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) through 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 modelling 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 degree of freedom hydraulic robot built by Boston Dynamics, joint level servos compute valve commands based on

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

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 compute q_d and $\dot{q}_d$ with IK.

Using full body inverse dynamics for force control has become a popular topic in recent humanoid research. Much of the research originates from [4]. A hierarchical approach for handling constraints and objectives is taken in [5], [6]. Ott et al. demonstrated a balancing controller on a torque controlled humanoid in [7]. Simple PD servos were used to generate a desired net ground reaction wrench. Forces are distributed among predefined contacts using optimization. [8] describes a recent effort of using floating base inverse dynamics and ZMP based pattern generation for dynamic walking. The presented ID formulation solves a smaller QP with decoupled dynamics. [9] first optimizes individual ground reaction forces and CoP for each contact and resulting admissible change in centroidal momenta. Then it solves a least square problem for the state acceleration. Joint torques are generated explicitly. Koole et al. [10] generate desired centroidal momenta change based on instantaneous capture points, and use QP to optimize for acceleration and contact forces. Although detail formulation differs in [11], the same QP inverse dynamics structure is used. [12] and [13] use orthogonal decomposition to project the allowable motions into the null space of the constraint Jacobian and minimize a combination of linear and quadratic costs in the contact constraints and the commands. [14] resolves redundancy in inverse dynamics using a kinematic task prioritization approach that ensures lower priority tasks always exist in the null space of higher priority ones. Contrary to many hierarchical null space projection approaches, we prefer using soft constraints by adding terms in the cost function with

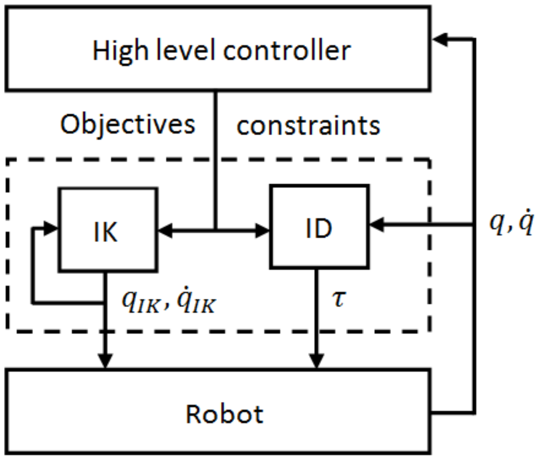


Fig. 1. The task dependent high level controller generates a set of desired objectives such as CoM or limb motion, and constraints such as CoP and joint limits. The proposed full body controller, which is contained by the dashed rectangle, takes the high level objectives and robot states $(q, \dot{q})$ 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 higher quality video of Atlas doing the described tasks is at `http://www.cs.cmu.edu/~sfeng/sfew_hum14.mp4`

high penalties. We gain numerical stability by sacrificing a small fraction of precision. We continue to use the same approach to ID that was previously developed in our group [1], [2], [15], [3]. Unlike most other approaches that solve a reduced form of inverse dynamics, we optimize acceleration, torque, and contact forces simultaneously on the full robot model. This design choice is very 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 make the QP problem higher dimensional than in other methods, but it is still solvable in real time with a standard QP solver. In our implementation, ID is operating at the acceleration and force level, thus it alone can not compute q_d or $\dot{q}_d$ to fully use the high bandwidth joint level controller. We could integrate $\ddot{q}$, the output from ID, to generate $\dot{q}_d$ and q_d, but we find this rapidly leads to constraint violation and instability.

Similar to ID, our IK is also formulated as a QP, where the unknowns are the velocities of the floating base and all the joints. At each time step, we solve for a set of $\dot{q}$ that obeys kinematic constraints and minimizes a combination of costs. q is computed by integrating $\dot{q}$ from the IK results. Our approach is similar in spirit to [16]. Although Mistry et al. also solve for $\dot{q}$, they compute it by carefully constructing and inverting a matrix composed of end effector and contact constraint Jacobians. This incremental approach of solving for $\dot{q}$ and integrating to obtain q is the primary difference between our approach and most traditional IK approaches such as [17], [18]. The incremental method can get stuck in local minima, but it does not produce discontinuous results.

The main contribution of this work is the successful application and many modifications of existing techniques to a physical system, 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.

