2013 13th IEEE-RAS International Conference on
Humanoid Robots (Humanoids).
October 15 - 17, 2013. Atlanta, GA

3D Walking Based on Online Optimization

Siyuan Feng, X Xinjilefu, Weiwei Huang and Christopher G. Atkeson

Abstract— **We present an optimization based real-time walking controller for a full size humanoid robot. The controller consists of two levels of optimization, a high level trajectory optimizer that reasons about center of mass and swing foot trajectories, and a low level controller that tracks those trajectories by solving a floating base full body inverse dynamics problem using Quadratic Programming. Our controller is capable of walking on rough terrain, and also achieves longer foot steps, faster walking speed, heel-strike and toe push-off. Results are demonstrated with Boston Dynamics' Atlas robot in simulation.**

I. Introduction

We consider a generalized walking controller that works on uneven terrain is essential to enabling humanoid robots to walk outside labs and be actually useful. Additionally, we argue that the controller needs to pay attention to a set of foot steps provided by either a planner or human operator. This is important in obstacle avoidance, rough terrain traversal, and walking in a cluttered indoor environment. Although recent research in hand-crafted policy based walking controllers [1], [2], [3] has shown robust performance on uneven terrain and against external perturbations, they are essentially walking blindly and are fundamentally incompatible with the requirement described above. On the other hand, traditional Zero Moment Point (ZMP) based designs [4], [5] have yet to show convincing performance on uneven terrain. We are proposing a walking controller based on online optimization that handles uneven terrain. Figure 2 shows a simulated Atlas robot using our controller to traverse rough terrain in DARPA's Virtual Robotics Challenge.

Our approach is rooted in model-based optimal control, as it takes a desired sequence of foot steps, uses optimization to generate a locally optimal trajectory, then tracks this trajectory using inverse dynamics. The high level controller performs online trajectory optimization using Differential Dynamic Programming (DDP)[6] with a simplified model that only reasons about the center of mass (COM) of the robot. We also use a quintic spline in Cartesian space to smoothly connect the given foot steps for the swing foot. The low level controller takes these trajectories as inputs and uses a Quadratic Programming (QP) based inverse dynamics solver to generate torques for all the joints. A schematic of the system is depicted in Figure 1.

The high level controller is similar in spirit to Preview Control proposed by Kajita et al. [4] in the sense that we are also using a COM model, reasoning about ZMP and using future information to guide the current trajectory. But our approach can be easily generalized to more complex nonlinear models and adjust foot steps while optimizing the COM trajectory. We do not explicitly generate inverse kinematics solutions or use high gain position controls to track joint trajectories, which results in compliant walking. We explicitly add the z dimension in our COM model to handle height variations on rough terrain. Like capture point methods [7], we take the next few steps into consideration during trajectory optimization, although we do not plan to come to rest at the end. Ogura et al. [8] investigated generating human like walking with heel-strike and toe-off by parametrizing the swing foot trajectory, and using a genetic algorithm to find the parameters. In contrast, we use very simple rules to guide the low level controller to achieve the same behaviors.

In the low level controller, we formulate the floating base inverse dynamics as a Quadratic Programming problem. We continue to use a formulation previously developed in our group [9], [10]. Unlike [11], [12] that use orthogonal decomposition to project the allowable motions into the null space of constraint Jacobian, and minimize any combination of linear and quadratic costs in the contact constraints and the commands, we directly optimize a quadratic cost in terms of state accelerations, torques and contact forces. We are also able to directly reason about inequality constraints such as center of pressure within the support polygon, friction and torque limits. Although it becomes a bigger QP problem, we are still able to solve it in real time. Hutter el al. [13] resolved redundancy in inverse dynamics using a kinematic task prioritization approach that ensures lower priority tasks always exist in the null space of higher priority ones. In contrast to their strictly hierarchical approach, we minimize a sum of weighted terms. We can directly specify the relative importance by adjusting the weights.

II. Trajectory generation

Given a sequence of desired foot steps, we assign uniform timing to them. We then plan a COM trajectory that minimizes stance foot ankle torque with DDP, which is an iterative trajectory optimization technique that updates the current control signals based on the spatial derivatives of the value function, and uses the updated controls to generate a trajectory for the next iteration. The swing foot trajectory is generated using a quintic spline between the starting and ending positions. For foot orientation, we take the yaw angle specified in the foot step sequence, assume 0 roll angle, and estimate the pitch angle by relative height change from consecutive foot steps. Body orientation at the end of the swing phase is computed by averaging the yaw

All the authors are with The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, {sfeng, xxinjile, cga}@cs.cmu.edu, huangwei@andrew.cmu.edu

