Optimization-based Full Body Control for the DARPA Robotics Challenge

Siyuan Feng, Eric Whitman, X. Xinjilefu, and Christopher G. Atkeson
Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania 15213
e-mail: sfeng@cs.cmu.edu, ewhitman@cs.cmu.edu, xxinjile@cs.cmu.edu

Received 8 March 2014; accepted 31 October 2014

We describe our full body humanoid control approach developed for the simulation phase of the DARPA Robotics Challenge (DRC), as well as the modifications made for the DARPA Robotics Challenge Trials. We worked with the Boston Dynamics Atlas robot. Our approach was initially targeted at walking, and it consisted 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 floating base full body inverse dynamics using quadratic programming. This controller is capable of walking on rough terrain, and it also achieves long footsteps, fast walking speeds, and heel-strike and toe-off in simulation. During development of these and other whole body tasks on the physical robot, we introduced an additional optimization component in the low-level controller, namely an inverse kinematics controller. Modeling and torque measurement errors and hardware features of the Atlas robot led us to this three-part approach, which was applied to three tasks in the DRC Trials in December 2013. © 2014 Wiley Periodicals, Inc.

1. INTRODUCTION

Originally targeted at rough terrain bipedal walking, we developed a walking control approach that can achieve a sequence of footstep targets and walk fast on level ground with no obstacles. Our approach is rooted in model-based optimal control, as it takes a desired sequence of foot steps, uses optimization to generate a trajectory for the center of mass (CoM), and then tracks this trajectory using inverse dynamics (ID) and inverse kinematics (IK). The controller consists of two levels. The high-level controller performs online trajectory optimization using differential dynamic programming (Jacobson & Mayne, 1970) with a simplified model that only reasons about the center of mass of the robot. We also use a quintic spline in Cartesian space to smoothly connect two consecutive foot steps for the swing foot. The low-level controller was originally designed to use inverse dynamics alone, and we added an inverse kinematics component to cope with modeling error when controlling the physical robot. For the DARPA Robotics Challenge (DRC) Trials, we redesigned the high-level controller to also handle ladder climbing and full body manipulation, and we successfully applied our approach to these problems as well. The two-stage optimization-based architecture separates the behavior level design process and full body control problem cleanly, and it offers us a versatile and powerful platform for rapidly developing multiple applications for the DRC. Figure 1 shows a diagram of our approach.

Direct correspondence to: Christopher G. Atkeson, cga@cmu.edu

2. RELATED WORK

For walking, the relationship between net ground reaction force and center of mass motion has been well studied over the past 50 years (Vukobratović & Borovac, 2004), and it has been widely used in gait generation. Orin & Goswami (2008) connected both linear and angular momentum, generalized velocity of the system, and net external wrench by introducing the centroidal momentum matrix, which inspired many of the later high-level controllers for balancing and walking. Our high-level controller for dynamic walking is similar in spirit to preview control proposed by Kajita et al. (2003) in the sense that we use a CoM model, reason about zero moment point (ZMP), and use future information to guide the current trajectory. Our approach can be easily generalized to nonlinear models, as opposed to the linear inverted pendulum model (LIPM) used in preview control. We explicitly add the vertical dimension in our CoM model to handle height variations on rough terrain. Like capture point methods (Pratt, Carff, Drakunov, & Goswami, 2006), we take the next few steps into consideration during trajectory optimization, although we do not plan to come to rest at the end. With a stronger emphasis on robustness, Urata et al. (2012) and Faraji, Pouya, & Ijspeert (2014) demonstrated three-dimensional (3D) push recovery and walking by reoptimizing foot step locations in a receding horizon fashion using LIPM dynamics. Although foot step locations can also be optimized during the our process, we currently focus on achieving a fixed sequence of desired foot steps.

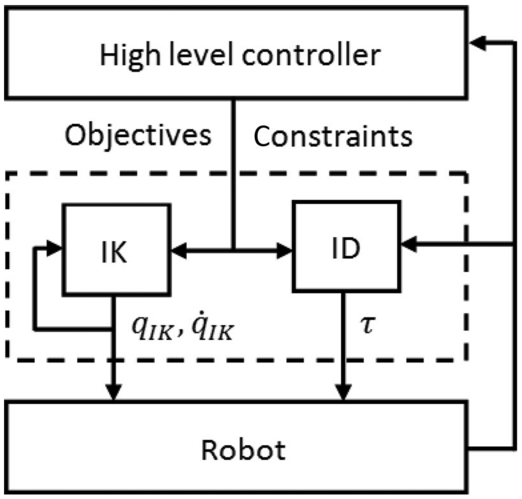


Figure 1. The task-dependent high-level controller generates a set of desired objectives such as center of mass or limb motion, and constraints such as center of pressure, friction, and joint limits. The generic high-level controller's input can range from a foot step sequence to a pregrasp pose or even operator commands depending on the specific application. The low-level full body controller, which is enclosed by the dashed rectangle, takes the high-level objectives and robot states $(q, \dot{q})$ as inputs and outputs of the desired position q_{IK}, velocity $\dot{q}_{\mathrm{IK}}$, and torque τ for each joint. Note that inverse kinematics uses its own internal states rather than the measured robot states.

Using full body inverse dynamics for force control has become a popular topic in recent humanoid research. This direction of research originated from Khatib (1987). Within this broad category, control designers can directly specify reference motions in task space, then rely on using convex optimization to handle constraints and solve for controls that best track the reference motions. Although detailed formulations differ, most active research has converged to formulating the floating base inverse dynamics as a quadratic programming (QP) problem. Hutter et al. (2012), Hutter et al. (2014), Herzog, Righetti, Grimminger, Pastor, & Schaal (2014), Saab et al. (2013), de Lasa et al. (2010), Escande et al. (2014), and Wensing & Orin (2013a) explored using a hierarchical approach to resolve redundant degrees of freedom in humanoid robots. These approaches typically ensure low priority objectives are within the null space of higher priority ones. A solution to resolve hierarchical quadratic programs presented in Escande et al. (2014) is more general and significantly faster than previous methods (Kanoun, Lamiraux, & Wieber, 2011). Although the method is currently applied to solve inverse kinematics, the authors claim it is also applicable to inverse dynamics. A hierarchical framework designed for humanoid robots to handle constraints and objectives is presented in Sentis, Park, & Khatib (2010) and Sentis & Khatib (2006). Contrary to these hierarchical approaches that have hard constraints, we prefer using soft constraints by adding corresponding terms in the cost function with high penalties. We gain numerical stability by sacrificing a small fraction of precision. There is also much interest in formulating a smaller optimization problem to reduce computation time. Contact forces can be removed from the equations of motion using orthogonal decomposition (Mistry, Buchli, & Schaal, 2010; Righetti, Buchli, Mistry, Kalakrishnan, & Schaal, 2013; Righetti, Buchli, Mistry, & Schaal, 2011). Ott, Roa, & Hirzinger (2011) demonstrated a balancing controller on a torque-controlled humanoid, in which simple Proportional Derivative (PD) servos were used to generate a desired net ground reaction wrench, which was then distributed among predefined contacts using optimization. Ramos, Mansard, Stasse, & Soueres (2012) described a recent effort using floating base inverse dynamics and ZMP-based pattern generation for dynamic walking. Their inverse dynamics formulation solves a smaller QP with decoupled dynamics. Lee & Goswami (2010) have a two-stage optimization setup. The first optimizes individual ground reaction forces and center of pressure (CoP) for each contact and the resulting admissible change in centroidal momenta. Then another least-squares problem is solved for the state acceleration. Joint torques are generated explicitly. Koolen et al. (2013) generated desired centroidal momenta change based on instantaneous capture points, and they used QP to optimize for acceleration and contact forces. Joint torques are then generated with explicit inverse dynamics. Kuindersma, Permenter, & Tedrake (2014) are similar in terms of optimization variables and torque generation, but a novel QP solver is implemented to exploit the observation that inequality constraints rarely change in this context. Zapolsky, Drumwright, Havoutis, Buchli, & C. (2013) applied QP-based inverse dynamics to a quadruped robot on a slippery surface. Without using constrained optimization, a novel approach to generate full body torques with a combination of gravity compensation and task-dependent attractors is proposed in Moro, Gienger, Goswami, Tsagarakis, & Caldwell (2013). We continue to use the formulation previously developed in our group (Stephens, 2011; Whitman, 2013; Whitman & Atkeson, 2010), which is similar to de Lasa et al. (2010), Bouyarmane & Kheddar (2011), and Bouyarmane, Vaillant, Keith, & Kheddar (2012). We directly optimize a quadratic cost in terms of state accelerations, torques, and contact forces on the full robot model. This design choice gives us the most flexibility in terms of trading off directly among physical quantities of interest. 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.

One traditionally popular approach to controlling humanoid robots is through inverse kinematics with stiff joint position tracking. On the other hand, inverse-dynamics-based approaches have gained increasing acceptance by providing compliant motions and robustness to external perturbations. However, the performance of inverse dynamics methods is heavily dependent on high-quality dynamic models, which are often very difficult to produce for a physical robot. In contrast, inverse kinematics based approaches only require kinematic models, which are much easier to generate in practice. Thus, we supplemented our original two-part implementation with an inverse kinematics controller for the physical robot.

On our Atlas robot, a hydraulic robot built by Boston Dynamics, independent joint level servos compute valve commands i based on

$$i = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + K_f(\tau_d - \tau) + c,$$

where q_d, $\dot{q}_d$, and τ_d are desired joint and torque values, q, $\dot{q}$, and τ are the measured values, and c contains the constant valve bias term plus some other auxiliary feedback terms. This joint level servo runs at 1 kHz, while we can update q_d, $\dot{q}_d$, and τ_d at 333 Hz. In previous work (Feng, Xinjilefu, Huang, & Atkeson, 2013; Stephens, 2011; Whitman, 2013), we focused on torque control with inverse dynamics that computes desired torques, τ_d. Inverse dynamics is operating at the acceleration and force level, thus it alone cannot compute q_d or $\dot{q}_d$ to fully use the high bandwidth joint level controller. We could integrate $\ddot{q}_d$, the output from inverse dynamics, to generate $\dot{q}_d$ and q_d, but we find this approach leads to oscillations and unstable behaviors on the physical robot. To take full advantage of the onboard high bandwidth joint PD servos, we need to compute q_d and $\dot{q}_d$ with inverse kinematics.

Our inverse kinematics is also formulated as a quadratic program, where the unknowns are the desired velocities of the floating base and all the joints $\dot{q}_{ik}$. At each time step, we solve for a set of $\dot{q}_{\text{IK}}$ that obeys kinematic constraints and minimizes a combination of costs. q_{IK} is computed by integrating $\dot{q}_{\text{IK}}$. The approach we take originates from the damped least-squares method that was first used by Wampler (1986) and Nakamura & Hanafusa (1986). We formulate inverse kinematics as a QP problem, and we treat the contacts as soft constraints. The approach of solving for $\dot{q}_{\text{IK}}$ and integrating to obtain q_{IK} is the primary difference between our approach and most traditional inverse kinematic approaches such as that of Kajita et al. (2003) and Hirai, Hirose, Haikawa, & Takenaka (1998). Although Mistry, Nakanishi, Cheng, & Schaal (2008) solved for $\dot{q}$ as well, it is computed by carefully constructing and inverting a matrix composed of end effector and contact constraint Jacobians. Computing $\dot{q}_{\text{IK}}$ and integrating it to compute q_{IK} can get stuck in local minima, but it does not produce discontinuous results, a problem with methods that compute q directly. Another advantage for this gradient based method is that it responds very rapidly to changes in the high-level commands.

