Optimization Based Controller Design and Implementation for the Atlas Robot in the DARPA Robotics Challenge Finals

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

***Abstract*— We describe the design and hardware implementation of our walking and manipulation controllers that are based on a cascade of online optimizations. A virtual force acting at the robot's center of mass (CoM) is estimated and used to compensated for modeling errors of the CoM and unplanned external forces. The proposed controllers have been implemented on the Atlas robot, a full size humanoid robot built by Boston Dynamics, and used in the DARPA Robotics Challenge Finals, which consisted of a wide variety of locomotion and manipulation tasks.**

I. Introduction

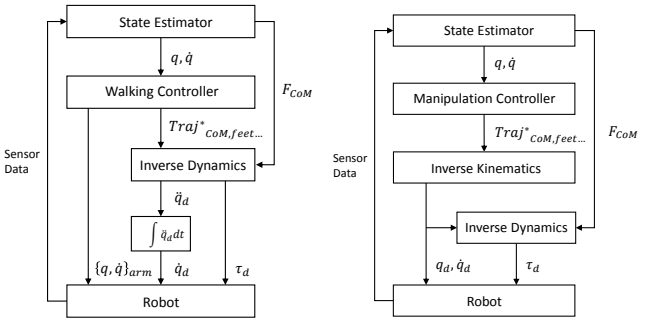
Versatility is one of the major advantages of optimization and model based approaches to humanoid locomotion and manipulation. This has been demonstrated convincingly throughout the DARPA Robotics Challenge (DRC) considering the short development time and the large number of different tasks that need to be completed. In this paper, we will present the design and implementation of our optimization based walking and manipulation controllers [1]. A general summary of our entire DRC effort is presented in [1]. Both the walking and manipulation controllers consist of a high level controller that optimizes for task space trajectories into the future and a low level full-body controller that generates instantaneous joint level commands that best track those trajectories while satisfying physical constraints. Hardware experiments are conducted with the Atlas robot built by Boston Dynamics. It has 30 actuated degrees of freedom (DoF), six for each leg, seven for each arm, three for the spine, and one for neck pitch. Most of them are hydraulic actuators with the exception of the neck and the last three joints on each arm which are electric.

Since the DRC Trials at the end of 2013, we have been focusing on implementing the previous walking controller [2] on the Atlas robot. We have also spent some effort on tuning gains and filter parameters on the hardware for better joint level tracking performance. The low level full-body controller is also redesigned to address an inconsistency issue we encountered during the DRC Trials [3]. The new architecture is summarized in Figure 1.

The foundation of our controller is a quadratic programming (QP) based inverse dynamics module. Like many others, it originates from [4]. Desired motions are specified in task space, and convex optimization is used to handle constraints and solve for controls that best track these motions. For controlling a high DoF system such as a humanoid robot, there are two main approaches for resolving redundancies: rank the objectives using different weights; or impose a strict hierarchy on the objectives. The hierarchical approaches [5]–[10] typically ensure low priority objectives are within the null space of higher priority ones. They are appealing in the sense that no low priority tasks can ever jeopardize higher priority ones. However in practice, only a small number of hierarchies are implemented, and objectives within the same hierarchy are still weighted. Furthermore, enforcing constraints for all hierarchies is typically computationally more expensive. We prefer the simpler and faster weighted approaches [11]–[15]. There is also much interest in formulating a smaller optimization problem. Contact forces can be removed using orthogonal decomposition [16]. Another approach is to optimize for force allocation among multiple contacts first and solve inverse dynamics in the second stage [17]–[20].

For mostly static manipulation tasks, we use an explicit inverse kinematics controller to first optimize for the full generalized state, which is then tracked using inverse dynamics. It is formulated as a QP optimizing for the generalized velocity, and the joint configuration is integrated from the joint velocity. Our approach is an extension to the damped least squares method used by [21] and is similar to [22] in spirit.

The Linear Inverted Pendulum Model (LIPM) is widely



(a) Walking controller (b) Manipulation controller

Fig. 1. For walking, in addition to tracking desired joint torques, the lower body joint level controllers have velocity control loops whose targets are the integrated accelerations optimized by inverse dynamics. The arm joints have position and velocity control loops, and the desired values come directly from the high level plan. For manipulation, full state is first solved by inverse kinematics, which is then tracked by inverse dynamics.

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

‡ is with Disney Research, 4720 Forbes Avenue, Suite 110, Pittsburgh, PA 15213, joohyung.kim@disneyresearch.com

[1]A high quality video accompanying this paper can be found at `https://youtu.be/ZzCPl8phG1g`

Preprint submitted to 2015 IEEE-RAS International Conference on Humanoid Robots.
Received July 7, 2015.

used for generating center of mass (CoM) trajectories for walking. Preview Control [23] is one of the most successful applications. Momentum based methods [12], [20] have been developed for balancing and walking. We use Differential Dynamic Programming [24] to optimize for a nominal CoM trajectory, together with a linear policy and local quadratic approximation of the value function, which are used to stabilize the robot around the nominal trajectory. [11] shares the same idea.