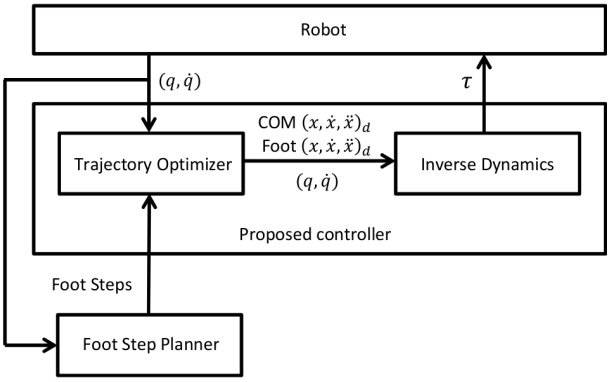


Fig. 1. The proposed controller consists of a trajectory optimizer and inverse dynamics solver. The former takes robot state $q, \dot{q}$ from a state estimator and desired foot steps from a planner, and produces desired COM and swing foot trajectories. The latter takes these trajectories and $q, \dot{q}$ and solves floating base full body inverse dynamics to generate torques.

angles from consecutive foot steps, assuming 0 roll angle, and a task specific pitch angle (e.g. leaning forward when climbing a steep ramp). Both body and foot orientation are represented as quaternions, and interpolated using spherical linear interpolation (slerp).

In the current implementation of the high level controller, we approximate the entire robot as a point mass, without considering angular momentum. The dynamics for the simple model are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}_t = \begin{bmatrix} \frac{(x_t - p_x)F_z}{mz_t} \\ \frac{(y_t - p_y)F_z}{mz_t} \\ \frac{F_z}{m} - g \end{bmatrix}.$$

The state, $X = (x, y, z, \dot{x}, \dot{y}, \dot{z})$, is the location and velocity of the center of mass. The control $u = (p_x, p_y, F_z)$ is the commanded center of pressure and force in the z direction. The current high level controller does not know about step length limits, and we are relying on the foot step planner to give us reasonable foot steps.

A. Differential Dynamic Programming

DDP applies Dynamic Programming along a trajectory. It can find globally optimal trajectories for problems with time-varying linear dynamics and quadratic costs, and rapidly converge to local optimal trajectories for problems with nonlinear dynamics or costs. This approach modifies (and complements) existing approximate Dynamics Programming approaches in these ways: 1) We approximate the value function and policy using many local models (quadratic for the value function, linear for the policy) along the trajectory. 2) We use trajectory optimization to directly optimize the sequence of commands $u_{0,N-1}$ and states $X_{0,N}$. 3) Refined local models of the value function and policy are created as a byproduct of our trajectory optimization process.

We represent value functions and policies using Taylor series approximations at each time step along a trajectory. For a state X^p, the local quadratic model for the value function

Fig. 2. The proposed controller tested in the Rough Terrain Task in DARPA's Virtual Robotics Challenge.

is

$$V^p(X) \approx V_0^p + V_X^p(X - X^p) + \frac{1}{2}(X - X^p)^T V_{XX}^p (X - X^p),$$

where X is some query state, V_0^p is the constant term, V_X^p is the first order gradient with respect to state evaluated at X^p, and V_{XX}^p is the second order spatial gradient evaluated at X^p. The local linear policy is

$$u^p(X) = u_0^p - K^p(X - X^p),$$

where u_0^p is a constant term, and K^p is the first derivative of the local policy with respect to state evaluated at X^p, which is also a gain matrix for a local linear controller. V_0, V_X, V_{XX} and K are stored along with the trajectory.

The one step cost function is

$$L(X, u) = 0.5(X - X^*)^T Q(X - X^*) + 0.5(u - u^*)^T R(u - u^*),$$

where R is positive definite, and Q is positive semi-definite. X^* is given as a square wave, instantly switching to the next foot step location and staying there for the entire stance with velocities equal zero. u^* is specified in a similar way with p_x and p_y being at the desired center of pressure in the world frame, and $F_z = mg$.

For each iteration of DDP, we propagate the spatial derivatives of the value function V_{XX} and V_X backward in time, and use this information to compute an update du to

the control signal. Then we perform a forward integration pass using the updated controls to generate a new trajectory. Although we are performing nonlinear trajectory optimization, due to analytical gradients of the dynamics, this process is fast enough in an online setting.

B. Initialization

Given the last desired center of mass location and desired center of pressure, (X^*, u^*) in the foot step sequence, we first compute a Linear Quadratic Regulator (LQR) solution at that point, and use its policy to generate an initial trajectory from the initial X_0. V_{XX} of this LQR solution is also used to initialize the backward pass.