The rest of the paper is organized as follows: Section 3 describes our high-level controller for walking that takes a sequence of desired foot steps and uses trajectory optimization to generate key reference motion. Section 4 introduces our full body controller that employs inverse dynamics and inverse kinematics. Sections 5 and 6 expand the previous section with a detailed formulation for each component. Section 7 explains how our state estimator works. Section 8 shows some results for walking in an idealized simulated environment. Section 9 describes our controller's use on the physical robot in three applications during the DARPA Robotics Challenge Trials. Sections 10 and 11 discuss limitations and future research directions and conclude the paper.

3. HIGH-LEVEL REFERENCE TRAJECTORY GENERATION FOR WALKING

In this section, we describe how our high-level behavior controller for walking generates a desired trajectory using trajectory optimization. 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 differential dynamic programming (DDP), which is an iterative trajectory optimization technique that updates the current control signals based on the spatial derivatives of the value function, and it uses the updated controls to generate a trajectory for the next iteration (Jacobson & Mayne, 1970). In the current implementation of the high-level controller, we approximate the entire robot as a point mass, without considering angular momentum or the effect of the swing leg. The dynamics of this simple model are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \dfrac{(x - p_x)F_z}{mz} \\ \dfrac{(y - p_y)F_z}{mz} \\ \dfrac{F_z}{m} - g \end{bmatrix}.$$

The state $X = (x, y, z, \dot{x}, \dot{y}, \dot{z})$ is the position and velocity of the CoM. The control $u = (p_x, p_y, F_z)$ is the commanded CoP and force in the z direction. The current high-level controller is not aware of step length limits, and we are relying on the foot step planner to produce a reasonable foot step sequence.

DDP applies dynamic programming along a trajectory. It can find globally optimal trajectories for problems with time-varying linear dynamics and quadratic costs, and it rapidly converges to locally optimal trajectories for problems with nonlinear dynamics or costs. This approach modifies (and complements) existing approximate dynamic programming approaches in the following 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^t, the local quadratic model for the value function is

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

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

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

where u_0^t is a constant term, and K^t is the first derivative of the local policy with respect to the state evaluated at X^t, and it is also the gain matrix for a local linear controller. V_0^t, V_X^t, V_{XX}^t, and K^t 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 semidefinite. 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 to 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 $F_z = mg$. In the VRC, we used $m =$ 95 kg. Q is a 6 by 6 diagonal matrix, where its diagonal entries are $1e{-}4$, $1e{-}4$, 1, $1e{-}2$, $1e{-}2$, and $1e{-}2$. R is a 3 by 3 diagonal matrix, where its diagonal entries are 1, 1, and $1e{-}6$.

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

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 we 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.

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

$$\begin{aligned}
Q_X^t &= L_X^t + V_X^t f_X^t,\\
Q_u^t &= L_u^t + V_X^t f_u^t,\\
Q_{XX}^t &= L_{XX}^t + V_X^t f_{XX}^t + f_X^{tT} V_{XX}^t f_X^t,\\
Q_{uX}^t &= L_{uX}^t + V_X^t f_{uX}^t + f_u^{tT} V_{XX}^t f_X^t,\\
Q_{uu}^t &= L_{uu}^t + V_X^t f_{uu}^t + f_u^{tT} V_{XX}^t f_u^t,\\
K^t &= (Q_{uu}^t)^{-1} Q_{uX}^t,\\
\delta u^t &= (Q_{uu}^t)^{-1} Q_u^t,\\
V_X^{t-1} &= Q_X^t - Q_X^t K^t,\\
V_{XX}^{t-1} &= Q_{XX}^t - Q_{Xu}^t K^t.
\end{aligned}$$

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

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

$$u_{\text{new}}^t = (u^t - \delta u^t) - K^t(X_{\text{new}}^t - X^t)$$

with $X_{\text{new}}^{t0} = X_0$. We terminate DDP when the cost-to-go at X_0 does not change significantly across iterations. This approach can be thought of as a generalized version of Kajita's preview control (Kajita et al., 2003). Figure 2 shows trajectories of the CoM generated with LQR policy and after DDP optimization.

Swing foot trajectory: 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 zero roll angle, and estimate the pitch angle by the relative height change from consecutive foot steps. Body orientation at the end of the swing phase is computed by averaging the yaw angles from consecutive foot steps, assuming zero 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).

4. FULL BODY CONTROL

This section introduces our use of quadratic programming to perform inverse dynamics and inverse kinematics. For many tasks, we specify desired Cartesian motions for specific locations on the robot (e.g., foot, hand, and CoM) in the high-level controller. The low-level controller takes these motions as inputs and computes physical quantities for each individual joint such as joint position, velocity, acceleration, and torque. Some of these outputs are then used

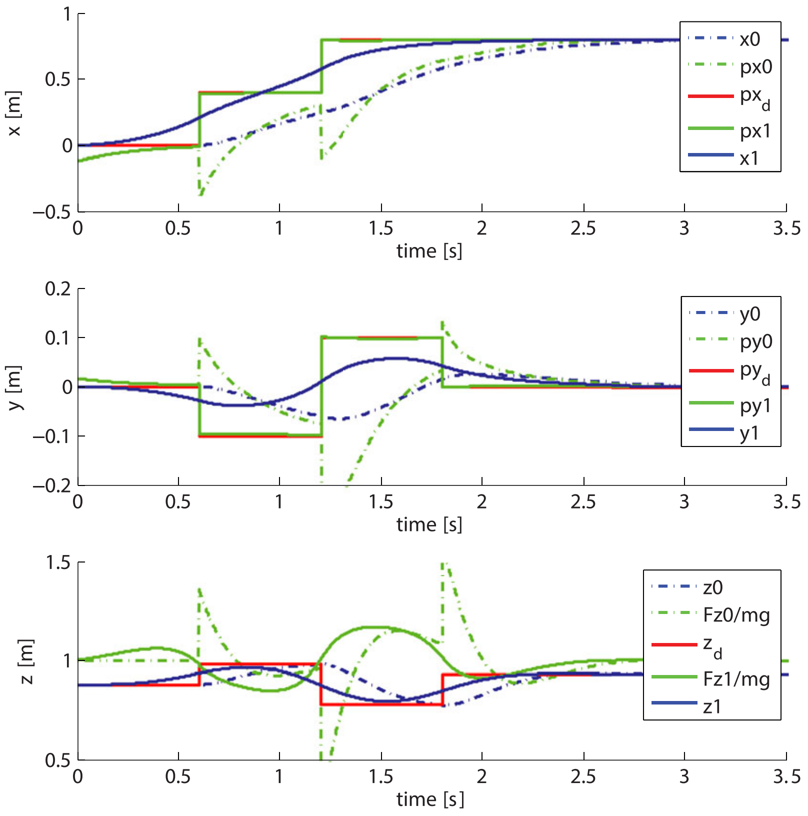


Figure 2. Desired CoP in the X and Y axes and CoM height trajectories are plotted with solid red lines from top to bottom, and they are referred to as px_d, py_d, z_d in the legend. The desired CoPs presented here are set to be at the middle of the stance foot, and the sharp changes are contact switching points. One can design a more complex desired CoP trajectory that has smooth transitions involving a double support phase. Trajectories shown by dashed lines are generated by the LQR policy. The state trajectories are $x0$, $y0$, $z0$, and the control trajectories are $px0$, $py0$, $\frac{Fz0}{mg}$, respectively. These are used to initialize DDP. The optimization results are $x1$, $y1$, $z1$ and $px1$, $py1$, $\frac{Fz1}{mg}$

as references in the joint level servos on the robot. Figure 1 shows a system block diagram. Joint position and velocity are computed separately from joint acceleration and torque. We refer to the former problem as inverse kinematics and the latter as inverse dynamics. Both are formulated as quadratic programming problems, whose general form is shown in Eq. (1).

$$\begin{aligned}\min_{\mathcal{X}} \quad & 0.5\mathcal{X}^T G\mathcal{X} + g^T\mathcal{X}, \\ s.t. \quad & C_E\mathcal{X} + c_E = 0, \\ & C_I\mathcal{X} + c_I \geq 0. \end{aligned} \tag{1}$$

The unknown, $\mathcal{X}$, and constraints, C_E, c_E, C_I, and c_I, are problem-specific, which we will elaborate on in the following sections. Both QP problems are solved at each time step in a 3 ms control loop with a standard solver. We add regularization terms for all the variables in both QP's cost functions to keep them numerically well conditioned.

For both problems, we optimize a cost function of the form $0.5\|A\mathcal{X} - b\|^2$. Thus $G = A^TA$ and $g = -A^Tb$. A and b can be decomposed into smaller blocks as

$$A = \begin{bmatrix} w_0A_0 \\ w_1A_1 \\ \vdots \\ w_nA_n \end{bmatrix}, b = \begin{bmatrix} w_0b_0 \\ w_1b_1 \\ \vdots \\ w_nb_n \end{bmatrix}. \tag{2}$$

Through this cost function, we specify a set of desired behaviors according to the high-level controller's goal, and we penalize the robot's deviation from the desired behavior. Each row in Eq. (2) emphasizes a certain desired behavior. w_i are weights that we use to express the relative priority among often overly constrained and potentially conflicting goals. During implementation, finding reasonable weights is fairly straightforward, and it takes fewer than one day of robot experiments.

5. INVERSE DYNAMICS

In this section, we describe how we compute inverse dynamics using quadratic programming. The equations of motion and the constraint equations for a floating base humanoid robot can be written as

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

where $(q,\dot{q})$ is the full state of the system including the six degrees of freedom (DOF) of 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 whose top six rows that correspond to the floating base are zeros and the rest form an identity matrix; τ is a vector of joint torques; $J^T(q)$ is the Jacobian matrix for all the contacts; and F is a vector of all contact forces in the world frame. The dimensions of F and J^T depend on the number of contacts. Given a state $(q,\dot{q})$, the equations of motion are linear in terms of $\mathcal{X} = [\ddot{q} \quad \tau \quad F]^T$. We use the equations of motions as equality constraints. The inequality constraints consist of various terms such as joint torque limits and contact force limits due to friction cone constraints and constraints keeping the CoP in the support polygon.

5.1. Inverse Dynamics Cost Function

The optimization process can be thought of as a least-squares minimization problem penalizing $\mathcal{X}$ for deviating from some desired $\mathcal{X}^*$. We list a few examples of the objectives that can be part of the optimization criterion by adding rows to Eq. (2).

