2014 14th IEEE-RAS International Conference on
Humanoid Robots (Humanoids)
November 18-20, 2014. Madrid, Spain

Optimization Based Full Body Control for the Atlas Robot

Siyuan Feng†, Eric Whitman‡, X Xinjilefu† and Christopher G. Atkeson†

***Abstract*— One popular approach to controlling humanoid robots is through inverse kinematics (IK) with stiff joint position tracking. On the other hand, inverse dynamics (ID) based approaches have gained increasing acceptance by providing compliant motions and robustness to external perturbations. However, the performance of such methods is heavily dependent on high quality dynamic models, which are often very difficult to produce for a physical robot. IK approaches only require kinematic models, which are much easier to generate in practice. In this paper, we supplement our previous work with ID-based controllers by adding IK, which helps compensate for modeling errors. The proposed full body controller is applied to three tasks in the DARPA Robotics Challenge (DRC) Trials in Dec. 2013.**

I. Introduction

Many humanoid applications can be decomposed into a two stage control problem: a behavior level controller that outputs high level commands and a low level controller that is responsible for generating joint commands. We believe that in order to fully utilize the workspace and be robust to external perturbations, the low level controller has to take full body kinematics and dynamics into consideration. In this paper, we present such a controller that solves full body inverse dynamics (ID) and inverse kinematics (IK) at each time step to track higher level objectives. Figure 1 shows a block diagram for the overall system. Both ID and IK are formulated as two separate Quadratic Programming (QP) problems, each with their own objectives and constraints.[1]

On our Atlas robot, a 28 degrees of freedom hydraulic robot built by Boston Dynamics, joint level servos compute valve commands based on

$$i = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + K_f(\tau_d - \tau) + c, \quad (1)$$

where $q_d, \dot{q}_d, \tau_d$ are desired joint position, velocity and torque, $q, \dot{q}, \tau$ are measured, and c contains the constant valve bias term plus some other auxiliary feedback terms. This joint level servo runs at $1kHz$, while we can update $q_d, \dot{q}_d$ and τ_d at $333Hz$. In previous work [1], [2], [3], we focused on torque control with ID that computes τ_d. To take full advantage of the on-board high bandwidth PD servo, we need to generate reference q_d and $\dot{q}_d$.

Using full body inverse dynamics for force control has become a popular topic in recent humanoid research. Much of the research originates from [4]. Within this broad category, reference motions in task space are specified, and convex optimization is used 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 QP problem. [5], [6], [7], [8], [9], [10], [11] explore 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. We prefer using soft costs as opposed to hard constraints for better numerical stability especially in situations where no feasible solutions exist for the specified objectives. There is also much interest in formulating a smaller QP problem to reduce computation time. Contact forces can be removed from the equations of motion using orthogonal decomposition [12]. Ott et al. [13] use simple PD servos to generate a desired net ground reaction wrench, which is then distributed among predefined contacts using optimization. [14] solves a smaller QP with decoupled dynamics. [15] has 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 square problem is solved for the state acceleration. Joint torques are generated explicitly. [16] generates desired centroidal momenta change based on instantaneous capture points, and uses QP to optimize for acceleration and contact forces. Joint torques are then generated with explicit inverse dynamics. [17] is similar in terms of optimization variables and torque

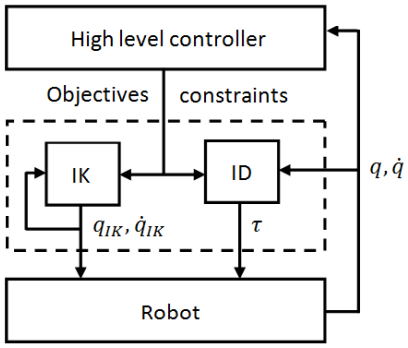


Fig. 1. The task dependent high level controller generates a set of desired objectives such as CoM or limb motion, and constraints such as CoP and joint limits. The proposed full body controller, which is contained by the dashed rectangle, takes the high level objectives and robot states $(q, \dot{q})$ as inputs and outputs position q_{IK}, velocity $\dot{q}_{IK}$ and torque τ for each joint, which are used as desired values, $q_d, \dot{q}_d$ and τ_d, in Eq. 1. Note that IK uses its own internal states rather than the measured robot states.

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

‡ is with Boston Dynamics / Google, 78 Fourth Avenue, Waltham, MA 02451, ewhitman@bostondynamics.com. Dr. Whitman did this work when he was a post-doc fellow at CMU.

