2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Decoupled State Estimation for Humanoids Using Full-body Dynamics

X Xinjilefu, Siyuan Feng, Weiwei Huang and Christopher G. Atkeson

Abstract— **We propose a framework to use full-body dynamics for humanoid state estimation. The main idea is to decouple the full body state vector into several independent state vectors. Some decoupled state vectors can be estimated very efficiently with a steady state Kalman Filter. In a steady state Kalman Filter, state covariance is computed only once during initialization. Furthermore, decoupling speeds up numerical linearization of the dynamic model. We demonstrate that these state estimators are capable of handling walking on flat ground and on rough terrain.**

I. INTRODUCTION

Unlike fixed base robot manipulators, humanoid robots are high degree of freedom dynamical systems with a floating base that can move around in complex environments. State estimation is an integral part of controlling such a system. For a controller using floating base inverse dynamics to compute feed-forward torques, the state estimator needs to provide the location, orientation, and linear and angular velocities of the floating base, as well as the angle and angular velocity of each joint.

In this paper, we propose a framework to use full-body dynamics for humanoid state estimation. Using full-body dynamics is currently too expensive to use in real time state estimation in the standard way. Our approach is to decouple the full body state vector into several independent state vectors. Each decoupled state vector can be estimated very efficiently by using a steady state Kalman Filter (KF). In a steady state KF, state covariance is computed only once during initialization. Furthermore, decoupling speeds up numerical linearization of the dynamic model. We trade partial information loss for a reduction in computational cost.

This paper is organized as follows. In Section II, we will review some related work in state estimation using the KF. In Section III, we formulate the full body state estimation problem as several decoupled state estimation problems. Section IV describes the implementation of each state estimator. In Section V, we show simulation results with the Atlas robot (see Fig. 1) walking using the Gazebo simulator, and actual robot results. Section VI discusses future work and the last section concludes this paper.

II. BACKGROUND

A. The Kalman Filter

Kalman Filter is a recursive filter that estimates the internal state of a linear dynamic system from a series of noisy measurements. To handle nonlinearity, the Extended Kalman Filter (EKF)[1] and later the Unscented Kalman Filter (UKF)[2] were invented. The EKF linearizes the nonlinear dynamics at the current mean estimate, and propagates the belief or information state covariance the same way as the KF. The UKF samples around the mean estimate based on the state covariance to create sampling points (sigma points), and propagates the mean and covariance of the belief or information state using the sigma points. We list the discrete time EKF equations below for future reference. The EKF equations are given in two steps:

Prediction step

$$x_k^- = f(x_{k-1}^+, u_{k-1}) \tag{1}$$

$$P_k^- = F_k P_{k-1}^+ F_k^T + Q_k \tag{2}$$

Update step

$$y_k = h(x_k^-) \tag{3}$$

$$\Delta y_k = z_k - y_k \tag{4}$$

$$S_k = H_k P_k^- H_k^T + R_k \tag{5}$$

$$K_k = P_k^- H_k^T S_k^{-1} \tag{6}$$

$$\Delta x_k = K_k \Delta y_k \tag{7}$$

$$x_k^+ = x_k^- + \Delta x_k \tag{8}$$

$$P_k^+ = (I - K_k H_k) P_k^- \tag{9}$$

The subscript k is the step index, the superscript "$-$" and "$+$" represent before and after the measurement update, capital letters are matrices, and lower case letters stand for vectors. F_k and H_k are Jacobian matrices of f and h linearized around the mean estimate. F_k is the state transition matrix, H_k is the observation matrix, P is the state covariance, and K_k is the Kalman Gain. z_k is the actual measurement, y_k is the predicted measurement, and Δy_k is called the innovation or measurement residual.

In this recursive formulation, one expensive operation is to compute S_k^{-1} in Eq. (6). If F_k and H_k are computed by numeric differentiation at each recursion, they are also computationally expensive. In the linear KF settings, if F_k, H_k, Q_k and R_k are time invariant (constant), then P_k and K_k will converge to their steady state values. It is uncommon for an EKF to have time invariant F_k and H_k. If we assume they are constant over a certain period of time or some states, we could formulate the recursive EKF problem as a steady state EKF problem.

