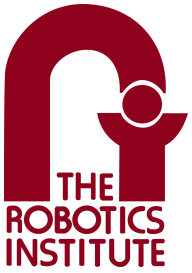


State Estimation for Humanoid Robots

CMU-RI-TR-15-20

Xinjilefu

Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy in Robotics

July, 2015

Thesis Committee: Professor Christopher G. Atkeson, chair

Professor Hartmut Geyer

Professor Alonzo Kelly

Professor Hannah Michalska (McGill University)

Abstract

This thesis focuses on dynamic model based state estimation for hydraulic humanoid robots. The goal is to produce state estimates that are robust and achieve good performance when combined with the controller.

Three issues are addressed in this thesis.

- How to use force sensor and IMU information in state estimation?
- How to use the full-body dynamics to estimate generalized velocity?
- How to use state estimation to handle modeling error and detect humanoid falling?

Hydraulic humanoid robots are force-controlled. It is natural for a controller to produce force commands to the robot using inverse dynamics. Model based control and state estimation relies on the accuracy of the model. We address the issue: "To what complexity do we have to model the dynamics of the robot for state estimation?". We discuss the impact of modeling error on the robustness of the state estimators, and introduce a state estimator based on a simple dynamics model, it is used in the DARPA Robotics Challenge Finals for fall detection and prevention.

Hydraulic humanoids usually have force sensors on the joints and end effectors, but not joint velocity sensors because there is no high velocity portion of the transmission as there are no gears. A simple approach to estimate joint velocity is to differentiate measured joint position over time and low pass filter the signal to remove noise, but it is difficult to balance between the signal to noise ratio and delay. To address this issue, we will discuss three ways to use the full-body dynamics model and force sensor information to estimate joint velocities. The first method efficiently estimates the full state through decoupling. It estimates the base variables by fusing inertial sensing with forward kinematics, and joint variables using forward dynamics. The second method estimates the generalized velocity using quadratic program. Force sensor information is also taken into account as an optimization variable in this formulation. The third method uses low cost MEMS IMUs to measure link angular velocities, and integrate that information into joint velocity estimation.

Some of these state estimators were used on the Atlas robot for full body control, odometry and fall detection and prevention. In the DARPA Robotics Challenge Finals, we achieved 12/14 points and had no fall or human intervention.

Acknowledgements

First and foremost, I would like to thank my advisor, Chris Atkeson, for all of his help and support throughout my time as a graduate student. Thanks are also due to the rest of my committee, Hartmut Geyer, Alonzo Kelly and Hannah Michalska for their advice and comments on this draft. I would also like to thank my entire lab group and my WPI-CMU DRC team for their help and feedback on my research, especial my lab mate Siyuan Feng for his help and discussions during the DRC project. My gratitude also goes to Mike Stilman for this LaTeXtemplate.

This material is based upon work supported in part by the DARPA Robotics Challenge program under DRC Contract No. HR0011-14-C-0011.

Table of Contents

List of Tables

List of Figures

List of Acronyms

BD Boston Dynamics.

CCP Corrected Capture Point.

CoM Center of Mass.

CoP Center of Pressure.

CP Capture Point.

DRC DARPA Robotics Challenge.

EKF Extended Kalman Filter.

IK Inverse Kinematics.

IMU Inertial Measurement Unit.

KF Kalman Filter.

LIPM Linear Inverted Pendulum Model.

LVDT Linear Variable Differential Transformers.

MEMS Microelectromechanical Systems.

MHE Moving Horizon Estimator.

QP Quadratic Program.

RMSE Root Mean Square Error.

SCKF Smoothly Constrained Kalman Filter.

SVD Singular Value Decomposition.

UKF Unscented Kalman Filter.

ZMP Zero Moment Point.

1 Introduction

This thesis focuses on state estimation for hydraulic humanoid robots. My goal is to produce state estimates that are robust and achieve good performance when combined with the controller. Different state estimators were used on the Atlas robot for full body control, odometry and fall detection and prevention. In the DARPA Robotics Challenge (DRC) Finals, we achieved 12/14 points and had no fall or human intervention.

The main idea is to use dynamic models for state estimation. There are three issues addressed throughout this thesis.

- How to use force sensor and Inertial Measurement Unit (IMU) information in state estimation?
- How to use the full-body dynamics to estimate generalized velocity?
- How to use state estimation to handle modeling error and detect humanoid falling?

Hydraulic humanoid robots are force-controlled. It is natural for a controller to produce force commands to the robot using inverse dynamics. Model based control and state estimation relies on the accuracy of the model. Chapter 2 addresses the issue: "To what complexity do we have to model the dynamics of the robot for state estimation?". We briefly discuss the impact of modeling error on the robustness of the state estimators in Chapter 2, and introduce a state estimator based on a simple dynamics model in Chapter 6, it was used

in the DRC finals for fall detection and prevention. Chapter 4 considers modeling error as a generalized force and estimates it. Hydraulic humanoids usually have force sensors on the joints and end effectors, but not joint velocity sensors because there is no high velocity portion of the transmission as there are no gears (this is the case for both Sarcos and Atlas humanoids). A simple approach to estimate joint velocity is to differentiate measured joint position over time and low pass filter the signal to remove noise. If the cut-off frequency of the low pass filter is set too high, the signal to noise ratio will be low; on the other hand, if the cut-off frequency is set too low, the delay will be significant. Both situations are difficult to handle. We introduce three methods to address this issue. We use the full-body dynamics model and force sensor information to estimate joint velocities in Chapter 3 and Chapter 4, and we use an IMU sensor network in Chapter 5. Chapter 3 discusses how to efficiently estimate the full state through decoupling. This chapter also introduces a state estimator to estimate the state of the floating base using forward kinematics. Chapter 4 formulates the generalized velocity estimation problem as a quadratic program. Force sensor information is also taken into account as an optimization variable in this formulation. Chapter 5 discusses how to integrate low cost Microelectromechanical Systems (MEMS) IMUs in joint velocity estimation.

This chapter covers related work in state estimation, and some of its robotics applications involving dynamic models.

1.1 State Estimation

State estimation arises in many fields of science and engineering. The goal of state estimation is to estimate the state of a dynamic system by combining knowledge from *a priori* information and online measurements. We first present a general formulation for state estimation called *full information estimation* in an optimization framework. It is called

full information because we consider all the available measurements.

Full Information Estimation

$$\underset{\mathcal{X}}{\text{minimize}} \quad J(\mathcal{X}) = \|x_0 - \bar{x}_0\|^2_{I_0} + \sum_{k=0}^{N-1} \|x_{k+1} - f(x_k, u_k)\|^2_{Q^{-1}} + \sum_{k=1}^{N} \|y_k - h(x_k)\|^2_{R^{-1}} \tag{1.1}$$

subject to

$$x_{k+1} = f(x_k, u_k) + w_k, \qquad k = 0, 1, \cdots, N-1 \tag{1.2}$$

$$y_k = h(x_k) + v_k, \qquad k = 1, 2, \cdots, N \tag{1.3}$$

$$c_{eq,k}(x_k) = 0, \qquad k = 1, 2, \cdots, N \tag{1.4}$$

$$c_{in,k}(x_k) \geq 0, \qquad k = 1, 2, \cdots, N \tag{1.5}$$

where $\mathcal{X} = \{x_0, x_1, \cdots, x_N\}$ is the vector of the sequence of the states over $N+1$ time steps. $\bar{x}_0$ is the given initial state. y_k's are the available measurements and N is the number of measurements. The notation $\|z\|^2_\Phi = z^T \Phi z$ is the weighted 2-norm of vector z using weighting matrix Φ. $J(\mathcal{X})$ in Eq (1.1) is the objective function which contains several terms. The first term penalizes the error covariance of the initial state, the second term penalizes deviations from the dynamic model, and the last term penalizes measurement errors. Eq (1.2) is the dynamics equation describing the evolution of the system state. Eq (1.3) is the observation equation (also called measurement equation) that generates the measurements from the state. w_k and v_k are termed process and measurement noise respectively. Q and R in Eq (1.1) are the process and measurement noise parameters, they represent the covariance of w_k and v_k respectively. They model uncertainty in the dynamics and measurement equation. I_0 is the initial information matrix, it is equivalent to the inverse of the initial state error covariance matrix P_0. Eq (1.4) and Eq (1.5) are

the equality and inequality constraint imposed on the system state, respectively. The full information estimation has provable stability and optimality properties in a theoretical sense [2], but in practice, it is generally computationally intractable because the dimension of $\mathcal{X}$ grows linearly with N, except for very simple cases. This formulation is valuable because it clearly defines the objective function for a state estimator. The full information estimation problem is formulated as a nonlinear constrained optimization problem, and different solution methods exist to solve the problem [3][4][5]. By imposing certain conditions on the general full information estimation problem, we recover some well-known state estimation techniques, such as the Kalman Filter (KF), the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), and the Moving Horizon Estimator (MHE). They will be discussed in detail in the following sections.

1.1.1 Linear Systems

In this section, we will discuss systems where Eq (1.2)-(1.5) are all linear. Our discussion is organized by whether there are constraints or not imposed on the system state, i.e., whether Eq (1.4)(1.5) are present.

The Kalman Filter

If Eq (1.2) and Eq (1.3) are linear, i.e.,

$$f(x_k, u_k) = Ax_k + Bu_k \tag{1.6}$$

$$h(x_k) = Cx_k \tag{1.7}$$

the noises are Gaussian with zero mean, and the initial condition is Gaussian with mean $\bar{x}_0$,

$$w_k \sim \mathcal{N}(0, Q), \quad v_k \sim \mathcal{N}(0, R), \quad x_0 \sim \mathcal{N}(\bar{x}_0, I_0^{-1}) \tag{1.8}$$

and there is no constraint on the state, then the full information estimation problem can be solved recursively, and this recursive estimator is termed the Kalman filter [6]. The Kalman filter is a recursive optimal state estimator for an *unconstrained linear system* subject to normally distributed state and measurement noise. The Kalman filter can be interpreted as a predictor-corrector method. On each time step, the prediction step estimates the mean and the covariance of the current state, and the update step corrects the predicted mean and covariance of the state using the new measurement. We list the Kalman filter procedure below:

The Kalman filter

Prediction step

$$x_{k+1}^- = Ax_k + Bu_k \tag{1.9}$$

$$P_{k+1}^- = AP_kA^T + Q_k \tag{1.10}$$

Update step

$$\Delta y_k = y_k - Cx_k^- \tag{1.11}$$

$$K_k = P_k^- C^T (CP_k^- C^T + R_k)^{-1} \tag{1.12}$$

$$x_k = x_k^- + K_k \Delta y_k \tag{1.13}$$

$$P_k = (I - K_k C) P_k^- \tag{1.14}$$

the superscript "$-$" represents before the measurement update. K_k is called the Kalman Gain, P_k is the *a posteriori* error covariance matrix, and Δy_k is called the innovation or measurement residual. One attractive property of the Kalman filter is it is recursive and suitable for online implementation. There are alternative formulations of the Kalman filter that provide numerical advantages under certain circumstances.

The *information filter* is an alternative implementation of the Kalman filter. In the Kalman filter, the error covariance matrix P is propagated through Eq (1.10) and Eq (1.14). The information filter propagates the inverse of P instead, and this matrix $I = P^{-1}$ is termed "the information matrix". If the size of the measurement m is much bigger than the number of state n, then it might be more efficient to use the information filter, because it involves inverting several $n \times n$ matrices, instead of inverting the $m \times m$ matrix in Eq (1.12) of the Kalman filter implementation. Also if we have zero certainty in terms of the initial state, there is a numerical difficulty to initialize the error covariance P_0 to infinity, but we can easily set I_0 to a zero matrix using the information filter.

The *sequential Kalman filter* is a Kalman filter implementation that avoids the matrix inversion operation in Eq (1.12) [7]. It works under the assumption that either the measurement noise covariance R_k is constant (time invariant $R_k = R$), or it is diagonal. The idea is to use the component of the measurement vector y_k one by one, thus the name "sequential". It replaces the matrix inversion by m scalar inversions, where m is the size of y_k.

The *square root Kalman filter* was first introduced to improve the precision of the Kalman filter in the late 60's [8][9][10], because the error covariance matrix P can become asymmetric or indefinite over time due to numerical error, or due to some components of the state vector being orders of magnitude larger than others (ill conditioned). A review of the early development of the square root filtering can be found in [11], and [12] gives a comprehensive review. The idea of square root filter is to propagate the square root of

the matrix P. P will always stay symmetric and positive semidefinite, avoiding numerical difficulties.

Linear System with State Constraint

If both the system and constraints are linear, i.e., Eq (1.2)-Eq (1.5), the full information estimation problem becomes a quadratic programming problem. Generally, we can think of this problem in two ways. The first way is to modify the Kalman filter formulation to account for the constraints, so that we can still take advantage of its recursive formulation and implement it online. A detailed discussion of this approach is delayed until Section 2.1. The other way to address the linear state constraint is by solving a quadratic program (Quadratic Program (QP)). Since we know the size of the full information estimation problem grows linearly with the number of measurements samples N, we can not afford to solve the problem over the entire horizon. A moving horizon estimator limits the measurements to the most recent M samples, thus fixes the size of the optimization problem. Early work on the moving horizon estimator can be found in [13][14][15]. The linear MHE is formulated as follows:

Linear Moving Horizon Estimator

$$\underset{\mathcal{X}}{\text{minimize}} \quad J(\mathcal{X}) = \theta_{N-M}(x_{N-M}) + \sum_{k=N-M}^{N-1} \|w_k\|^2_{Q^{-1}} + \sum_{k=N-M+1}^{N} \|v_k\|^2_{R^{-1}} \tag{1.15}$$

subject to

$$x_{k+1} = Ax_k + Bu_k + w_k, \qquad k = N-M, \cdots, N-1 \tag{1.16}$$

$$y_k = Cx_k + v_k, \qquad k = N-M+1, \cdots, N \tag{1.17}$$

$$C_{eq,k}x_k = d_{eq}, \qquad k = N-M+1, \cdots, N \tag{1.18}$$

$$C_{in,k}x_k \leq d_{eq}, \qquad k = N-M+1, \cdots, N \tag{1.19}$$

$\mathcal{X} = \{x_{N-M}, \cdots, x_N\}$ is the vector of all the state from time step $N-M$ to N. The formulation is very similar to that of the full information estimation, except in Eq (1.15), the first term is replaced by the arrival cost $\theta_{N-M}(x_{N-M})$, and the window size of the last two terms is M. The arrival cost θ summarizes the cost of all the past data $\{y_{k=0}^{N-M}\}$ on x_{T-N} [16]. To formulate the MHE as a quadratic program, we need the arrival cost to be a quadratic function of the state x_{N-M}. To take a small step back, if we know the system has no state constraint at all, then the Kalman filter solves the original full information estimation problem, and in the Kalman filter setting, the arrival cost is given by

$$\theta_{N-M}(x_{N-M}) = \|x_{N-M} - \bar{x}_{N-M}\|^2_{I_{N-M}} \tag{1.20}$$

where $\bar{x}_{N-M}$ is the Kalman filter estimated state, and $I_{N-M} = P^{-1}_{N-M}$ is the information matrix (the inverse of the error covariance) at time step $N-M$. The error covariance matrix P is computed recursively through Eq (1.10)-(1.14) starting from P_0.

