2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

Dynamic State Estimation using Quadratic Programming

X Xinjilefu, Siyuan Feng and Christopher G. Atkeson

***Abstract*— We propose a framework for using full-body dynamics for humanoid state estimation. It is formulated as an optimization problem and solved with Quadratic Programming (QP). This formulation provides two main advantages over a nonlinear Kalman filter for dynamic state estimation. QP does not require the dynamic system to be written in the state space form, and it handles equality and inequality constraints naturally. The QP state estimator considers modeling error as part of the optimization vector and includes it in the cost function. The proposed QP state estimator is tested on a Boston Dynamics Atlas humanoid robot.**

I. INTRODUCTION

Based on design and control strategies, there are commonly two types of humanoid robots: position-controlled robots and force-controlled robots. Position controlled robots are generally constructed using electric motors. They are usually high impedance, which allows them to track joint level and Cartesian coordinate targets very accurately.

On the other hand, force-controlled robots can achieve a much lower impedance through direct-drive actuation design. The advantage of a force-controlled robot is its compliance. They can better handle external force disturbances, such as a sudden push, or a geometric disturbance such as rough terrain. In addition to joint position sensing, these robots are generally equipped with force sensors, either at the joints or on the actuators. For example, the Atlas humanoid robot shown in Fig. 1, is designed by Boston Dynamics and equipped with pressure sensors on each side of the piston to compute actuator force.

The measured joint force or torque contains dynamic information of the robot. Using this information in a state estimator is not trivial. In a nonlinear Kalman filter framework, one could treat the measured joint force or torque as an input to the system dynamics [1]. However, these measurements are usually noisy, and the noise propagates through the dynamics equation. If instead of using the measured quantities, we choose to use the commanded joint force or torque as an input, then the measurement information is lost. The difficulty is that a nonlinear Kalman filter requires a state equation in the form of Eq. (1) for the process dynamics, but joint force or torque are acceleration level quantities that are difficult to express as part of the state x in the state equation.

$$\dot{x} = f(x, u) \tag{1}$$

Many humanoid robots have force torque sensors under their feet and on the wrist to measure contact forces and torques. Similar to the measured joint forces or torques, the measured contact forces or torques are at the acceleration level in the dynamics equation. They can either be treated as input [2], or be eliminated by projecting the state onto the null space of the constraint Jacobian using orthogonal decomposition [1][3][4].

Fig. 1. The Atlas humanoid robot is constructed by Boston Dynamics. It has 28 hydraulically actuated joints, and has competed in the DARPA Robotics Challenge Trials.

A force-controlled humanoid robot is often subject to many physical constraints, such as joint limits, joint speed limits, torque limits etc.. These constraints can be modeled as linear inequality constraints on the estimator state. A Kalman filter can handle a linear inequality constraint through estimation projection [5], which is essentially solving a QP on the a posteriori estimate.

The moving horizon estimator (MHE) is a more general formulation than the Kalman filter [6][5]. A linear MHE is formulated as a QP and it considers linear inequality constraints on the state naturally [7]. It is equivalent to a Kalman filter if the constraints are removed. A MHE still requires a state space formulation for the system equation, as in Eq. (1). The proposed estimator in this paper is inspired by the MHE. The difference is that a state space formulation

All the authors are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, USA {xxinjile, sfeng, cga}@cs.cmu.edu

is not required in our estimator.

The purpose of this paper is to address these state estimation issues for a force-controlled humanoid robot in a unified framework. We propose to formulate the state estimation problem as a QP problem. The full-body dynamics equation of the robot is used as a linear equality constraint, and various limits are treated as linear inequality constraints. The optimization variable is elaborated in Section III.

This paper is organized as follows. In Section II, we will describe the QP formulation in general. In Section III, we formulate the state estimation problem using QP and describe the objective function and constraints in detail. In Section IV, we show state estimation results of the Atlas robot walking using the Gazebo simulator, and in the real world. Section V discusses future work and the last section concludes this paper.