[1] A video of Atlas doing the described tasks is at `http://www.cs.cmu.edu/~sfeng/sfew_hum14.mp4`

- invalid link

978-1-4799-7174-9/14/$31.00 ©2014 IEEE

+ detailed literature review

generation, but a novel QP solver is implemented to exploit the observation that inequality constraints rarely change in this context.

We continue to use the same approach to ID that was previously developed in our group [1], [2], [3] that is similar to [9], [18]. We optimize for acceleration, torque, and contact forces simultaneously on the full robot model. This design choice is intuitive, and gives us the most flexibility in terms of trading off directly among physical quantities of interest. It also provides an easy way to properly manage all constraints on contact forces and joint torques. It does result in a bigger QP problem, but is still solvable in real time with a standard QP solver.

ID does not generate q_d or $\dot{q}_d$ directly for Eq. 1. We could integrate $\ddot{q}$, the output from ID, into $\dot{q}_d$ and q_d, but we find this rapidly leads to joint limit violation and instability due to delays and modeling errors. Thus, a separate inverse kinematics (IK) controller is introduced. Similar to ID, our IK is also formulated as a QP, where the unknowns are the generalized velocities, $\dot{q}$, of the floating base and all the joints. At each time step, we solve for $\dot{q}$ that obeys kinematic constraints and minimizes a combination of costs. q is computed by integrating $\dot{q}$. Our approach is similar in spirit to [19], in which $\dot{q}$ is computed by inverting a matrix composed of end effector and contact constraint Jacobian. Solving for $\dot{q}$ and integrating to obtain q is the primary difference between our approach and most traditional IK approaches such as [20], [21]. The incremental method can get stuck in local minima, but it does not produce discontinuous results.

Main Focus

The main contribution of this work is the successful application to a physical system and many modifications of existing techniques, as well as the lessons learned and intellectual challenges that arise from this endeavor.

II. Full Body Control

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 proposed low level controller takes these as inputs, and computes physical quantities for each individual joint such as joint position, velocity and torque. These outputs are then used as references in the joint level servos on the robot. Figure 1 shows a block diagram for the overall system. Joint position and velocity are computed separately from joint acceleration and torque. We refer to the former problem as inverse kinematics (IK) and the latter as inverse dynamics (ID). Both are formulated as Quadratic Programming (QP) problems.

+ makes a distinction

$$\begin{aligned} \min_{\mathcal{X}} \ & 0.5\mathcal{X}^T G\mathcal{X} + g^T\mathcal{X} \\ s.t. \ & C_E\mathcal{X} + c_E = 0 \\ & C_I\mathcal{X} + c_I \geq 0. \end{aligned} \tag{2}$$

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 $3ms$ control loop with a standard solver.

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

$$A = \begin{bmatrix} w_0 A_1 \\ w_1 A_1 \\ \vdots \\ w_n A_n \end{bmatrix}, b = \begin{bmatrix} w_0 b_0 \\ w_1 b_1 \\ \vdots \\ w_n b_n \end{bmatrix}. \tag{3}$$

Each row emphasizes a certain desired behavior with weight, w_i.

III. Inverse Dynamics

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

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

where $(q, \dot{q})$ is the full state of the system including a six DOF 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, where the first six rows that correspond to the six DOF 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, λ is a vector of all contact wrenches in the world frame, and x is a vector of contact position and orientation in Cartesian space. λ and J^T's dimensions depend on the number of contacts.

We can rewrite the equations of motion as

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \end{bmatrix} + h(q,\dot{q}) = 0. \tag{4}$$

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

For ID, $\mathcal{X} = \begin{bmatrix} \ddot{q} & \tau & \lambda \end{bmatrix}^T$. Eq. 4 is enforced as equality constraints. The inequality constraints consist of joint torque limits, contact force limits due to friction cone constraints, and contact torque limits due to center of pressure (CoP) remaining in the support polygon.

A. Cost function

We list a few examples of the objectives that can be plugged into the rows of Eq. 3.

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

$$\begin{aligned} A_{cart} &= \begin{bmatrix} J(q) & 0 & 0 \end{bmatrix} \\ b_{cart} &= \ddot{x}^* - \dot{J}(q,\dot{q})\dot{q}. \end{aligned} \tag{5}$$

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

$$\ddot{x}^* = K_p(x_d^* - x) + K_d(\dot{x}_d^* - \dot{x}) + \ddot{x}_d^*, \tag{6}$$

where x_d^*, $\dot{x}_d^*$ and $\ddot{x}_d^*$ are specified by a higher level controller, and x and $\dot{x}$ are computed by forward kinematics based on

the estimated current robot state. Many objectives such as CoM, hand, foot and torso motions are specified with this form. Depending on the tasks, we sometimes drop the rows in Eq. 5 that we do not want to track.