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

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 $g = -A^T b$. A and b can be decomposed into smaller blocks as

$$A = \begin{bmatrix} w_0 A_1 \\ w_1 A_1 \\ \vdots \\ w_n A_n \end{bmatrix}, b = \begin{bmatrix} w_0 b_0 \\ w_1 b_1 \\ \vdots \\ w_n b_n \end{bmatrix}. \tag{2}$$

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

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

where $(q, \dot{q})$ is the full state of the system including 6 DOF at the floating base, $M(q)$ is the inertia matrix, $h(q, \dot{q})$ is the sum of gravitational, centrifugal and Coriolis forces, S is a selection matrix, where the first 6 rows that correspond to the 6 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, F is a vector of all contact forces in the world frame, and x is a vector of contact position and orientation in Cartesian space. F and J^T's dimensions depend on the number of contacts.

We can rewrite the equations of motion as

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

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

Let $\mathcal{X} = \begin{bmatrix} \ddot{q} & \tau & F \end{bmatrix}^T$. We turn the equations of motions into the equality constraints. The inequality constraints consist of various terms such as joint torque limits and contact force limits due to friction cone constraints and center of pressure (CoP) remaining in the support polygon constraints.

A. Cost function

We list a few examples of the objectives that can be plugged into the rows of Eq. 2.

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

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

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

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

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 current robot state. Many objectives such as CoM, hand, foot motion and torso orientation are specified in this form. Depending on the objectives, we sometimes drop the rows in the Jacobian that we do not want to constrain.

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

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

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

We can penalize center of pressure deviation with

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

where (p_x^*, p_y^*) is the desired center of pressure in 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, 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 by

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

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

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

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

0 is used 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) Change in torques: To avoid high frequency oscillations, 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 output from the last time step.

B. *Constraints*

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

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

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

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

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 F into the foot frame.

IV. INVERSE KINEMATICS

Unlike traditional IK approaches that generate positions for the entire desired trajectory ahead of time, we compute desired velocities at each time step and integrate them to get desired 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 rather than using the real robot states, we use the internal states to compute the desired velocities. The internal states are set to the real robot states in the initialization stage. All the internal states are denoted with subscripts $_{ik}$.

A. *Cost function*

We list a few examples of the objectives that can be plugged into Eq. 2.

1) Cartesian space velocity: We penalize deviation from the desired Cartesian velocity with

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

where

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

We use a different set of K_p here than in ID.

The flow chart in Figure 1 shows that the actual physical robot state is not used by the IK. Without any such feedback, it is easy for the IK to diverge significantly from the measured position of the robot. In fact, it is nearly inevitable during walking. For example, suppose we wish to take a step of some specific length. The IK will advance almost exactly the desired amount. The real robot, however, might take a step of a significantly different length either because of tracking error or slipping. After several steps, this can add up to a large error. This becomes a problem because we wish to command desired locations (for e.g. feet, hands, or CoM) in world coordinates.

In order to tie the IK root position to reality, we use the contact positions as "anchor" points. We use a "leaky" integrator to adjust the desired contact position $x_{contact_d}^*$ towards the measured contact position $x_{contact}$,

$$x_{contact_d}^* = \alpha x_{contact} + (1 - \alpha) x_{contact_d}^*. \quad (3)$$

$x_{contact_d}^*$ is the input to IK, and is initialized to the IK's internal value upon establishing the contact. Since $x_{contact_d}^*$

essentially contains all the information about long term tracking error and state estimator drift, and IK will track $x^*_{contact_d}$ obeying all the kinematic constraints, we can use $x^*_{contact_d}$ to update IK's root position to match the state estimator's. ID is not affected since it ignores this term.

2) Direct tracking and regularization:

$$\begin{aligned} A_{state} &= I \\ b_{state} &= \dot{q}^*, \end{aligned}$$

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

3) Change in velocity:

$$\begin{aligned} A_{d\dot{q}} &= I \\ b_{d\dot{q}} &= \dot{q}_{prev}, \end{aligned}$$

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

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,

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

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.

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, 6 for each leg and arm, 3 for the back joints, and 1 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 [19].

A. Static walking

Given the short time frame for development for the DRC Trials, 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. 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 build a map with laser point clouds and select foot steps with either a human operator or some combination of heuristics and simple planning. 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 the 6 dimensional Cartesian space by commanding offsets in foot position and orientation. Once the operator is satisfied with the foot pose, a "continue" command is given, allowing the robot to lower the swing foot straight down until ground contact is detected. Because of forward kinematic errors in the robot, registering laser points while moving did not work well. Our main motivation was to avoid standing still and waiting for sufficient laser scanner data to accumulate. On the other hand, our approach requires more (and more difficult) input from the operator, and extends the single support phase unnecessarily since the operator commands are given during single support rather than double support.