II. QUADRATIC PROGRAMMING

A Quadratic Programming problem is to optimize a quadratic function of the optimization vector subject to linear constraints on the vector, as shown in Eq. (2)(3) and (4).

$$\underset{\mathcal{X}}{\text{minimize}} \quad \frac{1}{2}\mathcal{X}^T G\mathcal{X} + g^T\mathcal{X} \tag{2}$$

$$\text{subject to} \quad C_e\mathcal{X} + c_e = 0 \tag{3}$$

$$C_i\mathcal{X} + c_i \geq 0 \tag{4}$$

where $\mathcal{X}$ is the unknown vector of optimization variables, the rest of the variables are known vectors and matrices. In our QP state estimator formulation, we optimize a quadratic cost function of the form $||A\mathcal{X} - b||^2$. Therefore $G = A^T A$ and $g = -A^T b$, where A and b can be written in the block form as

$$A = \begin{bmatrix} \alpha_0 A_0 \\ \alpha_1 A_1 \\ \vdots \\ \alpha_n A_n \end{bmatrix}, b = \begin{bmatrix} \alpha_0 b_0 \\ \alpha_1 b_1 \\ \vdots \\ \alpha_n b_n \end{bmatrix} \tag{5}$$

α_i's are the weights associated with each block row. They are user specified to trade off between optimization variables. By construction, we always at least obtain a positive semidefinite matrix G which makes the QP problem convex. We use a dual active set QP solver based on Goldfarb-Idnani [8]. One disadvantage of the Goldfarb-Idnani method is it requires positive definite matrix G, thus the row rank of matrix A is at least the size of $\mathcal{X}$, and the weights must be non-zero. The main advantage of the solver is its speed. In our case, we can solve the QP problem at each time step in a $3ms$ control loop using a modern PC.

III. FULL-BODY DYNAMIC ESTIMATION USING QUADRATIC PROGRAMMING

The full body floating base dynamics of a humanoid robot with acceleration level contact constraints can be represented using the equation of motion and constraint equation as follows

$$M(q)\ddot{q} + h(q,\dot{q}) = S\tau + J_c^T(q)f \tag{6}$$

$$\dot{J}_c(q,\dot{q})\dot{q} + J_c\ddot{q} = \ddot{c} \tag{7}$$

where $q = [p_x, p_q, \theta_J]^T$ is the vector of base position, orientation and joint angles, $\dot{q} = [p_v, p_\omega, \dot{\theta}_J]^T$ is the vector of base velocity, angular velocity and joint velocities. $M(q)$ is the inertia matrix, $h(q,\dot{q})$ is the vector sum for Coriolis, centripetal and gravitational forces. $S = [0, I]^T$ is the selection matrix with zero entries corresponding to the base variables and the identity matrix corresponding to the joint variables. τ is the vector for actuating joint torques. $J_c(q)$ is the Jacobian matrix and $\dot{J}_c(q,\dot{q})$ is its derivative at contact points, and f is the vector of external forces at contact points in the world frame. c is the contact point's position and orientation in Cartesian coordinates.

A. Cost Function

We formulate the state estimation problem as minimizing the weighted sum of the squared estimate error.

The estimate error is composed of two parts, the modeling error w and the measurement error v. The objective function to be minimized is defined as a quadratic form of these errors:

$$\underset{\mathcal{X}_k}{\text{minimize}} \; w_k^T Q^{-1} w_k + v_k^T R^{-1} v_k \tag{8}$$

$\mathcal{X}$ is the optimization variable vector, its definition is given later in this section. The weighting matrices Q and R are positive definite, they play similar roles as the process and measurement noise covariance matrices in the Kalman filter setting. To simplify notation, we omit the time index k whenever it is clear from the context.

1) Modeling Error: Ideally, the rigid body dynamics equation describing the relationship between applied force and acceleration is given by Eq. (6). In a real humanoid robot system, due to modelling error, Eq. (6) will not be satisfied exactly, but within some modeling error bound. So the equality could be rewritten as an inequality within some upper and lower bounds. To maintain the equality form for the dynamics equations, we introduce a slack variable w, also known as the *modeling error*, as

$$M(q)\ddot{q} - S^T\tau - J^T f + h(q,\dot{q}) = w \tag{9}$$

The generalized acceleration can be approximated using a finite difference of the generalized velocity as

$$\ddot{q}_k = \frac{1}{\Delta t}(\dot{q}_{k+1} - \dot{q}_k) \tag{10}$$

Eq. (9) can be re-arranged using Eq. (10) as

$$\begin{bmatrix} \frac{M}{\Delta t} & -I & -S^T & -J^T \end{bmatrix} \begin{bmatrix} \dot{q}_{k+1} \\ w_k \\ \tau_k \\ f_k \end{bmatrix} = -h + \frac{M}{\Delta t}\dot{q}_k \tag{11}$$