5.1.1. Cartesian Space Acceleration

Since

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

we can penalize deviation from the desired Cartesian acceleration using

$$A_{\text{Cart}} = [J(q) \quad 0 \quad 0],$$
$$b_{\text{Cart}} = \ddot{x}^* - \dot{J}(q,\dot{q})\dot{q}.$$

The input $\ddot{x}^*$ is computed by

$$\ddot{x}^* = K_{\text{id}}(x_d^* - x) + D_{\text{id}}(\dot{x}_d^* - \dot{x}) + \ddot{x}_d^*, \tag{3}$$

where x_d^*, $\dot{x}_d^*$, and $\ddot{x}_d^*$ are specified by a higher-level controller, x and $\dot{x}$ are computed by forward kinematics based on the current robot state, and K_{id} and D_{id} are gains. Many objectives such as CoM, hand, foot, and torso motion and orientation are specified in this form. Depending on the objectives, we sometimes drop the rows in the Jacobian that we do not want to penalize. It is often much easier for the high-level controller to specify motions in a body's local frame. We transform the Jacobian by a rotation matrix in that case.

Rather than treating contacts as hard constraints, we find that using a soft penalty with a high weight is generally numerically more stable and faster to solve. For such contact costs, we disregard x_d^* and $\dot{x}_d^*$, and we set $\ddot{x}^* = 0$.

5.1.2. Center of Pressure Tracking

Given the forces and torques, ${}^bF, {}^bM$, specified in the foot frame by the high-level optimizer, 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 the center of pressure deviation with

$$A_{\text{cop}} = \left[0 \quad 0 \quad \begin{bmatrix} 0 & 0 & p_x^* & 0 & 1 & 0 \\ 0 & 0 & p_y^* & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \right],$$
$$b_{\text{cop}} = 0.$$

Here we have written out the portion of A_{cop} corresponding to the force component, a six-element vector F. Since F is in the world frame, we need to first transform the forces and torques into the foot frame. R is the 3×3 rotation matrix from the world frame to the foot frame. (p_x^*, p_y^*) is the desired center of pressure in the foot frame given by a high-level controller.

5.1.3. 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 using

$$A_{\text{weight}} = [0 \quad 0 \quad S_{\text{weight}}],$$
$$b_{\text{weight}} = 0,$$

where S_{weight} is a row vector with zeros, except the entry for F_{zl} is $1 - w^*$ and the entry for F_{zr} is $-w^*$.

5.1.4. Direct Tracking and Regularization

We can also directly penalize deviations of $\mathcal{X}$ from desired values with

$$A_{\text{state}} = I,$$
$$b_{\text{state}} = [\ddot{q}^* \quad \tau^* \quad F^*]^T.$$

Zero is used for a desired value if no target value is specified. This term is useful for directly controlling specific joints or forces. It also regularizes $\mathcal{X}$ to make the QP problem well-conditioned.

5.1.5. Change in Torques

To avoid high-frequency oscillations, we penalize changes in τ with

$$A_{d\tau} = [0 \quad I \quad 0],$$
$$b_{d\tau} = \tau_{\text{prev}},$$

where τ_{prev} is the output from the last time step. A similar term is used to penalize changes in contact force.

5.2. Constraints

Equations of motion are used as equality constraints. Torque limits can be easily added into the inequality constraints. Friction constraints are approximated by

$$|{}^bF_x| \le \mu^b F_z,$$
$$|{}^bF_y| \le \mu^b F_z.$$

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

$$d_x^- \le -{}^bM_y/{}^bF_z \le d_x^+,$$
$$d_y^- \le {}^bM_x/{}^bF_z \le d_y^+,$$

where ${}^bF$ and ${}^bM$ denote forces and torques in the foot frame, and d^- and d^+ are the sizes of the feet. The body frame forces and torques are computed by rotating F into the foot frame.

5.3. Parameters

Weights are summarized in Table I. w_{qdd} is for joint acceleration. w_{comdd} and w_{utorsowd} are for CoM position acceleration and upper torso orientation acceleration. w_{footdd} is for foot position and orientation acceleration. $w_{\text{reg}F}$ and $w_{\text{reg}Tau}$ are regularization weights for contact force and joint torques. w_w is for weight distribution. w_{cop} is for center of pressure. w_{dF} and $w_{d\tau}$ penalize changes in contact force and joint torques between two consecutive time steps. For CoM tracking, K_{id} and D_{id} in Eq. (3) have diagonal elements 15 and 0.4. For upper body orientation tracking, K_{id} and D_{id} in Eq. (3) have diagonal elements 5 and 0.3. For foot tracking, K_{id} and D_{id} in Eq. (3) have diagonal elements 5 and 0.3. Parameters are task-dependent, but they do not change much. These parameters are for the static walking task on the physical robot.

Table I. Weights for ID cost function.

w_{qdd}	w_{comdd}	w_{utorsowd}	w_{footdd}	w_{cop}
10^{-2}	1	1	1	10^{-2}
w_w	$w_{\text{reg}F}$	$w_{\text{reg}Tau}$	w_{dF}	$w_{d\tau}$
10^{-3}	10^{-4}	10^{-5}	10^{-4}	3×10^{-3}

6. INVERSE KINEMATICS

So far, neither the high-level behavior controller nor the inverse dynamics controller have specified desired velocities and positions for all the joints. Only task level quantities such as center of mass velocity and position have been specified. Due to modeling error and the availability of joint level servos, it is useful to specify targets for all the joints. Unlike traditional inverse kinematics approaches that generate desired positions for an entire desired trajectory ahead of time, we compute a desired velocity at each time step and integrate it to get a desired position. Computation is averaged over the duration of a motion, and the controller can be much more responsive to changes in the high-level commands.

For the inverse kinematics quadratic program, $\mathcal{X} = \dot{q}_{\text{IK}}$. The numerically integrated floating base and joint position vector is denoted by q_{IK}. Our inverse kinematics controller formulation is very similar to the inverse dynamics controller except rather than using the real robot states, we use internal states to compute the desired velocities in Eq. (4). The internal states are set to the real robot states in the initialization stage. All the internal states are denoted with the subscript IK.

6.1. Inverse Kinematics Cost Function

We list a few examples of the objectives that can be part of the optimization criterion by adding rows to Eq. (2).

6.1.1. Cartesian Space Velocity

We penalize deviation from a desired Cartesian velocity of a body part with

$$A_{\text{Cart}} = J(q_{ik}),$$
$$b_{\text{Cart}} = \dot{x}^*,$$

where

$$\dot{x}^* = K_{\text{IK}}(x_d^* - x_{ik}) + \dot{x}_d^*. \tag{4}$$

Like our inverse dynamics quadratic program, x_d^* and $\dot{x}_d^*$ are given by the high-level controller. The desired Cartesian target x_d^* is used to implement something we call "anchor" points. The diagram in Figure 1 shows that the actual physical robot state is not used by the inverse kinematics controller. Without any such feedback, it is easy for the inverse kinematics to diverge significantly from the measured position of the robot. This becomes a problem because we wish to command desired locations (for, e.g., feet, hands, or CoM) in world coordinates. To tie the inverse kinematics controller root position to reality, we use contact positions as "anchor" points. We use a "leaky" integrator to adjust the desired contact position $x^*_{\text{contact}_d}$ toward the measured contact position x_{contact},

$$x^*_{\text{contact}_d} = \alpha x_{\text{contact}} + (1-\alpha)x^*_{\text{contact}_d}. \tag{5}$$

$x^*_{\text{contact}_d}$ is the input to the inverse kinematics, and it is initialized to the inverse kinematics controller's internal value upon establishing the contact. Since $x^*_{\text{contact}_d}$ essentially contains all the information about long-term tracking error and state estimator drift, and the inverse kinematics will track $x^*_{\text{contact}_d}$ obeying all the kinematic constraints, we can use $x^*_{\text{contact}_d}$ to update the root position to match the state estimator's. The inverse dynamics controller is not affected since it ignores this term.

6.1.2. Direct Tracking and Regularization

$$A_{\text{state}} = I,$$
$$b_{\text{state}} = \dot{q}^*,$$

where $\dot{q}^*$ can be a target joint velocity or zero for regularization.

6.1.3. Change in Velocity

$$A_{d\dot{q}} = I,$$
$$b_{d\dot{q}} = \dot{q}_{\text{prev}},$$

where $\dot{q}_{\text{prev}}$ is the result from the previous time step. This term is useful to eliminate high-frequency oscillation.

6.2. Constraints

We do not impose equality constraints in the inverse kinematics QP. Inequality constraints mainly consist of joint limits. Depending on the application, we also add constraints in Cartesian space.

The joint limit constraints are

$$q^- \leq q_{\text{IK}} + \dot{q}dt \leq q^+,$$

where dt is the time step, and q^- and q^+ are the upper and lower joint limit. For Cartesian space position constraints,

$$x^- \leq x_{\text{IK}} + J(q_{\text{IK}})\dot{q}dt \leq x^+,$$

where x^- and x^+ are the upper and lower limits. Velocity constraints in joint space can be easily added, and Cartesian space velocity constraints need to be transformed by the Jacobian matrix.

6.3. Parameters

Weights are summarized in Table II. w_{qd} is for joint velocity tracking. w_{comd} and w_{utorsow} are for CoM velocity and upper torso angular velocity. w_{footd} is for foot linear and angular velocity. $w_{d\dot{q}}$ penalizes changes in velocity. The position gain K_{IK} used in Eq. (4) is 2 along the diagonal for the CoM, upper torso orientation, and foot tracking. α in Eq. (5) is 0.03. Parameters are task-dependent, but they do not change much except α for tasks that involve a very long stance phase such as manipulation. This set of parameters is for the static walking task on the physical robot.

Table II. Weights for IK cost function.

w_{qd}	w_{comd}	w_{utorsow}	w_{footd}	$w_{d\dot{q}}$
5×10^{-2}	5	1	10^2	5×10^{-1}

7. FLOATING BASE POSITION-VELOCITY STATE ESTIMATOR

The state estimator for the pelvis's position and velocity is based on Xinjilefu, Feng, Huang, & Atkeson (2014). For a floating base humanoid, the base has three translational and three rotational degrees of freedom. We designate the pelvis as the base link. For the Atlas robot, there is a six-axis inertial measurement unit (IMU) attached to the pelvis. The IMU measures angular velocity and linear acceleration, and it also provides an estimate of the pelvis orientation in the world frame. We use the orientation estimate from the IMU without modification. The pelvis state estimator estimates the global pelvis position p_x and linear velocity p_v. It is a multiple model Kalman filter with contact switching. We design a steady-state Kalman filter for each possible contact state in advance. The current contact state is specified by the controller. The IMU has known position and orientation offsets relative to the pelvis origin. These offsets have already been taken into account, so the following equations are offset-free.