For a discrete time steady state EKF, the steady state covariance P can be obtained through solving the Discrete

X Xinjilefu, Siyuan Feng and Christopher Atkeson are with the Robotics Institute, Carnegie Mellon University USA, Weiwei huang is with Institute for Inforcomm Research Singapore {xxinjile, sfeng, cga}@cs.cmu.edu, huangw@i2r.a-star.edu.sg

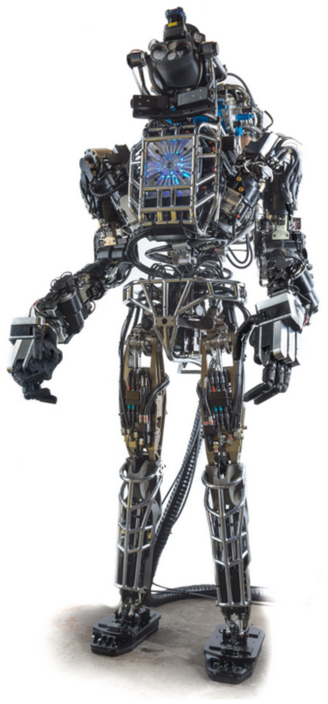

Fig. 1. The Atlas robot built by Boston Dynamics

time Algebraic Riccati Equation (DARE)

$$FPF^T - P - FPH^T(R+HPH^T)^{-1}HPF^T + Q = 0 \quad (10)$$

and the steady state Kalman Gain K is given by

$$K = PH^T(R + HPH^T)^{-1} \quad (11)$$

Given the steady state Kalman Gain K, we could formulate the steady state EKF as

Prediction step

$$x_k^- = f(x_{k-1}^+, u_{k-1}) \quad (12)$$

Update step

$$\Delta y_k = z_k - h(x_k^-) \quad (13)$$

$$x_k^+ = x_k^- + K\Delta y_k \quad (14)$$

B. The Kalman Filter in Legged Robot Locomotion

There is a lot of work on state estimation in legged robot locomotion, and it is not possible to list all the references here. We focus on the work that uses robot dynamics rather than kinematics for state estimation, because dynamics can predict generalized velocity.

In the work of [3], the Sarcos humanoid is modeled by a Linear Inverted Pendulum Model (LIPM) [4] and standing balance with unknown modeling errors is studied with an EKF. An H_2-norm optimal filter is introduced in [5] to estimate the pose and velocity of the links of a humanoid robot. They assume that the motion model is linear and all external forces are known. In locomotion state estimation of quadrupedal [6] and hexapedal [7] robots, hybrid EKFs are used and model transitions are determined by sensors. Sliding model observers are designed and implemented to estimate the absolute torso orientation of a 5-link biped robot during single support [8][9]. Leg kinematics and IMU data are fused to estimate the root pose for the quadruped robot StarlETH[10]. The base state estimator introduced in Section III-A uses a similar approach.

III. DECOUPLED STATE ESTIMATORS

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

$$\begin{aligned} M(q)\ddot{q} + h(q,\dot{q}) &= S\tau + J_c^T(q)f \\ \dot{J}_c(q,\dot{q})\dot{q} + J_c\ddot{q} &= \ddot{c} \end{aligned} \quad (15)$$

where $q = [p_x^T, p_q^T, \theta_J^T]^T$ is the vector of base position, orientation and joint angles, $\dot{q} = [p_v^T, p_\omega^T, \dot{\theta}_J^T]^T$ 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, centrifugal 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. c is the contact point's position and orientation in Cartesian coordinates.

It is possible to formulate a state estimator using the full body dynamics as the process equation, and using sensor information in the measurement equation. One difficulty in this formulation is computation speed. The degrees of freedom for a humanoid are usually over 30. It is not very efficient to do numerical differentiation every time step if we are using an EKF, nor to do multiple forward simulations if we are using an UKF. We could not implement the nonlinear KFs fast enough within one control step. Choosing appropriate filter parameters has also proved to be difficult in this setting, since the Kalman gain essentially depends on the covariance of all the states.