Due to the inequality state constraints, there is no analytical expression or recursive

formulation to compute the arrival cost in the MHE formulation. A solution is to approximate the arrival cost using the unconstrained error covariance matrix P through the recursive Kalman filter formulation. The approximated arrival cost is given by Eq (1.20). $\bar{x}_{N-M}$ is the estimated state by MHE at time $N-M$. Using the unconstrained error covariance to approximate the arrival cost can make the MHE suboptimal, because the constraint error covariance is always smaller than the unconstrained error covariance [17]. On the other hand, we never underestimate the cost, and if none of the constraints are active at that time step, the approximation is exact. Another solution is to set the arrival cost to zero. This is a very conservative view, because we assume we know nothing about the starting state. This implementation is also called a uniform prior, because it does not penalize any deviations of the initial state from the prior estimate.

The main issue with MHE is it is computationally expensive even for a small window size M.

1.1.2 Nonlinear Systems

We review state estimation algorithms for nonlinear systems in this section, including the Extended Kalman Filter, the Unscented Kalman filter, and the Moving Horizon Estimator.

The Extended Kalman Filter

The EKF is an extension of the Kalman filter to estimate the state of an unconstrained nonlinear system [18]. Its implementation is very similar to the Kalman filter. We linearize the process and measurement equation at the current mean estimate. The EKF is summarized below:

The extended Kalman filter

Prediction step

$$x_{k+1}^- = f(x_k, u_k) \tag{1.21}$$

$$P_{k+1}^- = F_k P_k F_k^T + Q_k \tag{1.22}$$

Update step

$$\Delta y_k = y_k - h(x_k^-) \tag{1.23}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{1.24}$$

$$x_k = x_k^- + K_k \Delta y_k \tag{1.25}$$

$$P_k = (I - K_k H_k) P_k^- \tag{1.26}$$

where $F_k = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_k,u_k}$ and $H_k = \left.\frac{\partial h(x)}{\partial x}\right|_{x_k^-}$

Unlike the Kalman filter, which is the exact and optimal solution for the unconstrained linear full information estimation problem, the EKF is suboptimal. There are alternative implementations of the EKF to improve the filtering accuracy.

The *iterated EKF* (IEKF) [19][20] re-linearize the measurement equation at the *a posteriori* estimate (as many times as desired), because x_k is supposed to be a better estimate than x_k^- after the measurement update. The IEKF performs better especially in the presence of significant nonlinearity in the measurement equation.

The *second-order EKF* [21][22] is an alternative implementation in an attempt to reduce the linearization error of the EKF. The second order EKF does the Taylor expansion of the nonlinear dynamics to second order and updates the *a priori* and *a posteriori* estimate

more accurately.

The EKF is widely applied in engineering fields because it is easy to implement. There are several difficulties that can not be addressed by the EKF. If the initial guess is far off, or the nonlinearity is strong, the EKF can diverge, because it relies on a first or second order Taylor expansion to compute error covariance, which is only accurate locally. Like the Kalman filter, the EKF assumes the process and measurement noise are uncorrelated white Guassian, which no longer holds after a nonlinear transformation. The unscented Kalman filter addresses these issues.

The Unscented Kalman Filter

The UKF [23][24][25] solves an unconstrained nonlinear state estimation problem. One difficulty of the EKF is it tries to propagate the covariance through a nonlinear function using linearization. One of the advantages of the UKF over the EKF is it does not introduce linearization error, which also makes the UKF a derivative-free method. Instead of using linearization, the UKF samples around the mean estimate based on the state covariance to create sampling points (sigma points), and propagates the mean of the state using the sigma points. The covariance of the state is then approximated using the propagated sigma points. The UKF method is summarized below:

The unscented Kalman filter

Prediction step

$$\mathcal{X}_k^{(0)} = x_k \tag{1.27}$$

$$\mathcal{X}_k^{(i)} = x_k + (\sqrt{(L+\lambda)P_k})^{(i)}, \qquad i = 1, 2, \cdots, L \tag{1.28}$$

$$\mathcal{X}_k^{(i)} = x_k - (\sqrt{(L+\lambda)P_k})^{(i)}, \qquad i = L+1, L+2, \cdots, 2L \tag{1.29}$$

$$\mathcal{X}_{k+1}^{-(i)} = f(\mathcal{X}_k^{(i)}, u_k), \qquad i = 0, 1, \cdots, 2L \tag{1.30}$$

$$x_{k+1}^{-} = \sum_{i=0}^{2L} w_m^{(i)} \mathcal{X}_{k+1}^{-(i)} \tag{1.31}$$

$$P_{k+1}^{-} = \sum_{i=0}^{2L} w_c^{(i)} (\mathcal{X}_{k+1}^{-(i)} - x_{k+1}^{-})(\mathcal{X}_{k+1}^{-(i)} - x_{k+1}^{-})^T + Q_k \tag{1.32}$$

Update step

$$\mathcal{Z}_k^{(i)} = h(\mathcal{X}_k^{-(i)}), \qquad i = 0, 1, \cdots, 2L \tag{1.33}$$

$$z_k = \sum_{i=0}^{2L} w_m^{(i)} \mathcal{Z}_k^{(i)} \tag{1.34}$$

$$P_{z_k, z_k} = \sum_{i=0}^{2L} w_c^{(i)} (\mathcal{Z}_k^{(i)} - z_k)(\mathcal{Z}_k^{(i)} - z_k)^T + R_k \tag{1.35}$$

$$P_{x_k, z_k} = \sum_{i=0}^{2L} w_c^{(i)} (\mathcal{X}_k^{-(i)} - x_k^{-})(\mathcal{Z}_k^{(i)} - z_k)^T \tag{1.36}$$

$$\Delta y_k = y_k - z_k \tag{1.37}$$

$$K_k = P_{x_k, z_k} P_{z_k, z_k}^{-1} \tag{1.38}$$

$$x_k = x_k^{-} + K_k \Delta y_k \tag{1.39}$$

$$P_k = P_k^{-} - K_k P_{x_k, z_k}^T \tag{1.40}$$

where the weights are

$$w_m^{(0)} = \frac{\lambda}{L+\lambda} \tag{1.41}$$

$$w_c^{(0)} = \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta) \tag{1.42}$$

$$w_m^{(i)} = w_c^{(i)} = \frac{1}{2(L+\lambda)}, \qquad i = 1, 2, \cdots, 2L \tag{1.43}$$

$$\lambda = \alpha^2(L+\kappa) - L \tag{1.44}$$

Eq (1.27)-(1.29) compute the $2L+1$ sigma points, where the column vector $\mathcal{X}^{(i)}$ is the i^{th} sigma point. Eq (1.30) propagates the sigmal points through the process dynamics, and Eq (1.31)-(1.32) compute the *a priori* state mean estimate and covariance using the weights given by Eq (1.41)-(1.43) [26]. In the update step, Eq (1.33) propagates the sigma points through the measurement equation, and Eq (1.34) computes the predicted measurement. Eq (1.35)(1.36) and Eq (1.38) compute the UKF Kalman gain. After computing the innovation Δy in Eq (1.37), the *a posteriori* state mean and covariance are updated using Eq (1.39)(1.40). L is the dimension of the system, α and κ controls the spread of the sigma points, β is related to the prior knowledge of the distribution of the state.

Van Der Merwe reviewed alternative UKF algorithms in his PhD dissertation [27]. These algorithms differ in how the sigma points and their weights are selected, and how the noise is modeled. In the original paper [23], the UKF formulation does not assume additive noise, and it augments the state mean and covariance by those of the noise. For computational efficiency, it is desirable to use fewer sigma points because both Eq (1.31) and Eq (1.34) can be expensive to evaluate. The *simplex sigma point set* [28] and the *spherical simplex sigma point set* [29] are proposed to reduce the number of sigma points needed. Because the UKF approximates the state statistics only accurately up to the third

moment [23], *higher order unscented filter* [30] was introduced to capture the statistics of order higher than three.

The UKF was also applied to constrained nonlinear systems. One idea is to project the unconstrained sigma points onto the constraint surface. A review of these approaches can be found in [31]. They mainly differ in at which step is the projection taken place. Another approach is to enforce the constraints during the update step, and solve the sigma points through nonlinear programming [32].

The Moving Horizon Estimator

The nonlinear MHE can be used to formulate a constrained nonlinear state estimation problem. The formulation is very similar to the linear MHE, except that any of the Eq (1.16)-Eq (1.19) can be nonlinear. The solution method falls into the category of nonlinear programming. The MHE approach and its variations have been used in the process control community on nonlinear models for two decades [33][34][35][36][37]. The research challenges in nonlinear MHE remain the computational complexity and approximating the arrival cost.

1.2 State Estimation in Legged 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 computer vision, Kalman filters using physics based models have been applied to tracking human body [38]. Stephens studied standing balance for the Sarcos humanoid with unknown modeling errors in [39], the robot is modeled by a Linear Inverted Pendulum

Model (LIPM), and EKFs are used to estimate the center of mass motion of the robot. A similar filter for a planar biped will be introduced in section 2.3.1, this work can be found in [40]. A H_2-norm optimal filter is introduced in [41] to estimate the pose and velocity of the links of a humanoid robot. It assumes that the motion model is linear and all external forces are known. In quadrupedal [42] and hexapedal [43] robot locomotion state estimation, hybrid EKFs are developed and model transitions are determined by inertial sensors or leg configuration sensors. Sliding model observers for a 5-link biped robot are designed and experimented to estimate the absolute orientation of the torso during the single support phase [44][45][46][47]. A planar Spring-Loaded Inverted Pendulum (SLIP) dynamic model was proposed in [48] for model-based state estimation in running. Leg kinematics and IMU data are fused to estimate the root pose for the quadruped robot StarlETH[49].

2

State Estimation With Simple Models

Most material in this chapter has already been published in [40].

Depending on the information required by the controller, state estimator models with different complexity can be developed. In this chapter. we compare two approaches to designing Kalman filters for walking systems. The first design uses Linear Inverted Pendulum Model (LIPM) dynamics, and the other design uses a more complete planar dynamics. We refer the first model as the *LIPM model*, and the second one as the *Five-link Planar model*. The corresponding Kalman filter designs are the LIPM Kalman Filter (KF) and the Five-link Planar KF. The filter based on the simpler LIPM design is more robust to modeling error. The more complex design estimates center of mass height and joint velocities, and tracks horizontal center of mass translation more accurately. We also investigate different ways of handling contact states and using force sensing in state estimation. In the LIPM filter, force sensing is used to determine contact states and tune filter parameters. In the Planar filter, force sensing is used to select the proper measurement equation.

Walking control strategies can be developed based on simplified models. The Linear Inverted Pendulum Model (LIPM) [50], shown in Fig. 2.1, is a very simplified model to describe a walking humanoid. It is explained in detail in Section 2.2.1. The LIPM is often used to generate a Center of Mass (CoM) trajectory that satisfies the Zero Moment Point (ZMP) constraint. The actual dynamics of the robot are more complex, for instance, the upper body rotation should be modeled using angular momentum and cannot be accurately

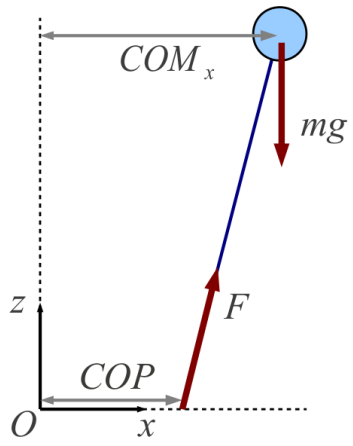


Figure 2.1: For the LIPM model, the body height is constant and the foot force always goes through the CoM

modeled by a point mass, and motion in the lateral and sagittal directions are coupled in 3D rather than decoupled. An interesting question is, what is the appropriate model to use for humanoid walking state estimation? Does modeling the rotation of the upper body, or modeling the dynamics of each link, help state estimation and control of humanoids? Intuitively, complex models use more information about the robot. Whether we have an improvement in the robot performance by using complex models depends on the accuracy of the models, and modeling errors can have different impact on different models.

Among the issues associated with state estimation of humanoid walking, two difficult ones are how to handle contact state and how to handle force sensors. The problem with contact state is that different contact states correspond to different dynamic models, making the walking system hybrid in nature. For example, in bipedal walking, the three basic dynamic models are double support, single support on the left and right foot; furthermore, the feet can either stick or slide with respect to the supporting surface. One way to handle hybrid system state estimation is to use Multiple Model Adaptive Estimation [51]. Knowing when to switch models is essential for this type of estimator to work. In this chapter, we try to avoid explicit model switching by using force sensor data. The force sensors mea-

sure contact forces and torques under each foot, and the data are used in several different ways in state estimation. In the LIPM KF, the force sensor data are used to compute the Center of Pressure (CoP), and to tune the process noise parameters (see section 2.3.1); in the Five-link Planar KF, the data are used as control inputs to the dynamics, and to select the measurement equations for the Five-link Planar KF (see section 2.3.2).

This chapter is organized as follows. In Section 2.1, we will review related work, and introduce the Kalman filter with state constraints. In Section 2.2, two models of the Sarcos humanoid robot are introduced. Section 2.3 describes the filter design of both models. In Section 2.4, we simulate both Kalman filters and compare their performance. Section 2.5 shows filtering results on robot data. Section 2.6 discusses future work and the last section concludes this chapter.

2.1 Kalman Filter with State Constraint

In bipedal walking, one or both feet are on the ground. This unilateral contact creates kinematic constraints for the biped system, and these constraints have to be accounted for in state estimation. Kalman filtering with state constraints is an open research area, and there have been many studies in the past few decades. A recent survey paper on this topic [52] introduces several algorithms to handle linear and nonlinear state constraints in a Kalman filter.

Suppose x_k is the state of the Kalman filter at time step k. Linear equality constraints are defined as

$$C_{eq}x_k = d_{eq} \tag{2.1}$$

and linear inequality constraints are defined as

$$C_{in}x_k \leq d_{in} \tag{2.2}$$

where C_{eq} and C_{in} are known matrices, and d_{eq} and d_{in} are known vectors.

Different ways to enforcing linear constraints in a Kalman filter include model reduction [53], perfect measurements [54][55][56], estimate projection [17], gain projection [57], and system projection [58]. Model reduction reduces the system parameterization by explicitly using the equality constraints, and does not apply to inequality constraints. Perfect measurements augments the measurement equation with the equality constraints without noise, and it also is not applicable to inequality constraints. The idea of estimation projection is to project the unconstrained estimate onto the constrained surface; for inequality constraints, a quadratic programming problem needs to be solved. Gain projection is effectively equivalent to estimate projection. The a posteriori estimate obeys the constraints by modifying the Kalman gain.

Nonlinear state constraints can be enforced in various ways. The easiest way is to linearize the constraints at the current estimate, and apply the algorithms for linear constraints. The resulting estimates only approximately satisfy the nonlinear constraints. To be more accurate, we could use the second order Taylor expansion of the constraints, and apply estimation projection methods. There are methods that are more difficult to implement, such as the Smoothly Constrained Kalman Filter (SCKF) [59] and Moving Horizon Estimator (MHE) [13]. Both methods use the nonlinear constraints in the augmented measurement equation.

2.2 System modeling

In the section, we will first describe two robot models, one with simplified dynamics and the other using more complete dynamics of the robot. Both models are planar.