Rather than treating contacts as hard constraints, we find that using a soft penalty with a high weight is generally more numerically stable. For such contact costs, we set $\ddot{x}^* = 0$ in Eq. 6.

2) Center of pressure tracking: CoP in the foot frame is

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

where ${}^bM$ and ${}^bF$ are contact forces and torques in the foot frame. We can penalize CoP deviation with

$$A_{cop} = \begin{bmatrix} 0 & 0 & \begin{bmatrix} 0 & 0 & p_x^* & 0 & 1 & 0 \\ 0 & 0 & p_y^* & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} \end{bmatrix}$$
$$b_{cop} = 0,$$

where (p_x^*, p_y^*) is the desired CoP in the foot frame given by a high level controller, and R is the rotation matrix from the world frame to the foot frame.

3) Weight distribution: In double support, we specify a desired weight distribution by $w^* = F_{zl}/(F_{zl} + F_{zr})$, where F_{zl} and F_{zr} are the vertical forces at the left and right foot. We add this term to the cost function with

$$A_{weigt} = \begin{bmatrix} 0 & 0 & S_{weight} \end{bmatrix}$$
$$b_{weigt} = 0.$$

S_{weight} is a row vector with zeros, except $S_{weight}(i_{Fzl}) = 1 - w^*$ and $S_{weight}(i_{Fzr}) = -w^*$, where i_{Fzl} and i_{Fzr} are indices correspond to Fzl and Fzr.

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

$$A_{state} = I$$
$$b_{state} = \begin{bmatrix} \ddot{q}^* & \tau^* & \lambda^* \end{bmatrix}^T. \tag{7}$$

Zero is used if no target value is specified except for the vertical forces, which are regularized using gravitational forces. This term is useful for directly controlling specific joints or forces. It also regularizes $\mathcal{X}$ to make the QP problem well conditioned.

5) Change in torques: To avoid high frequency oscillations in torque commands, we penalize changes in τ with

$$A_{d\tau} = \begin{bmatrix} 0 & I & 0 \end{bmatrix}$$
$$b_{d\tau} = \tau_{prev},$$

where τ_{prev} is from the last time step.

B. Constraints

Eq. 4 is used as equality 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 forces and torques at the feet need to obey CoP constraints,

$$d_x^- \leq -{}^bM_y/{}^bF_z \leq d_x^+$$
$$d_y^- \leq {}^b M_x/{}^bF_z \leq d_y^+, \tag{8}$$

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

IV. Inverse Kinematics

Unlike traditional IK approaches that generate positions for the entire desired trajectory ahead of time, we compute velocities at each time step and integrate them to get positions. The controller can be more responsive to changes in the high level commands, and computation is averaged across the course of motion.

For the IK QP, $\mathcal{X} = \dot{q}$, and the numerically integrated floating base position and joint position is denoted by q_{ik}. Our IK formulation is very similar to ID's except that the internal states are used for forward kinematics and generating target velocities. Separate internal states within IK are necessary to accumulate previous correctional velocities. This can lead to divergence between IK's internal state and the state estimator's, particularly the root position. Resetting IK's state to match the state estimator's is one option, but it introduces sudden discrete changes. We use "anchor" points instead, which is detailed in Section IV-C. The internal states are set to the real robot states in the initialization stage. All the internal states are denoted with subscripts $_{ik}$.

- not very clear

A. Cost function

We list a few examples of the objectives that can be plugged into Eq. 3.

1) Cartesian space velocity: We penalize deviation from the desired Cartesian velocity, $\dot{x}^*$, with

$$A_{cart} = J(q_{ik})$$
$$b_{cart} = \dot{x}^*, \tag{9}$$

where

$$\dot{x}^* = K_p(x_d^* - x_{ik}) + \dot{x}_d^*. \tag{10}$$

We use a different set of K_p here than in ID. Cartesian space tracking and contacts are both handled with Eq. 9 and Eq. 10. Although the weights and desired targets, x_d^* and $\dot{x}_d^*$ are different.

2) Direct tracking and regularization:

$$A_{state} = I$$
$$b_{state} = \dot{q}^*, \tag{11}$$

where $\dot{q}^*$ can be target joint velocity or 0 for regularization.

3) Change in velocity:

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

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

B. Constraints

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

The joint limit constraints are

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

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

$$x^- \leq x_{ik} + J(q_{ik})\dot{q}dt \leq x^+, \tag{12}$$

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 a Jacobian matrix.