We will introduce an alternative formulation, where we separate the full body dynamics into two parts: the dynamics of the base, and the dynamics of all the actuated joints. In this formulation, the base states are no longer correlated with the joint states. There is information loss due to the decoupling. This will be discussed in Section VI.

A. Base State Estimator

In a floating base humanoid, the base has 3 translational and 3 rotational degrees of freedom. We assume there is an 6-axis IMU attached to the base, which is common. The base state estimator estimates the base global position p_x, orientation p_q, linear velocity p_v, angular velocity p_ω, and the accelerometer bias b_a.

The base filter is modeled as a multiple model EKF with contact switching. We assume the base position and orientation coincide with the IMU. If there is an offset/misalignment, we can always find the fixed transformation between the IMU and the base. The orientation p_q of the base is represented with a quaternion. The angular velocity p_ω is expressed in the base frame. The discrete time process dynamics equations of the base are given by

$$x_k^- = \begin{bmatrix} p_{x,k}^- \\ p_{q,k}^- \\ p_{v,k}^- \\ p_{\omega,k}^- \\ b_{a,k}^- \end{bmatrix} = \begin{bmatrix} p_{x,k-1}^+ + p_{v,k-1}^+ \Delta t \\ p_{q,k-1}^+ + \frac{1}{2} G(p_{\omega,k-1}^+) p_{q,k-1}^+ \Delta t \\ p_{v,k-1}^+ + [R^T(p_{q,k-1}^+)(\tilde{a}_k - b_{a,k-1}^+) - g]\Delta t \\ p_{\omega,k-1}^+ + (\tilde{\omega}_k - \tilde{\omega}_{k-1}) \\ b_{a,k-1}^+ \end{bmatrix} \tag{16}$$

where Δt is the time step, $\tilde{a}$ and $\tilde{\omega}$ are the measured IMU proper acceleration and angular velocity, g is the gravity vector, and $R(\cdot)$ is the rotation matrix for the corresponding quaternion. $G(\cdot)$ is a 4×4 matrix that maps a quaternion to its rate. The fourth equation in Eq. (16) models the gyro bias explicitly in terms of the base angular velocity, because it is one of the base states we want to estimate directly.

The state vector has one more dimension than the state covariance due to the quaternion. We switch from quaternion to rotation vector representation during linearization. For a vector α representing a small angle, its incremental rotation matrix is given by

$$\begin{aligned} \Lambda(\alpha) &:= \exp(\alpha^\times) \\ &= \mathbb{I} + (\alpha^\times)\sin\|\alpha\| + (\alpha^\times)^2(1 - \cos\|\alpha\|) \end{aligned} \tag{17}$$

where $\alpha^\times$ is the cross product matrix of vector α, $\mathbb{I}$ is the identity matrix. The linearized state transition matrix is given by

$$F_k = \begin{bmatrix} \mathbb{I} & 0 & \Delta t\mathbb{I} & 0 & 0 \\ 0 & \Lambda(\Delta t p_{\omega,k-1}^+) & 0 & 0 & 0 \\ 0 & \Delta t\Big(R^T[\tilde{a} - b_{a,k-1}^+]\Big)^\times & \mathbb{I} & 0 & -\Delta t R^T \\ 0 & 0 & 0 & \mathbb{I} & 0 \\ 0 & 0 & 0 & 0 & \mathbb{I} \end{bmatrix} \tag{18}$$

The predicted covariance estimate follows Eq. (2).

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

In the double support phase (DS), we assume the feet are not moving to obtain the following measurements

$$z_{k,DS} = \begin{bmatrix} c_{l,k} \\ c_{r,k} \\ \dot{c}_{l,k} \\ \dot{c}_{r,k} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} c_{l,k-i} \\ \frac{1}{N}\sum_{i=1}^{N} c_{r,k-i} \\ 0 \\ 0 \end{bmatrix} \tag{19}$$

The first two equations say that the current foot position is the average of previous N time step foot positions, and the last two equations say the foot linear velocities are zero. Essentially the first and last two equations convey the same information: the feet are fixed. We decided to fuse the redundant information because the filter will not perform worse under this condition.