The process dynamics and prediction step of the pelvis Kalman filter are simply

$$x_k^- = \begin{bmatrix} p_{x,k}^- \\ p_{v,k}^- \end{bmatrix} = f(x_{k-1}^+) = \begin{bmatrix} p_{x,k-1}^+ + p_{v,k-1}^+ \Delta t \\ p_{v,k-1}^+ + a_{k-1}\Delta t \end{bmatrix}, \tag{6}$$

where x_k is the state estimate. The subscript k is the step index, and the superscript "$-$" and "$+$" represent before and after the measurement update. The net linear acceleration of the IMU a is transformed from the IMU frame to the world frame using the IMU orientation. It is straightforward to linearize the process dynamics to get the state transition matrix F_k,

$$F_k = \begin{bmatrix} I & \Delta t I \\ 0 & I \end{bmatrix}. \tag{7}$$

The measurement update step is slightly more complicated. There is no sensor directly measuring the position and velocity of the pelvis in world coordinates. We use the following assumptions in place of an actual measurement: we know the contact points, and we know how the contact points move in Cartesian space. These assumptions are not limited to walking, but we will use walking as an example. Let the point of the ankle joints of the left and right feet be c_l and c_r in Cartesian space, and let the corresponding velocities be $\dot{c}_l$ and $\dot{c}_r$. In the double support phase (DS), we assume the feet are not moving to obtain the following measurements:

$$z_{k,\mathrm{DS}} = \begin{bmatrix} c_{l,k} \\ c_{r,k} \\ \dot{c}_{l,k} \\ \dot{c}_{r,k} \end{bmatrix} = \begin{bmatrix} c_{l,\eta_l} \\ c_{r,\eta_r} \\ 0 \\ 0 \end{bmatrix}. \tag{8}$$

The time index η is the time step when the foot is detected to be firmly on the ground using the force sensors. The first two equations say that the current foot positions are fixed, and the last two equations say the foot linear velocities are zero. Essentially the first and last two equations convey the same information: the feet are fixed. We use the redundant information because the filter will not perform worse with the additional information, and it may perform better.

To write the measurement equations, we need the prediction to be a function of pelvis position and velocity. We use the floating base forward kinematics FK(·):

$$y_{k,\mathrm{DS}} = \begin{bmatrix} \mathrm{FK}_{c_l}(q_k) \\ \mathrm{FK}_{c_r}(q_k) \\ \mathrm{FK}_{\dot{c}_l}(q_k, \dot{q}_k) \\ \mathrm{FK}_{\dot{c}_r}(q_k, \dot{q}_k) \end{bmatrix}. \tag{9}$$

The observation matrix H_k is computed by linearizing Eq. (9), and it turns out to be the identity

$$H_{k,\mathrm{DS}} = I_{12\times 12}. \tag{10}$$

The stance foot is assumed to be fixed in the single support phase, so Eqs, (8), (9), and (10) are modified to account for contact switching.

8. SIMULATION RESULTS

We first describe results from the approach used in the simulation phase of the DARPA Robotics Challenge. At that time, the approach did not include the inverse kinematics in the low-level controller. Our approach was tested on a simulated Boston Dynamics Atlas robot in DARPA's Virtual Robotics Challenge setting (Figure 3). The simulation, which used a 1 KHz control loop, is based on Gazebo, which is produced by the Open Source Robotics Foundation. The simulated Atlas robot has 28 joints that can be used as pure torque sources: six for each leg and arm, three for the back joints, and one for neck pitch. We ran the simulation in real time on a computer with an Intel Xeon(R) CPU E5-2687W CPU with 32G memory. Our controller used two threads. One thread was dedicated to the low-level controller solving inverse dynamics at 1 KHz, and the other ran the high-level controller that optimizes the center of mass trajectory for the next three footsteps when the robot has taken a step. As for runtime, the high-level controller completed within 1–50 ms (depending on terrain complexity), and the low-level controller finished within 0.5 ms.

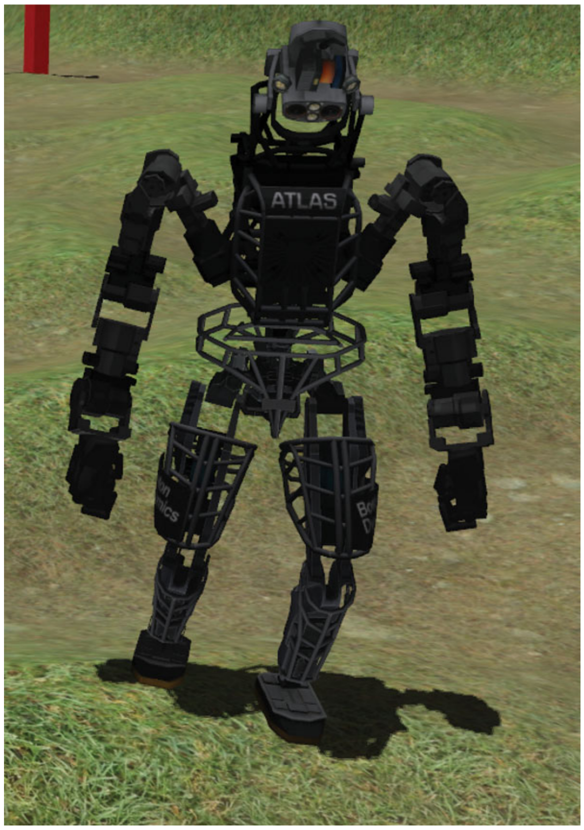


Figure 3. Our controller tested in simulation in the Rough Terrain Task in DARPA's Virtual Robotics Challenge

8.1. Flat Ground Walking

We are able to demonstrate toe-off and heel-strike behavior during flat floor walking by using simple heuristics to guide the low-level controller. For heel-strike, a desired touch down pitch angle is specified. For toe-off, we first change the reference point where the contact Jacobian is computed to the toe, constrain the CoP to be at the toe, and then specify a short but large pitch angular acceleration in the foot frame. With these heuristics, we have achieved a maximum step length of 0.8 m. Figure 4 shows a sequence of snapshots taken for walking with a 0.7 m step length, and a 0.8 s period on flat ground. The maximum speed we achieved is on average 1.14 m/s, with a 0.8 m step and a 0.7 s period on flat ground.

8.2. Rough Terrain

The controller can handle up to 0.4 rad inclined slopes, and it can continuously climb stair steps that are 0.2 m high and 0.4 m apart. We are able to successfully walk on the rough terrain environment provided in the Virtual Robotics Challenge as well. To traverse rough terrain, an A* planner

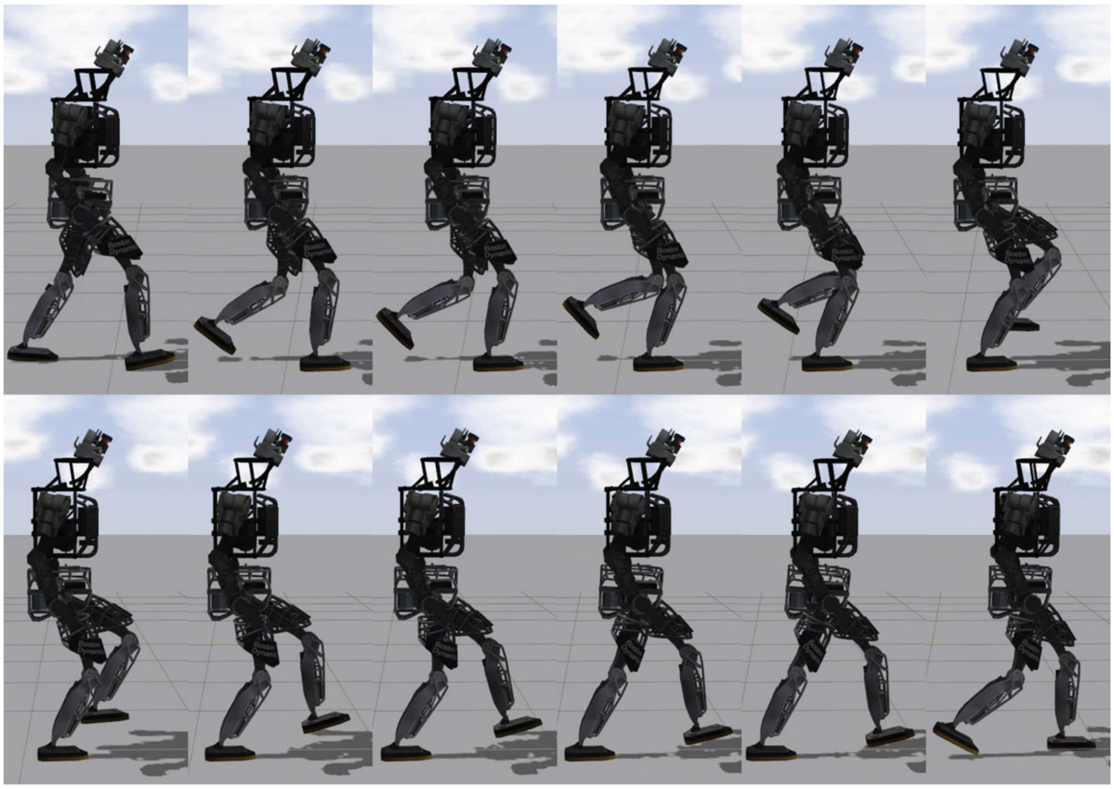


Figure 4. Snapshots taken for simulated walking on flat ground with 0.7 m step length and 0.8 s period

modified from Huang, Kim, & Atkeson (2013) 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 5 shows tracking performance.

9. ROBOT APPLICATIONS

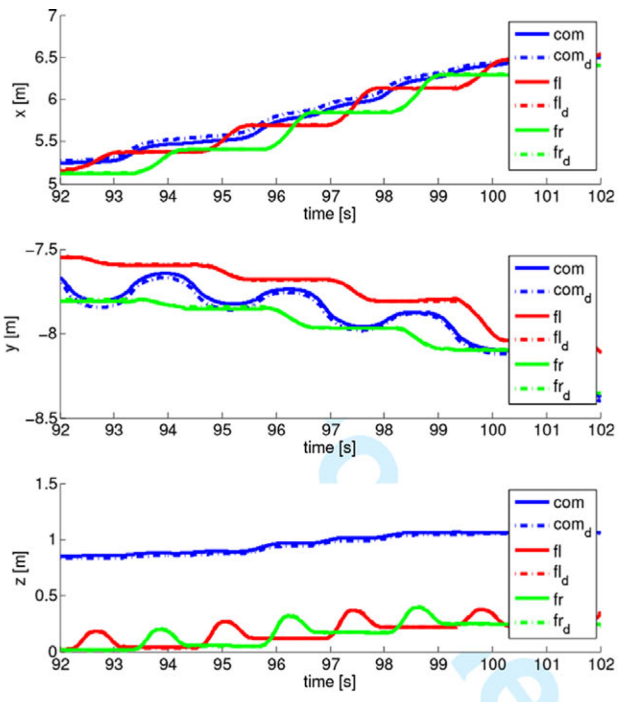
The full body controller with the inverse kinematics controller was tested on a Boston Dynamics Atlas robot at the DARPA Robotics Challenge Trials. Our rough terrain walking, full body manipulation, and ladder climbing controllers all used our control approach. Figures 6, 9, and 10 show the robot performing these three tasks.