We use Eq. (11) as an equality constraint in the QP, and define the vector of optimization variables as

$$\begin{aligned} \mathcal{X}_k &= \begin{bmatrix} \dot{q}_{k+1} & w_k & \tau_k & f_k \end{bmatrix}^T \\ &= \begin{bmatrix} [p_v & p_\omega & \dot{\theta}_J]_{k+1} & w_k & \tau_k & f_k \end{bmatrix}^T \end{aligned} \tag{12}$$

The second line is useful in terms of expressing the measurement error discussed in the next section III-A.2.

Using the notation in Eq. (5), the cost associated with the modeling error can be written as

$$A_w = \begin{bmatrix} 0 & I & 0 & 0 \end{bmatrix} \tag{13}$$
$$b_w = 0 \tag{14}$$

2) Measurement Error: The measurement error v is the difference between the measurement and a linear function of the optimization vector $\mathcal{X}$. In our case, the measurements are the joint velocities and torques measured by the sensors, contact force or torque from force or torque sensors, pelvis angular velocity and linear acceleration from the IMU, and an assumed zero stance foot velocity. The measurement error is expressed using the notation given in Eq. (5). Each term in the measurement error is explained in detail below

a) Joint Velocity: The joint velocity error is computed using

$$A_{\dot{\theta}_J} = \begin{bmatrix} [0 & 0 & I] & 0 & 0 & 0 \end{bmatrix} \tag{15}$$
$$b_{\dot{\theta}_J} = \dot{\theta}_{J,m} \tag{16}$$

where $\dot{\theta}_{J,m}$ is the measured joint velocity. The two leading zeros in matrix $A_{\dot{\theta}_J}$ corresponding to the base linear and angular velocities.

b) Pelvis Angular Velocity: The pelvis angular velocity is measured in the pelvis frame, since the IMU is rigidly attached to the pelvis. The error for pelvis angular velocity is computed using

$$A_{p_\omega} = \begin{bmatrix} [0 & I & 0] & 0 & 0 & 0 \end{bmatrix} \tag{17}$$
$$b_{p_\omega} = p_{\omega,m} \tag{18}$$

where $p_{\omega,m}$ is the IMU measured base angular velocity.

c) Pelvis Linear Acceleration: The pelvis linear acceleration a_m is measured by the IMU, we write the pelvis linear velocity at time $k+1$ as

$$p_{v,k+1} = a_m \Delta t + p_{v,k} \tag{19}$$

The "measured" pelvis velocity is given by

$$A_{p_v} = \begin{bmatrix} [I & 0 & 0] & 0 & 0 & 0 \end{bmatrix} \tag{20}$$
$$b_{p_v} = p_{v,k} + a_m \Delta t \tag{21}$$

where $p_{v,k}$ is the estimated pelvis velocity from previous time step.

d) Joint Torques: The torque on each joint is computed using the oil pressure sensor on each side of the piston. The measured joint torque error is given by

$$A_\tau = \begin{bmatrix} 0 & 0 & I & 0 \end{bmatrix} \tag{22}$$
$$b_\tau = \tau_m \tag{23}$$

e) Stance Foot Velocity: We assume the foot that has substantial z force on it is not moving with respect to the ground. The error in the measured foot velocity is computed by

$$A_{v_{side}} = \begin{bmatrix} J_{side} & 0 & 0 & 0 \end{bmatrix} \tag{24}$$
$$b_{v_{side}} = 0 \tag{25}$$

TABLE I

WEIGHTS USED IN THE COST FUNCTION

w	$\dot{\theta}_J$	p_ω	p_v	v_{side}
3e0	5e4	5e4	1e3	1e5
τ	$f_{side,z}$	$\tau_{side,x/y}$	$f_{side,x/y}$	$\tau_{side,z}$
1e1	1e0	2e0	1e-2	1e-2

where side is either left or right. In double support, both feet are assumed to have zero velocity.