C. Anchor points

Unlike ID, where we demand zero acceleration for the contacts, we introduce desired position targets in IK, which we refer to as anchor points. Since IK only relies on its internal states, without any external feedback, the internal states will diverge from the state estimator's. For example, suppose we wish to take a step of some specific length. The IK controller will hit the desired foot step almost perfectly if not violating any constraints. The real robot will not due to tracking errors. After taking several steps, the IK controller will end up in a significantly different location than the state estimator. Because the high level controllers generate desired Cartesian targets based on the state estimator, we need to compensate for this divergence within the IK controller. In order to tie IK's internal states to the state estimator's, we specify a Cartesian target, x^*_{anchor}, that slowly tracks the estimated contact pose, $x_{contact}$.

$$x^*_{anchor} = \alpha x_{contact} + (1-\alpha)x^*_{anchor}. \tag{13}$$

In Eq. 10, x^*_{anchor} replaces x^*_d, and $\dot{x}^*_d = 0$. $\alpha = 0.005$ for static walking. x^*_{anchor} is initialized to IK's internal value upon establishing the contact. Since x^*_{anchor} contains all the information about long term tracking error and state estimator drift, and IK tracks x^*_{anchor} well, we can use x^*_{anchor} to update IK's internal states to match the state estimator's. ID is not affected since target contact acceleration is always set to zero in Eq. 6.

V. Applications

The proposed full body controller is tested on Boston Dynamics's Atlas robot in the DARPA Robotics Challenge. Atlas has 28 hydraulic actuators, six for each leg and arm, three for the back joints, and one for neck pitch. Our rough terrain walking, ladder climbing and full body manipulation controllers are all targeted for it. For all three applications, the state estimator is based on [22].

A. Static walking

For the DRC Trials, we used a simple static walking strategy. Desired CoM motion is represented with a quintic spline. Swing foot trajectory is generated with piecewise splines. Figure 2 shows snapshots of the Atlas robot traversing piled cinder blocks with tilted tops, and CoM and feet trajectories are plotted in Figure 3. The desired CoP trajectory is generated using a Linear Inverted Pendulum Model (LIPM). Figure 4 shows CoP tracking for the Atlas robot stepping up piled cinder blocks.

Most other teams at the DRC Trials build a map with laser point clouds and select foot steps with either a human operator or some combination of heuristics and simple planning. Because of forward kinematic errors in the robot, registering laser points while moving does not work well. The robot has to be stationary to generate high fidelity maps. To avoid extensive periods of standing still, we provide our human operator with a live camera stream augmented with the current swing foot pose computed from forward kinematics, and let the operator "nudge" the swing foot around in Cartesian space by commanding offsets in desired foot position and orientation. Once the operator is satisfied with the foot pose, a "continue" command is issued, the robot puts the swing foot straight down until ground contact. On the downside, our approach requires more (and more difficult) input from the operator, and has much longer single support phase when the operator commands are issued.

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

1) Ankle torque controlled: To fully control CoP for better balancing, we control the stance ankles in pure torque mode. IK solutions for the stance ankle joints are ignored. The downside is that the ankle position errors propagate up the kinematic chain, and result in errors in swing foot tracking. An integrator on the desired swing foot position is used to compensate for this.

$$\begin{aligned} err_{swing} &= err_{swing} + K_i(x'_{swing_d} - x_{swing}) \\ x^*_{swing_d} &= x'_{swing_d} + err_{swing}, \end{aligned} \tag{14}$$

where x'_{swing_d} is the desired swing foot position from the operator or a foot step planner, x_{swing} is the computed position from forward kinematics, and $x^*_{swing_d}$ is used as inputs in Eq. 6 and Eq. 10.

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, toe-off is triggered when the rear knee approaches the joint angle limit (straight knee). During toe-off, we shift the rear foot reference point, where the Jacobian is computed, to the toe. The contact cost terms in Eq. 5 and Eq. 9 for the rear foot are modified. Position terms remain the same. For the rotational terms, we first

+ addresses a problem

method

+ points out shortcomings

+ explains using an example

+ states downside, and a way around it

+ states a problem, and a way around it

method

123

+ well-explained methods

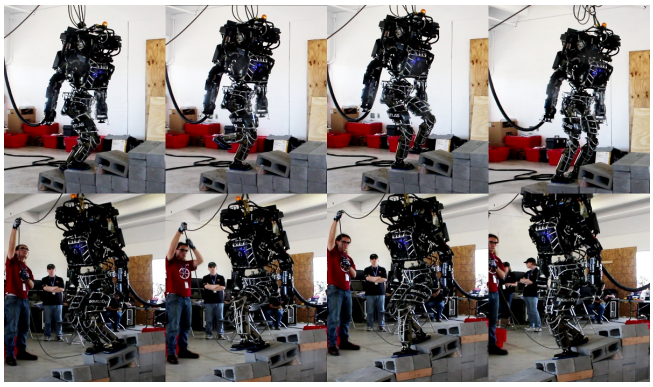


Fig. 2. These photos show the Atlas robot practicing for the segment 3 of the terrain task for the DRC Trials. The snapshots were taken every 5 seconds.