9.1. Rough Terrain

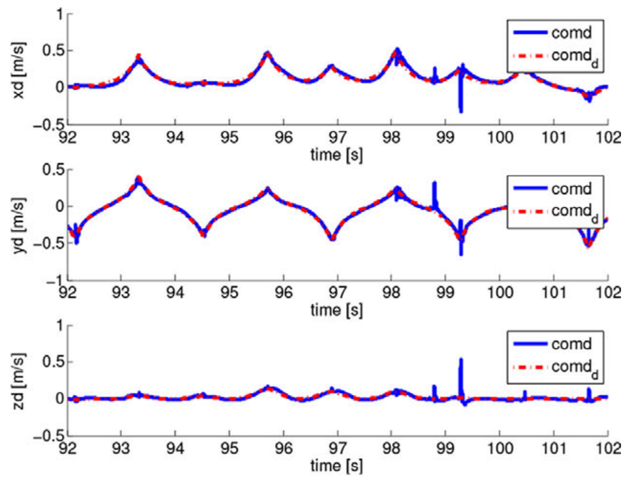
Given the short time frame for development for the DRC Trials and the cost of falls, we decided to use a simple static walking strategy. The high-level desired motions such as CoM and swing foot trajectories are generated with quintic splines. The given foot step locations are used as knot points for the splines. The desired CoP trajectory is generated using a linear inverted pendulum model (LIPM). Figure 6 shows a sequence of snapshots of the Atlas robot traversing the third segment of the rough terrain with tilted cinder blocks. and Figure 7 plots measured feet and CoM trajectories. Figure 8 shows CoP tracking in more detail.

In terms of operator control for walking, we provided our human operator with a live video stream from the robot's cameras augmented with the current swing foot pose computed from forward kinematics, and we let the operator "nudge" the swing foot around in the six-dimensional Cartesian space by commanding offsets in foot position and orientation. Once the operator was satisfied with the foot pose, a "continue" command was given, allowing the robot to lower the swing foot straight down until ground contact was detected. We chose this approach because we found that the operators easily understood the robot images, and we had more trouble understanding laser scan data. Laser scan data took time to accumulate, and they were most accurate when the robot stood still during data acquisition. Our goal was to avoid standing still waiting for data. On the other hand, our approach requires substantial input from the operator, and it extends the more risky single support phase unnecessarily since the operator commands were given during single support rather than double support. We plan to refine this approach in future work.

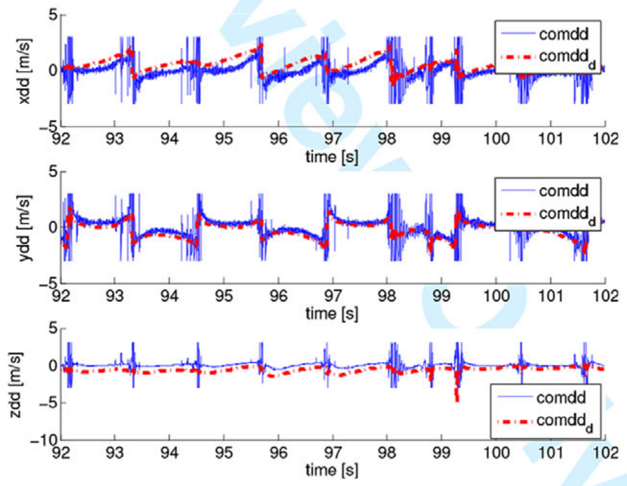
The following modifications to the full body controller as described above were made for the walking task:



(a) CoM and foot position tracking on rough terrain



(b) CoM velocity tracking on rough terrain



(c) CoM acceleration tracking on rough terrain

Figure 5. Comparison of actual and desired values during simulated walking. All actual traces are plotted by solid lines, and desired traces are shown by dashed lines. Subscripts in the legends refer to the desired traces. In (a), left foot traces are shown by red lines, right foot by green, and CoM are plotted in blue. The reference foot points are set to the heel. Part (b) shows velocity tracking of the CoM. In (c), actual CoM acceleration is computed by finite differencing the velocity trace, and it is truncated at $\pm 3\ \mathrm{m/s^2}$

9.1.1. Ankle Torque Controlled

To fully control the center of pressure for achieving better balancing and being more robust to perturbation, we control the stance ankle joints in pure torque mode. Inverse kinematics solutions for the stance ankle joints are ignored. The disadvantage is that the ankle angle errors propagate up the kinematic chain, and they result in significant errors in swing foot position tracking. An integrator on

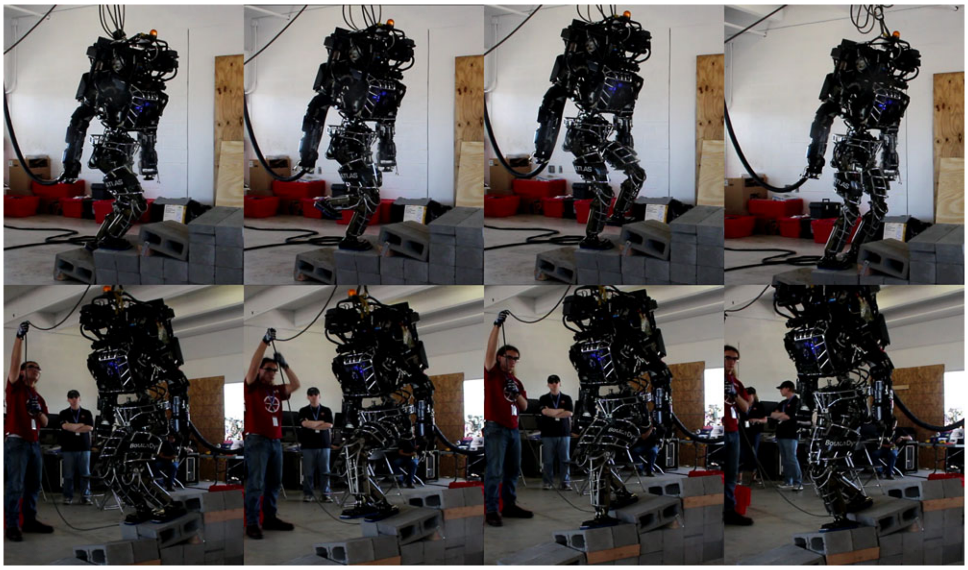


Figure 6. These photos show the Atlas robot practicing for segment 3 of the terrain task for DRC. The snapshots were taken every 5 s

the desired swing foot position is used to compensate for this,

$$\begin{aligned}\mathrm{err}_{\mathrm{swing}} &= \mathrm{err}_{\mathrm{swing}} + K_i(x'_{\mathrm{swing}_d} - x_{\mathrm{swing}}),\\ x^*_{\mathrm{swing}_d} &= x'_{\mathrm{swing}_d} + \mathrm{err}_{\mathrm{swing}},\end{aligned} \tag{11}$$

where x'_{swing_d} is the desired swing foot position, x_{swing} is the computed position from forward kinematics, and $x^*_{\mathrm{swing}_d}$ is used in the IK and ID controllers as inputs.

9.1.2. Toe-off

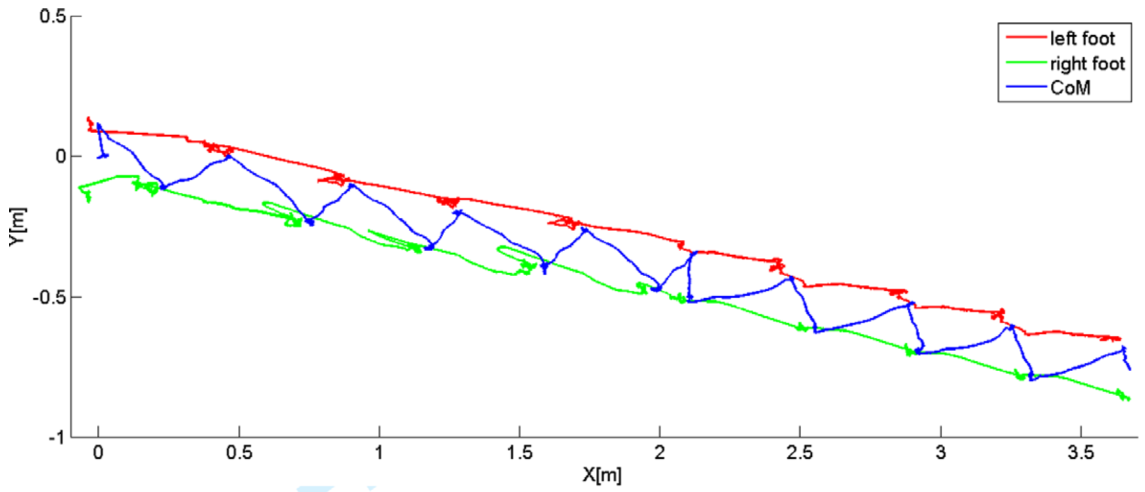
For static walking, the CoM needs to be completely shifted to the next stance foot during double support. When taking longer strides or stepping to a greater height, extending the rear leg knee alone is often insufficient to move the CoM all the way. Toe-off is one solution to this problem. During double support in our controller, toe-off is triggered when the rear knee approaches the joint angle limit (straight knee). Once triggered, special modifications are used in both the ID and IK controllers. We first move the rear foot reference point, where the Jacobian is computed to the toe. In the ID controller, the contact cost term for the rear foot is transformed to its local frame, and the row that corresponds to pitch angular acceleration cost is removed. We also constrain the allowed pitch torque to be zero. This effectively turns the rear foot contact into an unactuated pin joint around the pitch axis. In the IK controller, we transform the rear foot's pitch tracking error into the foot frame and drop the pitch term. A slightly bent rear knee angle is used to bias the inverse kinematics toward using ankle angle for a toe-off solution.

9.1.3. Integrator for Desired CoM Offset

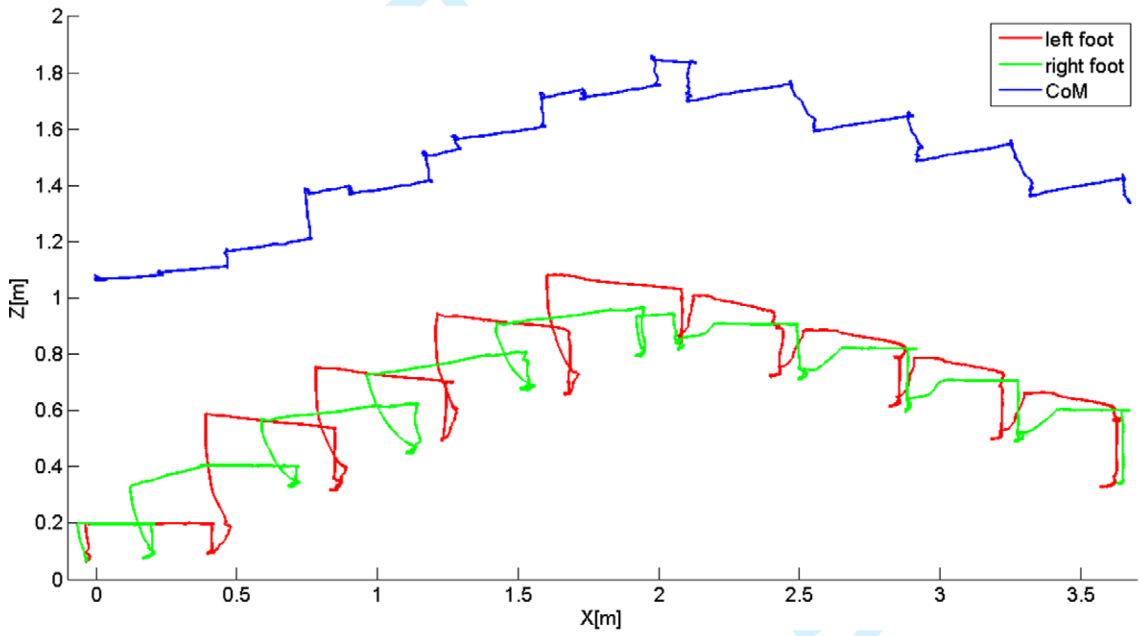
During static robot experiments, the measured CoM location, which is measured with foot force sensors, deviates from the model's prediction. We also believe this modeling error depends on the robot configuration. During the second half of double support and the full single support phase, we integrate this error and use it to offset the desired CoM location so that the true CoM matches the desired. Assuming the robot is moving slowly enough, we can approximate the true location of CoM with the measured CoP. The integrator is set up similarly to Eq. (11).