2.2.1 Linear Inverted Pendulum Model

LIPM assumes the CoM is at a constant height, therefore the total vertical force must be equal to the body weight at all times. The dynamic equation of LIPM in the horizontal direction is given by Eq. (2.3)

$$\ddot{x}_{com} = \frac{g}{z_0}(x_{com} - x_{cop}) \tag{2.3}$$

where x_{com} is the horizontal position of the CoM, x_{cop} is the position of CoP, g is gravity, and z_0 is a constant representing the height of CoM. In Eq. (2.3), x_{cop} is the total CoP, it can be further written as the weighted sum of the CoP under each foot. Assuming point feet for the model, the CoP under each foot is the same as the foot location.

$$x_{cop} = \alpha x_L + (1 - \alpha)x_R \tag{2.4}$$

where x_L and x_R are the horizontal positions of the left and right foot, respectively. α represents the distribution of vertical forces between the two feet, and is given by

$$\alpha = \frac{F_{zL}}{F_{zL} + F_{zR}} \tag{2.5}$$

where F_{zL} and F_{zR} are the vertical ground reaction force under left and right foot, respectively. It is reasonable to believe that the left foot is on the ground when α is close to 1, and the right foot is on the ground when α is near zero. An alternative CoM dynamics

equation is obtained by substituting Eq. (2.4) into Eq. (2.3):

$$\ddot{x}_{com} = \frac{g}{z_0}[x_{com} - \alpha x_L - (1-\alpha)x_R] \tag{2.6}$$

2.2.2 Planar Five-link Model

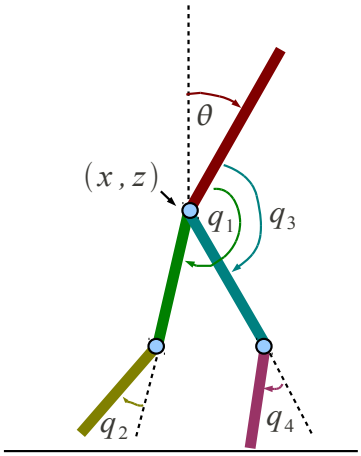


Figure 2.2: Schematic of planar biped robot

A more complete walking model of our Sarcos humanoid robot is the planar five-link model. It consists of a torso link and two identical legs with knee joints (Figure 2.2). The variables (x, z) represents torso position, θ represents torso orientation, q_is are the internal joint angles, $i = 1, \cdots, 4$. The planar model is actuated at the hip joints and knee joints. The point feet coincide with the actuated ankles, which apply torque when they are in contact with the ground. A real world implementation of this model is the “Rabbit" robot [60]. It is equipped with gears at both knee joints and hip joints, but the ankle joints of “Rabbit" are unactuated. In our case, we use this model to approximate the 2D walking motion of the Sarcos robot in the sagittal plane with the upper limb joints servoed to a

fixed position. The dynamic model of this planar biped can be represented as

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + h(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \boldsymbol{\tau} + J^T(\boldsymbol{q}) f_{ext} \tag{2.7}$$

where the generalized coordinates are defined as $\boldsymbol{q} = \begin{bmatrix} x & z & \theta & q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T$, $\dot{\boldsymbol{q}}$ is the vector of generalized velocities and $\ddot{\boldsymbol{q}}$ is the vector of the generalized acceleration. $M(\boldsymbol{q})$ is the inertia matrix, $h(\dot{\boldsymbol{q}}, \boldsymbol{q})$ is the vector accounting for the gravity, centripetal and Coriolis forces, and $\boldsymbol{\tau}$ is the vector for the actuating torques at the hip and knee joints, where the first three components of this vector is zero since those degrees of freedom are not actuated. The last term $J^T(\boldsymbol{q})f$ represents external forces/torques applied to the system, where $J(\boldsymbol{q})$ is the contact Jacobian matrix mapping the applied force/torque locations on the robot in world coordinates to the generalized coordinates, and f is the vector of all external forces/torques applied to the robot represented in world coordinates.

Eq. (2.7) is a compact description of the robot dynamics, but it can still represent the hybrid nature of the dynamics. During walking the robot experiences different phases, such as double support or single support phases. Each of these phases corresponds to different forms of Jacobians and applied forces.

2.2.3 Sensors

The sensors on the robot are potentiometers measuring joint angles and load cells measuring joint torques on each joint, the force/torque sensors to measure the contact force and torque under each foot, and an IMU mounted on the torso. The IMU measures linear accelerations and angular velocities of the torso in torso coordinates, and also provides absolute orientation of the body by internally filtering measured accelerations and angular velocities. The LIPM KF uses the orientation measurement from the IMU, and the Five-link Planar KF uses the acceleration and angular velocity measurements. With these

measurements, we would like to estimate the CoM position and velocity. In the Five-link Planar case, we would like to estimate the internal joint velocities with minimum delay as well so that we can control the robot better.

2.3 Filter design

In this section, we design Kalman filters for each robot model described in Section 2.2.

2.3.1 LIPM Kalman Filter

We adopt the filter designed in [39] to the planar biped. The main idea is to approximate the full robot dynamics with the LIPM dynamics, and estimate the position and velocity of CoM directly. An Extended Kalman filter is constructed from Eq. (2.6), where the equation is discretized and put into state space form as

$$\boldsymbol{x}_{k+1}^{L} = \boldsymbol{f}_{LIPM}(\boldsymbol{x}_{k}^{L}) \tag{2.8}$$

where $\boldsymbol{x}_k^L = [x_{com} \ \dot{x}_{com} \ x_L \ x_R \ \alpha]^T$ is the state at time step k, the superscript L stands for LIPM. It is assumed that there is no external input and the system is driven by disturbances. The measurement model includes CoM position relative to both feet derived from robot kinematics, and the parameter α computed from the force sensor measurement:

$$\boldsymbol{y}_{k}^{L} = \boldsymbol{h}_{LIPM}(\boldsymbol{x}_{k}^{L}) = [x_{com} - x_L \ \ x_{com} - x_R \ \ \alpha]^T \tag{2.9}$$

This measurement model assumes that the relative CoM position to the foot can be computed from the data. Because the absolute torso orientation is extracted from the IMU, by assuming the torso position at the origin, we can compute the CoM and feet positions

by forward kinematics, since all the joint angles are measured by the sensors. Then we simply use the differences as the measurement. α is computed using the data from force sensors under each foot (Eq. (2.5)).

In [39], state dependent process noise is assumed for the foot positions. The rational is that when the foot is on the ground, its position should have lower process noise than when it is in the air. Whether the foot is in the air or on the ground is determined by the state variable α. We avoid using multiple models by introducing α (which depends on the force sensor data) and tuning the process noise.

Eq. (2.8)(2.9) together with the noise model form the process and measurement equations for the LIPM KF model.

2.3.2 Five-link Planar KF

A nonlinear Kalman filter can be constructed using the full body dynamics equation in Eq. (2.7) by first rewriting it in the control affine form as

$$\ddot{\boldsymbol{q}} = -M^{-1}(\boldsymbol{q})h(\dot{\boldsymbol{q}}, \boldsymbol{q}) + M^{-1}(\boldsymbol{q})(\boldsymbol{\tau} + J^T(\boldsymbol{q})f_{ext}) \tag{2.10}$$

The generalized coordinates $\boldsymbol{q}$ can be divided into two parts $\boldsymbol{q} = [\boldsymbol{q}_{base}^T, \boldsymbol{q}_{joint}^T]^T$, where the base vector is $\boldsymbol{q}_{base} = [x\ z\ \theta]^T$ and the joint vector is $\boldsymbol{q}_{joint} = [q_1\ q_2\ q_3\ q_4]^T$.

We assume the joint torques (with noise) are measured by load cells. The joint torque vector $\boldsymbol{\tau}$ together with the reaction force vector f_{ext} measured by the foot force sensors are treated as input the Five-link Planar KF, defined by $\boldsymbol{u} = [\boldsymbol{\tau}^T, f_{ext}^T]^T$. Eq. (2.10) can be

put into state space form as

$$\begin{aligned}\dot{\boldsymbol{x}}^P &= \begin{bmatrix} \dot{\boldsymbol{q}} \\ -M^{-1}h(\dot{\boldsymbol{q}},\boldsymbol{q}) + [M^{-1}, M^{-1}J^T]\boldsymbol{u} \end{bmatrix} \\ &= \boldsymbol{f}(\boldsymbol{x}^P) + \boldsymbol{g}(\boldsymbol{x}^P)\boldsymbol{u} \end{aligned} \tag{2.11}$$

with $\boldsymbol{x}^P = [\boldsymbol{q}^T \ \ \dot{\boldsymbol{q}}^T]^T$ defined as the state vector, superscript P stands for "Planar".

Since the input $\boldsymbol{u}$ to the process dynamics is given, we do not have to deal with potential multiple models in process dynamics. We rewrite Eq. (2.11) in the discretized form to obtain the filter process equation as follows:

$$\boldsymbol{x}^P_{k+1} = \boldsymbol{f}_{Planar}(\boldsymbol{x}^P_k) + \boldsymbol{g}_{Planar}(\boldsymbol{x}^P_k)\boldsymbol{u} \tag{2.12}$$

The measurement data used in the Five-link Planar KF come from two sources: sensors and kinematic constraints.

The sensor data are joint position measurements from potentiometers (four from $\boldsymbol{q}_{joint}$ and two from ankle joints), the linear accelerations in torso coordinates and angular velocity of the torso from IMU ($[\ddot{x}_t \ \ \ddot{z}_t \ \ \dot{\theta}]$), where the subscript t stands for "torso".

The kinematic constraints are as follows: in single support, the supporting foot has to be on the ground, therefore its vertical position is set to the ground level (we assume the surface is solid so there is no up-down motion of the supporting foot); in double support both feet are on the ground. The ground level is treated as known and set to 0. We make an additional assumption that the supporting foot is not sliding relative to the ground, which means its horizontal position is fixed. However, this constraint is not trivial to implement directly because we need to keep tracking the standing foot position. We impose an equivalent constraint that the velocity of the standing foot is zero. Both kinematic constraints are treated as perfect measurements.

Given the measurement data, we set the measurement equation accordingly. The linear accelerations are functions of filter states $\boldsymbol{x}^P$ and input $\boldsymbol{u}$ (using forward dynamics), the torso angular velocity $\dot{\theta}$ as well as the four joint positions $\boldsymbol{q}_{joint}$ are filter states. The rest of the measurement equation depends on the phase the robot is in. In single support, we use the ankle joint of the standing foot, its vertical position and horizontal velocity as measurements (all computed from the torso using forward kinematics); in double support phase, we use these measurements for both feet. The filter update equation is multiple-model as the equations used are different in different phases. We determine the phase based on the force sensor data (Eq. (2.5)). The measurement equation is given by

$$\boldsymbol{y}_k^P = \boldsymbol{h}_{Planar}(\boldsymbol{x}_k^P) = \begin{cases} [\ddot{x}_t \;\; \ddot{z}_t \;\; \dot{\theta} \;\; \boldsymbol{q}_{joint}^T \;\; \phi_L \;\; z_L \;\; \dot{x}_L]^T & \text{LSS} \\ [\ddot{x}_t \;\; \ddot{z}_t \;\; \dot{\theta} \;\; \boldsymbol{q}_{joint}^T \;\; \phi_R \;\; z_R \;\; \dot{x}_R]^T & \text{RSS} \\ [\ddot{x}_t \;\; \ddot{z}_t \;\; \dot{\theta} \;\; \boldsymbol{q}_{joint}^T \;\; \phi_L \;\; z_L \;\; \dot{x}_L \;\; \phi_R \;\; z_R \;\; \dot{x}_R]^T & \text{DS} \end{cases} \tag{2.13}$$

where ϕ is the ankle joint position, z_L, z_R are the foot vertical positions, $\dot{x}_L, \dot{x}_R$ are the horizontal foot velocities, L and R stand for left and right respectively.

We refer to Eq. (2.12) and Eq. (2.13) with additive noise as the Five-link Planar KF model. As both equations in the Five-link Planar model are nonlinear, nonlinear Kalman filters such as EKF or UKF (unscented Kalman filter) may be applied. An EKF is implemented here.

The Five-link Planar KF model estimates generalized coordinates and generalized velocities. Because we are also interested in the CoM position relative to the standing foot as well as CoM velocity, these quantities can be generated by forward kinematics.

2.4 Simulation results

2.4.1 Simulation Parameters

In this section, we will show and compare the simulation results of both Kalman filters. The parameter values used in the simulation are given in Table 2.1.

Table 2.1: Parameters used in dynamic simulation

	torso	thigh	calf
mass(kg)	50.0	5.676	6.8952
inertia about CoM(kgm^2)	1.5	0.0959	0.1535
length(m)	0.8	0.3918	0.3810
local CoM(m)	0.2868	0.1898	0.2384

The filters are updated at a frequency of $400Hz$. The simulated sensor data has additive Gaussian noise with constant bias, and their noise parameters are based on the actual sensor units equipped with the Sarcos Humanaoid robot shown in Table 2.2.

Table 2.2: Noise parameters for simulated sensor data

	$\boldsymbol{q}_{joint}$ (rad)	$\dot{\theta}$ (rad/s)	$\ddot{x}_t, \ddot{z}_t$ (m/s^2)	f_{ext} (N)	τ (Nm)
Bias	$1e^{-2}$	$1e^{-2}$	$1e^{-1}$	1	0.5
Std σ	$1e^{-3}$	$3.2e^{-2}$	$1.28e^{-1}$	$1e^{-1}$	$1e^{-1}$

2.4.2 Kalman Filter Comparison

Table 2.3: LIPM KF noise parameters

	x_{com}	$\dot{x}_{com}$	x_L, x_R	α
Q_{LIPM}	$1e^{-10}$	$1e^{-9}$	$1e^{-10}/1e^{-6}$	$1e^{-5}$

	$x - x_L$, $x - x_R$	α
R_{LIPM}	$1e^{-6}$	$1e^{-6}$

Table 2.4: Five-link Planar KF noise parameters

	x, z	θ	$\boldsymbol{q}_{joint}$	$\dot{x}, \dot{z}$	$\dot{\theta}$	$\dot{\boldsymbol{q}}_{joint}$
Q_{Planar}	$1e^{-8}$	$1e^{-8}$	$1e^{-10}$	$1e^{-6}$	$1e^{-8}$	$1e^{-5}$

	$\ddot{x}_t, \ddot{z}_t$	$\dot{\theta}$	$\boldsymbol{q}_{joint}$	ϕ
R_{Planar}	$1e^{-2}$	$1e^{-4}$	$1e^{-6}$	$1e^{-6}$

The choice of noise parameter values for the filter determines its performance. The process noise covariance Q reflects how confident we are about the process model, and the measurement noise covariance R reflects how much we trust the measurement model. The values are chosen in an attempt to minimize the root mean squared (RMS) error of the filter states. For the LIPM Kalman filter, the chosen noise parameters are given in Table 2.3, and noise parameters for the Five-link Planar KF are shown in Table 2.4. The simulated sensor data has constant biases, they should be considered as modeling errors rather than random noises, and they can be dealt with directly by introducing additional states to the filter (double the number of states). [39] discusses the effect of modeling error

on state estimation. We do not model these offsets in either Kalman filter, and try to compensate their effects through tuning the noise parameters.