To write the measurement equations we need the observation to be a function of the base states. They are given by the floating base forward kinematics $FK(\cdot)$,

$$y_{k,DS} = \begin{bmatrix} FK_{c_l}(q_k^-) \\ FK_{c_r}(q_k^-) \\ FK_{\dot{c}_l}(q_k^-, \dot{q}_k^-) \\ FK_{\dot{c}_r}(q_k^-, \dot{q}_k^-) \end{bmatrix} \tag{20}$$

In Eq. (20), we used filter states from the joint state estimator, which will be discussed in the next section.

The observation matrix H_k is computed by linearizing Eq. (20). This can be done numerically. It is also possible to write the entries of H_k symbolically in terms of the base states.

In single support phase, we assume the stance foot is fixed, so Eq. (19)(20) and H_k are modified to account for contact switching. Contact switching will be discussed in Section IV.

When we compute Eq. (7), the innovation multiplied by the Kalman gain is one dimension less than the state vector. We need to switch from rotation vector to quaternion. Suppose α is a rotation vector, then its corresponding quaternion is given by

$$\xi(\alpha) = \begin{bmatrix} \cos\frac{1}{2}\|\alpha\| \\ \sin\frac{1}{2}\|\alpha\| \frac{\alpha}{\|\alpha\|} \end{bmatrix} \tag{21}$$

Therefore the quaternion component of the updated state estimate is given by

$$p_{q,k}^+ = \xi(\Delta\phi_k) p_{q,k}^- \tag{22}$$

where $\Delta\phi$ is the fourth to sixth components of Δx in Eq. (7).

B. Joint State Estimator

The joint state estimator is composed of two filters: the joint position filter and the joint velocity filter. The reason to separate the joint state estimator into two filters is to simplify computation during linearization, and the joint position θ_J is treated as constant in the joint velocity filter.

1) Joint Position Filter: The process dynamics and state transition matrix are

$$x_k^- = \theta_{J,k}^- = \theta_{J,k-1}^+ + \dot{\theta}_{J,k-1}^+ \Delta t \tag{23}$$

$$F_k = \mathbb{I} \tag{24}$$

We assume each joint angle is measured. The measurement equation and observation matrix are also trivial,

$$y_k = \theta_{J,k}^- \tag{25}$$

$$H_k = \mathbb{I} \tag{26}$$

2) Joint Velocity Filter: The joint velocity filter uses the full body dynamics to estimate joint velocities. Define

$$\ddot{q} = [\ddot{x}_b^T, \ddot{\theta}_J^T]^T \tag{27}$$

$\ddot{x}_b = [\dot{p}_v^T, \dot{p}_\omega^T]^T$ is the base linear and angular acceleration in Cartesian coordinate. The process dynamics is derived from Eq. (15), assuming $\ddot{c} = 0$ and reorganizing it as

$$\begin{bmatrix} M_{x_b} & -J_c^T & M_{\theta_J} \\ J_{c,x_b} & 0 & J_{c,\theta_J} \end{bmatrix} \begin{bmatrix} \ddot{x}_b \\ f \\ \ddot{\theta}_J \end{bmatrix} = \begin{bmatrix} S\tau - h(q,\dot{q}) \\ -\dot{J}_c\dot{q} \end{bmatrix} \tag{28}$$