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

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

$$\begin{aligned} err_{swing} &= err_{swing} + K_i(x'_{swing_d} - x_{swing}) \\ x^*_{swing_d} &= x'_{swing_d} + err_{swing}, \end{aligned} \tag{4}$$

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

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 ID and IK. We first move the rear foot reference point, where the Jacobian is computed to the toe. In ID, 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 IK, 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 as desired to bias IK towards using ankle angle for a toe-off solution.

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

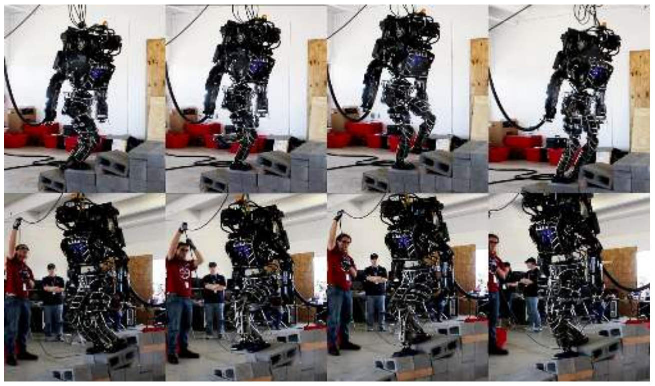


Fig. 2. These photos show the Atlas robot practicing for the segment 3 of the terrain task for DRC. The snapshots were taken every 5 seconds.

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. 4.

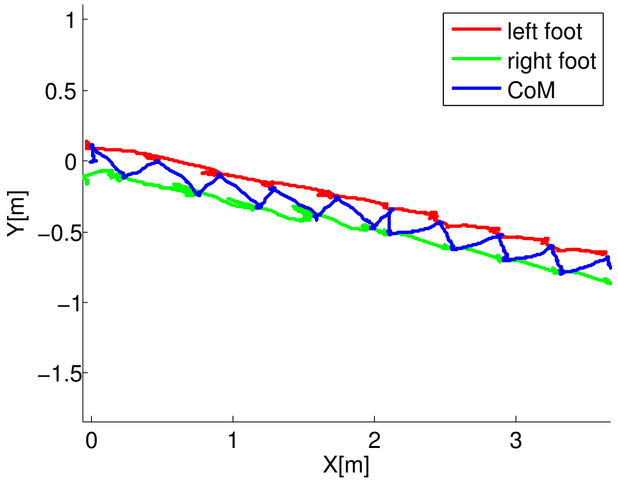
B. Full body manipulation

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. These commands are used to update the desired IK position. We use equality constraints in the IK 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 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 IK position. We then use PD gains comparing the measured and IK positions (of hands, CoM, etc.) to produce input desired acceleration for the ID. Figure 7 shows a picture of the Atlas robot performing the valve task during the DRC Trials.

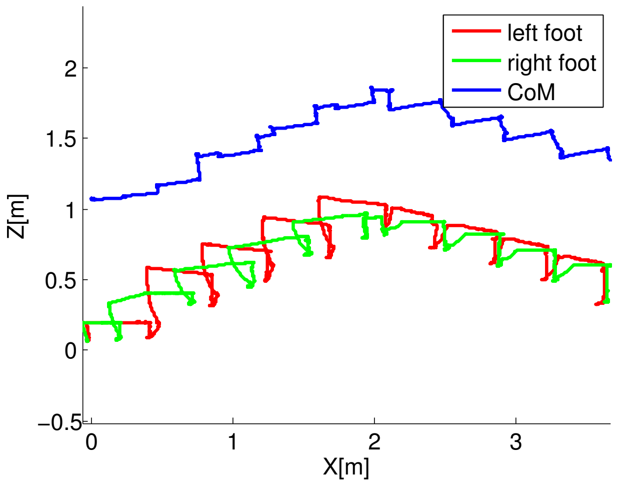
We make a few small changes to the basic full body control algorithm:

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. 3 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.

2) Allowing rotation: For some tasks, we only care about the position of the hand, and the orientation is unimportant.