A variety of foot step planners are implemented for different scenarios. In most flat ground walking cases, we use an A^* based [25] planner that uses an action set of more aggressive foot steps. Interpolation based planners are also implemented. We rely on their repeatability for certain tasks, such as sidestepping through an open door. Special purpose planners are implemented for the terrain and stair tasks. The terrain planner first fits a grid of blocks to the laser point cloud using the operator's alignment as an initial guess. It then classifies each cinder block based on surface normal. A discrete search is performed to generate the actual foot step sequence using a set of transitions that have predefined costs. For the stairs, a model is generated and fitted. Human guidance can then be used to modify the generated foot step sequence. The operator manually selects the most suitable planner during the Finals. All planned foot steps can be adjusted by the operator, and our interface allows manual foot step inputs as well.

II. Full-body controller

In most cases, the high level controller outputs desired Cartesian motions for specific parts on the robot (e.g. foot, hand and CoM). The low level controller takes these as inputs and generates joint level commands such as joint position, velocity and torque, which are then used as desired values in the joint level controllers.

A. State estimation

Before going into more details about the full-body controller, we want to briefly describe our state estimator. More details can be found in [26]. Joint position is sensed by linear variable differential transformers (LVDTs) on the actuators, and velocity is the low pass filtered finite difference of positions. Actuator force is measured using oil pressure in the piston chambers. Pelvis angular velocity and orientation are taken directly from an inertial measurement unit (IMU) mounted on the pelvis. Pelvis linear velocity and position are estimated using an Extended Kalman Filter based on the IMU acceleration measurements and forward kinematics [27].

In addition to the pelvis estimator, we also implement an estimator that explains the modeling error on the CoM level using LIPM dynamics, for which the modeling error can be treated as a combination of an external force and a CoM offset. We treat it as an external force and compensate for it in the inverse dynamics controller.

This estimator is especially helpful when compensating for unplanned slow changing external forces applied at unknown locations on the robot, which is quite likely when operating

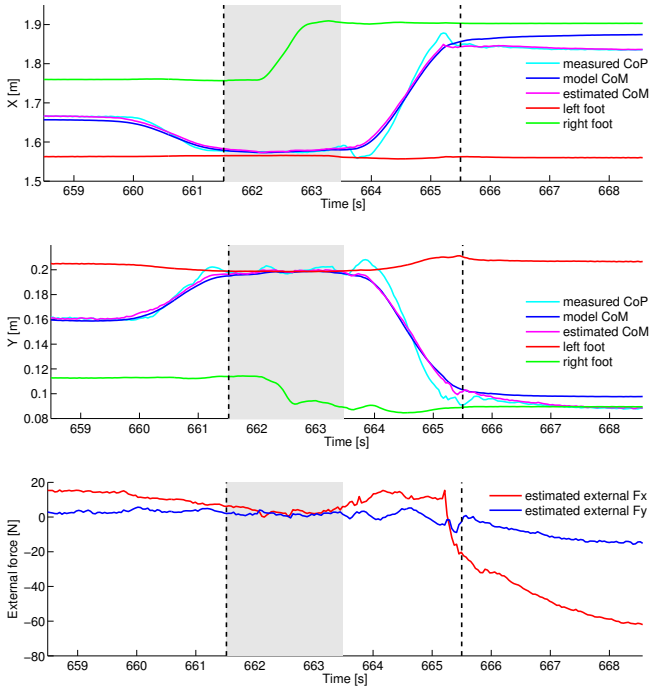


Fig. 2. Atlas was caught on the door frame when sidestepping through it during the dress rehearsal at the DRC Finals. The walking controller properly delayed liftoff and remained in double support when the CoM modeling error estimator (described in Section II-A) detected anomaly. Single support phase is shown by the shaded area, and the black dashed lines indicate the planned liftoff time. Estimated CoM is the sum of the model CoM and the estimated CoM offset.

in tight spaces. It also handles relatively small dynamic forces well when walking, e.g. dragging a tether or pushing through a spring loaded door. Thanks to the estimator, very little tuning is done for our mass model. The most significant contribution of this estimator is that it can detect when a large external force is being applied that might push the robot over. It saved our robot from falling twice at the DRC Finals, where no safety belay was allowed. On the dress rehearsal day, the robot was caught on the door frame when sidestepping through, and the walking controller stopped to enable manual recovery. Figure 2 shows data from this experiment. During our second run in the Finals, our electric forearm mechanically failed when the cutting motion was initiated for the drill task. The uncontrolled forearm wedged the drill into the dry wall and applied a large backward force. The manipulation controller properly froze and saved the robot from falling. The human operator was able to recover from an otherwise catastrophic scenario.

B. Unified full-body controller

It is necessary to generate joint level kinematic targets for the robot implementation, since inverse dynamics alone performs poorly when facing modeling errors. For the DRC Trials [3], we implemented the low level controller using independent inverse kinematics (IK) and inverse dynamics (ID) modules. An independent IK controller was suitable for the static motions in the Trials, and was preferred for

Preprint submitted to 2015 IEEE-RAS International Conference on Humanoid Robots.
Received July 7, 2015.

easy implementation. However, we were concerned about inconsistency between the two modules due to their different sets of constraints. For the Finals, two versions of the full-body controllers are implemented. When walking, we no longer use IK for joint level kinematic targets. For the lower body joints, we numerically integrate the joint accelerations from ID into desired velocities, which are tracked by the joint level controllers. Arm motions are always specified in joint space during walking, so no extra computation is needed for the arm joints. For manipulation, IK is used first to generate the full state that best tracks the desired Cartesian commands. ID is then used to track the IK's state. Although IK does not enforce dynamic constraints, we argue that by properly positioning the CoM, running into a dynamic constraint is very unlikely during manipulation for a strong robot like Atlas. These design choices are motivated by the observation that the legs on Atlas are much better engineered than the arms. The leg joints have less sensor noise and friction. For the arm joints, we were unable to use higher velocity gains and failed to achieve decent joint tracking without the position control loop. An explicit IK is also preferred as opposed to double integration for stability and accuracy reasons.