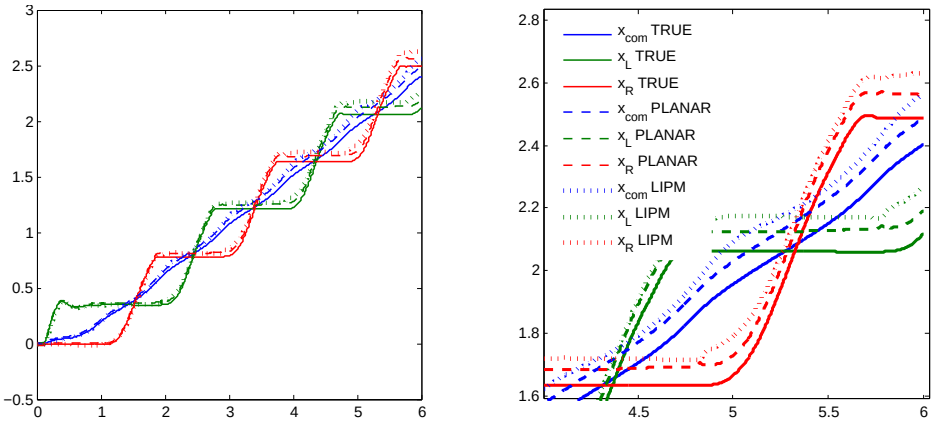


Figure 2.3: CoM and feet positions, from ground truth, Five-link Planar KF and LIPM KF; figure on the right is a blowup view of the plot on the left

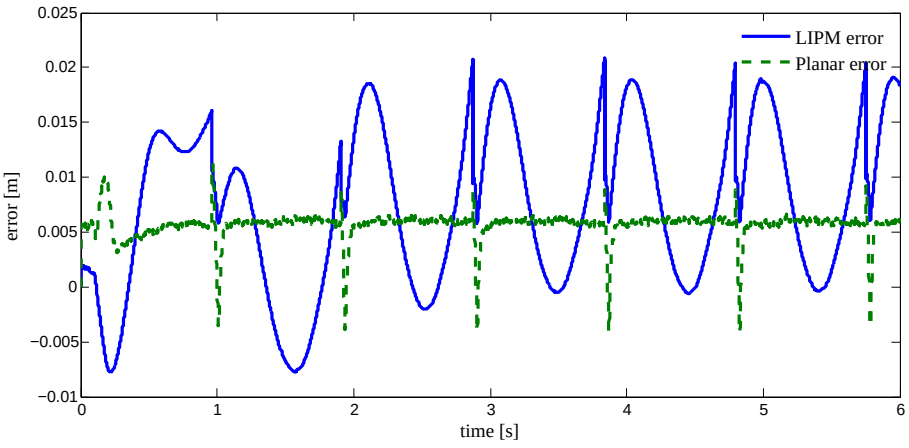


Figure 2.4: Total position error on $x_{com} - x_{cop}$

Before working on the robot data, we simulate normal speed forward walking using a five-link planar humanoid model for 6 seconds, and perform Kalman filtering on the simulated sensor data. The results are shown below. In Fig. 2.3, we plot CoM position x_{com} in blue, left foot position x_L in green and right foot position x_R in red. The solid lines

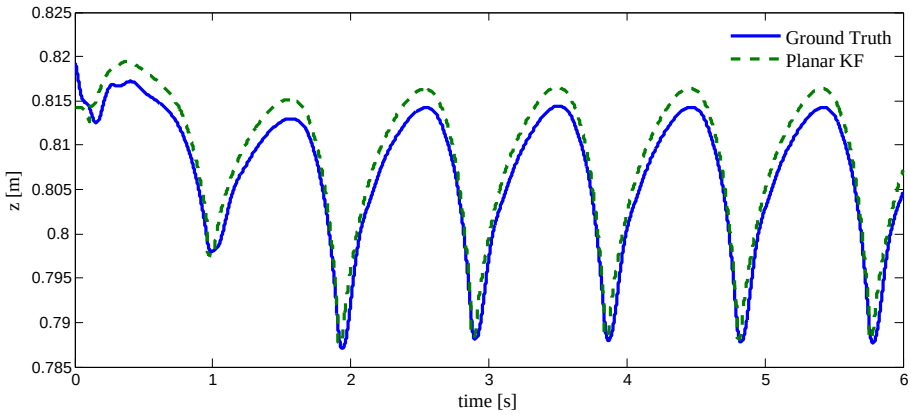


Figure 2.5: CoM vertical position

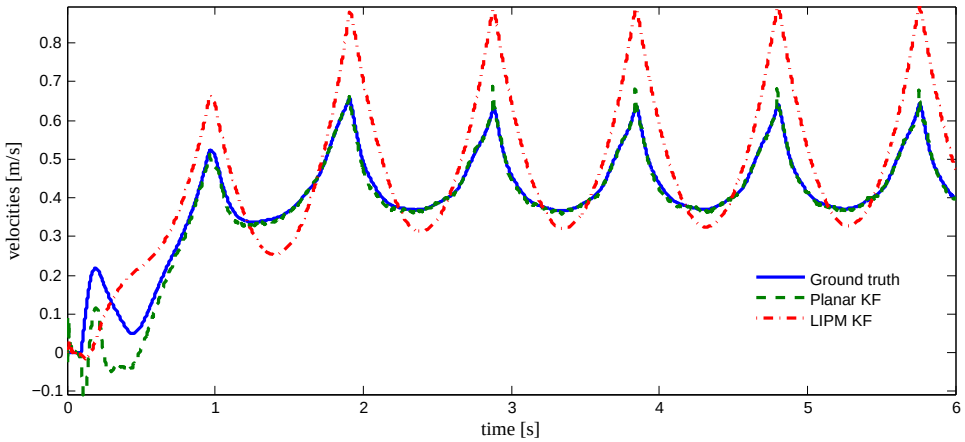


Figure 2.6: CoM horizontal velocity, ground truth, Five-link Planar KF and LIMP KF

are ground truth from simulation, dashed lines are estimates from the Five-link Planar KF, and dotted lines are estimates from the LIPM Kalman filter. Both estimates drift away from the ground truth, and the LIPM KF diverges faster than the Five-link Planar KF. Fig. 2.4 shows the total error between CoM and CoP defined by

$$error = [(x_{com} - x_{cop})_{est} - (x_{com} - x_{cop})_{true}] \tag{2.14}$$

where the subscript “est" stands for “estimated". Both filters reach a minimum error in double support phase, and the Five-link Planar KF in general performs better than the LIPM KF in estimating the relative horizontal position. We notice the constant offset in Five-link Planar KF error trace due to measurement biases. Although the LIPM model assumes CoM at a constant height, the Five-link Planar KF is able to estimate the CoM height, shown in Fig. 2.5. Both Kalman filters can estimate horizontal CoM velocity, the results compared to the ground truth are shown in Fig. 2.6. The Five-link Planar KF clearly outperforms the LIPM KF in this situation, mainly due to the torso rotating around the hip joint during walking, and the LIPM KF does not take this into account. It reveals the limitation of over simplified models in predicting complex dynamic behaviors.

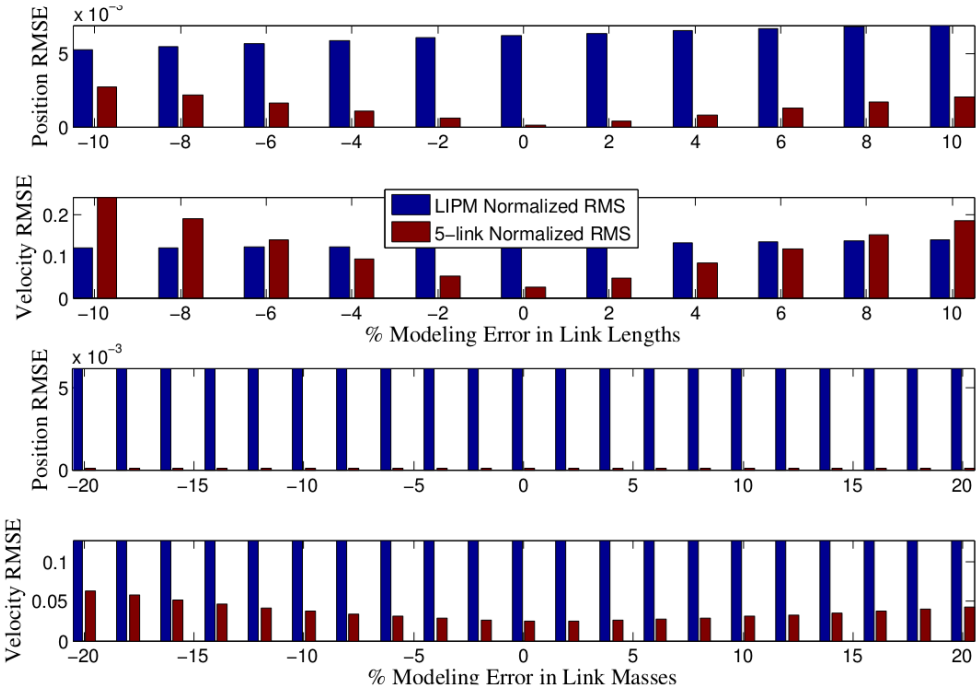


Figure 2.7: CoM position and velocity RMSE vs. modeling error in link lengths and masses

The performance of the filters depends both on the choice of noise covariances and modeling error. To test the robustness of the filters, we introduce modeling error in the

simulation and tune the noise covariance to have the best performance. The results are shown in Fig. 2.7. For the same amount of modeling error (±10%) on link lengths and (±20%) on link masses, the LIPM KF prediction on the CoM position and velocity is less influenced by the modeling error, because its dynamic model is simpler and less dependent on the model parameters. However, this does not mean the LIPM KF is more accurate than the Five-link Planar KF given the modeling error. In fact, the Five-link Planar model has smaller Root Mean Square Error (RMSE) in the given modeling error range.

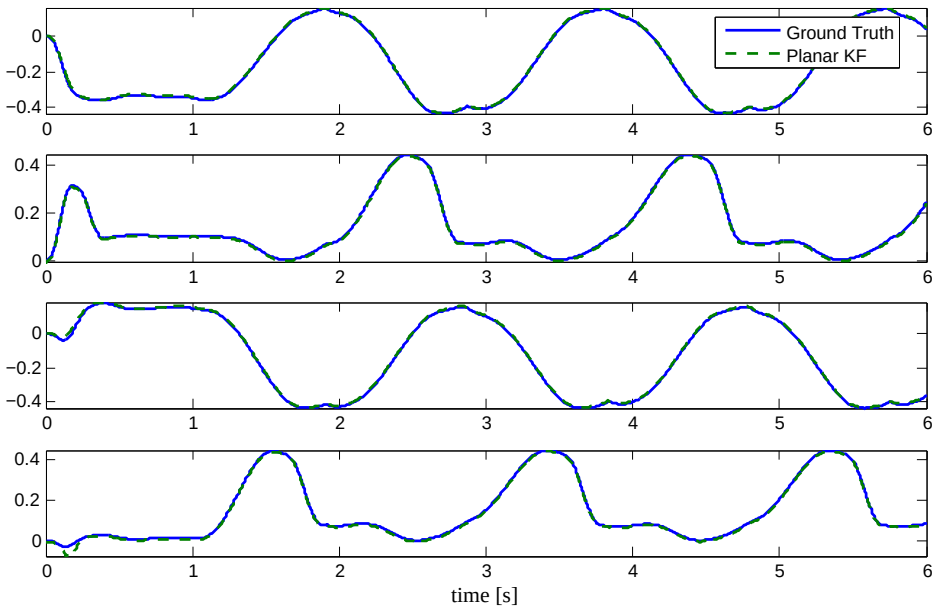


Figure 2.8: Joint positions from Five-link Planar KF and ground truth. From top to bottom: left hip, left knee, right hip, right knee

Besides above comparisons of both Kalman filters, we also show the estimated internal joint positions and velocities by the Five-link Planar KF in Fig. 2.8 and Fig. 2.9. It appears from these figures that the modeling error can not be eliminated without being explicitly addressed in the filter.

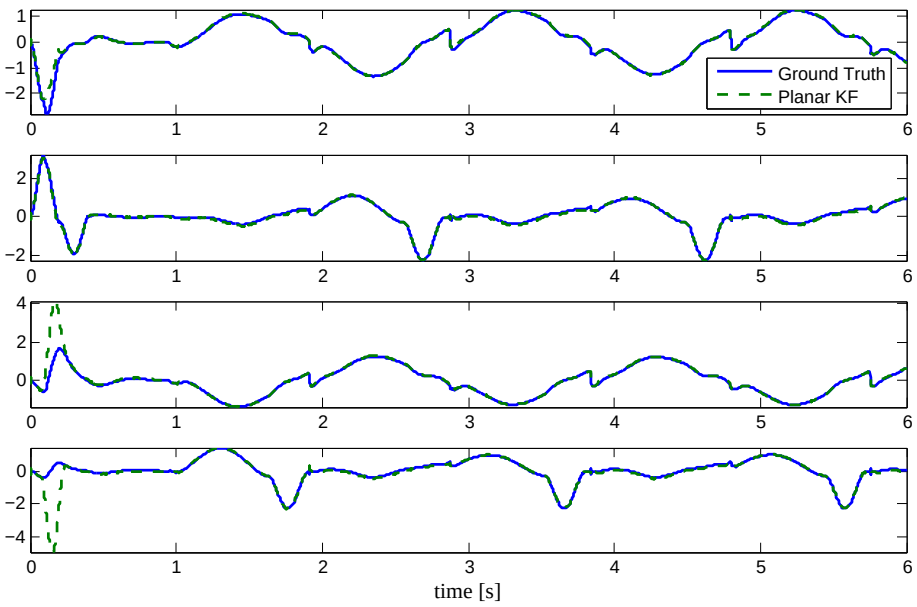


Figure 2.9: Joint velocities from Five-link Planar KF and ground truth. From top to bottom: left hip, left knee, right hip, right knee

2.5 Experiment on robot data

Both filters are tested using robot sensor data. The data is collected from Sarcos Humanoid robot trying to walk in place, while all the upper body joints are servoed to fixed positions. Due to asymmetry and slipping, the robot is drifting forward and slightly towards left. Since both Kalman filters are planar and the robot motion is in 3D, we assume the motion in the sagittal and coronal plane are decoupled, and simply use the sensor data in the sagittal plane. This is a reasonable assumption for forward walking. The robot has real planar feet rather than point feet, and we treat them as point feet located right at the ankle joint, by mapping the measured force/torque under each foot to the location of the ankle joints. Fig. 2.10 shows the estimated CoM position from both Kalman filters, and Fig. 2.11 shows the estimated velocity of CoM. The velocity is slightly delayed for the

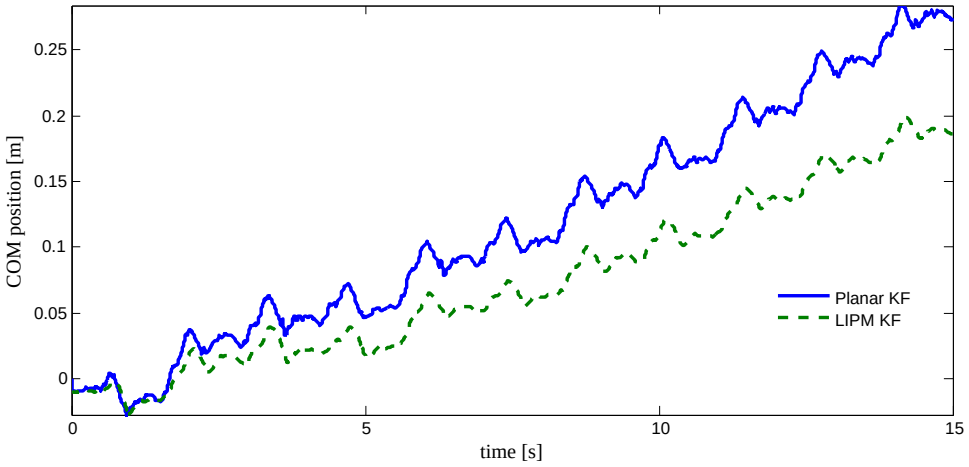


Figure 2.10: Robot CoM horizontal position, Five-link Planar KF and LIMP KF

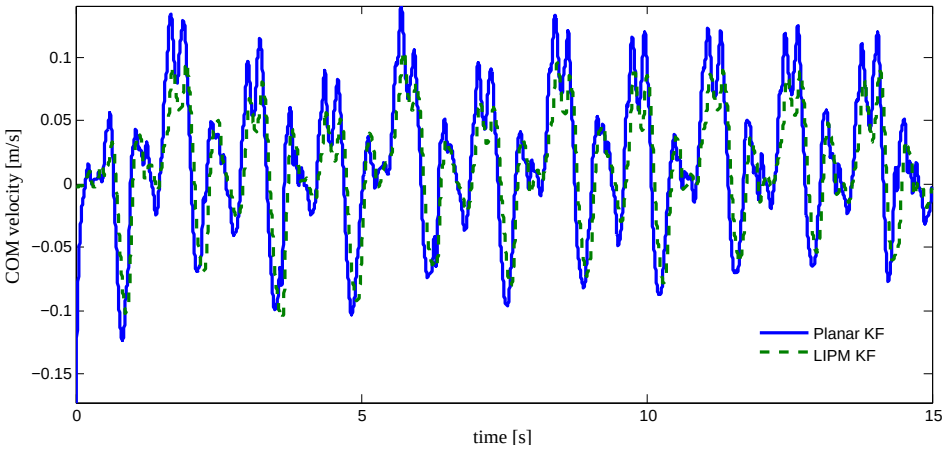


Figure 2.11: Robot CoM horizontal velocity, Five-link Planar KF and LIMP KF

LIPM KF.