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

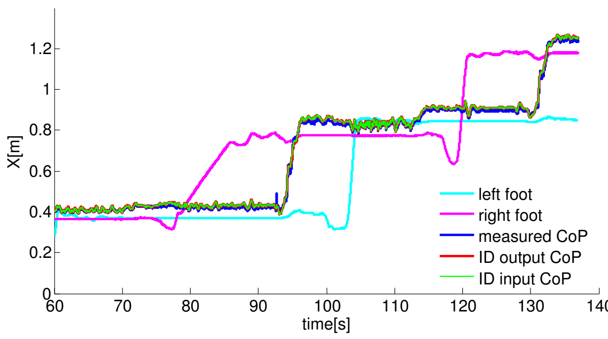


(b) Measured feet and CoM trajectories in XZ plane for terrain task

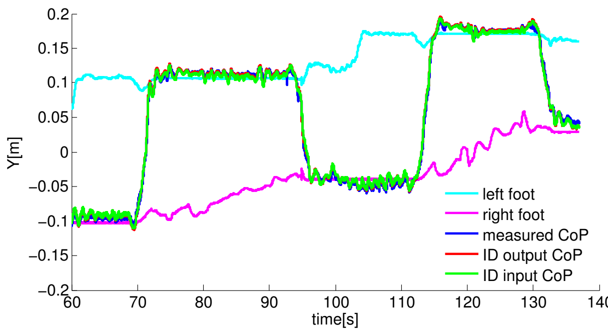
Fig. 3. These plots show the Atlas robot traversing segment 3 of the terrain task. X axis is the forward direction, Y points to the robot's left, and Z points upward. Left and right foot positions are shown with red and green lines, and center of mass is plotted in blue. The robot walks in a straight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

For such cases, we can turn the weight for the hand orientation equations in the IK to 0, 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 must maintain its position while keeping that axis normal to the wall. Allowing the controller the freedom to rotate around one axis can drastically 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 IK equations concerning hand orientation into this basis and remove the one corresponding to the specified axis.



(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 $1cm$.

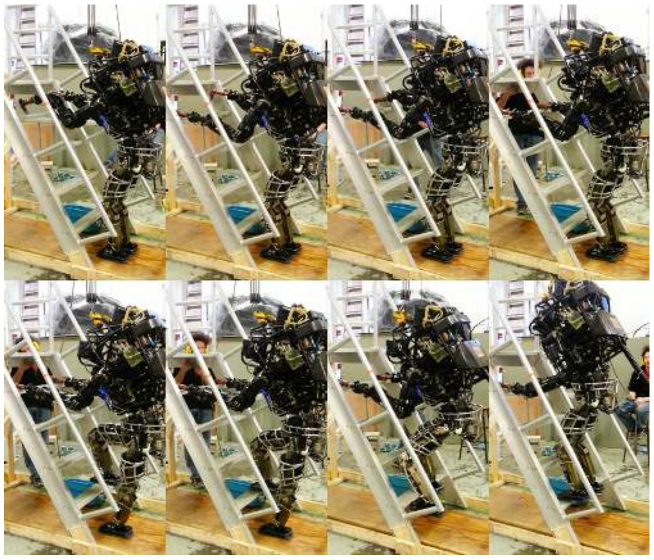
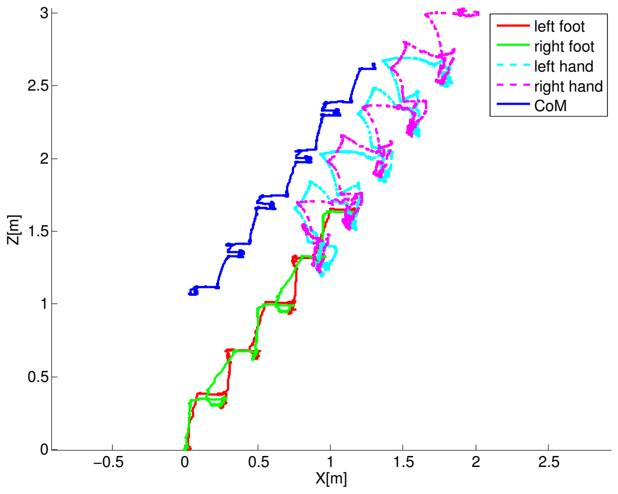
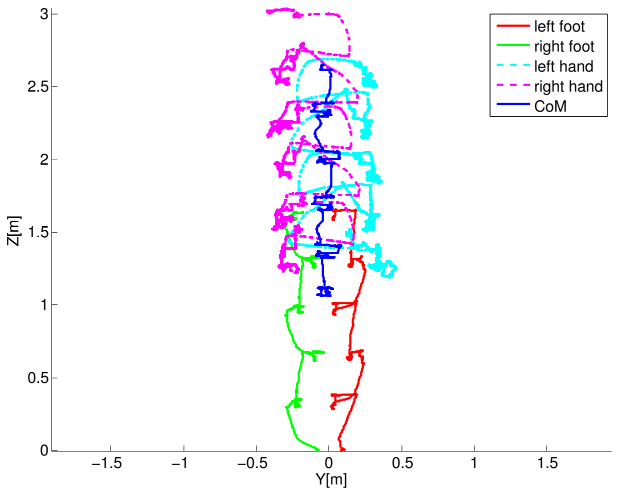


Fig. 5. These photos show the Atlas robot climbing the top half of the same ladder as used in DRC. The snapshots were taken every 13 seconds. The top row shows repositioning of the hook hands, and the bottom row shows stepping up one tread. Most of the climbing motions are scripted. After each limb's rough repositioning, the operator can 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 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.