Both ID and IK are formulated as quadratic programing (QP) problems. The actual implementation does not differ greatly from [3], so we will only present the new features here.

C. Inverse dynamics

1) Reduction in QP size: Let $M(q)$ be the inertia matrix, $h(q, \dot{q})$ be the sum of gravitational and Coriolis and centrifugal forces, τ stand for a vector of joint torques, $J(q)$ be the Jacobian matrix for all the contacts, and λ be a vector of all contact wrenches in the world frame. As pointed out by Herzog, et al. [7], the equations of motion can be decoupled into two parts:

$$\begin{aligned} M_u(q)\ddot{q} + h_u(q, \dot{q}) &= J_u^T(q)\lambda \\ M_l(q)\ddot{q} + h_l(q, \dot{q}) &= \tau + J_l^T(q)\lambda, \end{aligned} \tag{1}$$

where the top represents the six rows of the floating base, and the bottom corresponds to the actuated DoF. τ is thus linearly dependent of $\ddot{q}$ and λ as

$$\tau = M_l(q)\ddot{q} + h_l(q, \dot{q}) - J_l^T(q)\lambda, \tag{2}$$

and can be eliminated from the QP to reduce the problem size by almost a half. Eq. 2 is used in the cost terms and inequality constraints to replace τ, and the top part of Eq. 1 is used as the equality constraints.

2) Approximated value function for the CoM: A new term is added to the cost function to approximate the value function of the CoM state with a pure quadratic. Let $X = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$ be the current CoM state, and we can approximate the cost-to-go for the next time step with

$$\begin{aligned} V(X') &\approx 0.5(X' - X^*)^T V_{XX}^*(X' - X^*), \\ X' &= \begin{bmatrix} x + \dot{x}dt + 0.5(J(q)\ddot{q} + \dot{J}(q)\dot{q})dt^2 \\ \dot{x} + (J(q)\ddot{q} + \dot{J}(q)\dot{q})dt \end{bmatrix} \\ &= \begin{bmatrix} 0.5J(q)dt^2 \\ J(q)dt \end{bmatrix} \ddot{q} + \begin{bmatrix} x + \dot{x}dt + 0.5\dot{J}(q)\dot{q}dt^2 \\ \dot{x} + \dot{J}(q)\dot{q}dt \end{bmatrix}, \end{aligned} \tag{3}$$

where V_{XX}^* and X^* are the value function's second order derivative and the setpoint. Eq. 3 can be further rewritten to fit the cost term form in [3].

3) Centroidal momentum matrix: In the previous implementation, desired change in centroidal momentum was expressed as a cost term using the net external wrench from the contact forces. On the other hand, it is also linear with respect to the generalized acceleration [28]. In the current implementation, the latter form is preferred since we can then freely add virtual forces in the generalized coordinate or in the Cartesian space that can be potentially useful for handling modeling errors.

4) Structural change smoothing in QP: In the previous implementation, the ID QP changes dimension based on number of contacts. Discrete changes can also happen due to constraint manifold changes such as during toe-off, when the foot frame CoP is constrained at the toe. Such changes will cause sudden jumps in the outputs that can induce undesired oscillations on the physical robot. These jumps are caused by structural changes in the problem specification, and cannot be handled by just adding cost terms that penalize changes. Our solution is to maintain the same QP dimension and gradually blend the constraints over a short period of time. We always assume the largest number of contacts, but heavily regularize the magnitude of the contact force and relax the acceleration constraints for the fake contacts. Inequality constraint surfaces are changed smoothly when a change is required.

5) CoM level modeling error: The estimated virtual force is applied at the CoM and added directly in Eq. 1 to be properly handled by ID.

D. Inverse kinematics

The main purpose of the IK controller is to maintain a feasible full-body kinematic state that best tracks the high level controller's objectives during manipulation. The ID controller then tracks IK's full state instead of the incomplete targets from the high level controller.

1) Self collision avoidance: For many manipulation tasks, especially for human in the loop tele-operation, where proper collision checking is usually not performed, the vanilla IK controller often ends up in a self colliding state. Even in situations where desired collision free full-body trajectories are generated with proper motion planning algorithms, avoiding self collision in the IK controller is still useful. Trajectories generated by typical motion planners often consist of a sequence of key frames that is orders of magnitudes too sparse for control purposes. Interpolation at the controller

level is necessary for smooth motions. Furthermore, the controller needs to modify the original plan to maintain balance in unexpected situations during execution. On the other hand, depending on how the particular planner handles collisions, the planned trajectory can be arbitrarily close to a collision surface. Any small variation can invalidate the collision free property. Thus, it would be ideal to enforce collision avoidance constraints in the inverse kinematics controller.