compute

$$\begin{aligned} A^{ID}_{Rot} &= \begin{bmatrix} RJ_{rot}(q) & 0 & 0 \end{bmatrix} \\ b^{ID}_{Rot} &= R(\ddot{x}^*_{Rot} - \dot{J}_{rot}(q,\dot{q})\dot{q}) \\ A^{IK}_{Rot} &= RJ_{rot}(q_{ik}) \\ b^{IK}_{Rot} &= R\dot{x}^*_{Rot} \end{aligned} \tag{15}$$

for ID and IK respectively, where R is the rotation matrix from world to foot frame. Only the first and third rows in Eq. 15, that correspond to roll and yaw motion are added to the cost functions. $\ddot{x}^*_{Rot}$ is set to zero, and $\dot{x}^*_{Rot}$ is computed with Eq. 13. We also set $d_y^- = d_y^+ = 0$ in Eq. 8. This effectively turns the rear foot contact into an unactuated pin joint around the pitch axis. A slightly bent rear knee angle is given as desired in Eq. 7 and Eq. 11 as well.

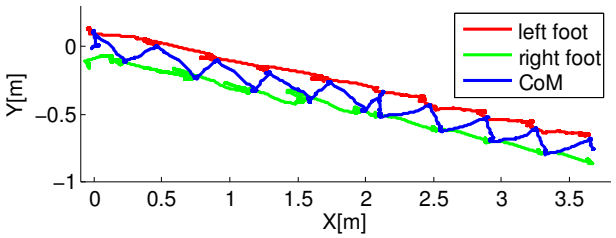
3) Integrator for desired CoM offset: Assuming the robot is moving slowly enough, we can approximate the true CoM with the measured CoP, and compute $CoM_{err} = CoP - CoM_{model}$. During the second half of double support and single support phase, we integrate CoM_{err} to offset the desired CoM so that the true CoM is at the middle of the supporting foot. The integrator is set up similarly to Eq. 14.

B. Full body manipulation

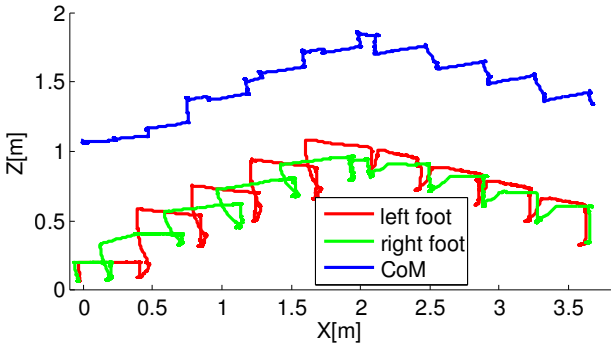
During full body manipulation, the operator gives commands requesting either direct joint angles or target Cartesian poses for one or both hands. These commands are used as x^*_d in Eq. 6 and Eq. 10. We use equality constraints in the IK QP to enforce directly specified joint angles. For large Cartesian motions, the new target is connected to the current target through splines. For small motions, we use the "nudge" method as described above for precise foot placement. Figure 7 shows a picture of the Atlas robot performing the valve task during the DRC Trials.

We make a few small changes to the basic full body controller:

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 IK position diverging from the estimated robot position. However, the leaky integrator in Eq. 13 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 IK prevent further sliding. However, when we constantly update the IK to the measured position, it can then constantly slide farther. We therefore disable this integrator during manipulation.



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



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

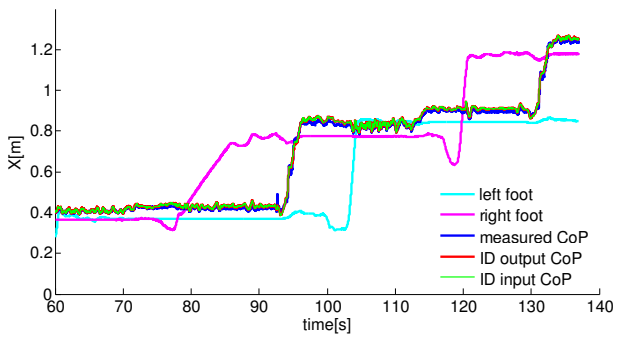
Fig. 3. These plots show the Atlas robot traversing segment 3 of the terrain task. X axis is the forward direction, Y points to the robot's left, and Z points upward. Left and right foot positions are shown with red and green lines, and center of mass is plotted in blue. The robot walks in a straight line in reality. Without external observation, our state estimator drifts significantly as shown in the top plot.