We could use the stance foot velocity as an equality constraint. This constraint is the integral of Eq. (7) given $\ddot{c} = 0$, and it is a stronger constraint than Eq. (7). We find that using a soft penalty with a relative large weight is faster to solve than using the equality constraints.

f) Contact Force and Torque: The Atlas humanoid robot has a 3-axis force/torque sensor under each foot to measure the vertical contact force, and roll and pitch contact torques in the foot frame. We do not have measurement of the horizontal contact forces and yaw torque, thus we put very low weights on these errors (we did not put zero weights on them due to solver limitations).

$$A_{f_{side}} = \begin{bmatrix} 0 & 0 & 0 & R_{side} \end{bmatrix} \tag{26}$$
$$b_{f_{side}} = \begin{bmatrix} 0 & 0 & f_{zm} & \tau_{xm} & \tau_{ym} & 0 \end{bmatrix}^T \tag{27}$$

where R_{side} is a 6 by 6 block diagonal matrix, with the rotation matrix representing foot orientation in the world frame on the diagonal.

Table. I shows the weights used in the cost function.

B. Constraints

In addition to the equality constraints given by Eq. (11) in Section III-A.1, the system is subject to several linear inequality constraints.

The first inequality constraint is for the modelling error w. We notice that w is defined as a generalized force. The limit on modelling error reflects our confidence on the dynamic model. We can turn the dynamic equation Eq. (6) into an equality constraint by setting the limits on w to zero.

Other constrains include torque limits, joint limits and unilateral contact constraint on the vertical force. The torque limit constraints are defined by the maximum torque on each joint.

The joint limit constraints can be written as

$$\theta_{lower} \leq \theta_J + \Delta t \dot{\theta}_J \leq \theta_{upper} \tag{28}$$

Some rotary joints can rotate freely and thus have no joint limits.

The unilateral contact constraint on vertical reaction force can be written as

$$f_z \geq 0 \tag{29}$$

which means the force can only push but not pull.

IV. RESULTS

We test our QP state estimator both in simulation and on real hardware.

A. Implementation

In the walking controller, we servo the robot upper body joints to a fixed desired position. The dynamic model used in the QP state estimator is implemented using SD/Fast.

In the QP state estimator, contact forces and torques are unknown optimization variables, the size of which depends on the contact state (single or double support). We determine contact state by thresholding the measured vertical forces under each foot. In each state estimation step, we solve a QP of the appropriate size.

We assume the joint angles are known and the joint sensors are well calibrated, the base orientation is provided by the IMU mounted on the pelvis. The base position is estimated as in [1].

The QP solver is based on a modified version of the QuadProg++ library.

B. Simulation Results

We used the Gazebo simulator provided by the Open Source Robotics Foundation for the DARPA Virtual Robotics Challenge. The simulator physics is based on the Open Dynamic Engine. We have tested our QP state estimator on various walking patterns. The simulated Atlas robot can walk straight forward, backward and turn on flat ground, and walk up and down slopes and stairs. It can perform these tasks either statically or dynamically. The controller for dynamic walking is based on our full-body inverse dynamics controller, where a feedforward torque is computed for each joint. At the high level, we plan a center of mass trajectory based on desired foot step location using Differential Dynamic Programming; at the low level, we solve inverse dynamics using QP to track the desired center of mass trajectory [9]. The controller for static walking is based on simultaneously solving inverse dynamics and inverse kinematics [10]. We show the simulation results in dynamic walking on a flat ground as an example. The robot is walking at an average speed of 0.4 meter per second.

From Fig. 2 to 4, the simulated Atlas robot went through one walking cycle, starting from left foot single support to the next left foot single support. Double support phase happened at around 17 second and 19.8 second, and lasted for about 0.1 second. During the short double support phase, both heel-strike and toe-off happened. Fig. 2 shows the base velocities in ground truth, estimated by a decoupled Kalman filter [1] and by the QP state estimator. Due to heel-strike and toe-off, the zero velocity assumption for the stance foot is invalid, therefore both state estimators show greater errors during double support. The estimated vertical velocity displays less error for the decoupled KF than for the QP state estimator during swing foot touch-down, because the decoupled KF uses the kinematic model only for the base velocity estimation, and the QP state estimator uses full-body dynamics. Thus a large vertical force during impact introduces more velocity error for the QP state estimators than for the decoupled KF. Fig. 3 plots the estimated joint velocities for all the right leg joints. Both the decoupled KF and the QP state estimators obtain very similar results. We