2.6 Discussion

This chapter has compared two ways of designing Kalman filter for a walking system. Several issues will be addressed in future research.

First, both the LIPM KF and the Five-link Planar KF are abstractions of the robot. In reality, a 3D dynamic model is desirable. It is straightforward to extend the LIPM model to 3D because each direction is still decoupled. We can deal with walking without turning by using two uncoupled planar filters in the sagittal and coronal plane. For more complex behaviors, it may be better to use a true 3D model with full body dynamics.

In the Five-link Planar KF, the force sensor measurements under each foot are treated as control inputs to the process and measurement model. The advantage is we avoid multiple model filtering for the filter process model. The obvious drawback is that the force measurement is noisy. A subtle issue is information loss, because the reaction forces reflect system states. If the reaction forces are treated as constraints, they can be explicitly represented as a function of robot state [61][62], and incorporated into the measurement model. This approach requires the correct constraint to be imposed, which is another issue we want to investigate. The Five-link Planar KF can be extended to uneven ground by constraining the standing foot velocity to zero rather than a known ground level. The Five-link Planar KF assumes the non-slipping condition is satisfied by the standing foot, and this assumption is often violated by our Sarcos robot. Therefore it is crucial to detect slipping and perform state estimation under slipping conditions. In [63] a slip observer was proposed to detect skids that would occur while walking on unexpected slippery floor. Recently, a slip detection algorithm using boosted innovation of an UKF was introduced in [64].

2.7 Conclusion

This chapter presented two ways of designing Kalman filters for a walking humanoid robot system, and compared their performances in terms of estimating the CoM position and velocity. Both filters perform similarly in terms of predicting relative CoM position, and the Five-link Planar KF has better performance on predicting the CoM velocity. The advantage of using the LIPM KF is that it is simple to design and implement, and can be easily generalized to other humanoid walking systems. On the contrary, the Five-link Planar KF is very specific to the system since the entire rigid body dynamics is taken into account. However, being able to predict the motion of each joint makes the Five-link Planar KF more versatile and capable of predicting more general behaviors other than walking.

3

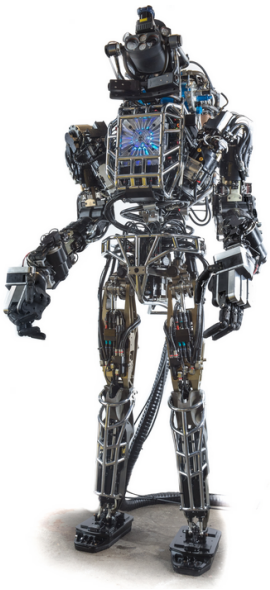
Decoupled State Estimation

Most material in this chapter has been published in [1].

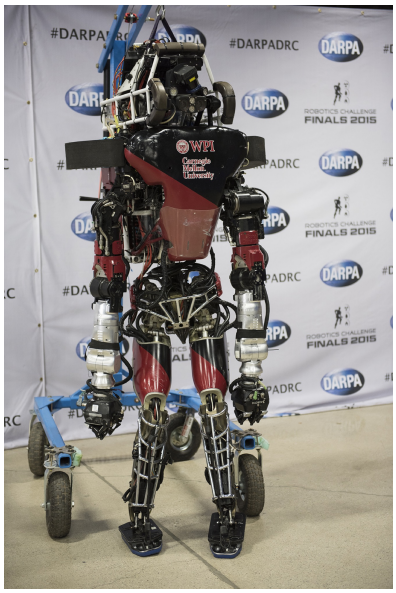
This chapter addresses the issue of estimating the full state of the Atlas robot. We use a forward kinematic model to estimate the states of the floating base, and full body dynamics to estimate joint velocities. The main idea is to decouple the full body state vector into several independent state vectors. Some decoupled state vectors can be estimated very efficiently with a steady state Kalman filter. Decoupling also speeds up numerical linearization of the dynamic model. In a steady state Kalman filter, state covariance is computed only once during initialization. We demonstrate that these state estimators are capable of handling walking on flat ground and on rough terrain.

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.

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. Decoupling speeds up numerical linearization of the



(a) First generation has 6 DoF hydraulic arms and 28 joints.



(b) Second generation has 7 DoF arms (the lower arms are electrical) and 30 joints.

Figure 3.1: Both generation of the Atlas humanoid robot constructed by Boston Dynamics. The first generation was used in the DRC Trials, and second generation was used during the DRC Finals. Most data and results shown in this chapter was from the first generation unless explicitly pointed out.

dynamic model. In a steady state Kalman filter, state covariance is computed only once during initialization. We trade partial information loss for a reduction in computational cost.

This chapter is organized as follows. In Section 3.1, we will review the formulation of the Extended Kalman Filter (EKF). In Section 3.2, we formulate the full body state estimation problem as several decoupled state estimation problems. Section 3.3 describes the implementation of each state estimator. In Section 3.4, we show simulation results with the Atlas robot (see Fig. 3.1) walking using the Gazebo simulator, and actual robot results. Section 3.5 discusses future work and the last section concludes this chapter.

3.1 The Extended Kalman Filter

The 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 EKF[18] and later the Unscented Kalman Filter (UKF)[24] were invented. The EKF linearizes the nonlinear dynamics at the current estimate, and propagates the belief or information state covariance the same way as the Kalman filter. The UKF samples around the current 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

$$x_k^- = f(x_{k-1}^+, u_{k-1}) \tag{3.1}$$

$$P_k^- = F_k P_{k-1}^+ F_k^T + Q_k \tag{3.2}$$

Update step

$$y_k = h(x_k^-) \tag{3.3}$$

$$\Delta y_k = z_k - y_k \tag{3.4}$$

$$S_k = H_k P_k^- H_k^T + R_k \tag{3.5}$$

$$K_k = P_k^- H_k^T S_k^{-1} \tag{3.6}$$

$$\Delta x_k = K_k \Delta y_k \tag{3.7}$$

$$x_k^+ = x_k^- + \Delta x_k \tag{3.8}$$

$$P_k^+ = (I - K_k H_k) P_k^- \tag{3.9}$$

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. Eq. (3.1) and Eq. (3.3) are the process and measurement equation, respectively. 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 error 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. (3.6). If F_k and H_k are computed by numeric differentiation at each recursion, they are also computationally expensive. In the linear Kalman filter 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 or H_k. If we assume they are both constant over a certain period of time or for some states, we could formulate the recursive EKF problem as a steady state Kalman filter problem.

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

by solving the Discrete time Algebraic Riccati Equation (DARE)

$$FPF^T - P - FPH^T(R + HPH^T)^{-1}HPF^T + Q = 0 \tag{3.10}$$

and the steady state Kalman Gain K is given by

$$K = PH^T(R + HPH^T)^{-1} \tag{3.11}$$

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

Prediction step

$$x_k^- = f(x_{k-1}^+, u_{k-1}) \tag{3.12}$$

Update step

$$\Delta y_k = z_k - h(x_k^-) \tag{3.13}$$

$$x_k^+ = x_k^- + K\Delta y_k \tag{3.14}$$

3.2 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

$$\begin{aligned} M(q)\ddot{q} + h(q, \dot{q}) &= S\tau + J_c^T(q)f \\ \dot{J}_c(q, \dot{q})\dot{q} + J_c(q)\ddot{q} &= \ddot{c} \end{aligned} \tag{3.15}$$

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 world 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 computationally expensive 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 Kalman filters fast enough within one control step. Choosing appropriate filter parameters has also proved to be difficult in this setting, since the Kalman gain 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 3.5.

3.2.1 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

$$x_k^- = \begin{bmatrix} p_{x,k}^- \\ p_{q,k}^- \\ p_{v,k}^- \\ p_{\omega,k}^- \\ b_{a,k}^- \end{bmatrix} = \begin{bmatrix} p_{x,k-1}^+ + p_{v,k-1}^+ \Delta t \\ p_{q,k-1}^+ + \frac{1}{2} G(p_{\omega,k-1}^+) p_{q,k-1}^+ \Delta t \\ p_{v,k-1}^+ + [R^T(p_{q,k-1}^+)(\tilde{a}_k - b_{a,k-1}^+) - g]\Delta t \\ p_{\omega,k-1}^+ + (\tilde{\omega}_k - \tilde{\omega}_{k-1}) \\ b_{a,k-1}^+ \end{bmatrix} \tag{3.16}$$

where Δt is the time step, $\tilde{a}$ and $\tilde{\omega}$ are the measured IMU 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 the derivative of its elements. The fourth equation in Eq. (3.16) 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 $\alpha = [\alpha_x, \alpha_y, \alpha_z]$ representing a small rotation, its incremental rotation matrix is given by

$$\Lambda(\alpha) := \exp(\alpha^\times) = \mathbb{I} + (\alpha^\times) \sin \|\alpha\| + (\alpha^\times)^2 (1 - \cos \|\alpha\|) \tag{3.17}$$

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

$$F_k = \begin{bmatrix} \mathbb{I} & 0 & \Delta t\mathbb{I} & 0 & 0 \\ 0 & \Lambda(\Delta t p^+_{\omega,k-1}) & 0 & 0 & 0 \\ 0 & \Delta t \left(R^T[\tilde{a} - b^+_{a,k-1}]\right)^\times & \mathbb{I} & 0 & -\Delta t R^T \\ 0 & 0 & 0 & \mathbb{I} & 0 \\ 0 & 0 & 0 & 0 & \mathbb{I} \end{bmatrix} \tag{3.18}$$

The predicted covariance estimate follows Eq. (3.2).

The update step is slightly more complicated. There is no sensor directly measuring the position and velocity of the base in world coordinates. 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, we assume the feet are not moving to obtain the following measurements

$$z_{k,DS} = \begin{bmatrix} c_{l,k} \\ c_{r,k} \\ \dot{c}_{l,k} \\ \dot{c}_{r,k} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} c_{l,k-i} \\ \frac{1}{N}\sum_{i=1}^{N} c_{r,k-i} \\ 0 \\ 0 \end{bmatrix} \tag{3.19}$$

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 $FK(\cdot)$,

$$y_{k,DS} = \begin{bmatrix} FK_{c_l}(q_k^-) \\ FK_{c_r}(q_k^-) \\ FK_{\dot{c}_l}(q_k^-, \dot{q}_k^-) \\ FK_{\dot{c}_r}(q_k^-, \dot{q}_k^-) \end{bmatrix} \tag{3.20}$$

In Eq. (3.20), 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. (3.20). 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. (3.19)(3.20) and H_k is modified to account for contact switching. Contact switching will be discussed in Section 3.3.

When we compute Eq. (3.7), the innovation multiplied by the Kalman gain is one dimension less than the state vector. We need to switch from a rotation vector to quaternion. Suppose α is a rotation vector, then its corresponding quaternion is given by

$$\xi(\alpha) = \begin{bmatrix} \cos \frac{1}{2} \parallel \alpha \parallel \\ \sin \frac{1}{2} \parallel \alpha \parallel \frac{\alpha}{\|\alpha\|} \end{bmatrix} \tag{3.21}$$

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

$$p_{q,k}^+ = \xi(\Delta\phi_k) p_{q,k}^- \tag{3.22}$$

where $\Delta\phi$ is the fourth to sixth components of Δx in Eq. (3.7).

3.2.2 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.

Joint Position Filter

The process dynamics and state transition matrix are

$$x_k^- = \theta_{J,k}^- = \theta_{J,k-1}^+ + \dot{\theta}_{J,k-1}^+ \Delta t \tag{3.23}$$

$$F_k = \mathbb{I} \tag{3.24}$$

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

$$y_k = \theta_{J,k}^- \tag{3.25}$$

$$H_k = \mathbb{I} \tag{3.26}$$

Joint Velocity Filter

The joint velocity filter uses the full body dynamics to estimate joint velocities. Define

$$\ddot{q} = [\ddot{x}_b^T, \ddot{\theta}_J^T]^T \tag{3.27}$$

$\ddot{x}_b = [\dot{p}_v^T, \dot{p}_\omega^T]^T$ is the base linear and angular acceleration in Cartesian coordinate. The process dynamics is derived from Eq. (6.9), assuming $\ddot{c} = 0$ and reorganizing it as

$$\begin{bmatrix} M_{x_b} & -J_c^T & M_{\theta_J} \\ J_{c,x_b} & 0 & J_{c,\theta_J} \end{bmatrix} \begin{bmatrix} \ddot{x}_b \\ f \\ \ddot{\theta}_J \end{bmatrix} = \begin{bmatrix} S\tau - h(q,\dot{q}) \\ -\dot{J}_c\dot{q} \end{bmatrix} \tag{3.28}$$

Rewrite Eq. (3.28) as

$$A_1 \begin{bmatrix} \ddot{x}_b \\ f \end{bmatrix} + A_2\ddot{\theta}_J = b \tag{3.29}$$

where

$$A_1 = \begin{bmatrix} M_{x_b} & -J_c^T \\ J_{c,x_b} & 0 \end{bmatrix} \tag{3.30}$$

$$A_2 = \begin{bmatrix} M_{\theta_J} \\ J_{c,\theta_J} \end{bmatrix} \tag{3.31}$$

$$b = \begin{bmatrix} S\tau - h(q,\dot{q}) \\ -\dot{J}_c\dot{q} \end{bmatrix} \tag{3.32}$$

In [62] and [65], 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. (3.29) for $\ddot{\theta}_J$, we perform a QR decomposition on A_1

$$A_1 = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1 \tag{3.33}$$

where the matrix Q is orthogonal. Multiply Eq. (3.29) by Q_2^T

$$Q_2^T A_2 \ddot{\theta}_J = Q_2^T b \tag{3.34}$$

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

$$x_k^- = \dot{\theta}_{J,k}^- = \dot{\theta}_{J,k-1}^+ + (Q_2^T A_2)^{-1} Q_2^T b \Delta t \tag{3.35}$$

The state transition matrix is the linearization of Eq. (3.35), given by

$$F_k = \mathbb{I} + \Delta t (Q_2^T A_2)^{-1} Q_2^T \left. \frac{\partial b}{\partial \dot{\theta}_J} \right|_{\dot{\theta}_{J,k-1}^+} \tag{3.36}$$

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

$$y_k = x_k^- = \dot{\theta}_{J,k}^- \tag{3.37}$$

$$H_k = \mathbb{I} \tag{3.38}$$

3.3 Implementation

We implement all the state estimators mentioned in Section 3.2 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.

