporates a portion of the new foot placement position at any time, and regulates the velocity towards the original interpolated one.

### III. ROBOT EXPERIMENTS

Robot experiments are done with the Atlas robot built by Boston Dynamics, which has 30 actuated degrees of freedom, six for each leg, seven for each arm, three for the spine, and one for neck pitch. Most are hydraulic actuators with the exception of the neck and the last three joints on each arm which are electric. To simplify the experimental setup, the next desired foot step is always updated with respect to the current stance foot location, so that the robot only tries to maintain balance rather than an absolute position after disturbances. This is also the simplest receding-horizon foot step planner.

The overall goal is to keep walking under strong disturbances. The first set of experiments are conducted with external kicks applied by humans around Atlas’ pelvis, and the second set requires Atlas to walk over a strip of unstructured terrain made of loose rubble. Snapshots of these experiments are shown in Figure 2 and Figure 4(a).
Unfortunately, we did not have instruments to measure the magnitude or the duration of these perturbations, and we are unable to estimate the net impulse due to noisy estimates of the CoM velocity. Fast walking is also attempted, and the fastest walking speed we have achieved is around 0.6m/s. The walking controller is the same for all these experiments, and the CoM planner and the full body controller have few differences from [1].

A. Push Recovery

Figure 3 shows plots of the CoM states and the optimized foot steps when recovering from pushes in the coronal plane. The first thing worth noticing is that we are not controlling the CoM velocity very well, which is quite oscillatory during the single support phases. These oscillations also have a direct impact on the optimized foot placement shown with the orange lines in Figure 3(c). The oscillations in foot placement require a large amount of smoothing and damping, otherwise they will cause large swing foot motions that induce further CoM oscillations. This motivates for only partially updating the position part of the swing foot acceleration computation in Eq. 5. However, this heuristic can introduce a large tracking error when the swing foot needs to move fast. Obviously, there is room for improvements for our CoM and swing foot tracking, but this also shows that during high cadence dynamic walking, precise control is marginal, the robot can maintain dynamic balance as long as it can put down the swing foot somewhere reasonable at the right time. The effectiveness of CoP control is minimal comparing to taking steps.

B. Walking on Rubble

Figure 4 shows foot steps taken by Atlas when walking over a loose rubble field that is roughly 3m by 0.9m. This experiment is particularly challenging, especially for the state estimator, because the fundamental assumption of stationary contacts is often violated. Physically, the actual support region for the stance foot is shrunk significantly, and the robot is effectively walking on stilts. Estimating the actual support is close to impossible in this case. It will very difficult to walk over this kind of terrain statically since precise CoP control will be incredibly hard. Many of the position controlled humanoids achieve compliance with feedbacks on the measured ground reaction wrench. We also speculate that these approaches will not work well on this terrain because of noisy measurements and limited bandwidth for their force feedback.

C. Fast Walking

We want to explore how fast our Atlas can walk in the last set of experiments, and the fastest walking speed we achieved is roughly at 0.6m/s. Data for this experiment are plotted in Figure 5. This speed is computed by averaging the estimated CoM velocity over a couple cycles once the robot starts walking forward. We think a hardware limit stopped us from going faster. The onboard pump is unable to deliver the amount of flow at the desired pressure when the robot is walking that fast. Our Atlas starts going down as it walks, which is also evident from large torque tracking errors for the stance knee. We are not able to walk for a longer distance due to limited lab space. We do not think this is the absolute limit for Atlas’ walking speed, since a more powerful pump (potentially offboard) can easily solve this issue. On the other hand, our current implementation requires large knee torques since it maintains a bent stance knee to avoid singularities. Walking with straighter knees will reduce the power requirement, and we might be able to push the speed limit further.

IV. DISCUSSION AND FUTURE WORK

Once the foot placement controller is implemented, minimal tuning is necessary for the reset of the system for successful robot deployment, which is somewhat surprising. From our DRC experiences, precise control of the CoM state is critical for a mostly static walking controller. This requires accurate and low delay state estimation as well as good CoP and force control at the contacts, which all require extensive tuning on the hardware. Correctly estimating the
CoM modeling errors [29] also plays an important role. In contrast, the control authority through foot placement is magnitudes bigger than controlling the CoP for dynamic walking. We no longer require fine control over the CoM states, which greatly reduces performance requirements for the low level controllers and state estimators. When planned in a receding-horizon fashion, foot placement does not have to be very precise as long as the robot can roughly capture itself in the next step. Even without explicitly formulated, the multi-step recovery strategy emerges naturally with the current implementation. Dynamic walking is much easier in the sense that its error tolerance for all the individual components is much larger than for static walking. On the other hand, timing is important. Our early work [30] shows combining foot placement and step timing can greatly improve the stability margin, and we want to implement this in the near future.

Simple models are easy to understand and provide us with important intuitions, and they are also powerful tools for analysis. Using simple models imposes structures on the underlying problem and reduces the search space. Although
the current hierarchical architecture is mostly motivated by computational costs, we think planning with simple models can still be beneficial even with unlimited computing power because of the simplicity and added structures. On the other hand, due to their limited expressive powers, planners using simple models either generate potentially infeasible plans or become overly conservative. We still need to find a better balance between model complexity and performance.

V. CONCLUSION

The previous walking controller based on CoM trajectory optimization and full body inverse dynamics is complemented by a receding-horizon component that rapidly optimizes foot placement in this work. With the complete controller, our Atlas can withstand large external pushes, walk over unstructured terrain made by loose rubble, and achieve a top speed of 0.6m/s for flat ground walking.

ACKNOWLEDGEMENT

This material is based upon work supported in part by the DARPA Robotics Challenge program under DRC Contract No. HR0011-14-C-0011 and the US National Science Foundation (ECCS-0824077 and IIS-0964581).

REFERENCES