9.2. Full Body Manipulation

During full body manipulation, the operator gives a series of commands requesting either direct joint angles for one or both arms or target Cartesian locations for one or both hands. The controller itself does not try to reason about grasping. A pool of task-specific preferred pregrasp poses for the end-effector is manually determined offline. During each manipulation task, the operator would pick the most suitable pregrasp based on the situation. These commands are used to update the desired inverse kinematics controller position. We use equality constraints in the inverse kinematics QP formulation to enforce directly specified joint angles. For large Cartesian motions, we transition the desired locations through splines starting at the current target and



(a) Measured feet and CoM trajectories in XY plane for terrain task



(b) Measured feet and CoM trajectories in XZ plane for terrain task

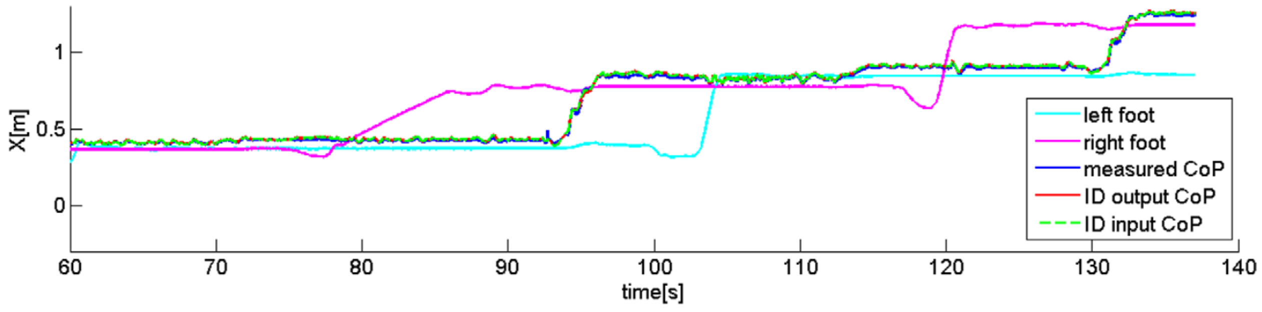
Figure 7. These plots show the Atlas robot traversing segment 3 of the terrain task. The X axis is the forward direction, Y points to the robot's left, and Z points upward. Left and right foot positions are shown in red and green lines, and center of mass is plotted in blue. The robot walks in a straight line along $y = 0$ in reality. Our state estimator drifts significantly as shown in the top plot

ending at the new target. For small motions, we use the "nudge" method as described above for precise foot placement: single keyboard taps result in small instantaneous changes in the desired position. We then use PD gains comparing the measured and desired positions (of hands, CoM, etc.) to produce desired accelerations for the inverse dynamics. Figure 9 shows our robot turning a valve in the DRC Trials.

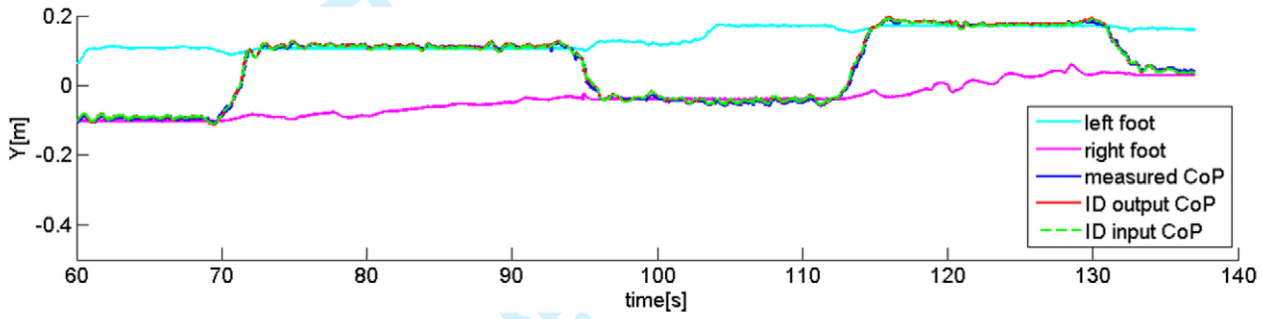
We made a few small changes to the basic full body control algorithm:

9.2.1. No Anchoring

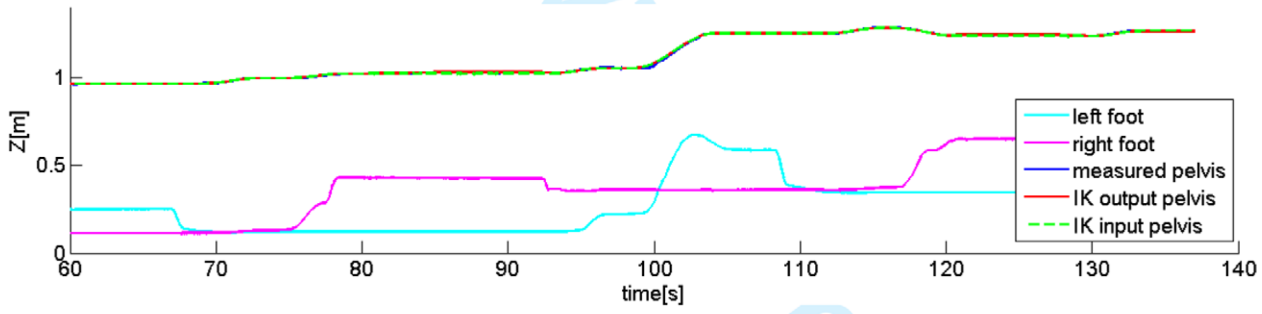
During manipulation, we keep both feet planted and do not take any steps. Accordingly, we do not have to worry about the inverse kinematics position diverging from the



(a) CoP in X axis



(b) CoP in Y axis



(c) Root position in Z axis

Figure 8. The top plot shows CoP in the *X* (forward) direction, the middle plot shows CoP in the *Y* (side) direction, and the bottom plot shows *Z* (vertical positions). These data were collected when the robot was stepping up the cinder block piles. Measured CoP is plotted in solid blue. Desired CoP given by the high-level controller is shown in dashed green. ID's output CoP is shown in solid red. These traces are very similar. Cyan and magenta lines represent left and right foot position computed through forward kinematics, respectively. CoP tracking is within 1 cm. The last row shows pelvis height tracking. Measured root height (state estimator's pelvis position) is shown by a solid blue line. Input to IK is shown by dashed green, and IK's result is plotted in solid red. These traces are very similar as well. Cyan and magenta lines represent left and right foot height

estimated robot position. However, the leaky integrator in Eq. (5) can result in a failure mode characterized by a constant velocity sliding of the foot. We call this failure mode a "chase condition," and it occurs when the contact friction is too low to keep the feet from sliding on its own (usually because very little weight is on one of the feet). Normally, the foot would slide a small amount, but then the position gains from the inverse kinematics controller prevent further sliding. However, when we constantly update the inverse kinematics controller to the measured position, it can then constantly slide farther. We therefore disable this integrator during manipulation.

9.2.2. Allowing Rotation

For some tasks, we only care about the position of the hand, and hand orientation is unimportant. For such cases, we can turn the weight for the hand orientation equations in

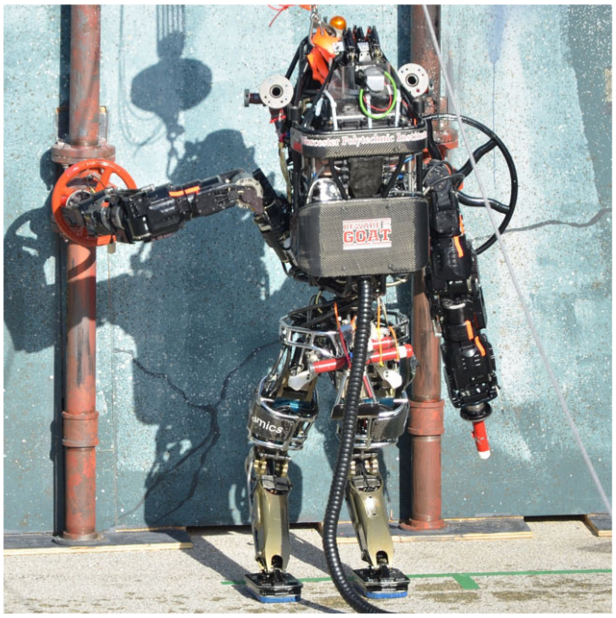


Figure 9. The Atlas robot in full body manipulation mode doing the valve task during the actual DRC Trials

the inverse kinematics QP to zero, or we can remove the equations entirely. For some tasks, we can allow free rotation around one vector, but not otherwise. For example, while drilling through a wall, the robot can freely rotate the drill around the drilling axis, but it must maintain its position while keeping that axis normal to the wall. Allowing the controller the freedom to rotate around one axis can greatly increase the available workspace.

To allow rotation about one axis, we first construct a basis of three orthogonal unit vectors including the desired free-to-rotate-about axis. We then rotate the inverse kinematic equations concerning hand orientation into this basis and remove the one corresponding to the specified axis.

9.3. Ladder Climbing

The underlying controller for ladder climbing is similar to that used for manipulation. Due to Atlas's kinematics (the knee colliding with a tread during the climb), the robot's center of mass must sometimes be behind the polygon of support, and there is a tipping moment. We chose to counteract that tipping moment by grasping higher treads with the hands. To make up for weak fingers, we built hook hands for this purpose. The majority of the motion is scripted ahead of time, with only the final placement of the hands and feet controlled by the operator. For each limb, the hand or foot is automatically moved to approximately the desired position by placing it relative to the other hand or foot. Then, the operator uses the keyboard to precisely place the limb with 1 cm increments. The correct vertical height is found automatically, using force sensors to detect contact for the feet and position sensing when contact is known to have already occurred for the hands. Figure 10 shows a sequence of snapshots of ladder climbing, and Figure 11 plots measured hands, feet, and CoM trajectories during the actual ladder task at the DRC Trials.