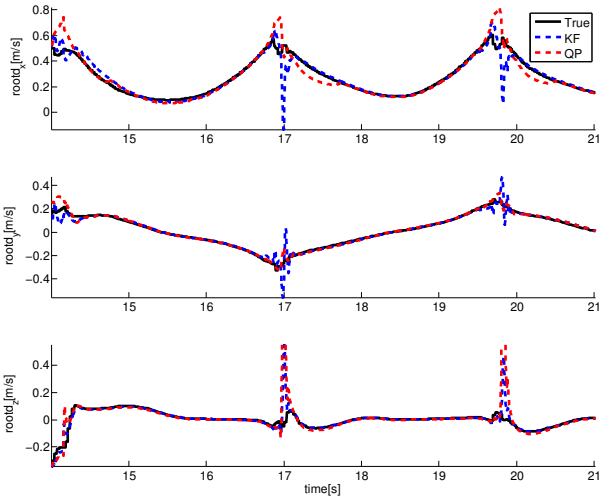

Fig. 2. Simulation data: base velocity from ground truth, decoupled EKF and QP state estimator. Due to heel-strike and toe-off, both state estimators have errors during contact phase transition

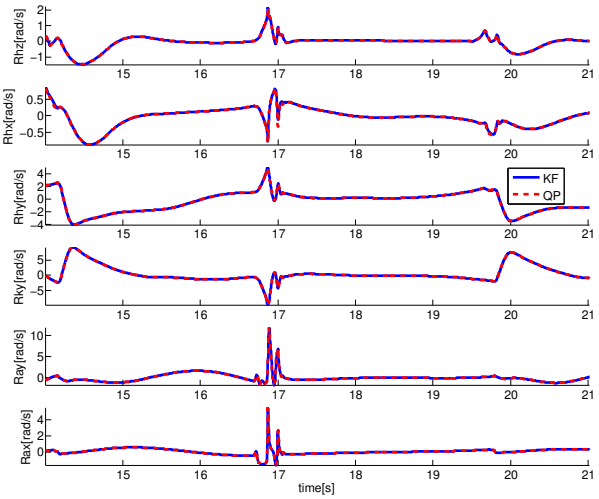

Fig. 3. Simulation data: estimated joint velocities using decoupled KF and QP state estimator. From top to bottom are the right hip yaw, roll, pitch, knee pitch, ankle pitch and roll angular velocities

compare the measured and estimated vertical contact force in Fig. 4. The QP estimated contact forces tracks measurements reasonably well during single support phase. At heel-strike, the touch-down foot contacted the ground and bounced back, resulting in a large force spike which is not well tracked by the QP state estimator, because the equality constraint Eq.(9) will be violated with such a large force. The results indicate that the performance of the QP state estimator is comparable to that of the decoupled KF, and the QP state estimator is able to track measurements well in simulation.

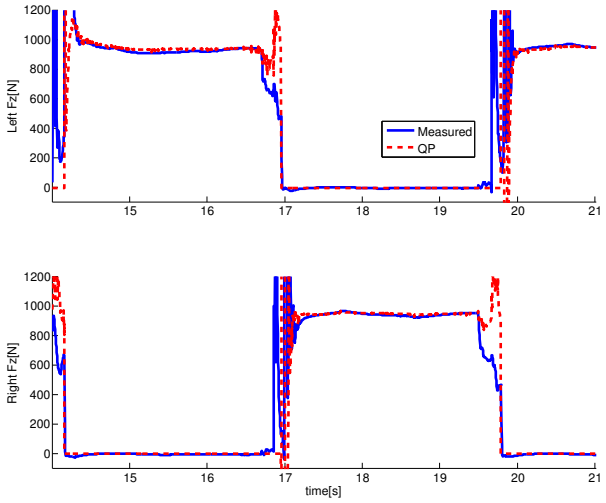

Fig. 4. Simulation data: measured and estimated contact forces. From top to bottom are the left foot and right contact force in the z direction respectively