3.3.1 Base State Estimator

The base state estimator is implemented as a recursive EKF. One filter step follows Eq. (3.1)-(3.9) and Eq. (3.16)-(3.22). The quaternion states need some additional operations. We normalize the quaternion after Eq. (3.16) 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 the latter in simulation.

3.3.2 Joint Position Filter

The joint position filter is implemented with a steady state EKF, since both the state transition (Eq. (3.24)) and observation (Eq. (3.26)) matrix are constant. During the filter initialization stage, we pre-compute the steady state Kalman Gains (Eq. (3.11)) by solving the corresponding DARE with constant F, H, Q and R. Then the filter prediction step involves Eq. (3.12), and the update step is based on Eq. (3.13) and (3.14). Essentially the covariance matrix computation is not needed.

3.3.3 Joint Velocity Filter

The joint velocity filter is expensive to implement recursively at each time step due to the numerical linearization in Eq. (3.36). Instead, we compute Eq. (3.36) 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. (3.35).

3.3.4 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 minimum impact on the other filters. This makes tuning individual filters much easier compared to tuning one large filter with all the correlated states.

3.3.5 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 [66]. The planner uses the estimated base position at a much lower frequency to build a map, see[67] for details.

3.4 Results

3.4.1 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.

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. 3.2 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. 3.3 shows spikes in the estimated vertical velocity due to a large impact at foot touch-down. Fig. 3.6 shows some joint velocities of the right leg. Since there is no ground truth joint velocity information available in the Gazebo simulator, we compare our estimation to the sensor values. Joint positions are very consistent with measurements so we did not show any plots.

3.4.2 Hardware Results

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

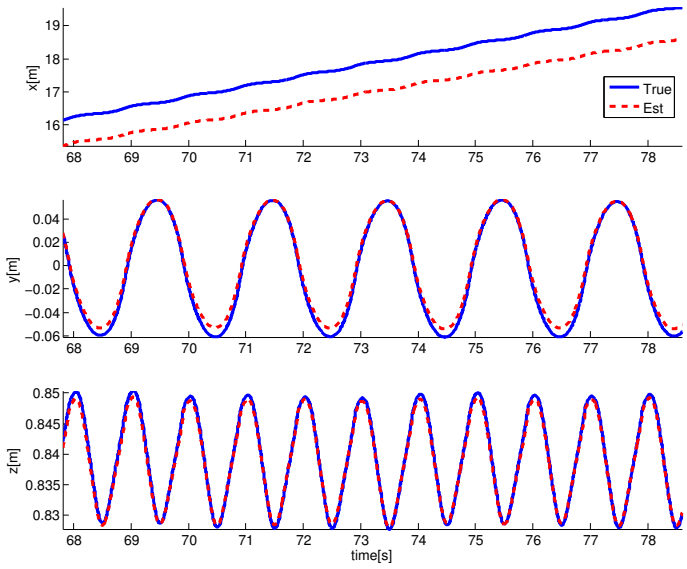


Figure 3.2: Simulation data: ground truth and estimated base position

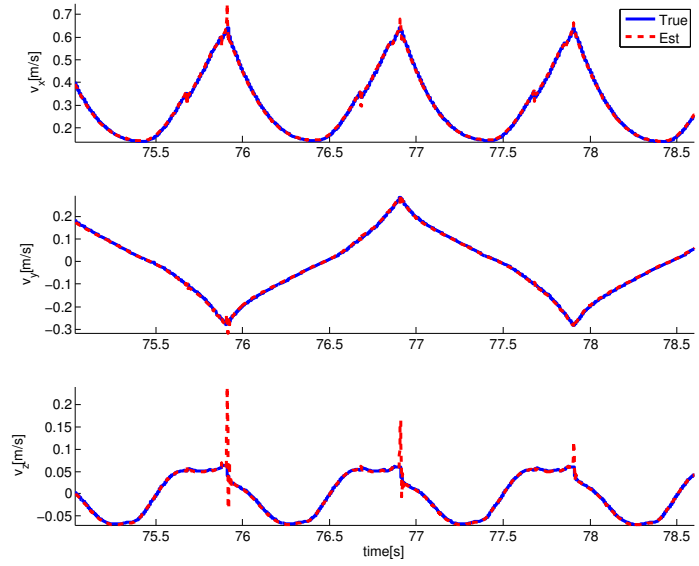


Figure 3.3: Simulation data: ground truth and estimated base velocity

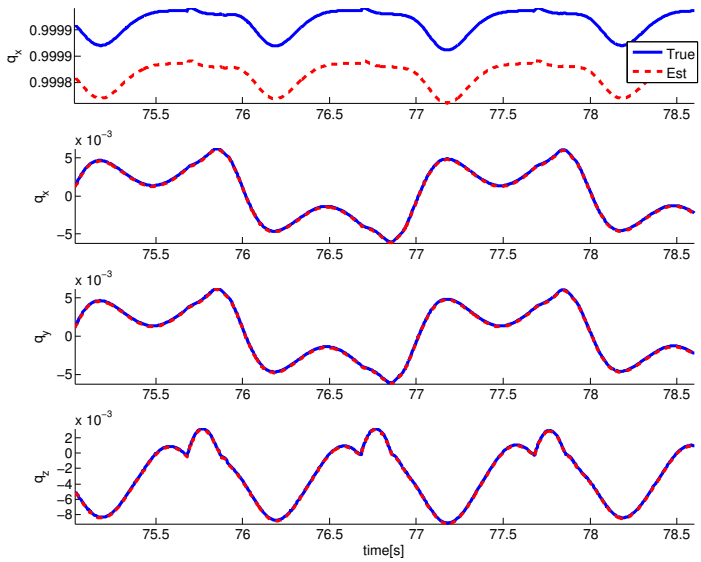


Figure 3.4: Simulation data: ground truth and estimated base orientation for the base orientation quaternion.

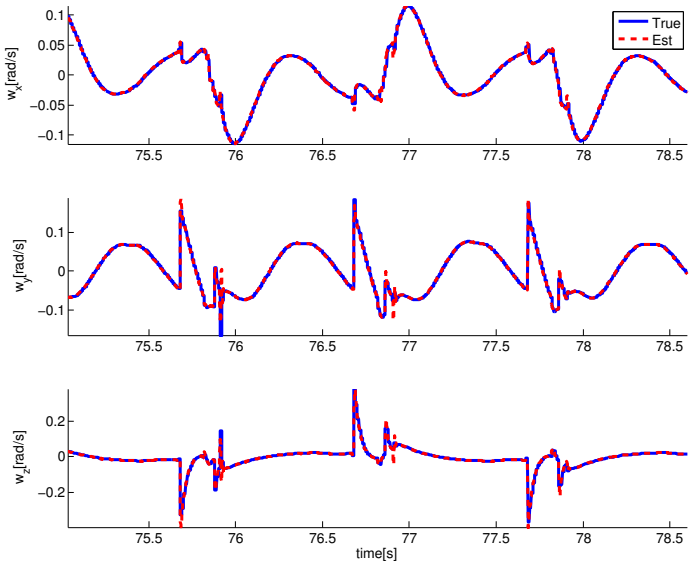


Figure 3.5: Ground truth and estimated base angular velocity

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. 3.7 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. 3.8 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 Kalman filtered states are less noisy than the measurement. There is no delay

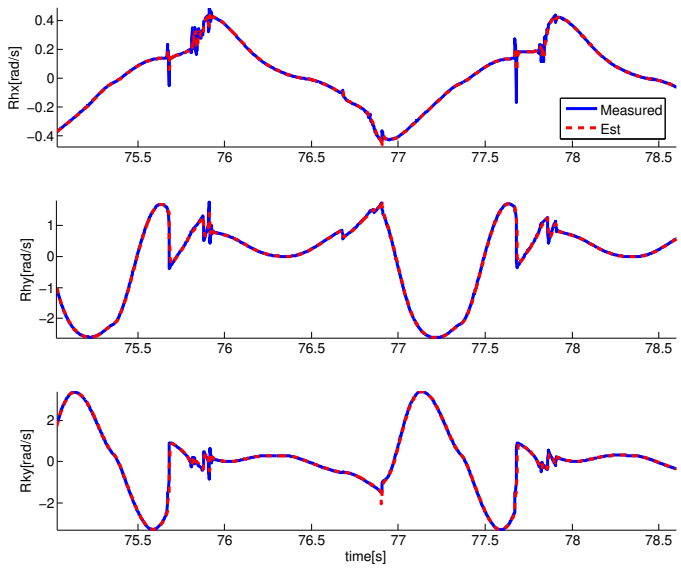


Figure 3.6: Measured and estimated joint velocities, from top to bottom are the right hip roll, pitch and knee pitch angle velocities

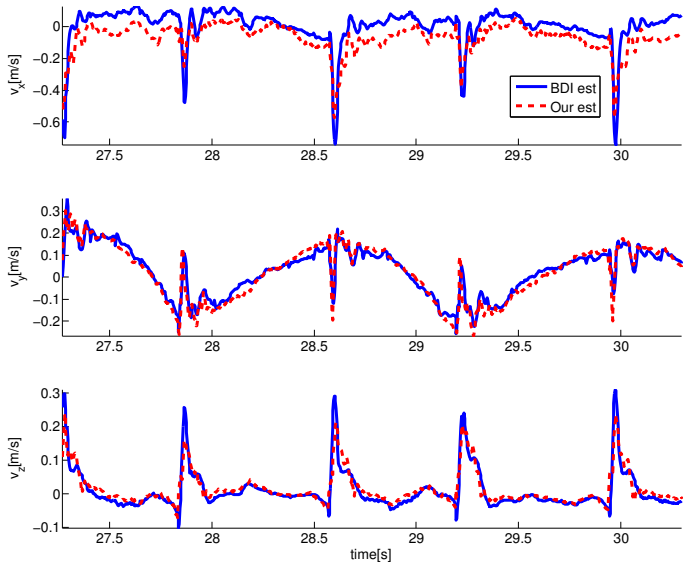


Figure 3.7: Robot data: base velocities from Boston Dynamics state estimator and our base state estimator

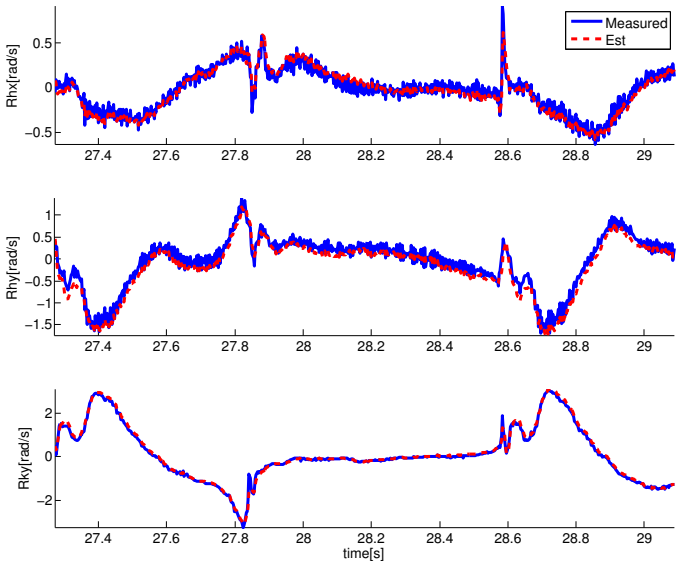


Figure 3.8: Robot data: measured joint velocities vs. estimated joint velocities from joint velocity filter. From top to bottom are the right hip roll, pitch and knee pitch angular velocities

between the Kalman 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.

3.4.3 Results After Hardware Update

Our Atlas robot received a hardware update several months before the DARPA Robotics Challenge (DRC) Finals. Boston Dynamics re-sized actuators in the hip, knee and back to address strength, backlash and compliance issues. We were able to benchmark the odometry performance of the base state estimator in an outdoor environment in preparation

for the DRC Finals. We controlled the Atlas robot to walk forward continuously in a parking lot with an average step length of 0.4m. The experiment was carried out several times with different walking speeds and step lengths to check repeatability. It took the Atlas robot about 25 steps to traverse a course of roughly 10m. We compared the travel distance estimated by the base state estimator and measured using a tape measure. The position error for all the experiments was below 2%, on average each step had an error below 1cm. A very similar state estimator performance was report by Team MIT in [68].

The reliability and accuracy of our state estimator was tested in the DRC Finals. During both days of the competition, our Atlas robot successfully traversed the entire concrete course (about 10m). The course setup is shown in Fig. 3.9.

The Atlas was registered to the environment using LIDAR and stereo vision.

Door traversal can be broken down into four sub-tasks; door detection, walk to the door, door opening, and walk through the door. Door detection locates the door and finds its normal. Wrist mounted cameras assisted in finding the door handle, maintaining contact, and providing information to the human supervisor. The handle is turned, and in the DRC Finals with a push door, the other arm is used to push the door open. We had the robot walk sideways through the doorway to maximize clearance, taking into account the large sideways sway of the robot as it walked.

For the manipulation tasks, the operator looks at the 3D point cloud generated by the LIDAR, and manually commands the robot to go to the vicinity of the object being manipulated. Then the robot walks towards the target blindly following the foot steps generated by the planner.

For the terrain task, the robot has to traverse the terrain composed of flat and tilted cinder blocks. For the stair task, the robot needs to go up an industrial ladder with 4 stair steps. 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. The operator's input of a

Figure 3.9: Picture taken from the second run of team WPI-CMU in the DRC Finals, showing part of the course setup. At the bottom of the picture is the rough terrain, in the middle is the manipulation tasks setup, and on the top right corner is the Atlas robot passing through the door.

desired goal is then snapped to a safe location. It then classifies each cinder block based on its surface normal. A discrete search is perform to generated 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 DRC Finals.

The robot walks blindly for about 6-10 steps for the terrain task, and 8 footsteps (4 stair steps) for the stair task following the foot step plans.

3.5 Discussion

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 cannot estimate the entire 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. (6.9). 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 [40]. We have shown in the

chapter that the decoupled state estimator reduces the noise level on the joint velocities of the Atlas.

Our robot has a LIDAR and stereo cameras that can be integrated with the base state estimator to provide odometry information, such as described in [68][69]. The visual odometry algorithm usually computes a relative pose measurements, we can augment the filter state with the pose at previous measurement instance (stochastic cloning [70][71]). Vision based localization algorithms usually rely on feature detection and matching to determine motion change. The vision system will improve odometry if it provides more accurate information than the interoceptive sensors.