C. Backward pass

Given a trajectory, one can integrate the value function and its first and second spatial derivatives backwards in time to compute an improved value function and policy. We utilize the "Q function" notation from reinforcement learning: $Q^t(X,u) = V^{t+1}(f(X,u)) + L^t(X,u)$ where t is the time index. The backward pass of DDP can be expressed as

$$\begin{aligned}
Q^t_X &= L^t_X + V^t_X f^t_X \\
Q^t_u &= L^t_u + V^t_X f^t_u \\
Q^t_{XX} &= L^t_{XX} + V^t_X f^t_{XX} + f^{tT}_X V^t_{XX} f^t_X \\
Q^t_{uX} &= L^t_{uX} + V^t_X f^t_{uX} + f^{tT}_u V^t_{XX} f^t_X \\
Q^t_{uu} &= L^t_{uu} + V^t_X f^t_{uu} + f^{tT}_u V^t_{XX} f^t_u \\
K^t &= (Q^t_{uu})^{-1} Q^t_{uX} \\
\delta u^t &= (Q^t_{uu})^{-1} Q^t_u \\
V^{t-1}_X &= Q^t_X - Q^t_X K^t \\
V^{t-1}_{XX} &= Q^t_{XX} - Q^t_{Xu} K^t.
\end{aligned}$$

Derivatives are taken with respect to the subscripts, and evaluated at (X, u).

D. Forward pass

Once we have computed the local linear feedback policy K^t and updates for controls δu^t, we integrate forward in time using

$$u^t_{new} = (u^t - \delta u^t) - K^t(X^t_{new} - X^t)$$

with $X^{t0}_{new} = X_0$. We terminate DDP when the cost-to-go at X_0 does not change much across iterations. This approach can be thought of as a generalized version of Kajita's preview control [4]. Figure 3 shows trajectories of COM before and after DDP optimization. The original trajectories are plotted with dashed lines, and results are plotted with solid lines. State trajectories are shown in blue, and controls are in green. In the plots, F_z is normalized by body weight.

E. Double support

A short double support phase is planned to allow weight shifting and explicit controls on the touch down foot and lift off foot.

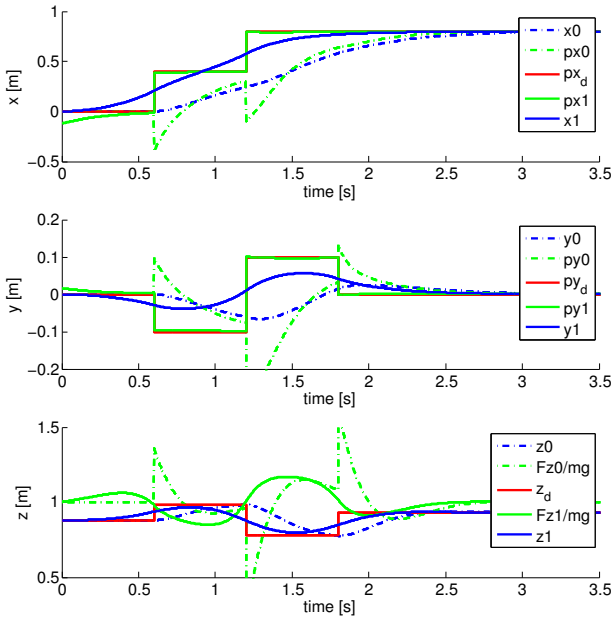


Fig. 3. Desired center of pressure in XY plane and COM height trajectories are plotted with solid red lines. Initial trajectories before DDP optimization are plotted in dashed lines, and optimized ones are shown in solid lines. States x, y and z are shown in blue, and controls px, py and $\frac{F_z}{mg}$ are plotted in green.

1) Heel-strike: We command the landing foot's orientation to have a pitch angle and allowed the center of pressure to move quickly from the heel to somewhere in the middle of the foot.

2) Toe push-off: During the lift off phase, we move the foot reference point to the toe, and instead of commanding the angular acceleration to be zero, we request a large pitch angular acceleration in the foot frame, set the desired center of pressure to be at the toe, and let the low level controller handle the rest. A more elegant approach is perhaps eliminating the row in A_{cart} and b_{cart} in (1) that corresponds to pitch angular acceleration entirely and let QP figure out the rest.

III. Full Body Inverse Dynamics

For the low level control, we use an inverse dynamics approach to solve for torques that would best satisfy the given desired motion. We formulate the full body floating base inverse dynamics problem as a Quadratic Programming (QP) problem, and apply a standard QP solver at each time step to generate a set of torque commands.