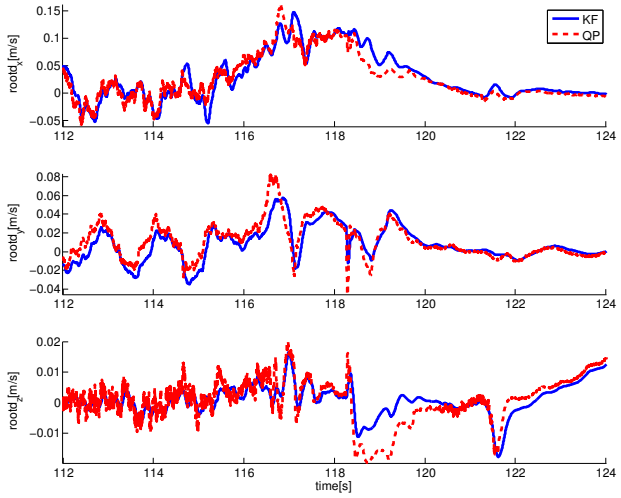

Fig. 5. Robot data: decoupled Kalman filter [1] and QP state estimator estimated base velocity

C. Robot Results

The QP state estimator is also tested on the data collected during the Atlas robot performing a static walking task. The robot was walking up and then walking down several stairs of tilted cinder blocks, as shown in Fig. 8. The controller and state estimator used during the experiment is discussed in detail in [10][1].

We show 12 seconds of data from Fig. 5 to Fig. 7, where Atlas went from left foot single support to double support at around $114.5s$, and from double support to right foot single support at around $121.5s$. Fig. 5 plots the pelvis velocity estimated using the decoupled Kalman filter and the QP state estimator. The QP estimated velocities show more noise but less delay. We plot the measured and estimated joint velocities of all the right leg joints in Fig. 6. The QP state estimator demonstrates good tracking performance on the joint velocities. Fig. 7 is the plot of the measured and estimated right leg joint torques. We notice that there is high frequency chattering in all joint torques. We believe the chattering is caused by our controller trying to compensate for backlash by applying a velocity damping torque, along with various phase shifts and time delays, that drives the actuator to go in opposite directions. The QP state estimator reduces the chattering almost by half.

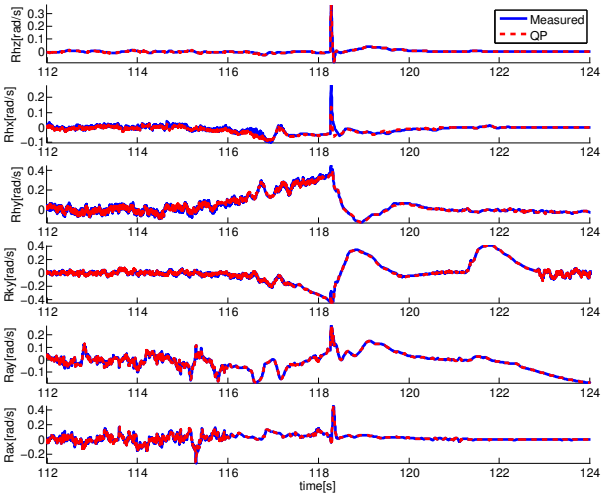

Fig. 6. Robot data: measured and QP estimated joint velocities, from top to bottom are the right hip yaw, roll, pitch, knee pitch, ankle pitch and roll angular velocities

V. DISCUSSION AND FUTURE WORK

The QP state estimator is very similar to a MHE with a window size of one, and an arrival cost of zero. The arrival cost summarizes the cost of all the past data, and a zero arrival cost is also called a uniform prior. Using a zero arrival cost is conservative because we assume we know nothing about previous states.

Compared to the recursive Kalman filter, a QP state estimator offers several advantages. It naturally handles linear constraints in the problem formulation. It is particularly useful for system with many kinematic and dynamics constraints, such as a humanoid robot. A QP state estimator is able to estimate variables not belonging to the state in the state space model, such as joint torques and contact forces. In the joint level servo for the Atlas robot, the valve command is computed using measured joint torque, and filtering its noise helps stabilize the controller. The dynamics equation Eq. (6) is linear in the unknown variables which makes implementation simpler than a nonlinear Kalman filter.