Rewrite Eq. (28) as

$$A_1 \begin{bmatrix} \ddot{x}_b \\ f \end{bmatrix} + A_2\ddot{\theta}_J = b \tag{29}$$

where
$A_1 = \begin{bmatrix} M_{x_b} & -J_c^T \\ J_{c,x_b} & 0 \end{bmatrix}$, $A_2 = \begin{bmatrix} M_{\theta_J} \\ J_{c,\theta_J} \end{bmatrix}$, $b = \begin{bmatrix} S\tau - h(q,\dot{q}) \\ -\dot{J}_c\dot{q} \end{bmatrix}$.
In [11] and [12], an orthogonal decomposition is used to project motion into the orthogonal complement of the contact Jacobian where the inverse dynamics is solved. We are taking a similar approach here by projecting the allowable motion into the orthogonal complement of A_1. To solve Eq. (29) for $\ddot{\theta}_J$, we perform a QR decomposition on A_1

$$A_1 = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1 \tag{30}$$

where the matrix Q is orthogonal. Multiply Eq. (29) by Q_2^T

$$Q_2^T A_2 \ddot{\theta}_J = Q_2^T b \tag{31}$$

and we can solve for $\ddot{\theta}_J$. Since Q_2 and A_2 are not functions of $\dot{\theta}_J$, and only b is a function of $\dot{\theta}_J$, we only have to modify b during numerical linearization. Now we write the process dynamics of the joint velocity filter as

$$x_k^- = \dot{\theta}_{J,k}^- = \dot{\theta}_{J,k-1}^+ + (Q_2^T A_2)^{-1} Q_2^T b \Delta t \tag{32}$$

The state transition matrix is the linearization of Eq. (32), given by

$$F_k = \mathbb{I} + \Delta t (Q_2^T A_2)^{-1} Q_2^T \left. \frac{\partial b}{\partial \dot{\theta}_J} \right|_{\dot{\theta}_{J,k-1}^+} \tag{33}$$

We assume the joint angle velocities are measured. The measurement equation and observation matrix are given by

$$y_k = x_k^- = \dot{\theta}_{J,k}^- \tag{34}$$

$$H_k = \mathbb{I} \tag{35}$$

IV. IMPLEMENTATION

We implement all the state estimators mentioned in Section III in an EKF framework. At each time step, every filter follows two steps: the prediction step, and the update step. Both prediction and update steps are synchronized across different filters, such that we have access to all the priori states ("$-$") before the update step, and to all the posterior states ("$+$") before the prediction step. As for each filter, the implementation is slightly different. The time step for simulation is $1ms$, and $3.33ms$ on the actual robot.

A. Base State Estimator

The base state estimator is implemented as a recursive EKF. One filter step follows Eq. (1)-(9) and Eq. (16)-(22). The quaternion states need some additional operation. We normalize the quaternion after Eq. (16) to make sure it is unity.

The filter has multiple observation models corresponding to different contact states, and there are two ways to specify the contact state. One way is to use force sensors on the foot, the other is to use the desired contact states from the controller. We used the former to process robot data, and latter in simulation.

B. Joint Position Filter

The joint position filter is implemented with a steady state EKF, since both the state transition (Eq. (24)) and observation (Eq. (26)) matrix are constant. During the filter initialization stage, we pre-compute the steady state Kalman Gains (Eq. (11)) by solving the corresponding DARE with constant F, H, Q and R. Then the filter prediction step involves Eq. (12), and the update step is based on Eq. (13) and (14). Essentially the covariance matrix computation is not needed.

C. Joint Velocity Filter

The joint velocity filter is expensive to implement recursively at each time step due to the numerical linearization in Eq. (33). Instead, we compute Eq. (33) and the steady state Kalman Gain through DARE in a separate thread, and update them whenever new values are available, each update usually takes less than $10ms$. Each time step we use the commanded torque as τ in the prediction step of Eq. (32).

D. Filter Parameters

Each filter has its own set of process and measurement noise covariance parameters. Since the state estimators are decoupled, a set of parameters for one filter has no impact on the other filters. This makes tuning individual filters much easier compared to tuning one large filter with all the correlated states.

E. Controller and Planner

These state estimators provide estimated base position, velocity, orientation and angular velocity, as well as joint positions and velocities to the controller, which is described in detail in [13]. The planner uses the estimated base position at a much lower frequency to build a map, see[14] for details.