2) Allowing rotation: For some tasks, we only care about the position of the hand, and the orientation is unimportant. For such cases, we can remove the rotational rows entirely in Eq. 5 and Eq. 9. 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 must maintain its position and keeping that axis normal to the wall. Allowing the controller the freedom to rotate around one axis can drastically increase the available workspace. The modification is similar to Eq. 15. R is constructed from a basis of three orthogonal unit vectors including the desired free-to-rotate-about axis, and the row that corresponds to the rotation axis is dropped.

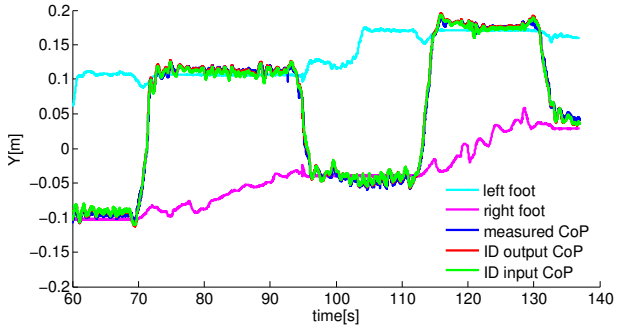
+ explains using an example

C. Ladder climbing

The underlying controller for ladder climbing is similar to that used for manipulation, but the majority of the motion is scripted ahead of time with only the final placement of the hands and feet controlled by the operator. Figure 5 shows



(a) CoP in X axis



(b) CoP in Y axis

Fig. 4. 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). This data was 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 with dashed green. ID's output CoP is shown with 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 $1cm$.

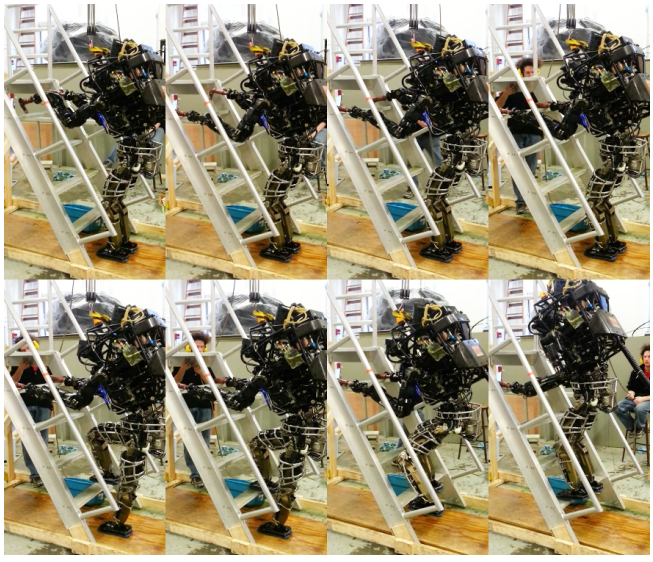
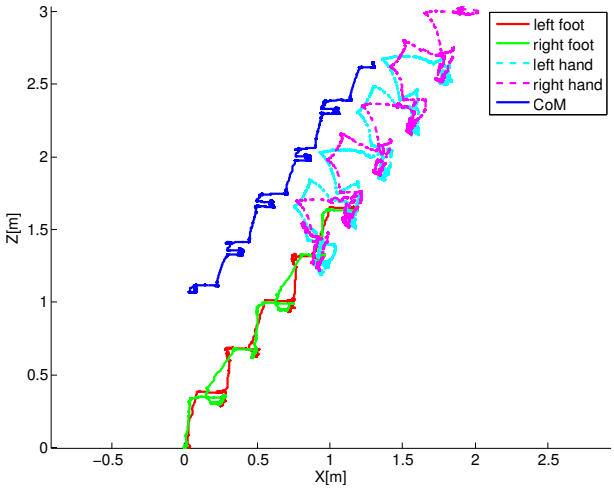
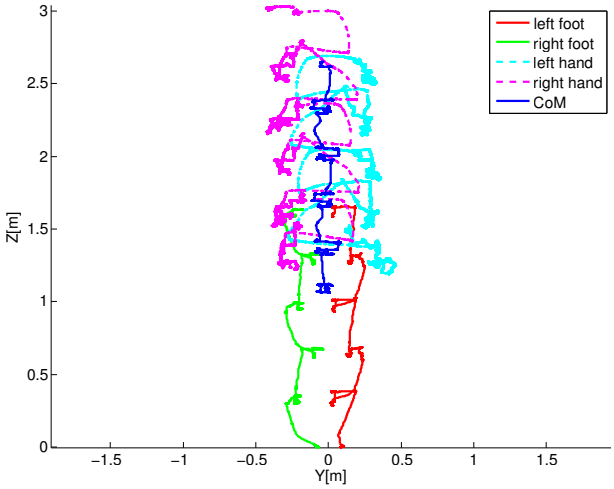