A. QP formulation

The equations of motion and constraints equations for a floating base humanoid robot can be described as

$$\begin{aligned}
M(q)\ddot{q} + h(q,\dot{q}) &= S\tau + J^T(q)F \\
J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} &= \ddot{x}.
\end{aligned}$$

$(q, \dot{q})$ is the full state of the system including a 6 DOF joint at the floating base. $M(q)$ is the inertia matrix. $h(q,\dot{q})$ is the sum of gravitational, centrifugal and Coriolis forces. S is a

selection matrix that is the identity matrix except for the first 6 degrees of freedom that corresponds to the floating base. τ is the joint torque vector. $J^T(q)$ is the stacked Jacobian matrix for all the contacts. F is the vector of all stacked contact forces. x is the contact location and orientation in Cartesian space. The sizes of F and J^T change depends on the number of contacts.

We can rewrite the equations of motion as

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ F \end{bmatrix} + h(q, \dot{q}) = 0.$$

Given a state $(q, \dot{q})$, the equations of motion are linear in terms of $\begin{bmatrix} \ddot{q} & \tau & F \end{bmatrix}^T$.

Let $\mathcal{X} = \begin{bmatrix} \ddot{q} & \tau & F \end{bmatrix}^T$. We can now formulate the inverse dynamics as a Quadratic Programming problem

$$\begin{aligned} \min_{\mathcal{X}} \ & \mathcal{X}^T G \mathcal{X} + g^T \mathcal{X} \\ s.t. \ & CE\mathcal{X} + ce^T = 0 \\ & CI\mathcal{X} + ci^T >= 0. \end{aligned}$$

The equality constraints for this QP problem are the equations of motions described above, and the inequality constraints consist of various terms such as joint torque limits and ground force limits due to friction cone and center of pressure remaining in the support polygons. The optimization criteria can be thought as a big least square minimization problem penalizing $\mathcal{X}$ for deviating from some desired $\mathcal{X}^*$.

B. QP optimization criteria

Essentially, we are optimizing a cost function of the form $\|A\mathcal{X} - b\|^2$, where $\mathcal{X} = \begin{bmatrix} \ddot{q} & \tau & F \end{bmatrix}^T$. A and b can be broken down row-wise into smaller blocks, each emphasizing certain desired behaviors with different weights. We will present a few concrete examples.

1) Cartesian space accelerations: Since

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q, \dot{q})\dot{q},$$

we can penalize deviation from desired Cartesian space accelerations using

$$\begin{aligned} A_{cart} &= \begin{bmatrix} J(q) & 0 & 0 \end{bmatrix} \\ b_{cart} &= \ddot{x}^* - \dot{J}(q, \dot{q})\dot{q}. \end{aligned} \tag{1}$$

This is useful for task level controls such as COM, swing foot or hand tracking. For COM and swing foot, the desired accelerations are specified by the high level trajectory optimizer described in Section II. For the stance foot, the desired accelerations are set to zero. Rather than treating the stance foot accelerations as hard constraints, we find that using a soft penalty with high weights is generally more stable and faster to solve.

2) Net external force and center of mass motion: The relationship between COM linear acceleration and net external force is

$$F_{lin} = m(\ddot{x}_{com} + \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T),$$

where m is the total mass, and g is the gravitational acceleration. Net torque and change in angular momentum $\dot{L}$ are related in a similar way

$$F_{ang} + (x_{contact} - x_{com}) \times F_{lin} = I\dot{\omega}_{com},$$

where I is the total moment of inertia, and $\dot{\omega}_{com}$ is the angular acceleration. We can compute

$$\begin{aligned} A_{comF} &= \begin{bmatrix} 0 & 0 & \begin{bmatrix} I_{3\times3} & 0 \end{bmatrix} \end{bmatrix} \\ b_{comF} &= m(\ddot{x}^*_{com} + \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T) \end{aligned}$$

and

$$\begin{aligned} A_{comTau} &= \begin{bmatrix} 0 & 0 & \begin{bmatrix} Cross(x_{contact}, x_{com}) & I_{3\times3} \end{bmatrix} \end{bmatrix} \\ b_{comTau} &= I\dot{\omega}^*_{com}. \end{aligned}$$

$I_{3\times3}$ is a 3 by 3 identity matrix. $\ddot{x}^*_{com}$ and $\dot{\omega}^*_{com}$ are desired linear and angular accelerations. $Cross(x_{contact}, x_{com})$ is the matrix representing a cross product between $(x_{contact} - x_{com})$ and F_{lin}.