V. RESULTS

A. Simulation Results

We test our state estimators together with the controller and planner on a simulated Atlas robot. The Gazebo simulator is based on Open Dynamic Engine and is provided by the Open Source Robotics Foundation for the DARPA Virtual Robotics Challenge. The dynamics involved in the state estimator is implemented using SD/Fast.

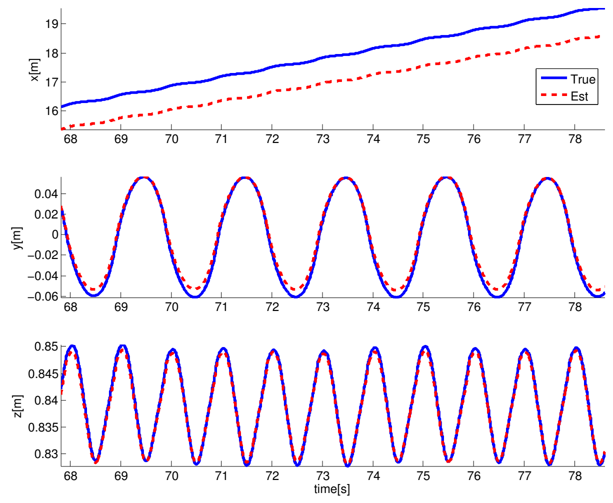
Fig. 2. Simulation data: ground truth and estimated base position

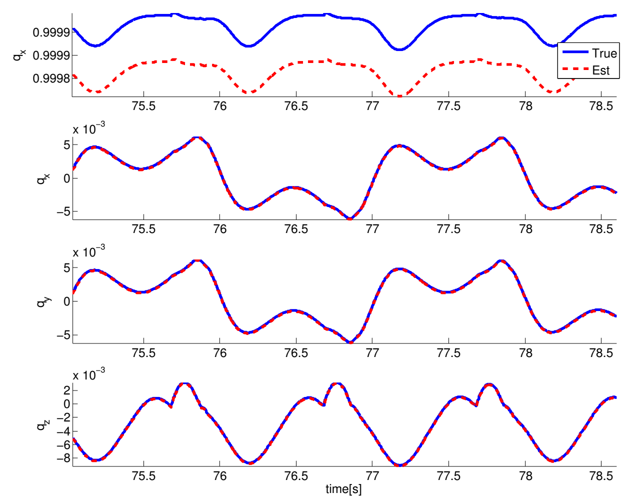
Fig. 4. Simulation data: ground truth and estimated base orientation in quaternion.

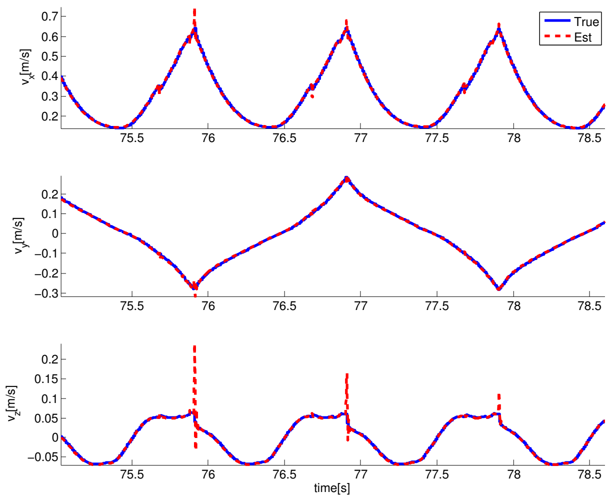
Fig. 3. Simulation data: ground truth and estimated base velocity

We have tested the state estimators on different walking patterns and tasks. The simulated Atlas robot can walk straight and turn on flat ground, and walk up and down slopes. It also walks on rough terrain with different local geometry. We show the results of walking on a flat ground as an example, the traces are very similar in other scenarios.

Fig. 2 is the estimated base position vs. ground truth. The estimated position drifted about 0.6 meters in the forward direction. This happens because there is no actual sensor information to correct position drift. This does not have any impact on the controller or planner as long as they are consistent with the state estimator. The estimated base orientation, velocity, and angular velocity are quite consistent with the ground truth. Fig. 3 shows spikes in the estimated vertical velocity due to a large impact at foot touch-down.