Fig. 5. These photos show the Atlas robot climbing the top half of the same ladder as used in the DRC Trials. The snapshots were taken every 13 seconds. The top row shows re-positioning of the hook hands, and the bottom row shows stepping up one tread. Most of the climbing motions are scripted. After each limb's rough re-positioning, the operator can fine adjust its final position with "nudge" commands that are small offsets in Cartesian space.



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



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

Fig. 6. These two plots show the Atlas robot climbing the first five treads during the actual run at the DRC Trials. 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 in cyan and magenta dashed lines. Center of mass is shown with solid blue line. While approaching the ladder, the robot has its arms outside of the ladder railings, and we have to first raise both arms all the way up and around to get them both between the railings. The swinging motions at the beginning of the hand trajectories are results of this motion.

snapshots of a complete cycle of the Atlas robot climbing the ladder. CoM and limb trajectories from the actual run during the DRC Trials are plotted in Figure 6. 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 $1cm$ 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.

Once on the steps, only the toes of the feet are supported, so we adjust the CoP constraint accordingly in Eq. 8. Having all of the weight on the toes makes the robot vulnerable to rotational slipping, causing unexpected yaw rotations. After

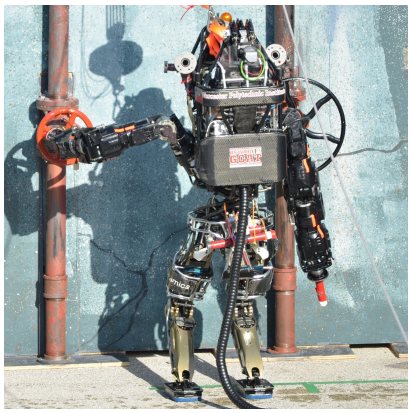


Fig. 7. The proposed low level controller is applied on the Atlas robot to turn a valve in the DRC Trials.

such slip occurs, we need to inform the IK controller. This issue is analogous the root position divergence problem in the walking case, but for orientation instead. In order to correctly place the hands on the next rung to recover from such rotations, we must rotate the IK solution to match the measured orientation. We therefore periodically rotate the IK solution such that the feet are aligned with the measured feet orientations, and then the robot will reorient its upper body towards 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 in Section V-B.1. This reorienting serves a similar purpose to Eq. 13, but for rotation instead of translation. To avoid chase conditions, this modification is disabled if there is not significant (about 20%) weight on the foot.

1) Elbow management: We climb the ladder by placing the hands (hooks made from pipes) on the steps. 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 rung) and adding inequality constraints to the IK using Eq. 12. 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 (in only the lateral direction) with a low weight. To prevent the problem from becoming over-constrained by elbow management, we use low weights for commanding hand orientation. Specifically, we use Eq. 15 to rotate the hand orientation equations into a basis containing the hand axis, a pitch-like vector, and a yaw-like vector. We use a very low, but non-linear weight for rotation about the hand axis (roll-like), allowing it roll about 45 degrees nearly freely, but preventing it from rolling much farther.

2) Hand to CoM integration: Our forward kinematics based on the robot model is inaccurate. One result is that if the hands are resting on one rung and the robot steps up one step 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 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 similar to Section V-A.3. 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 rung.

VI. Discussion and Future Work

During early development on the real robot, we found using ID alone is insufficient to generate the desired motions on the physical robot, especially for the swing leg and arms. We attribute this to modeling errors and joint stiction and friction. Some control based on kinematics is necessary to achieve accurate foot and hand placement. We have briefly experimented with naively integrating desired accelerations from ID into desired velocity and position, which resulted in unstable overall behaviors. We did not investigate this further for the sake of development time, but we observed qualitatively the same behaviors in simulation when $20ms$ delays of the torque command were added. Thus, a separate IK pass was introduced as a temporary solution. Inconsistency between ID and IK solutions becomes our major concern. A failure mode that we have observed is due to IK and ID having separate constraints. For example, when ID is unable to produce the demanded CoM acceleration due to limited friction, CoM will physically diverge from the desired trajectory. However, IK is unaware of such constraints, and it will keep generating physically unrealistic answers. We have also experimented with heavily biasing IK solutions to match the acceleration from ID, which also resulted in unstable behaviors. This, in the limit, is equivalent to integrating the acceleration. We think one approach to resolve this issue is to replace our current ID/IK combination with Receding Horizon Control on the full dynamic model similar to [23], which is close to, but not yet, computationally tractable in a real-time setting.