One challenge is how to do it robustly. Trusting the vision system too much makes the algorithm vulnerable to outliers, such as feature mismatching, tracking moving features or motion blur during walking, etc.. We can robustify the filter by rejecting measurement with a self-reported error bigger than some threshold, or if the Chi-squared test on the renovation is bigger than some threshold.

Another challenge is how to handle delayed measurements from the vision system, as the vision algorithm takes time (30ms - 100ms in our case). We save all the states and measurements in a buffer, and rerun the filter once the delayed measurement is available [72].

Given our odometry was sufficiently accurate for the DRC tasks, we chose to be conservative and did not use the vision system directly for odometry.

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.

3.6 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.

The base state estimator provided accurate odometry information. The enabled our robot to traverse the entire course without using exteroceptive sensing during locomotion in the DRC Finals.

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.

4

Dynamic State Estimation using Quadratic Programming

Most material in this chapter has been published in [73].

This chapter addresses the issue of estimating the generalized velocity using joint and end effector force measurements. It is formulated as an optimization problem and solved with Quadratic Program (QP). This formulation provides two main advantages over a non-linear 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 optimization vector and includes it in the cost function. The proposed QP state estimator is tested on a Boston Dynamics Atlas humanoid robot. 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 heavily geared electric motors. They are usually high impedance and stiff, which allow 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 disturbances in the form of force disturbances, such as a sudden push, or a geometric disturbances such as rough terrain to walk through. 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. 3.1, 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, however 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 will propagate 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. (4.1) for the process dynamics, but joint force or torque are acceleration level quantities that can not be expressed as part of the state x in the state equation.

$$\dot{x} = f(x, u) \tag{4.1}$$

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 [40], or be eliminated by projecting the state onto the null space of the constraint Jacobian using orthogonal decomposition [1][62][65].

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 state estimator state. A Kalman filter can handle a linear inequality constraint through estimation projection [52], which is essentially solving a QP on the a posteriori estimate.

A Moving Horizon Estimator (MHE) is a more general formulation than Kalman filter [74][52]. A linear MHE is formulated as a QP and it considers linear inequality constraints on the state naturally [16], it is equivalent to a Kalman filter if the inequality constraints are removed. A MHE still requires a state space formulation for the system equation, as in Eq. (4.1). The proposed estimator in this chapter is inspired by the MHE.

The purpose of this chapter is to address the above mentioned issues on state estimation of 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 an linear equality constraint, and various limits are treated as linear inequality constraints. The optimization variables are elaborated in Section 4.2.

This chapter is organized as follows. In Section 4.1, we will describe the QP formulation in general. In Section 4.2, we formulate the state estimation problem using QP and describe the objective function and constraints in detail. In Section 4.3, we show state estimation results of the Atlas robot walking using the Gazebo simulator, and in the real world. Section 4.4 is the discussion and the last section concludes this chapter.

4.1 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. (4.2)(4.3) and (4.4).

$$\underset{\mathcal{X}}{\text{minimize}} \quad \frac{1}{2}\mathcal{X}^T G \mathcal{X} + g^T \mathcal{X} \tag{4.2}$$

$$\text{subject to} \quad C_e \mathcal{X} + c_e = 0 \tag{4.3}$$

$$C_i \mathcal{X} + c_i \geq 0 \tag{4.4}$$

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

$$A = \begin{bmatrix} \alpha_0 A_0 \\ \alpha_1 A_1 \\ \vdots \\ \alpha_n A_n \end{bmatrix}, b = \begin{bmatrix} \alpha_0 b_0 \\ \alpha_1 b_1 \\ \vdots \\ \alpha_n b_n \end{bmatrix} \tag{4.5}$$

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 the Goldfarb-Idnani method [75]. One disadvantage of the Goldfarb-Idnani method is it requires a positive definite matrix G, thus the row rank of the 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 CPU chip type "i7-3452".

4.2 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

$$M(q)\ddot{q} + h(q, \dot{q}) = S\tau + J_c^T(q)f \tag{4.6}$$

$$\dot{J}_c(q, \dot{q})\dot{q} + J_c(q)\ddot{q} = \ddot{c} \tag{4.7}$$

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, 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 in the world frame. c is the contact point's position and orientation in Cartesian coordinates.

4.2.1 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:

$$\underset{\mathcal{X}_k}{\text{minimize}}\ w_k^T Q^{-1} w_k + v_k^T R^{-1} v_k \tag{4.8}$$

$\mathcal{X}$ is the optimization variable vector, its definition is given later in this section. The weight 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.

Modeling Error

Ideally, the rigid body dynamics equation describing the relationship between applied force and acceleration is given by Eq. (4.6). In a real humanoid robot system, due to modelling error, Eq. (4.6) 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

$$M(q)\ddot{q} - S^T\tau - J^T f + h(q,\dot{q}) = w \tag{4.9}$$

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

$$\ddot{q}_k = \frac{1}{\Delta t}(\dot{q}_{k+1} - \dot{q}_k) \tag{4.10}$$

Eq. (4.9) can be re-arranged using Eq. (4.10) as

$$\begin{bmatrix} \frac{M}{\Delta t} & -I & -S^T & -J^T \end{bmatrix} \begin{bmatrix} \dot{q}_{k+1} \\ w_k \\ \tau_k \\ f_k \end{bmatrix} = -h + \frac{M}{\Delta t}\dot{q}_k \tag{4.11}$$

We use Eq. (4.11) as an equality constraint in the QP, and define the vector of optimization variables as

$$\begin{aligned}\mathcal{X}_k &= \begin{bmatrix} \dot{q}_{k+1} & w_k & \tau_k & f_k \end{bmatrix}^T \\ &= \begin{bmatrix} [p_v & p_\omega & \dot{\theta}_J]_{k+1} & w_k & \tau_k & f_k \end{bmatrix}^T \end{aligned} \tag{4.12}$$

The second line is useful in terms of expressing the measurement error discussed in the next section 4.2.1.

Using the notation in Eq. (4.5), the cost associated with the modeling error can be written as

$$A_w = \begin{bmatrix} 0 & I & 0 & 0 \end{bmatrix} \tag{4.13}$$

$$b_w = 0 \tag{4.14}$$

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. (4.5). Each term in the measurement error is explained in detail below

Joint Velocity The joint velocity error is computed using

$$A_{\dot{\theta}_J} = \begin{bmatrix} [0 & 0 & I] & 0 & 0 & 0 \end{bmatrix} \tag{4.15}$$

$$b_{\dot{\theta}_J} = \dot{\theta}_{J,m} \tag{4.16}$$

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

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

$$A_{p_\omega} = \begin{bmatrix} [0 & I & 0] & 0 & 0 & 0 \end{bmatrix} \tag{4.17}$$

$$b_{p_\omega} = p_{\omega,m} \tag{4.18}$$

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

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

$$p_{v,k+1} = a_m \Delta t + p_{v,k} \tag{4.19}$$

The "measured" pelvis velocity is given by

$$A_{p_v} = \begin{bmatrix} [I & 0 & 0] & 0 & 0 & 0 \end{bmatrix} \tag{4.20}$$

$$b_{p_v} = p_{v,k} + a_m \Delta t \tag{4.21}$$

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

Table 4.1: 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

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

$$A_\tau = \begin{bmatrix} 0 & 0 & I & 0 \end{bmatrix} \tag{4.22}$$

$$b_\tau = \tau_m \tag{4.23}$$

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

$$A_{v_{side}} = \begin{bmatrix} J_{side} & 0 & 0 & 0 \end{bmatrix} \tag{4.24}$$

$$b_{v_{side}} = 0 \tag{4.25}$$

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. (4.7) given $\ddot{c} = 0$, and it is a stronger constraint than Eq. (4.7). We find that using a soft penalty with a relative large weight is faster to solve than using the equality constraints.

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).

$$A_{f_{side}} = \begin{bmatrix} 0 & 0 & 0 & R_{side} \end{bmatrix} \tag{4.26}$$

$$b_{f_{side}} = \begin{bmatrix} 0 & 0 & f_{zm} & \tau_{xm} & \tau_{ym} & 0 \end{bmatrix}^T \tag{4.27}$$

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. 4.1 shows the weights used in the cost function.

4.2.2 Constraints

In addition to the equality constraints given by Eq. (4.11) in Section 4.2.1, 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. (4.6) 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

$$\theta_{lower} \leq \theta_J + \Delta t \dot{\theta}_J \leq \theta_{upper} \tag{4.28}$$

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

$$f_z \geq 0 \tag{4.29}$$

which means the force can only push but not pull.

4.3 Results

We test our QP state estimator both in simulation and on real hardware. For the walking controller and state estimator, we servo the robot upper body joints to a fixed desired position, thus we could model the entire upper body as a single rigid body in the dynamic equation to simplify computation. The dynamic model 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 the QuadProg++ library.

4.3.1 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 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 [66]. The controller for static walking is based on simultaneously solving inverse dynamics and inverse kinematics [76]. 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. 4.1 to 4.4, 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. 4.1 shows the base velocities in ground truth, estimated by the 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. 4.2 plots the estimated joint

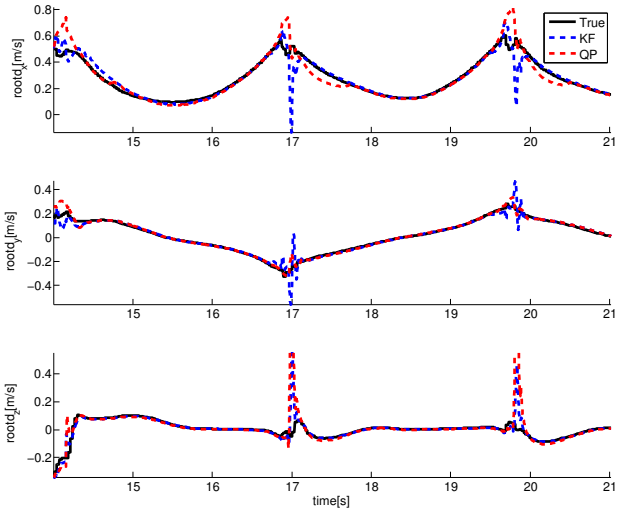


Figure 4.1: Simulation data: base velocity from ground truth, decoupled EKF and QP state estimator. Due to heel-strike and toe-off, both state estimators have errors during contact phase transition

velocities for all the right leg joints. Both the decoupled KF and the QP state estimators obtain very similar results. We compare the measured and estimated vertical contact force in Fig. 4.4. The QP estimated contact forces track measurements reasonably well during the single support phases. At heel-strike, the touch-down foot contacted the ground and was bounced back, resulting in a large force spike which is not well tracked by the QP state estimator, because the equality constraint Eq.(4.9) 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.

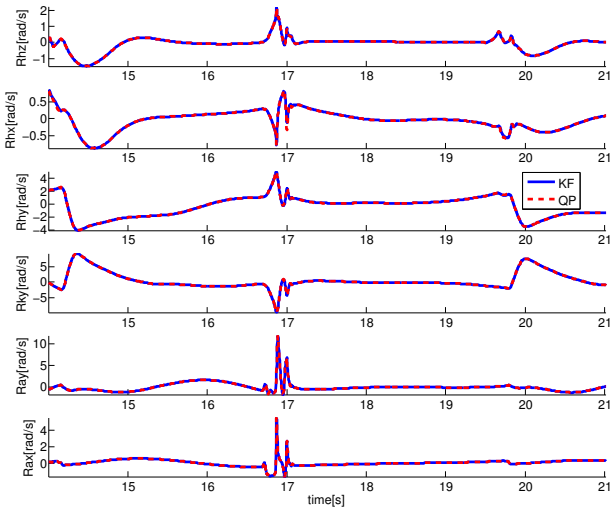


Figure 4.2: Simulation data: estimated joint velocities using decoupled KF and QP state estimator. From top to bottom are the right hip yaw, roll, pitch, knee pitch, ankle pitch and roll angular velocities

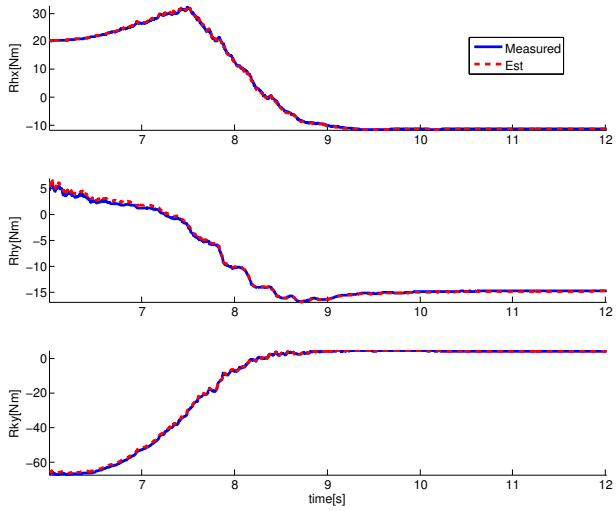


Figure 4.3: Simulation data: measured and estimated joint torques, from top to bottom are the right hip roll, pitch and knee pitch torques

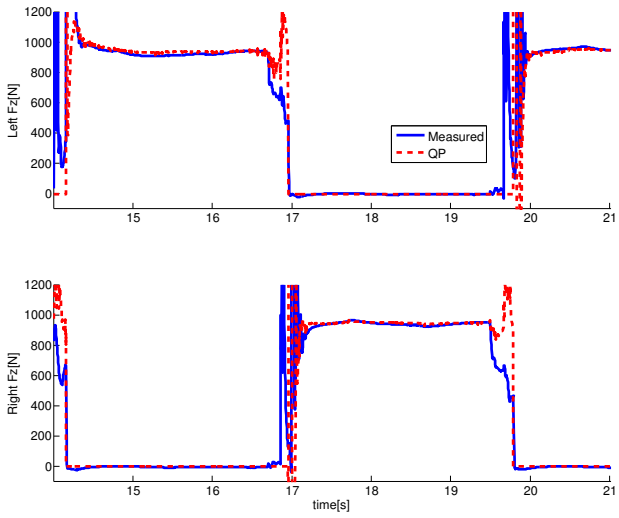


Figure 4.4: Simulation data: measured and estimated contact forces. From top to bottom are the left foot and right contact force in z direction respectively

4.3.2 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. 4.8. The controller and state estimator used during the experiment is discussed in detail in [76][1].

We show 12 seconds of data from Fig. 4.5 to Fig. 4.7, 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. 4.5 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. 4.6. The QP state estimator demonstrates good tracking performance on the joint velocities. Fig. 4.7 is the plot of the measured and estimated right

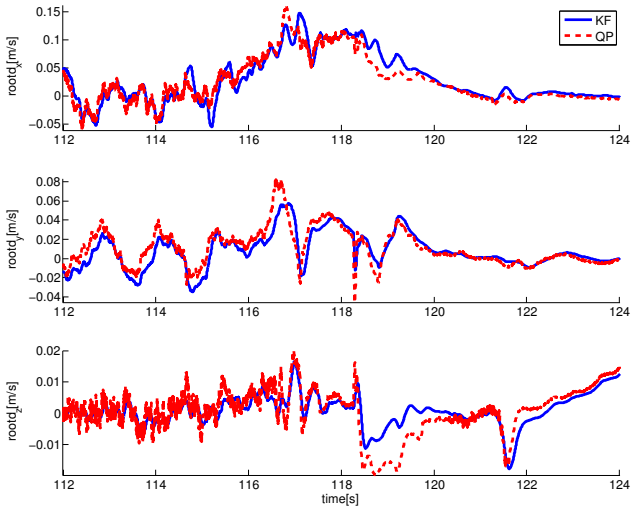


Figure 4.5: Robot data: decoupled Kalman filter [1] and QP state estimator estimated base velocity

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 during data processing.