Once on the steps, only the toes of the feet are supported, so we adjust the CoP constraint accordingly. Having all of the weight on the toes makes the robot vulnerable to rotational slipping, causing unexpected yaw rotations. To correctly place the hands on the next step to recover from such rotations, we must rotate the inverse kinematics solution to match the measured orientation. We therefore periodically rotate the IK solution such that the feet are aligned with the measured feet orientations, allowing the robot to reorient its upper body toward the ladder and correctly reach targets in the real world. It would have been preferable to update the orientation continuously, but periodic updates were easier from a software engineering perspective. Additionally, periodic updates are less susceptible to the "chase condition" problem described above. This reorienting serves a similar purpose to Eq. (5), but for rotation instead of translation. To avoid chase conditions, we disable Eq. (5) if there is not significant (about 20%) weight on the foot.

9.3.1. Elbow Management

The robot's shoulders are nearly as wide as the railings, so the necessity of shifting weight from side to side results in a significant danger of bumping the arms on the railings. We avoid such collisions by estimating the railing location based on the hand location (based on the assumption that the hand is pushed up against the side of the step) and adding inequality constraints to the inverse kinematics quadratic program. The inequality constraints limit how far outward each elbow can move in the lateral direction. Additionally, when we wish to intentionally lean on the railing, we provide a desired elbow location (only in the lateral direction) with a low weight. To prevent becoming overconstrained by elbow management, we use low weights for commanding hand orientation. Specifically, we rotate the hand orientation equations into a basis containing the hand axis, a pitchlike vector, and a yawlike vector. We use a very low but nonlinear weight for rotation about the hand axis (roll-like), allowing it to roll about 45 degrees nearly freely, but preventing it from rolling much farther.

9.3.2. Hand to CoM Integration

Our robot model had inaccurate forward kinematics. One result is that if the hands are resting on one step and the robot steps up one rung on the ladder, even though the true position of the hands will not have moved, the measured position will have moved several centimeters. If not accounted for, this will push the CoM far from the desired

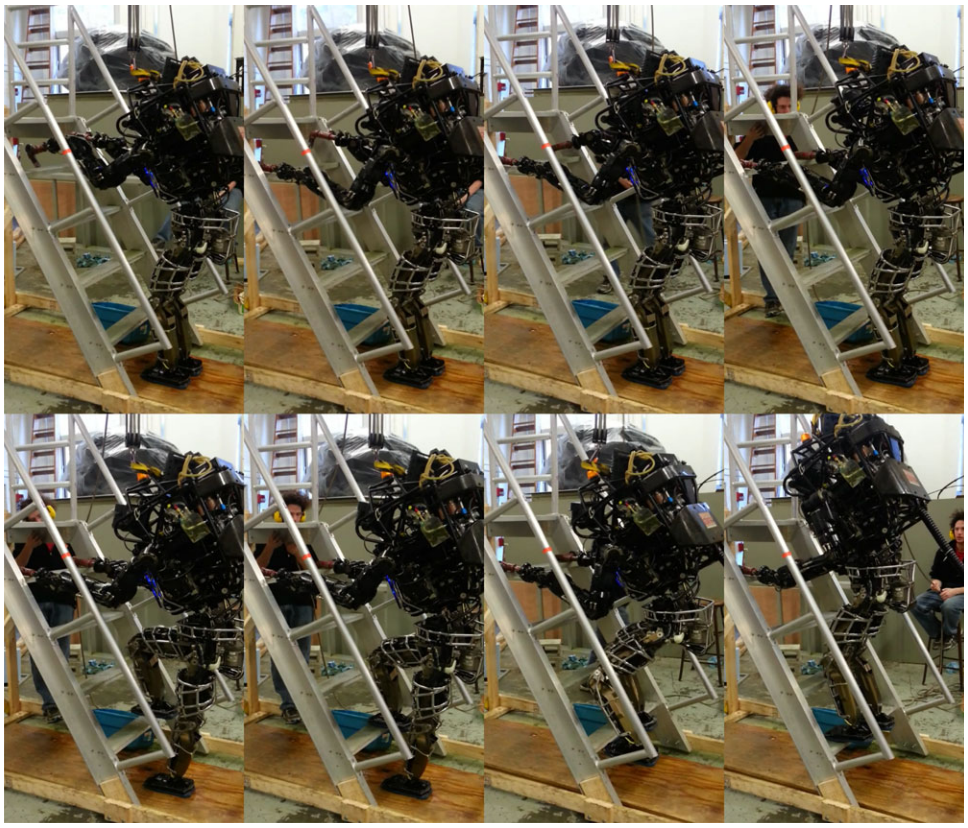


Figure 10. These photos show the Atlas robot climbing the top half of the same ladder as used in DRC. The snapshots were taken every 13 s. The top row shows repositioning of the hook hands, and the bottom row shows stepping up one tread. Most of the climbing motions are scripted. After each limb's rough repositioning, the operator can adjust its final position with "nudge" commands that are small offsets in Cartesian space

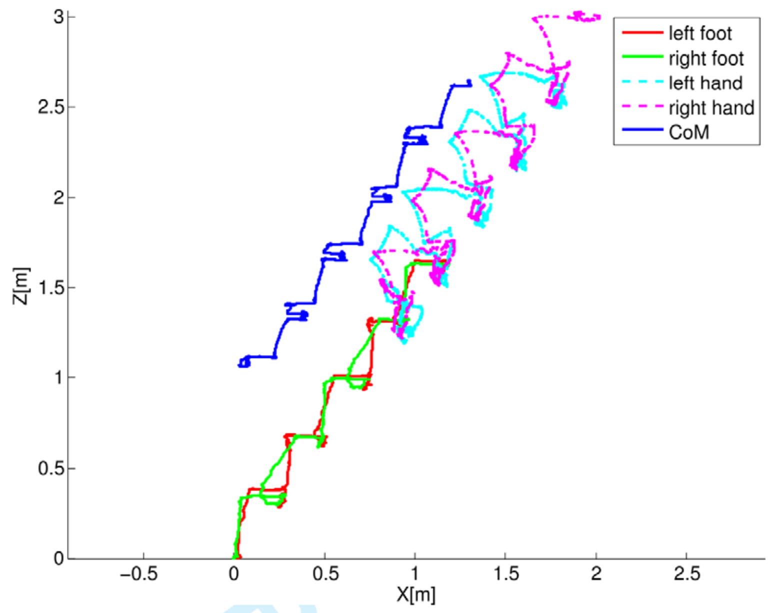
location, eventually resulting in failure. We therefore introduce an integrator that gradually adjusts the desired position of both hands in the horizontal plane based on the deviation between the measured and desired CoM position. Essentially, we are using the arms to pull or push the CoM into the desired position. To avoid unintentionally rotating the robot, this integrator is only active while both hands are in contact with the step (not during hand repositioning).

10. DISCUSSION AND FUTURE WORK

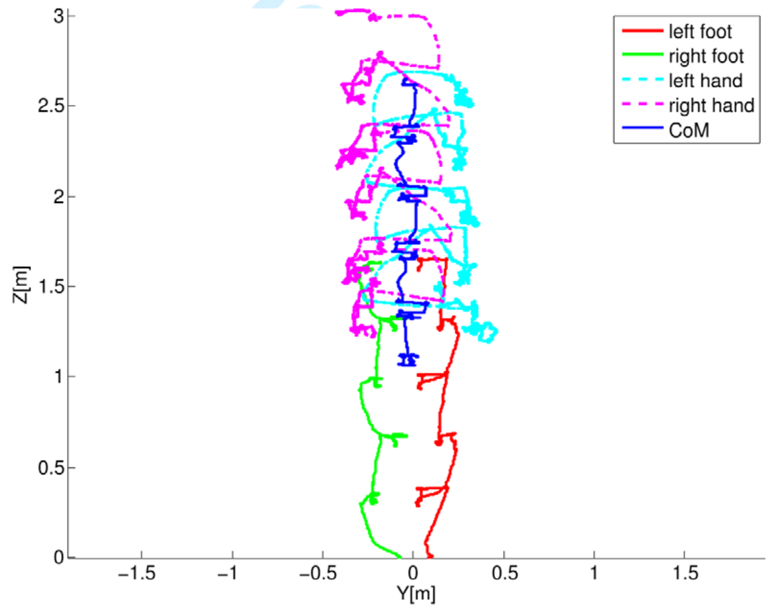
One contribution with this work is demonstrating that we can use online multilevel optimization to tackle a wide range of practical humanoid problems. By decomposing into a simplified behavior level controller that only reasons about relevant aspects of a task, and a generic low-level full body controller that tracks the high-level commands while filling in the details, we can focus on designing high-level controllers and rapidly develop for several applications simultaneously. On the other hand, we did not push for task level autonomy in the DRC Trials. Most of our desired high-level trajectories are manually scripted offline, and simply played back on the robot using the controller, which ignores most of the external environment. For the DRC in 2015, we are working on integrating autonomous perception and planning (Berenson, Srinivasa, Ferguson, & Kuffner, 2009) to generate more intelligent high-level trajectories and avoid collisions.

Reasoning about CoM motion alone appears to be sufficient to guide the inverse dynamics controller through moderate rough terrain with height and surface normal changes. We were surprised that taking into account angular momentum or swing leg dynamics in the high-level model was not necessary. Another interesting discovery is that explicitly adding toe-off and heel-strike is easy. We have shown that, with a very straightforward design, we can achieve these behaviors and push step length and walking speed closer to those of humans in simulation.

The current walking controller does restrict the upper body to a predefined policy for its orientation. On the other hand, we believe freeing the upper body and utilizing its angular momentum will drastically increase the stability



(a) Measured limb and CoM trajectories in the XZ plane



(b) Measured limb and CoM trajectories in the YZ plane

Figure 11. These two plots show the Atlas robot climbing the first five treads during the actual run at the DRC Trials. The X axis is the forward direction, Y points to the robot's left, and Z points upward. Left and right foot positions are shown with red and green solid lines, and left and right hand positions are plotted with cyan and magenta dashed lines. The center of mass is shown with a solid blue line

margin, especially in the single support phase. The current walking controller is also maintaining constant CoM height throughout the entire walking cycle when on level ground, which is undesirable for balancing or agility. An immediate line of research is to expand the point mass model to a more sophisticated one. We can include angular momentum and orientation (Whitman, Stephens, & Atkeson, 2012) and swing leg dynamics. We can also use a dead-beat controller for the 3D spring loaded inverted pendulum (SLIP) model (Wensing & Orin, 2013b; Wu & Geyer, 2013) as the high-level controller to capitalize on extensive research of robust running and walking using a 3D SLIP model. We will explore including the foot step locations and timing as part of the trajectory optimization. These choices can be treated as parameters in DDP, and they can be optimized along with the trajectory. The foot step planner would output footstep cost functions, and the trajectory optimizer would have the freedom to pick the foot steps that are the most suitable. This is one way to have the footstep planner implicitly take 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 value function as part of the optimization criteria to incorporate a notion of the future similar to Kuindersma et al. (2014). One way to generate this value function is to perform full body DDP on a periodic walking pattern (Liu, Atkeson, & Su, 2013). We can also utilize the value function from the CoM trajectory optimization part.