In the interest of complexity and computational cost, only approximate self collision avoidance is implemented. We use capsules as collision shapes for the major limbs and torso. Figure 3 shows the collision shapes we use for the Atlas robot. Given the current configuration q_{ik} of the robot, for any two capsules of interest, $\mathcal{C}_a$ and $\mathcal{C}_b$, we first find the closest two points on their surfaces respectively, c_a and c_b. Assuming c_a and c_b will remain the closest points on $\mathcal{C}_a$ and $\mathcal{C}_b$ for the next time step, we can construct a frame O_a, whose origin is at c_a, the Z axis has the same direction as $c_b - c_a$, and the Y axis is perpendicular to both $c_b - c_a$ and $\mathcal{C}_a$'s principal axis d_a.

$$\begin{aligned} Z_a &= \frac{c_b - c_a}{|c_b - c_a|} \\ Y_a &= Z_a \times d_a \\ X_a &= Y_a \times Z_a \\ R_a &= \begin{bmatrix} X_a & Y_a & Z_a \end{bmatrix} \end{aligned} \tag{4}$$

Let $H_a = \begin{bmatrix} R_a & c_a \\ 0 & 1 \end{bmatrix}$ be the homogeneous transformation from the world frame to O_a, and we use ${}^a c_b$ to denote the position of c_b specified in frame O_a. Collision free between $\mathcal{C}_a$ and $\mathcal{C}_b$ can be approximated by enforcing ${}^a c_b$ to keep a minimum value in its Z coordinate. Let $'$ denote variables for the next time step. Assuming the orientation for O'_a is the same as O_a, we have

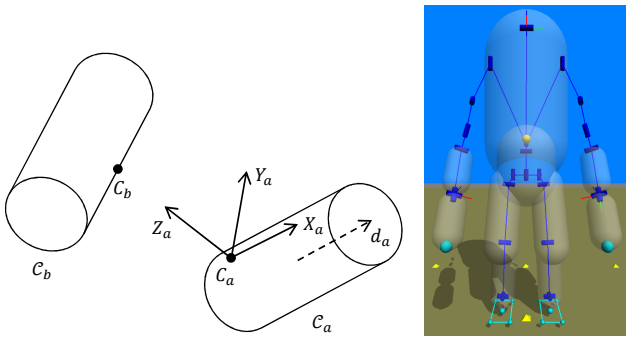
$$\begin{aligned} H'_a &= \begin{bmatrix} R_a & c_a + J_a \dot{q} dt \\ 0 & 1 \end{bmatrix} \\ c'_b &= c_b + J_b \dot{q} dt \\ \begin{bmatrix} {}^a c'_b \\ 1 \end{bmatrix} &= {H'_a}^{-1} \begin{bmatrix} c'_b \\ 1 \end{bmatrix} \\ \Rightarrow {}^a c'_b &= R_a^{-1}(c_b + J_b \dot{q} dt) - R_a^{-1}(c_a + J_a \dot{q} dt) \\ &= R_a^{-1}(J_b - J_a)\dot{q} dt + R_a^{-1}(c_b - c_a), \end{aligned} \tag{5}$$

where J_a and J_b are Jacobian computed at c_a and c_b. Eq. 5 is linear with respect to $\dot{q}$, and the inequality constraint for avoiding collision is

$${}^a c'_b[Z] \geq \epsilon, \tag{6}$$

where $[Z]$ takes only the Z component of ${}^a c'_b$, and ϵ is some positive safety margin.

Due to the kinematics of the Atlas robot and empirical observations, we only consider a subset of all possible self collisions to speed up computation.



(a) Coordinate system for setting up a collision constraint

(b) Simplified collision shapes for the Atlas robot

Fig. 3. Figure 3(a) is an illustration for Eq. 4 and Eq. 5. $\mathcal{C}_a$ and $\mathcal{C}_b$ are the two collision shapes of interest. c_a and c_b are the two closest points on $\mathcal{C}_a$ and $\mathcal{C}_b$. d_a is a unit vector that represents the principle axis of $\mathcal{C}_a$. X_a, Y_a and Z_a are defined in Eq. 4. Figure 3(b) shows the collision shapes we used for the Atlas robot represented by transparent grey capsules.

III. High Level Controllers

A brief summary of the walking and manipulation high level controllers is presented in this section. The walking controller is based on previous simulation work [2], and the manipulation controller is largely the same as in [3].

A. Walking controller

Given a sequence of desired foot steps, the walking controller optimizes a CoM trajectory using Differential Dynamic Programming [24], which is a local iterative trajectory optimization technique that can be applied to nonlinear dynamics. A nonlinear point mass model that includes the Z dimension is used for trajectory optimization to take height changes into account. The CoM trajectory is replanned during every single support phase for the next two foot steps. The swing foot trajectory is generated by a quintic spline from the liftoff pose to the desired touch down pose. Figure 5 shows snapshots of the robot walking over a pile of tilted cinder blocks and a standard staircase.