3) Center of pressure tracking: Given the forces and torques, ${}^bM, {}^bF$, specified in foot frame, the location of the center of pressure in the foot frame is

$$p = \begin{bmatrix} -{}^bM_y / {}^bF_z \\ {}^bM_x / {}^bF_z \end{bmatrix}.$$

We can penalize center of pressure deviation with

$$\begin{aligned} A_{cop} &= \begin{bmatrix} 0 & 0 & \begin{bmatrix} 0 & 0 & p^*_x & 0 & 1 & 0 \\ 0 & 0 & p^*_y & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \end{bmatrix} \\ b_{cop} &= 0, \end{aligned}$$

where (p^*_x, p^*_y) is the desired center of pressure in foot frame, and R is the rotation matrix from the world frame to the foot frame.

4) Weight distribution: In double support, it is often desirable to specify the desired weight distribution $w^* = F_{zl}/(F_{zl} + F_{zr})$. We add this term to the cost function with

$$\begin{aligned} A_w &= \begin{bmatrix} 0 & 0 & I_w \end{bmatrix} \\ b_w &= 0, \end{aligned}$$

where I_w is a row vector with zeros, except $I_w(3) = 1 - w^*$ and $I_w(9) = -w^*$.

5) Direct tracking and regularization: We can also directly penalize $\mathcal{X}$ from desired values with

$$\begin{aligned} A_{state} &= \begin{bmatrix} I & I & I \end{bmatrix} \\ b_{state} &= \begin{bmatrix} \ddot{q}^* & 0 & 0 \end{bmatrix}^T, \end{aligned}$$

where $\ddot{q}^*$ is the desired joint accelerations. This is useful for directly controlling the upper body joints, as well as regularizing $\mathcal{X}$ to make the QP problem well conditioned.

The final cost function uses the vertically concatenated As and bs each multiplied with their corresponding weights, ws.

$$A = \begin{bmatrix} w_{cart}A_{cart} \\ w_{comF}A_{comF} \\ \vdots \end{bmatrix}, b = \begin{bmatrix} w_{cart}b_{cart} \\ w_{comF}b_{comF} \\ \vdots \end{bmatrix}$$

Weights are summarized in Table I. w_{qdd} is for joint acceleration. w_{com} and $w_{utorsowd}$ are for COM position

TABLE I

WEIGHTS FOR QP COST FUNCTION

w_{qdd}	w_{comdd}	$w_{utorsowd}$	w_{footdd}
10^{-1}	6×10^{-1}	1	1
w_{comF}	w_{comTau}	w_{regF}	w_{regTau}
10^{-2}	10^{-2}	10^{-6}	10^{-6}
w_w	w_{cop}	w_{Fd}	w_{Taud}
10^{-3}	8×10^{-3}	10^{-3}	10^{-1}

acceleration and upper torso orientation acceleration. w_{footdd} is for foot position and orientation acceleration. w_{comF} and w_{comTau} are for net external force and torque. w_{regF} and w_{regTau} are regularization weights for contact force and joint torques. w_w is for weight distribution. w_{cop} is for center of pressure. w_{Fd} and w_{Taud} penalize changes in contact force and joint torques between two consecutive time steps.

C. Constraints

Torque limits can be easily added into the inequality constraints. Friction constraints are approximated by

$$|^bF_x| \leq \mu^bF_z$$
$$|^bF_y| \leq \mu^bF_z.$$

The center of pressure also has to be under the feet, which can be written as

$$d_x^- \leq -^bM_y/^bF_z \leq d_x^+$$
$$d_y^- \leq^b M_x/^bF_z \leq d_y^+,$$

where ^bF and ^bM denotes forces and torques in the foot frame, and d^- and d^+ are the sizes of foot. Equations of motion are treated as equality constraints in the QP problem.

IV. IMPLEMENTATION

A. Trajectory replanning

The high level trajectory planner replans whenever a new step is taken. It optimizes a COM trajectory for 3 consecutive steps, although we will only use the trajectory that corresponds to the first step. Because sometimes DDP cannot finish in one time step ($1ms$), we run DDP for the next 3 foot steps on a separate thread, and use the trajectory that was computed most recently in the low level controller. As for runtime, DDP completes within 1 to 50ms (depending on terrain complexity), and QP finishes within $0.5ms$.

B. Desired acceleration for the low level controller

We store the local linear policies in addition to the COM trajectory, and use them to compute the desired COM accelerations as an input to the low level controller. Since we have a spline representation (quintic for position, and slerp for orientation) for the swing foot trajectory, we can compute its desired acceleration with

$$\ddot{x}_{swing} = -K_p(x - x^*) - K_d(\dot{x} - \dot{x}^*) + \ddot{x}^*,$$

where x^*, $\dot{x}^*$, $\ddot{x}^*$ are on the spline. Upper body desired joint accelerations are also computed in a similar PD-servo fashion.