In general, we find that inverse dynamics plays an important role for heavily loaded joints such as the stance leg, particularly the ankle joints for a precise center of pressure control. However, on lightly loaded limbs (e.g., swing leg or arms), friction and unmodeled dynamics tend to dominate the rigid-body model torques that inverse dynamics produces, so inverse kinematics is more helpful for swing leg and arm tracking. We considered explicitly switching between control modes as the contact state changed, but the switching transients are hard to manage. Performing both inverse dynamics and inverse kinematics concurrently for all joints eliminated the switching issue, and we found that the appropriate control mode dominates at the appropriate time. In comparison, another approach is to solve inverse dynamics alone and integrate desired accelerations into desired velocities and positions. When faced with substantial modeling errors and delays, this approach can easily lead to inaccurate behaviors. We think one principled approach to resolve this issue is to implement Receding Horizon Control with the full dynamic model similar to Tassa, Erez, & Todorov (2012) and Erez et al. (2013), which is close to, but not yet computationally tractable for implementing on a robot in a real-time setting.

Since our full body controller decouples inverse dynamics and inverse kinematics into two independent quadratic programs, consistency becomes a concern. We can introduce additional terms in the cost function of the inverse kinematics to bias solutions toward accelerations computed by inverse dynamics. This method, when taken to its extreme, is equivalent to just integrating the acceleration. Empirically, results from both quadratic programs are consistent with each other when the overall controller is tracking the desired trajectory well, since the high-level commands are always consistent. On the other hand, when either is unable to track the desired trajectory due to constraint violation, the other is often unaware of the situation. For example, if the inverse dynamics controller is unable to produce the desired CoM acceleration due to limited ground forces because of friction cone constraints, the CoM will start diverging from the desired trajectory. However, such constraints are not implemented in the inverse kinematics controller, and it will keep tracking the desired CoM velocity. Once the divergence starts, the inverse dynamics controller will demand more CoM acceleration because of the increasing position error, which is impossible to achieve because of the constraints. Since both the ID and IK controllers are local greedy optimizers, they do not have the ability to correct this problem alone. We expect that a successful solution to this problem will involve the high-level controller foreseeing and preventing these situations by considering both kinematic and dynamic constraints during the planning stage.

Due to the tight time line for the DRC Trials, we have not conducted systematic system identification procedures on the robot. We hope to increase the quality of both kinematic and dynamic models in the near future. As a matter of fact, all the leg joint level sensing on the Atlas robot, such as positioning, velocity (numerically differentiated from position), and torque, are pretransmission. This hardware design choice reduces jitter in the low-level joint control, but it introduces problems for forward kinematics and torque control. Better state estimation is necessary to achieve more accurate position tracking and force control.

11. CONCLUSIONS

We have designed a full body controller for a full size humanoid robot with online optimization. A high-level controller guides the robot with trajectories that are optimized online using simplified models. A low-level controller solves full body floating base inverse dynamics and inverse kinematics by formulating each as a quadratic programming problem. Inverse dynamics provides us with a tool to perform compliant motions and dynamic behaviors. Inverse kinematics helps us to battle modeling errors, and it makes the overall controller applicable to real hardware. The controller was successfully tested on the Atlas robot in rough terrain walking, ladder climbing, and full body manipulation in the DARPA Robotics Challenge Trials.

ACKNOWLEDGMENTS

This material is based upon work supported in part by the U.S. National Science Foundation (ECCS-0824077 and IIS-0964581) and the DARPA M3 and Robotics Challenge programs.

REFERENCES

Berenson, D., Srinivasa, S., Ferguson, D., & Kuffner, J. (2009). Manipulation planning on constraint manifolds. In IEEE International Conference on Robotics and Automation (pp. 625–632), Kobe, Japan.

Bouyarmane, K., & Kheddar, A. (2011). Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 4414–4419), San Francisco.

Bouyarmane, K., Vaillant, J., Keith, F., & Kheddar, A. (2012). Exploring humanoid robots locomotion capabilities in virtual disaster response scenarios. In 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 337–342), Osaka, Japan.

de Lasa, M., Mordatch, I., & Hertzmann, A. (2010). Feature-based locomotion controllers. ACM Transactions on Graphics, 29(4), 131:1–131:10.

Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., & Todorov, E. (2013). An integrated system for real-time model-predictive control of humanoid robots. In 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA.

Escande, A., Mansard, N., & Wieber, P.-B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. The International Journal of Robotics Research, 33(7), 1006–1028.

Faraji, S., Pouya, S., & Ijspeert, A. (2014). Robust and agile 3D biped walking with steering capability using a footstep predictive approach. In Robotics: Science and Systems (RSS), Berkeley, CA.

Feng, S., Xinjilefu, X., Huang, W., & Atkeson, C. (2013). 3D walking based on online optimization. In 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA.

Herzog, A., Righetti, L., Grimminger, F., Pastor, P., & Schaal, S. (2014). Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Chicago.

Hirai, K., Hirose, M., Haikawa, Y., & Takenaka, T. (1998). The development of Honda humanoid robot. In Proceedings of the IEEE International Conference on Robotics and Automation (vol. 2, pp. 1321–1326), Leuven, Belgium.

Huang, W., Kim, J., & Atkeson, C. (2013). Energy-based optimal step planning for humanoids. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 3124–3129), Karlsruhe, Germany.

Hutter, M., Hoepflinger, M. A., Gehring, C., Bloesch, M., Remy, C. D., & Siegwart, R. (2012). Hybrid operational space control for compliant legged systems. In Robotics: Science and Systems (RSS), Sydney, NSW, Australia.

Hutter, M., Sommer, H., Gehring, C., Hoepflinger, M., Bloesch, M., & Siegwart, R. (2014). Quadrupedal locomotion using hierarchical operational space control. The International Journal of Robotics Research, 33(8), 1047–1062.

Jacobson, D. H., & Mayne, D. Q. (1970). Differential dynamic programming. Amsterdam: Elsevier.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In Proceedings of the IEEE International Conference on Robotics and Automation (vol. 2, pp. 1620–1626), Taipei, China.

Kanoun, O., Lamiraux, F., & Wieber, P. B. (2011). Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality tasks. IEEE Transactions on Robotics, 27(4), 785–792.

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal of Robotics and Automation, 3(1), 43–53.

Koolen, T., Smith, J., Thomas, G., Bertrand, S., Carff, J., Mertins, N., Stephen, D., Abeles, P., Englsberger, J., McCrory, S., van Egmond, J., Griffioen, M., Floyd, M., Kobus, S., Manor, N., Alsheikh, S., Duran, D., Bunch, L., Morphis, E., Colasanto, L., Hoang, K.-L. H., Layton, B., Neuhaus, P., Johnson, M., & Pratt, J. (2013). Summary of Team IHMC's Virtual Robotics Challenge entry. In 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA.

Kuindersma, S., Permenter, F., & Tedrake, R. (2014). An efficiently solvable quadratic program for stabilizing dynamic locomotion. In IEEE International Conference on Robotics and Automation, Hong Kong, China.

Lee, S.-H., & Goswami, A. (2010). Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3157–3162), Taipei, China.

Liu, C., Atkeson, C. G., & Su, J. (2013). Biped walking control using a trajectory library. Robotica, 31, 311–322.

Mistry, M., Buchli, J., & Schaal, S. (2010). Inverse dynamics control of floating base systems using orthogonal decomposition. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 3406–3412), Anchorage, AK.

Mistry, M., Nakanishi, J., Cheng, G., & Schaal, S. (2008). Inverse kinematics with floating base and constraints for full body humanoid robot control. In 8th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 22–27), Daejung, Korea.

Moro, F. L., Gienger, M., Goswami, A., Tsagarakis, N. G., & Caldwell, D. G. (2013). An attractor-based whole-body motion control (WBMC) system for humanoid robots. In 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA.

Nakamura, Y., & Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. Journal of Dynamic Systems, Measurement, and Controls, 108, 163–171.

Orin, D., & Goswami, A. (2008). Centroidal momentum matrix of a humanoid robot: Structure and properties. In IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 653–659), Nice, France.

Ott, C., Roa, M., & Hirzinger, G. (2011). Posture and balance control for biped robots based on contact force optimization. In 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 26–33), Bled, Slovenia.

Pratt, J., Carff, J., Drakunov, S., & Goswami, A. (2006). Capture point: A step toward humanoid push recovery. In 6th

IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 200–207), Genoa, Italy.

Ramos, O., Mansard, N., Stasse, O., & Soueres, P. (2012). Walking on non-planar surfaces using an inverse dynamic stack of tasks. In 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 829–834), Osaka, Japan.

Righetti, L., Buchli, J., Mistry, M., Kalakrishnan, M., & Schaal, S. (2013). Optimal distribution of contact forces with inverse-dynamics control. The International Journal of Robotics Research, 32(3), 280–298.

Righetti, L., Buchli, J., Mistry, M., & Schaal, S. (2011). Inverse dynamics control of floating-base robots with external constraints: A unified view. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 1085–1090), Shanghai, China.

Saab, L., Ramos, O., Keith, F., Mansard, N., Soueres, P., & Fourquet, J. (2013). Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. IEEE Transactions on Robotics, 29(2), 346–362.

Sentis, L., & Khatib, O. (2006). A whole-body control framework for humanoids operating in human environments. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation (pp. 2641–2648), Orlando, FL.

Sentis, L., Park, J., & Khatib, O. (2010). Compliant control of multicontact and center-of-mass behaviors in humanoid robots. IEEE Transactions on Robotics, 26(3), 483–501.

Stephens, B. (2011). Push recovery control for force-controlled humanoid robots. Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Tassa, Y., Erez, T., & Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 4906–4913), Vilamoura, Portugal.

Urata, J., Nshiwaki, K., Nakanishi, Y., Okada, K., Kagami, S., & Inaba, M. (2012). Online walking pattern generation for push recovery and minimum delay to commanded change of direction and speed. In IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3411–3416).

Vukobratović, M., & Borovac, B. (2004). Zero-moment point—Thirty five years of its life. International Journal of Humanoid Robotics, 01(01), 157–173.

Wampler, C. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. IEEE Transactions on Systems, Man and Cybernetics, 16(1), 93–101.

Wensing, P., & Orin, D. (2013a). Generation of dynamic humanoid behaviors through task-space control with conic optimization. In IEEE International Conference on Robotics and Automation (ICRA) (pp. 3103–3109), Karlsruhe, Germany.

Wensing, P., & Orin, D. (2013b). High-speed humanoid running through control with a 3d-slip model. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5134–5140), Tokyo, Japan.

Whitman, E. (2013). Coordination of multiple dynamic programming policies for control of bipedal walking. Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Whitman, E., & Atkeson, C. (2010). Control of instantaneously coupled systems applied to humanoid walking. In 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 210–217), Nashville, TN.

Whitman, E., Stephens, B., & Atkeson, C. (2012). Torso rotation for push recovery using a simple change of variables. In 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (pp. 50–56), Osaka, Japan.

Wu, A., & Geyer, H. (2013). The 3-D spring-mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments. IEEE Transactions on Robotics, 29(5), 1114–1124.

Xinjilefu, X., Feng, S., Huang, W., & Atkeson, C. (2014). Decoupled state estimation for humanoids using full-body dynamics. In IEEE International Conference on Robotics and Automation, Hong Kong, China.

Zapolsky, S., Drumwright, E., Havoutis, I., Buchli, J., & Semini, C. (2013). Inverse dynamics for a quadruped robot locomoting along slippery surfaces. In International Conference on Climbing and Walking Robots (CLAWAR), Sydney, Australia.