1) Differential Dynamic Programming: In addition to the optimized trajectory, DDP also produces a linear policy and local second order approximation for the value function along the trajectory, which are used to stabilize the CoM motion. Let X^t and u^t be the state and control on the optimized trajectory at time step t, X be the estimated state, and $X_e = X - X^t$. We can compute the control and approximated the cost-to-go with

$$\begin{aligned} V(X) &\approx V_0^t + V_X^t X_e + \frac{1}{2} X_e^T V_{XX}^t X_e \\ u(X) &= u^t - K^t X_e. \end{aligned} \tag{7}$$

$u(X)$ can be expressed as CoM acceleration and used as an input to ID. X^t and V_{XX}^t are plugged into Eq. 3.

2) Step timing: Reliability is our primary objective for the DRC Finals, so a more static walking style is preferred. Having a low cadence allows us to arbitrarily pause while walking and gives the operator a chance to recover when

Preprint submitted to 2015 IEEE-RAS International Conference on Humanoid Robots.
Received July 7, 2015.

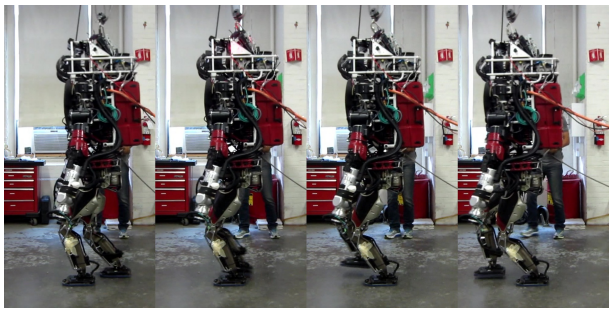


Fig. 4. Atlas walking at $0.4m/s$ with $0.8s$ per step. These snapshots are taken every $0.2s$.

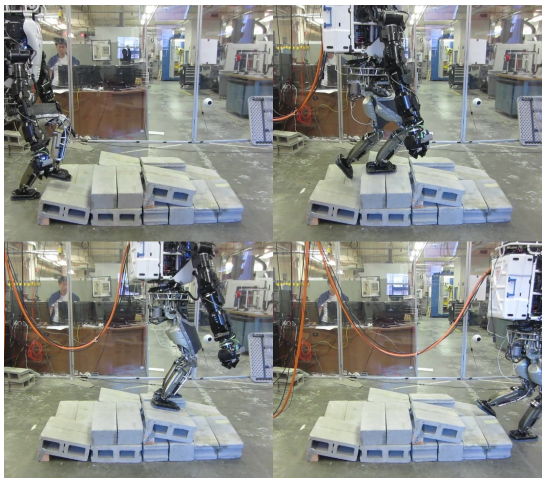
things go wrong. In the DRC Finals, we use a nominal step length of $40cm$ and $4s$ for each step ($8s$ for a complete stride of left and right steps). Among the top three Atlas teams that developed their own walking controllers, we have the lowest cadence but take the most aggressive foot steps. On the other hand, the controller is capable of more dynamic walking, and we have achieved $0.4m/s$ walking reliably by just speeding up the cadence to $0.8s$ per step. Snapshots of the experiment are shown in Figure 4.

3) Improved toe-off criteria: Toe-off is triggered by detecting a stance leg approaching knee straight. However, this is not a sufficient condition for toe-off. For some of the more challenging configurations or more dynamic walking, blindly switching to toe-off mode without considering dynamic feasibility will result in the robot falling. When a near straight knee configuration is detected, we prepare for toe-off by adding a high weight cost term in ID that moves the CoP in the relevant foot's frame to be at its toe without changing the constraint. Once the optimized CoP is close to the toe, we then start shrinking the CoP constraint to be exactly at the toe.

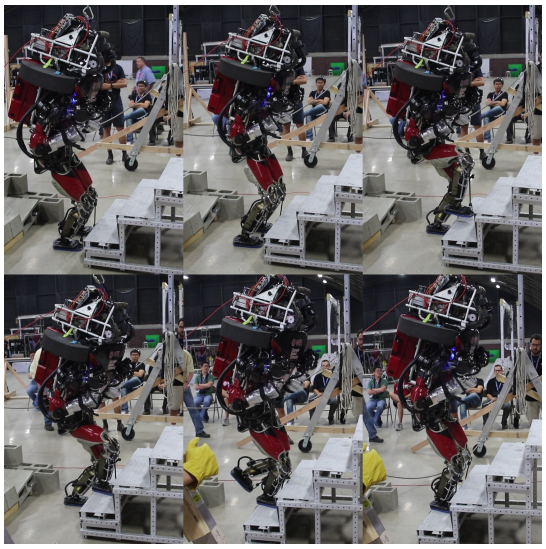
4) Offset for the planned CoM trajectory: In the current implementation, we only replan the CoM trajectory at every single support phase. We need a heuristic to modify the planned CoM trajectory to compensate for inevitable foot placement errors. We define $x_{off} = x_{foot}^{actual} - x_{foot}^{planned}$ in the horizontal plane, where x_{foot}^{actual} is the actual location for the touch down foot, and $x_{foot}^{planned}$ is the planned. x_{off} is computed on every time step and used to offset the planned CoM and CoP trajectories. To ensure smoothness, x_{off} is gradually ramped up over a short period of time after touchdown. During our first official run at the Finals, our robot did not lift its swing foot high enough when stepping down on the cinder block course. The swing foot scraped a cinder block halfway through and triggered touch down much earlier than planned. The walking controller was able to handle this situation despite the large x_{off}.