The current implementation works well for static behaviors. Being static allows us to use various integrators to compensate for modeling errors easily. It also greatly reduces the effects of all kinds of delays. We want to achieve more dynamic behaviors in the near future to gain speed and improve stability. Dynamic walking is one of our top priority goals. Modeling errors and delays stopped our early attempt to walk dynamically with the walking controller presented in [3]. The high level controller needs to rapidly re-optimize foot step timing and location in a receding horizon fashion to account for tracking errors and perturbation. We are actively experimenting with explicitly accounting for torque delays in our inverse dynamics formulation. The current low level controllers are optimizing greedily for the current time step.

Thus, we want to include a value function that captures the future cost similar to [17].

Due to the tight timeline for the DRC Trials, we have not conducted many system identification procedures on the robot. We hope to increase the quality for both kinematic and dynamic models in the near future. All the leg joint level sensing on the Atlas robot such as position, velocity (numerically differentiated from position) and torque are pre-transmission. This hardware design choice alleviates jitter in the low level joint control, but introduces problems for forward kinematics and torque control. Unmeasured stiction greatly degrades performance of torque control. Better state estimation technology is necessary to achieve more accurate position tracking and force control.

VII. Conclusions

We modified previous work to implement the proposed controller on the Atlas robot. Our approach uses both ID and IK in the full body controller. The ID module provides compliant motion and robustness against perturbation. The IK module helps us battle modeling errors and makes the controller applicable to real hardware. The combined low level controller abstracts away the details about the physical system and provides mechanisms to realize and trade off among high level controllers' potentially conflicting objectives while obeying various constraints. We have successfully demonstrated our approach on three different challenging real life applications, uneven terrain traversal, ladder climbing, and manipulation during the DRC Trials. We are the only Atlas team that was able to climb the ladder reliably, and one of the two Atlas teams that implemented their own walking controller during the Trials.

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 the Robotics Challenge programs.

References

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

[2] E. Whitman, "Coordination of multiple dynamic programming policies for control of bipedal walking," Ph.D. dissertation, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2013.

[3] S. Feng, X. Xinjilefu, W. Huang, and C. Atkeson, "3D walking based on online optimization," in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, Atlanta, GA, USA, 2013.

[4] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, pp. 43–53, February 1987.

[5] M. Hutter, M. A. Hoepflinger, C. Gehring, M. Bloesch, C. D. Remy, and R. Siegwart, "Hybrid operational space control for compliant legged systems," in *Robotics: Science and Systems (RSS)*, Sydney, NSW, Australia, July 2012.

[6] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1047–1062, 2014.

[7] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*, Chicago, IL, USA, Sept 2014.

[8] L. Saab, O. Ramos, F. Keith, N. Mansard, P. Soueres, and J. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *Robotics, IEEE Transactions on*, vol. 29, no. 2, pp. 346–362, April 2013.

[9] M. de Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 131:1–131:10, Jul. 2010.

[10] P. Wensing and D. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, Karlsruhe, Germany, May 2013, pp. 3103–3109.

[11] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, Orlando, FL, USA, May 2006, pp. 2641–2648.

[12] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

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

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

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

[16] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. McCrory, J. van Egmond, M. Griffioen, M. Floyd, S. Kobus, N. Manor, S. Alsheikh, D. Duran, L. Bunch, E. Morphis, L. Colasanto, K.-L. H. Hoang, B. Layton, P. Neuhaus, M. Johnson, , and J. Pratt, "Summary of Team IHMCs Virtual Robotics Challenge entry," in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, Atlanta, GA, USA, 2013.

[17] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *Robotics and Automation, 2014. ICRA '14. IEEE International Conference on*, Hong Kong, China, 2014.

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

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

[20] 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, Taipei, China, Sept 2003, pp. 1620–1626 vol.2.

[21] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of Honda humanoid robot," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, Leuven, Belgium, 1998, pp. 1321–1326 vol.2.

[22] X. Xinjilefu, S. Feng, W. Huang, and C. Atkeson, "Decoupled state estimation for humanoids using full-body dynamics," in *Robotics and Automation, 2014. ICRA '14. IEEE International Conference on*, Hong Kong, China, 2014.

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