The modelling error can be expressed explicitly in the QP state estimator. It is a tool to directly manipulate the dynamic model. In the Kalman filter framework, the modeling error

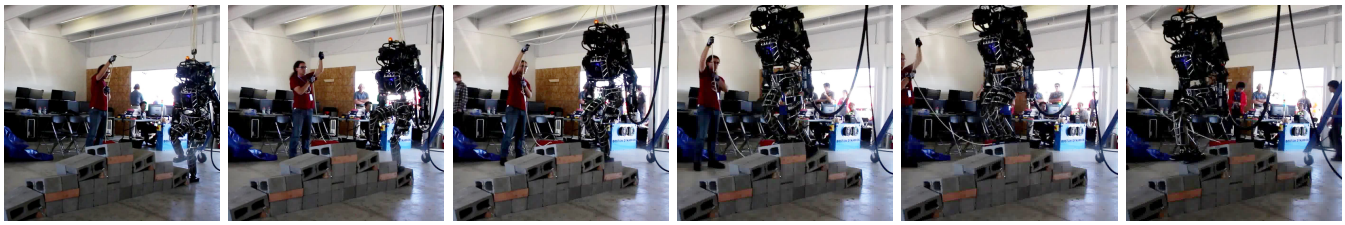

Fig. 8. Snapshots of the Atlas robot walking up and then walking down the tilted cinder blocks.

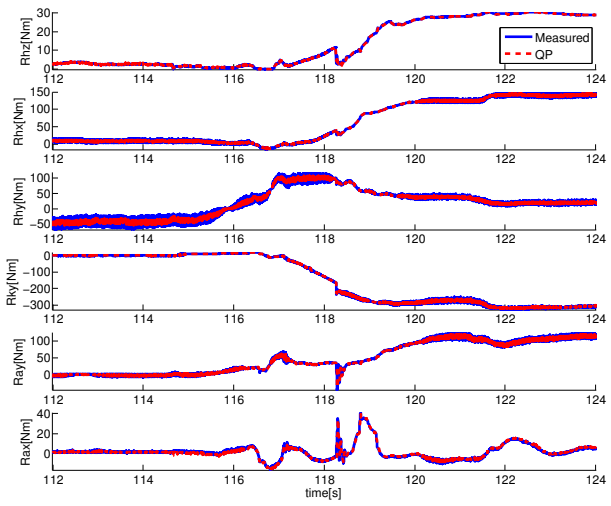

Fig. 7. Robot data: measured and QP estimated joint torque, from top to bottom are the right hip yaw, roll, pitch and knee pitch, ankle pitch and pitch torques

is represented by the process noise, which is assumed to be drawn from a zero-mean normal distribution and independent across time. In Section IV-C, the dynamic model used in the QP state estimator is not quite the same as the actual robot. The modelling error comes from several sources. The hydraulic oil flow during motion introduces error in mass distribution, there are unmodelled stiction and viscous friction in the actuators, etc.. By penalizing modelling error in the cost function, we traded following the dynamic equation exactly off for tracking measurements.

For the future, we will test our QP state estimator with the controller on the Atlas robot.

VI. CONCLUSIONS

We introduced a framework of using full-body dynamics for humanoid state estimation. The main idea is to formulate the dynamic state estimation problem as a QP. This formulation provides two main advantages over nonlinear Kalman filter: it does not require the dynamic system to be written in the state space form, and it handles equality and inequality constraints naturally. The QP state estimator optimizes modeling error directly. We tested the proposed QP state estimator successfully both on simulated and real Atlas robot data.

ACKNOWLEDGMENT

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

REFERENCES

[1] X. Xinjilefu, S. Feng, W. Huang, and C. Atkeson, "Decoupled state estimation for humanoids using full-body dynamics," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 195–201.

[2] X. Xinjilefu and C. Atkeson, "State estimation of a walking humanoid robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 3693–3699.

[3] 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.

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

[5] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," *Control Theory & Applications, IET*, vol. 4, no. 8, pp. 1303–1318, 2010.

[6] C. V. Rao, J. B. Rawlings, and D. Q. Mayne, "Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations," *Automatic Control, IEEE Transactions on*, vol. 48, no. 2, pp. 246–258, 2003.

[7] C. V. Rao, J. B. Rawlings, and J. H. Lee, "Constrained linear state estimation a moving horizon approach," *Automatica*, vol. 37, no. 10, pp. 1619–1628, 2001.

[8] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical programming*, vol. 27, no. 1, pp. 1–33, 1983.

[9] 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*, 2013.

[10] S. Feng, E. Whitman, X. Xinjilefu, and C. Atkeson, "Optimization-based full body control for the DARPA Robotics Challenge," *Journal of Field Robotics*, p. submitted, 2014.