B. Robot Data Processing

We also tested our state estimator on data collected from an actual Atlas robot. The data was taken when the Atlas was walking in place for about one minute, and it was controlled by the Boston Dynamics walking controller. The Boston Dynamics controller uses its own state estimator for the base state estimation. We do not know whether the joint positions and velocities are estimated in the Boston Dynamics controller by any state estimator, or they are simply low pass filtered. Implementing our own state estimator will allow us to do state estimation under a wider range of conditions, including our own walking control as well as filter joint velocities. The details of the Boston Dynamics state estimator are secret, so we can not improve it. Fig. 5 compares the base linear velocity estimated by the Boston Dynamics state estimator, and our decoupled state estimators. Both estimators have similar noise characteristics and the same amount of delay. Fig. 6 plots some joint velocities of the right leg joints. We compare the raw sensor data with the filtered joint velocities from the joint velocity filter. We believe the raw sensor data is the finite difference of the measured joint position with some low pass filtering. It is visible that the filtered states are less noisy than the measurement. There is no delay between the filtered and measured velocities by computing the cross correlation between traces.

The dynamic model used in our state estimator for robot data was identified through robot experiments and is different from the simulation model.

VI. DISCUSSION AND FUTURE WORK

One obvious flaw of decoupling the full state of the robot is information loss. There is always a trade-off between accuracy and computational cost, in our case we favor computational cost because we can not estimate the entire

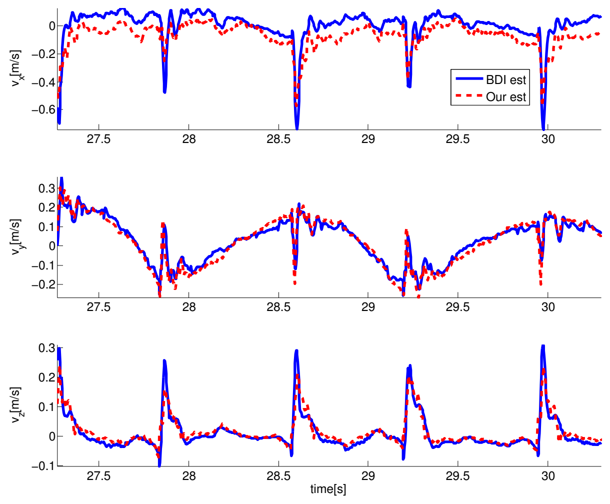

Fig. 5. Robot data: base velocities from Boston Dynamics (BDI) state estimator and our base state estimator

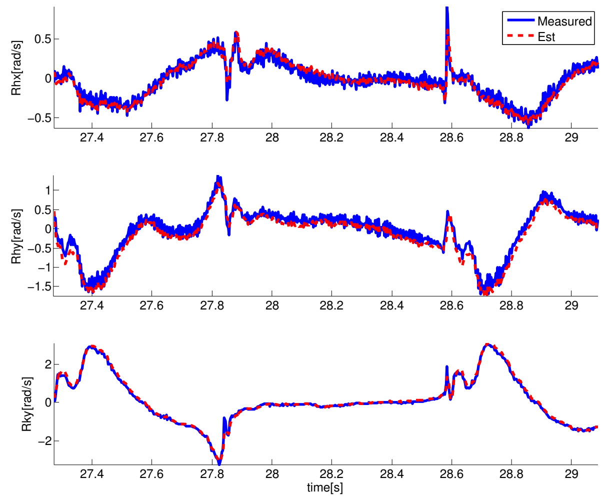

Fig. 6. Robot data: measured joint velocities vs. estimated joint velocities from joint velocity filter. From top to bottom are the right hip roll, pitch and knee pitch angular velocities

state fast enough using one big EKF. An interesting question is, to what degree and in what way can we decouple the state to get optimal performance. It is possible to decouple the joint velocity filter into several filters based on the tree structure of the floating base dynamics (for example, a left leg filter and a right leg filter).