C. Body orientation tracking

In practice, we discover that controlling upper torso orientation is better than directly controlling pelvis orientation because this allows the low level controller to use the spine joints in addition to the leg joints. Upper torso orientation tracking is also done with a Jacobian similar to COM position tracking.

D. Foot step orientation

We only require foot step yaw angle from the foot step planner. To generate a desired foot step orientation, we use the specified yaw angle, assume 0 roll angle, and estimate the pitch angle by relative height change from consecutive foot steps. On rough terrain, orientation tracking is disabled on the landing foot once we detect a substantial F_z on the force sensor.

E. Stance leg damping

In DARPA's Virtual Robotics Challenge, rough terrain is represented by a triangular mesh. If the robot steps onto a pointed region, the foot contact geometry no longer remains rectangular, and the foot will "rock" on the "spike". We handle this problem by having explicit joint damping on the stance leg, and adding a term to the QP's cost function that penalizes large torque changes from previous time step.

F. Joint limits

When some joint is close to its limit, we command a large acceleration in the other direction to push it back with

$$\ddot{q}^* = -10 \tan\left(\pi\left(\frac{q - q_{lower}}{q_{upper} - q_{lower}} - 0.5\right)\right).$$

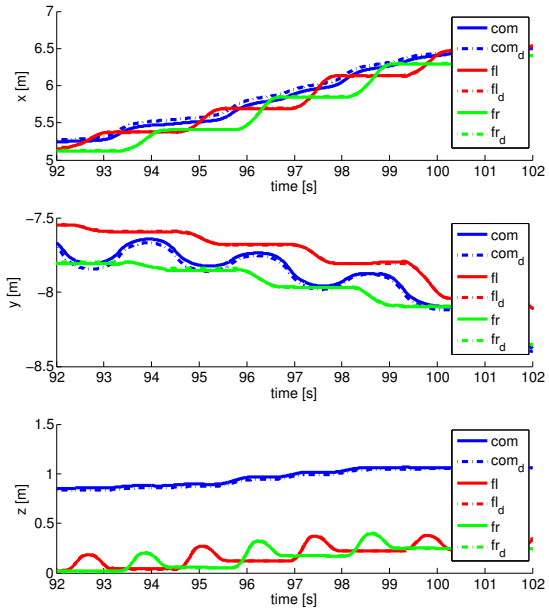
G. State estimation

The proposed controller needs estimates of the root position, velocity, orientation, angular velocity and all the joint positions and velocities. All this information is gathered by several state estimators, which are discussed in depth in [14].

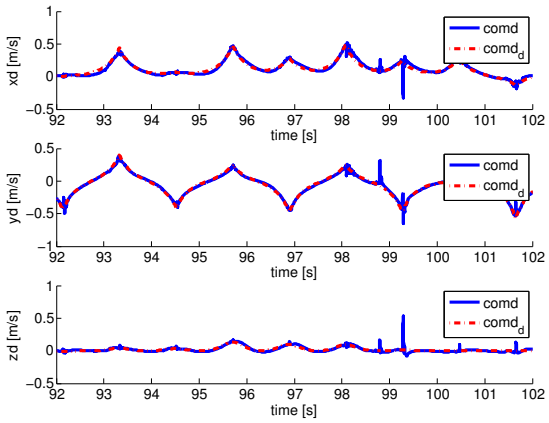
V. RESULTS

The proposed walking controller is tested on a simulated Boston Dynamics' Atlas robot in DARPA's Virtual Robotics Challenge setting. The Gazebo simulator is produced by the Open Source Robotics Foundation. We run the simulation at real time on a computer with Intel Xeon(R) CPU E5-2687W CPU with 32G memory. This setup is necessary for simulating Atlas in the Gazebo simulator at real time alone. Our proposed controller uses two threads. One thread is dedicated to the QP solver running at $1KHz$, and the other computes COM trajectory only when the robot has taken a step. The QP solver finishes within $0.5ms$, and trajectory optimizer can take up to $50ms$ depends on the given foot steps.
[1]