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 snapshots of a complete cycle of the Atlas robot climbing the ladder. 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 $1cm$ 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

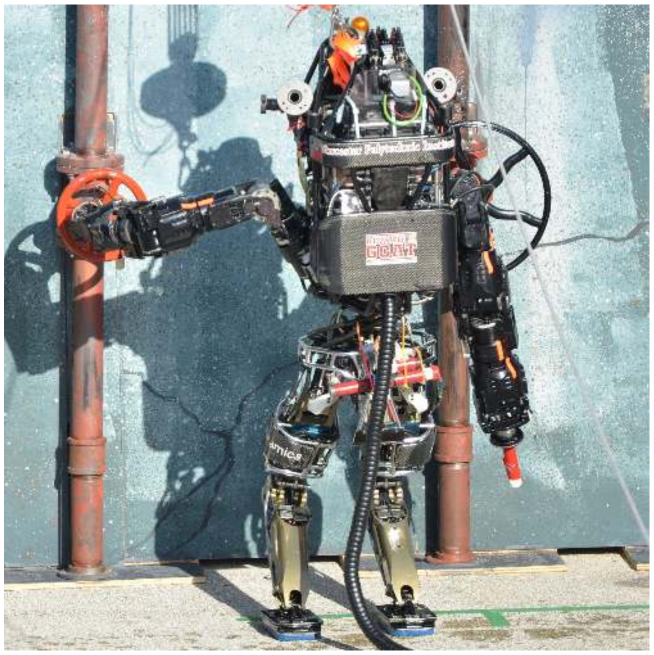


Fig. 7. The proposed low level controller is applied on the Atlas robot to turn a valve in the DRC Trials.

hands.

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. 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, allowing the robot to 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 above. This reorienting serves a similar purpose to Eq. 3, but for rotation instead of translation. To avoid chase conditions, we disable Eq. 3 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. 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 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 robot model had inaccurate forward kinematics. 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. 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 (not during hand repositioning). CoM and limb trajectories from the actual run during the DRC Trials are plotted in Figure 6.

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. We attribute this to modeling errors and joint stiction and friction, especially for the swing leg. Some control based on kinematics is necessary to achieve accurate foot placement. We have briefly experimented with naively integrating desired accelerations from ID into desired velocity and position, which resulted in unstable overall behaviors. Thus, a separate IK pass was introduced as a temporary solution to these problems. Inconsistency between their answers becomes our major concern. A failure mode we observe 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, the CoM will physically diverge from the desired trajectory. However, IK is unaware of such constraints, and it will keep generating the physically unrealistic answers. We have also experimented with heavily biasing IK solutions to match the acceleration from ID, which also resulted in unstable behaviors, and, to its limit, is equivalent to just 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 [20], 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 modelling 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. Modelling errors and system delay stopped our early attempt to walk dynamically with the walking controller presented in [3]. The high level controller needs to rapidly re-optimize for step timing and location in a receding horizon fashion to account for delays and maximize stability. 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. We want to include the value function that captures the future cost similar to [11].

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 positioning, 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 use both ID and IK in the low level controller. The inverse dynamics module provides us compliant motion and robustness against perturbation. The inverse kinematics module helps us battle modelling errors and makes our approach applicable to the 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 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, August 2011.

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

[3] S. Feng, X. Xinjilefu, W. Huang, and C. Atkeson, "3d walking based on online optimization," in *Humanoid Robots, 2013. Humanoids 2013. 13th IEEE-RAS International Conference on*, 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] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 483–501, June 2010.

[6] 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*, May 2006, pp. 2641–2648.

[7] 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*, Oct 2011, pp. 26–33.

[8] 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*, Nov 2012, pp. 829–834.

[9] 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*, Oct 2010, pp. 3157–3162.

[10] 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*, 2013.

[11] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *Robotics and Automation, 2013. ICRA '13. IEEE International Conference on*, 2013.

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

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

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

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

[16] 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, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, 2008, pp. 22–27.

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

[18] 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, 1998, pp. 1321–1326 vol.2.

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

[20] 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*, 2012, pp. 4906–4913.