In the base state estimator, we assumed the stance foot was fixed on the ground. This assumption is often violated when we have push-off and heel-strike, or when the foot is slipping which happens a lot in rough terrain simulation. To handle push-off, we move the stance foot reference point from the ankle joint to the toe of that foot, since we know the exact moment of push-off from the controller. Similarly, we move the stance foot reference point to the heel during heel-strike. To handle slipping, we have to detect it first. One way to detect slipping is to learn a slipping model for $\ddot{c}$ in Eq. (15). Another way is to use the innovation, which is the difference between assumed stance foot position-velocity and predicted stance foot position-velocity. To handle slip, we could put a threshold on the innovation, and increase the measurement noise covariance R accordingly. At the same time, we need to switch back to the recursive EKF since R is no longer time invariant. The second approach was implemented, and it made the base velocity trace smoother with fewer spikes.

It is not a surprise that the state estimator works well in simulation, since the state estimator dynamic model is nearly identical to the simulator model. When it comes to real hardware, modeling error is inevitable. Preliminary results show that state estimator with simplified models could work on the Sarcos humanoid [15]. We have shown in the paper that the decoupled state estimator reduces the noise level on the joint velocities of the Atlas. Our future work will be implementing the decoupled state estimator on the Atlas together with our controller and planner.

The decoupled framework provides a modular option to state estimation. We could identify different swing and stance leg dynamics, and switch the filters in different phases without affecting other filters.

VII. CONCLUSIONS

We introduced a framework to estimate the full state of a humanoid robot. The main idea is to decouple the full body state vector into several independent state vectors. These state estimators are implemented on a simulated Atlas Robot, and tested successfully walking on flat ground and rough terrain.

We also showed preliminary results on processing real robot data with the decoupled state estimators. The results indicate a reduction on the measured joint velocity noise.

The main advantage of this approach over a single full EKF is speed, because we are dealing with lower dimensional systems, and using the steady state EKF speeds up numerical linearization of the robot dynamics.

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] B. Anderson and J. B. Moore, "Optimal filtering," *Prentice-Hall Information and System Sciences Series, Englewood Cliffs: Prentice-Hall, 1979*, vol. 1, 1979.

[2] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *AeroSense'97*. International Society for Optics and Photonics, 1997, pp. 182–193.

[3] B. J. Stephens, "State estimation for force-controlled humanoid balance using simple models in the presence of modeling error," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3994–3999.

[4] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1405–1411.

[5] L. Pongsak, M. Okada, and Y. Nakamura, "Optimal filtering for humanoid robot state estimators," in *Proceedings of SICE System Integration Division Annual Conference (SI2002), 2P13-04*, 2002.

[6] S. P. Singh and K. J. Waldron, "A hybrid motion model for aiding state estimation in dynamic quadrupedal locomotion," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 4337–4342.

[7] P.-C. Lin, H. Komsuoglu, and D. E. Koditschek, "Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits," *Robotics, IEEE Transactions on*, vol. 22, no. 5, pp. 932–943, 2006.

[8] V. Lebastard, Y. Aoustin, and F. Plestan, "Finite time observer for absolute orientation estimation of a five-link walking biped robot," in *American Control Conference, 2006*. IEEE, 2006, pp. 6–pp.

[9] Y. Aoustin, F. Plestan, and V. Lebastard, "Experimental comparison of several posture estimation solutions for biped robot rabbit," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1270–1275.

[10] M. Bloesch, M. Hutter, M. Hoepflinger, and C. Gehring, "State estimation for legged robots-consistent fusion of leg kinematics and IMU," in *Robotics: Science and Systems, 2012. RSS 2012.* IEEE, 2012.

[11] 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*. IEEE, 2010, pp. 3406–3412.

[12] 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*. IEEE, 2011, pp. 1085–1090.

[13] 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*. IEEE, 2013.

[14] W. Huang, X. Xinjilefu, S. Feng, and C. Atkeson, "Humanoid Robot Step Planning with Dynamic Cost," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, submitted.

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