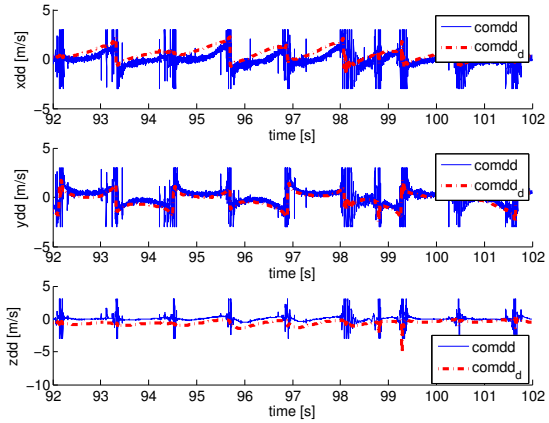
[1]Video showing walking using our proposed controller: `http://www.youtube.com/watch?v=lP52k_ZkOHM`



(a) COM and foot position tracking on rough terrain



(b) COM velocity tracking on rough terrain



(c) COM acceleration tracking on rough terrain

Fig. 4. All actual traces are plotted in solid lines, and desired traces are shown in dashed lines. In 4(a), left foot traces are shown in red lines, right foot in green, and COM are plotted in blue. The reference foot points are set to the heel. 4(b) shows velocity tracking of COM. In 4(c), actual COM acceleration are computed by finite differencing the velocity trace, and capped at $\pm 3m/s^2$.

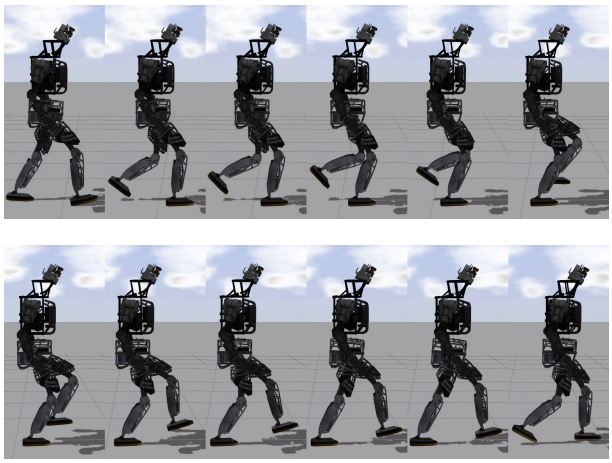


Fig. 5. Snapshots taken for walking on flat ground with $0.7m$ step length, and $0.8s$ period.

A. *Flat ground walking*

We are able to demonstrate toe push-off and heel-strike behavior on flat floor walking. We have achieved a maximum step length of $0.8m$. Figure 5 shows a sequence of snapshots taken for walking with $0.7m$ step length, and $0.8s$ period on flat ground. The maximum speed we have achieved is on average $1.14m/s$ with $0.8m$ step and $0.7s$ period on flat ground.

B. *Rough terrain*

The proposed controller can handle up to 0.4 rad inclined slopes, and continuously climb stair steps that are $0.2m$ high and $0.4m$ apart. We are able to successfully walk on the rough terrain environment provided in the Virtual Robotics Challenge as well. In order to traverse rough terrain, an A* planner [15] is used to provide sequences of foot steps, which are given as input to the proposed controller. The high level controller takes the foot steps and generates desired foot and COM trajectories for the low level controller to track. Figure 4 shows tracking performance for a segment of the experiment.

VI. Discussion and Future Work

The most important contribution of this work is demonstrating that we can perform online trajectory optimization and inverse dynamics in real time to handle rough terrain and achieve close to human like step length and speed on a 3D full size humanoid robot. Reasoning about COM motion alone is enough to guide the inverse dynamics controller, which is a greedy optimizer, through moderate rough terrain with height and surface normal changes. Another important point is that our approach does not impose a predefined criteria on the set of allowable foot steps since trajectories are optimized online. Control and foot step planning can be treated as almost completely independent problems. This feature allows us to develop the foot step planner separately. Another interesting discovery was explicitly adding toe push-off and heel-strike between single stance phases. We have

shown that, with a very simple-minded design, we can achieve these behaviors and push step length and walking speed closer to those of humans. We will further explore this issue in future work.

For flat ground walking, the proposed controller is maintaining constant COM height and torso pitch angle throughout the entire walking cycle, which is undesirable for achieving higher speed and larger step length. We think the point mass model is also limiting performance since it ignores angular momentum and swing leg dynamics completely. The current trajectory optimizer implementation uses less than 10% of its allowable computation time. An immediate line of research is to expand the COM model to a more sophisticated one. We can include angular momentum and orientation [16], switch to a 3D Spring Loaded Inverted Pendulum (SLIP) model [17] to capture natural arc like motion observed in human walking, or add a telescoping leg to capture some simple notion of stance leg configuration [18]. Another direction would involve simultaneously optimizing many trajectories. Apart from changing models, we can also take the foot steps (location and timing) as part of the optimization. These choices can be treated as parameters in DDP, and be optimized along with the trajectory. We imagine the foot step planner will hand us foot steps along with viable regions, and the trajectory optimizer will have the freedom to pick the foot steps that are most suitable. This will be one way to have the planner implicitly taking robot dynamics into consideration.