B. Manipulation controller

The operator can directly specify desired Cartesian motions for the hands, the pelvis and the upper torso. Direct joint commands for the upper body can also be issued. Figure 6 shows pictures of Atlas doing manipulation tasks.



(a) Walking over cinder blocks



(b) Walking up a staircase

Fig. 5. Snapshots of Atlas walking over cinder blocks and climbing a staircase. The top ones are taken every $12s$, and the bottom ones are taken every $2s$.

For the Finals, we add a new feature that takes in full state trajectories planned by an external motion planner, TrajOpt, developed by Schulman et al. [29] These trajectories are specified by sequences of key frames. Since the centroidal momentum [28] is linear with respect to the generalized velocity, which is approximately the finite difference between two consecutive key frames, we can compute the minimum duration between two key frames by bounding the maximum magnitude of the centroidal momentum. The controller then linearly interpolates between two key frames using this duration. Intuitively, the trajectory slows down when it generates larger whole body motions and speeds up otherwise.

For the drill task in the Finals, in order to make a reliable cut, the robot needs to press the drill firmly against the dry wall with a small amount of constant force. Since the arms are essentially position driven for end effector accuracy, we servo the hand Cartesian target based on the wrist force

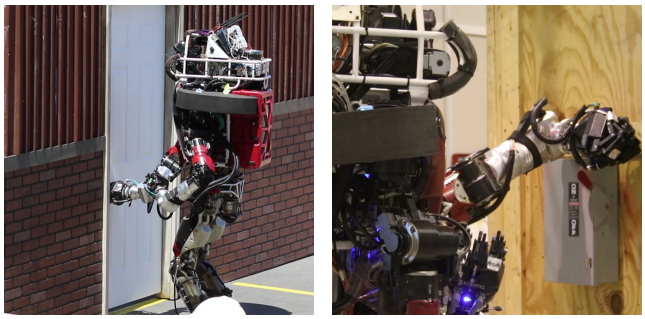


Fig. 6. Motions that need to satisfy nontrivial constraints, such as turning a door handle, are planned using TrajOpt. The operator can directly issue Cartesian or joint commands, which are useful for human-in-the-loop visual servoing and fault recovery. The DRC surprise tasks are completed using direct Cartesian commands.

torque sensor.

When the manipulation controller detects a near falling scenario, it will terminate all ongoing trajectories, maintain its current posture and block future commands until the operator attempts manual recovery. Fall detection [26] is based on a capture point [30] estimator combined with the external force estimator discussed in Section II-A.

Since a full fledged IK is running inside the manipulation controller, it can be used as a standalone kinematic tool for previewing commands before sending them to the actual robot.

IV. Hardware Implementation

A. Joint level controllers

On the Atlas robot, all the leg, back and upper arm joints are hydraulic. The joint level controllers compute valve commands based on

$$\begin{aligned} i = & K_{qp}(q_d - q) + K_{qi}\sum(q_d - q)dt \\ & + K_{qd}(\dot{q}_d - \dot{q}) + F_{qd}\dot{q} + F_{qd_d}\dot{q}_d \\ & + K_{\tau p}(\tau_d - \tau) + F_{\tau_d}\tau_d \\ & + F_{const}. \end{aligned} \tag{8}$$

In Eq. 8, K_* stands for gains for feedback terms, F_* stands for feedforward gains, F_{const} is a constant bias term, and subscript $_d$ denotes desired values. We use K_{qp}, K_{qd}, $K_{\tau p}$, F_{qd} and F_{const} in our implementation. The electric forearms share the same interface, but we only use the K_{qp}, K_{qd} and K_{qi} terms.

During implementation, we found that specifying gains in actuator coordinate gives us better joint tracking performance. The actuator gains need to be transformed to fit the interface's joint space specification. This is done using the transmission information disclosed by Boston Dynamics. We use actuator gains for all the linearly actuated joints, which are all the leg joints and spine pitch and roll joints. Let l denote piston length, and e_* for tracking error for the following equations. Valve commands computed in either the joint or actuator space should be the same, thus for the K_{qd} term,

$$\begin{aligned} K_{qd}e_{\dot{q}} &= K_{ld}e_{\dot{l}} \\ K_{qd}\frac{dq}{dl}e_{\dot{l}} &= K_{ld}e_{\dot{l}} \\ K_{qd} &= \frac{dl}{dq}K_{ld}, \end{aligned} \tag{9}$$

since $\dot{q} = \frac{dq}{dl}\dot{l}$. Similarly, $F_{qd} = \frac{dl}{dq}F_{ld}$.

For the torque feedback term, the virtual work produced in both spaces should be the same, and $\tau = \frac{dl}{dq}F$, thus

$$\begin{aligned} K_{\tau p}e_{\tau} &= K_{Fp}e_F \\ K_{\tau p}\frac{dl}{dq}e_F &= K_{Fp}e_F \\ K_{\tau p} &= \frac{dq}{dl}K_{Fp}. \end{aligned} \tag{10}$$

$\frac{dl}{dq}$ is a function of q and specified in a table by Boston Dynamics. Linear interpolation is used for smoothing.