Our low level controller only greedily optimizes for the current time step given a set of desired values to track. We could potentially add some form of a value function as part of the optimizing criteria to incorporate a notion of the future in the low level controller. This value function can come from performing full-blown DDP on a periodic walking pattern [19], or once again from the trajectory optimization part.

Our biggest concern in applying this approach to real hardware is handling modeling error. One option is to adopt a multiple model approach [20] during optimization and simultaneously consider a distribution of models.

VII. Conclusions

We have designed a real time walking controller for a 3D full size humanoid robot with online optimization. The low level controller solves full body floating base inverse dynamics by formulating it as a Quadratic Programming problem. The high level controller guides it with trajectories that are optimized online using simplified models of the robot. We have successfully applied this approach to both flat ground and rough terrain walking, which is demonstrated using a simulated Boston Dynamics' Atlas robot. We have also achieved more human like foot step length and walking speed by explicitly adding toe push-off and heel-strike in a very simple way.

Acknowledgement

This material is based upon work supported in part by the US National Science Foundation (ECCS-0824077, and IIS-0964581) and the DARPA M3 and Robotics Challenge programs. Special thanks to Eric Whitman for his help in implementation and enlightening discussions.

References

[1] S. Song and H. Geyer, "Generalization of a muscle-reflex control model to 3d walking," in *IEEE Engineering in Medicine and Biology Society (EMBC'13)*, 2013.

[2] J. M. Wang, S. R. Hamner, S. L. Delp, and V. Koltun, "Optimizing locomotion controllers using biologically-based actuators and objectives," in *ACM Transactions on Graphics*, 2012, pp. Vol. 31, No. 4, Article 25.

[3] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers for uncertain inputs and environments," in *ACM Transactions on Graphics*, 2010, pp. Vol. 29, No. 4, Article 73.

[4] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, 2003, pp. 1620–1626 vol.2.

[5] T. Takubo, Y. Imada, K. Ohara, Y. Mae, and T. Arai, "Rough terrain walking for bipedal robot by using ZMP criteria map," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, 2009, pp. 788–793.

[6] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. Elsevier, 1970.

[7] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, 2006, pp. 200–207.

[8] Y. Ogura, K. Shimomura, H. Kondo, A. Morishima, T. Okubo, S. Momoki, H. ok Lim, and A. Takanishi, "Human-like walking with knee stretched, heel-contact and toe-off motion by a humanoid robot," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 3976–3981.

[9] B. Stephens, "Push recovery control for force-controlled humanoid robots," Ph.D. dissertation, The Robotics Institute, Carnegie Mellon University, Pittsburgh, August 2011.

[10] E. Whitman and C. Atkeson, "Control of instantaneously coupled systems applied to humanoid walking," in *Humanoid Robots (Humanoids), IEEE-RAS International Conference on*, 2010, pp. 210–217.

[11] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 3406–3412.

[12] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse dynamics control," *International Journal of Robotics Research*, pp. 280–298, 2013.

[13] M. Hutter, M. Hoepflinger, C. Gehring, C. D. R. M. Bloesch, and R. Siegwart, "Hybrid operational space control for compliant legged systems," in *Proc. of the 8th Robotics: Science and Systems Conference (RSS)*, 2012.

[14] X. Xinjilefu, S. Feng, W. Huang, and C. Atkeson, "Decoupled state estimation for humanoid using full-body dynamics," in *International Conference on Robotics and Automation*, 2014, p. submission.

[15] W. Huang, S. Feng, X. Xinjilefu, and C. Atkeson, "Autonomous path planning of a humanoid robot on unknown rough terrain," in *International Conference on Robotics and Automation*, 2014, p. submission.

[16] E. C. Whitman, B. J. Stephens, and C. G. Atkeson, "Torso rotation for push recovery using a simple change of variables," in *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[17] A. Wu and H. Geyer, "The 3-D spring-mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments," *Robotics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–11, 2013.

[18] P. A. Bhounsule, "Gait planning and control of a compass gait walker based on energy regulation using ankle push-off and foot placement," in *International Conference on Robotics and Automation*, 2014, p. submission.

[19] C. Liu, C. G. Atkeson, and J. Su, "Biped walking control using a trajectory library," *Robotica*, 2012.

[20] E. Whitman and C. Atkeson, "Multiple model robust dynamic programming," in *American Control Conference (ACC)*, 2012, pp. 5998–6004.