The ankle roll and pitch joints on Atlas are mechanically coupled. Let q_y and q_x stand for the ankle pitch and roll joint angle, and l_l and l_r be the left and right piston length. The relationship between the joint and actuator velocity is

$$\begin{bmatrix} \dot{l}_l \\ \dot{l}_r \end{bmatrix} = J \begin{bmatrix} \dot{q}_y \\ \dot{q}_x \end{bmatrix}, J = \begin{bmatrix} \frac{dl_l}{dq_y} & \frac{dl_l}{dq_x} \\ \frac{dl_r}{dq_y} & \frac{dl_r}{dq_x} \end{bmatrix}. \tag{11}$$

The valve commands for the left and right ankle actuators are computed by

$$\begin{bmatrix} i_l \\ i_r \end{bmatrix} = J \begin{bmatrix} i'_{q_y} \\ i'_{q_x} \end{bmatrix} + \begin{bmatrix} F_{q_y} \\ F_{q_x} \end{bmatrix}_{const}, \tag{12}$$

where i'_{q_y} and i'_{q_x} are computed with Eq. 8 without the F_{const} term.

Thus for the velocity term,

$$\begin{aligned} \mathbf{K}_{ld}\mathbf{e}_{\dot{l}} &= J\mathbf{K}_{qd}\mathbf{e}_{\dot{q}} = J\mathbf{K}_{qd}J^{-1}\mathbf{e}_{\dot{l}} \\ \Rightarrow \mathbf{K}_{qd} &= J^{-1}\mathbf{K}_{ld}J \end{aligned} \tag{13}$$

where $\mathbf{K}_{ld}$ and $\mathbf{K}_{qd}$ are 2×2 gain matrices, and $\mathbf{e}_{\dot{l}}$ and $\mathbf{e}_{\dot{q}}$ are 2×1 error vectors. Let $\mathbf{K}_{qd} = \begin{bmatrix} k_{yy} & k_{yx} \\ k_{xy} & k_{xx} \end{bmatrix}$,

$$\begin{aligned} \begin{bmatrix} i_l \\ i_r \end{bmatrix}_{qd} &= J\mathbf{K}_{qd}\mathbf{e}_{\dot{q}} \\ &= J\left(\begin{bmatrix} k_{yy} & 0 \\ 0 & k_{xx} \end{bmatrix}\begin{bmatrix} e_{\dot{q}_y} \\ e_{\dot{q}_x} \end{bmatrix} + \begin{bmatrix} k_{yx}e_{\dot{q}_x} \\ k_{xy}e_{\dot{q}_y} \end{bmatrix}\right) \\ \Rightarrow K^y_{qd} &= k_{yy} \\ K^x_{qd} &= k_{xx} \\ F^y_{const} &+= \frac{dl_l}{dq_y}k_{yx}e_{\dot{q}_x} + \frac{dl_l}{dq_x}k_{xy}e_{\dot{q}_y} \\ F^x_{const} &+= \frac{dl_r}{dq_y}k_{yx}e_{\dot{q}_x} + \frac{dl_r}{dq_x}k_{xy}e_{\dot{q}_y}. \end{aligned} \tag{14}$$

The components for F_{qd} terms are computed in the same way.

For deriving torque gains, the piston force and joint torque are related by

$$\boldsymbol{\tau} = J^T\mathbf{F}, \tag{15}$$

Preprint submitted to 2015 IEEE-RAS International Conference on Humanoid Robots.
Received July 7, 2015.

where $\boldsymbol{\tau}$ and $\mathbf{F}$ are vectors. The derivation is similar to Eq. 13,

$$\begin{aligned} \mathbf{K}_{Fp}\mathbf{e}_F = J\mathbf{K}_{\tau p}\mathbf{e}_\tau = J\mathbf{K}_{\tau p}J^T\mathbf{e}_F \\ \mathbf{K}_{\tau p} = J^{-1}\mathbf{K}_{Fp}J^{-T}, \end{aligned} \tag{16}$$

and the individual terms are computed same as in Eq. 14.

B. Joint velocity filtering

Joint position and torque are directly measured on the actuator side with LVDTs and pressure sensors inside the pistons, then transformed to joint space. Velocity is computed by filtering the finite difference of positions. In addition to Boston Dynamics' default filters for joint position, velocity, torque and error in torque, second and third order low pass filters with custom parameters are provided. We have the option to replace the default filters with these custom ones in the joint level controllers. In our implementation, we use second order Butterworth filters for joint velocities and keep the default filters for the rest. We found that Boston Dynamics' default velocity filter is very sensitive to sharp velocity changes and amplifies the actual changes. It also filters slow signals more and induces a larger delay. For us, the first attribute leads to undesired rumbling when we use higher F_{qd} or K_{qd} gains. We are able to drastically increase the velocity gains with our simple low pass filters. The cutoff frequencies are summarized in Table I.

TABLE I

CUTOFF FREQUENCIES FOR SECOND ORDER BUTTERWORTH JOINT VELOCITY FILTERS [HZ]

Back z	Back y	Back x	Neck y		
10	10	10	2		
Hip z	Hip x	Hip y	Knee y	Ankle y	Ak. x
10	12.5	15	15	15	15
Shoulder z	Sh. x	Elbow y	El. x	3 Elec. Wrists	
7	7	5	5	10	

C. Joint elasticity compensation

Due to pre-transmission joint position sensing and compliance in the mechanism, forward kinematics is not great on Atlas. For stationary single stance, there can be up to $3cm$ offsets between the model CoM from forward kinematics and the measured CoP. A simple linear torque dependent heuristic to reduce the effects of joint compliance is proposed by [12], which is also employed in our controller.

D. Control loop frequency

The joint level controllers run at $1kHz$, and we can receive states and issue commands at the same frequency. On the other hand, one cycle of our state estimator and control loop takes about $1.5ms$. Stock Ubuntu 12.04 is used on our control computers, and we have observed very consistent and uniform computation time when the X server (for graphical interface) is disabled. Thus we did not pursue a real-time operating system in favor of easier development. In the final implementation, the state estimator runs within the robot communication loop run at $1kHz$, while the controller runs on a separate thread at $500Hz$. We are able to get much more uniform control ticks than what we had for the DRC Trials. The uniform control loop enable us to numerically integrate computed joint accelerations into velocities, which are used as the desired values in the joint level controllers.

V. DISCUSSION AND FUTURE WORK

To be computationally feasible for real-time control, a cascade of smaller optimizations is usually favored, which for us and many others eventually boiled down to reasoning about long term goals using a simplified model and a one-step constrained convex optimization that uses the full kinematics and dynamics. Although effective, this approach suffers from the common problem that plagues hierarchical systems: how to generate a different high level plan when something goes wrong at a lower level, which is not even modeled or represented in the high level planner? We think the ideal solution is to include as complete a model as possible in the high level planner and replan as often as possible, but it is unfortunately currently computationally impractical. A second alternative is to give limited "foresight" to the low level controller and guide it towards the original high level goal in that limited horizon. We have experimented with providing the approximated local value function computed by DDP to the inverse dynamics controller, but we have yet to observe a significant increase in either performance or stability. We think this is due to the fact that we are not previewing far ahead enough in time. For walking, we had the most issues with kinematic related constraints, such as rear ankle joint limit when taking a step down and knee singularity when it goes straight. Just adding inequality constraints on accelerations is not sufficient in most cases. In the end, we worked around these issues with special purpose heuristics that operate in joint space and incur increasingly higher costs as the system approaches these constraints. Although crude, it is a way of injecting some notion of the future into greedy local optimization. Some high level changes are also necessary, such as limiting step length and allowing toe-off in various situations. A third direction is to insert a simpler trajectory optimizer that has a shorter horizon, but replans much more rapidly. We are currently exploring this idea by adding a receding horizon control component at the 1kHz time scale rather than at the 1Hz time scale, as is currently done.

One of our goals is coming closer to human behavior in terms of speed, robustness, full body locomotion, and versatility. A very simple step recovery behavior based on capture point has been implemented. We think high cadence dynamic walking, heel strike, and toe push off are essential to fast robust walking. To achieve this, better system identification will further bridge the gap between simulation and hardware and improve the overall performance. Optimizing angular momentum in the high level controller will also benefit fast walking. It will increase the safety margin for balancing as well.

Preprint submitted to 2015 IEEE-RAS International Conference on Humanoid Robots.
Received July 7, 2015.

VI. Conclusions

In this paper, we presented our controller design and hardware implementation on the Atlas robot for the DARPA Robotics Challenge Finals. Our main progress since the DRC Trials are: a new walking controller that is based on a cascade of online optimizations, hardware related parameter tuning that enabled successful robot application, and an estimator for CoM level modeling errors and unplanned external forces. Although it was tuned to be slow and static for reliability, the walking controller is capable of more dynamic and faster walking with a simple parameter change. The new estimator also serves as an anomaly detector that triggers recovery behaviors when necessary. It successfully prevented catastrophic falls at the Finals, where no safety belay was allowed. Thanks to it, we were the only competitive walking humanoid team at the Finals that did not fall or require a physical human intervention.

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] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, J. P. Graff, P. He, A. Jaeger, J. Kim, K. Knoedler, L. Li, C. Liu, X. Long, F. Polido, M. Gennert, T. Padir, G. G. Tighe, and X. Xinjilefu, "NO FALLS, NO RESETS: Reliable Humanoid Behavior in the DARPA Robotics Challenge," in *submission to Humanoid Robots (Humanoids), 2015 15th IEEE-RAS International Conference on*, 2015.

[2] 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.

[3] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, 2015.

[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] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *preparation for Autonomous Robots*, 2015.

[12] T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot atlas," *preparation for International Journal of Humanoid Robotics*, 2015.

[13] K. Bouyarmane and A. Kheddar, "Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, San Francisco, CA, USA, Sept 2011, pp. 4414–4419.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] N. Mansard, "A dedicated solver for fast operational-space inverse dynamics," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 4943–4949.

[20] 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.

[21] C. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 16, no. 1, pp. 93–101, Jan 1986.

[22] 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.

[23] 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.

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

[25] J. Chestnutt, "Navigation planning for legged robots," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, December 2007.

[26] X. Xinjilefu, S. Feng, and C. Atkeson, "Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention," in *submission to Humanoid Robots (Humanoids), 2015 15th IEEE-RAS International Conference on*, 2015.

[27] 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.

[28] D. Orin and A. Goswami, "Centroidal momentum matrix of a humanoid robot: Structure and properties," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Nice, France, Sept 2008, pp. 653–659.

[29] J. Schulman, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems (RSS)*, Berlin, Germany, June 2013.

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