4.4 Discussion

Compared to the recursive Kalman filter, the QP state estimator offers several advantages. It naturally handles linear constraints in the problem formulation. It is particularly useful for systems 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

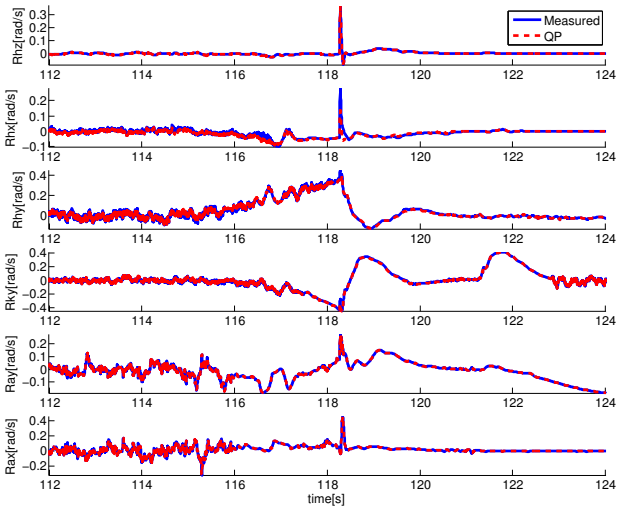


Figure 4.6: Robot data: measured and QP estimated joint velocities, from top to bottom are the right hip yaw, roll, pitch, knee pitch, ankle pitch and roll angular velocities

robot, the valve command is computed using measured joint torque, and filtering its noise helps stabilize the controller. The dynamics equation Eq. (4.6) 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 is represented by the process noise, which is assumed to be drawn from a zero-mean normal distribution and independent across time. In Section 4.3.2, 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.

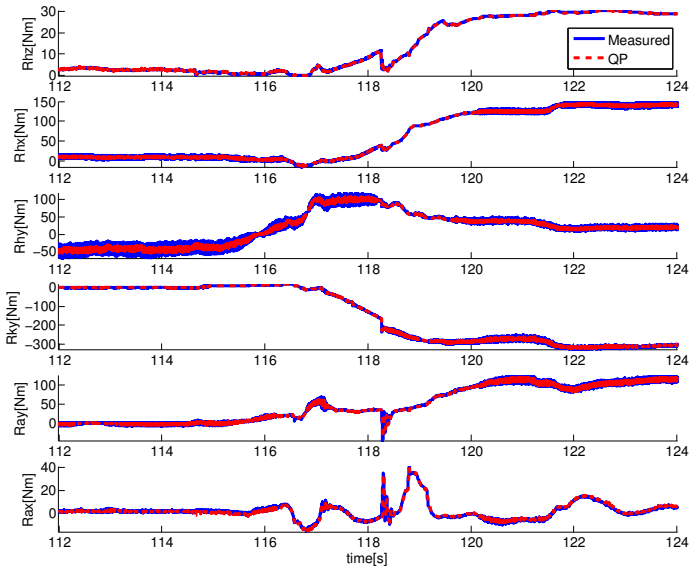


Figure 4.7: Robot data: measured and QP estimated joint torque, from top to bottom are the right hip yaw, roll, pitch and knee pitch, ankle pitch and pitch velocities

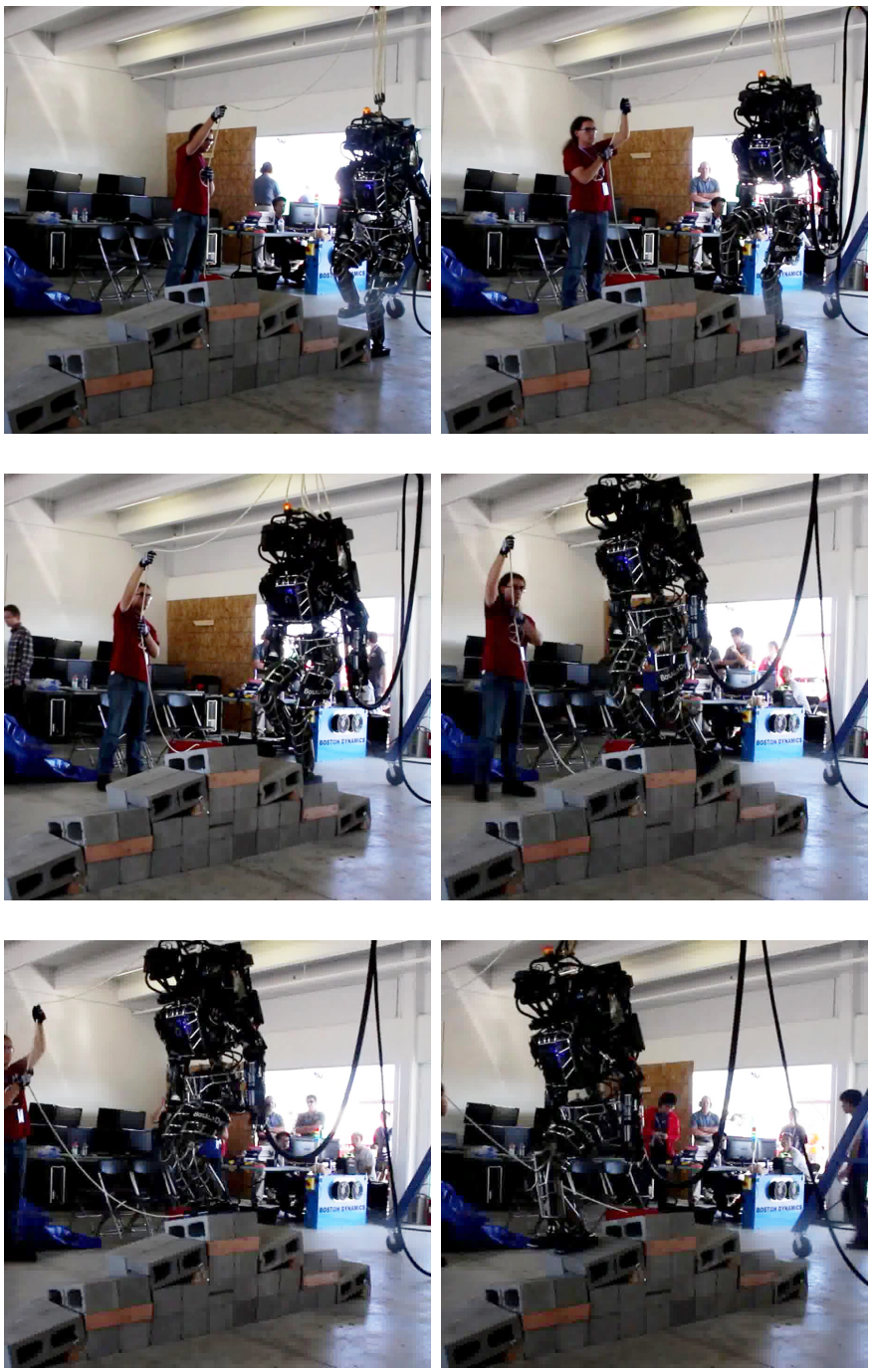


Figure 4.8: Snapshots of the Atlas robot walking up and then walking down the tilted cinder blocks.

4.5 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.

5

IMU Network

This chapter focuses on enhancing the sensing capability of humanoid robots with distributed IMU sensors. We believe super-human sensing (whole body vision systems, for example) is a useful research area which could greatly improve humanoid robustness and performance.

5.1 Joint Sensing

In the Atlas robot, most joint position sensing is done on the actuator side, either by Linear Variable Differential Transformers (LVDT)s (leg joints) or by potentiometers (shoulder and elbow joints), and transformed into the joint space. For each arm, the two shoulder joints and two elbow joints are rotational and hydraulic driven. Because these joint positions are sensed by potentiometers, and do not account for backlash, play and structural deformation, the accuracy of the arm forward kinematics using these sensors are poor. This has a huge impact on manipulation when used open-loop. Fortunately, there is an additional set of incremental encoders to measure position change post-joint transmission. These redundant encoders require calibration. They improve the accuracy of forward kinematics because their signal to noise ratio of are also higher than the pre-transmission potentiometers, and the position measurement is after backlash and play.

There is no joint velocity sensing available to the Atlas robot. Velocity is computed by

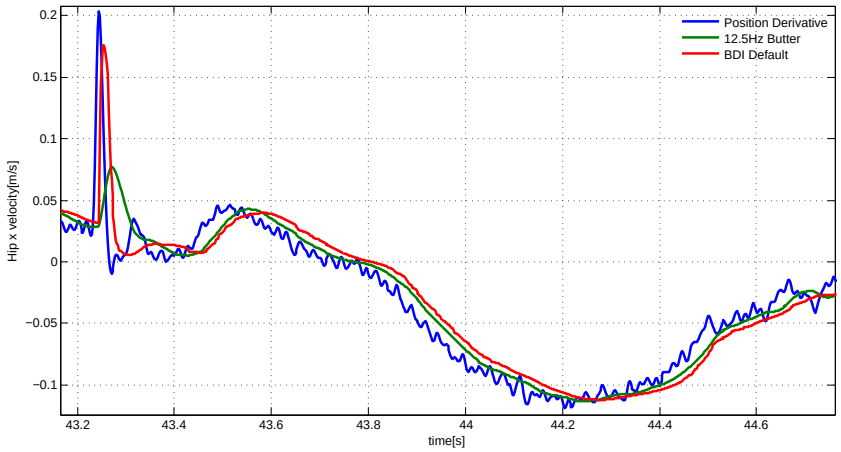


Figure 5.1: Comparison of the velocity traces of the left hip roll joint. Blue: position derivative filtered by an acausal low pass filter (50Hz cutoff) with no phase shift. Green: second order Butterworth filter with a cutoff frequency of 12.5Hz. Red: Boston Dynamics' default filter.

filtering the finite difference of positions. In addition to Boston Dynamics (BD)'s default filters for joint position, velocity, torque and error in torque, up to third order digital 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 joint level controller, we use second order Butterworth filters for joint velocities. We believe that BD's default velocity filter is adaptive to the frequency spectrum of the velocity signal. It is very sensitive to sharp velocity changes and amplifies the actual changes, and it filters slow signals more and induces a larger delay. In Fig. 5.1, we compare the velocity traces of the left hip roll joint from position derivatives, BD provided filter, and the second order Butterworth filter with a cutoff frequency of 12.5Hz. There is a large velocity change right after 43.2s, the BD filter's response is faster than the low pass without much attenuation. After 43.4s, the velocity change slows down and the BD filter displays a larger delay than the low pass filter. For our controller, the first attribute leads

Table 5.1: 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	

to undesired oscillations when we use higher feed-forward or feedback velocity gains. We are able to drastically increase the velocity gains with our customized low pass filters. The cutoff frequencies are summarized in Table. 5.1. To determine the cutoff frequency, we first minimize the cross covariance between the low pass filtered velocity and the BD's default velocity for walking, then we tune them together with the velocity gains to achieve reliable velocity tracking performance. The group delay is a measure of the time delay of the amplitude envelopes of various frequency components. For all the leg joints, the velocity delay in the passband is between 15ms and 27ms. The group delays of two second order Butterworth filters with different cutoff frequencies are shown in Fig. 5.2.

5.2 IMU Sensor Network

MEMS IMUs are low cost and ubiquitous. A 6-axis IMU measures linear accelerations using accelerometer and angular velocities using gyroscope. By fusing these sensor measurements, we can also obtain the orientation of the IMU relative to the gravity vector, up to an arbitrary yaw rotation using sensor fusion techniques, this will be discussed later.

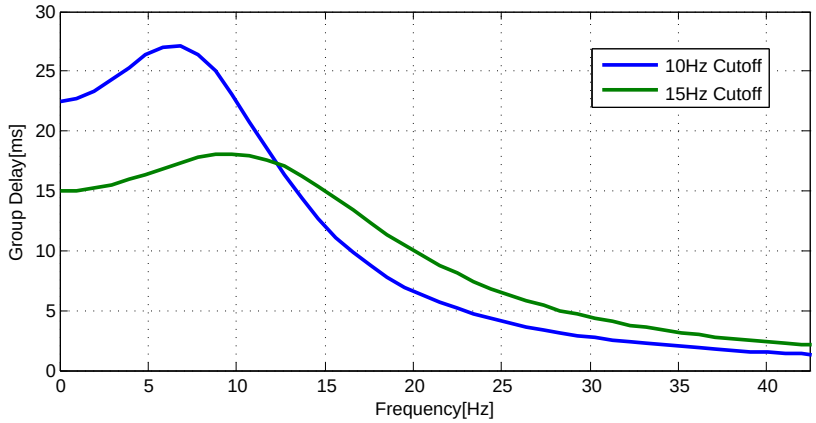


Figure 5.2: Group delays of two second order Butterworth filters. The cutoff frequencies are 10Hz and 15Hz.

5.2.1 IMU Sensor Specifications

We used MPU-6050 6-axis IMUs from InvenSense Inc. Before installing these IMUs on the robot, we collected some data to evaluate their performance.

Typically, the accelerometer is calibrated with a dividing head, and gyroscope is calibrated with a rate table [77]. Given we did not have access to any of this equipment, we could not really determine the accelerometer bias and scale factor for each axis, nor the scale factor for each axis of the gyroscope. We rotated the IMU by hand to roughly align the axises with the gravity vector to approximate the 6-point tumble test, and observed the total magnitude of the accelerometer readings. In each opposite direction, the total sensor reading can vary up to $\pm 50mg$ (the zero-g bias offset is $\pm 50mg$ in X,Y direction and $\pm 80mg$ in Z direction from the data sheet). This indicates a non-zero acceleration bias in that direction. We recorded these bias offsets and used them as constants. Because the acceleration measurements are really inaccurate, we will not use them to determine the accelerometer orientation with respect to gravity. Similarly we collected angular velocity

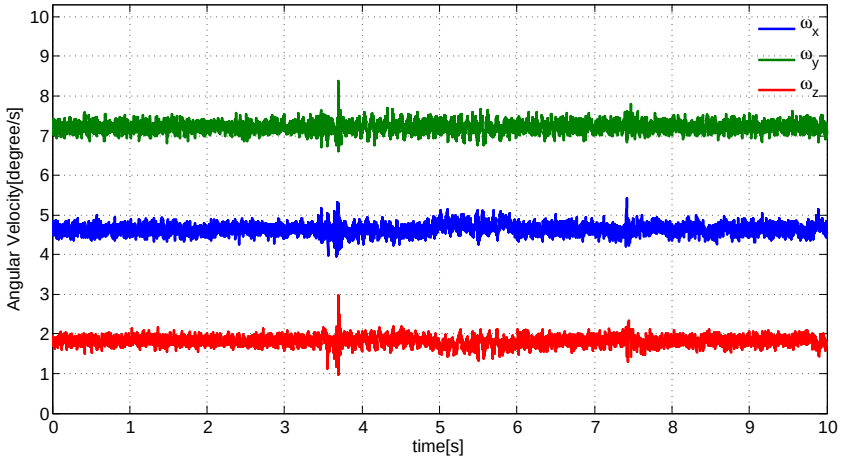


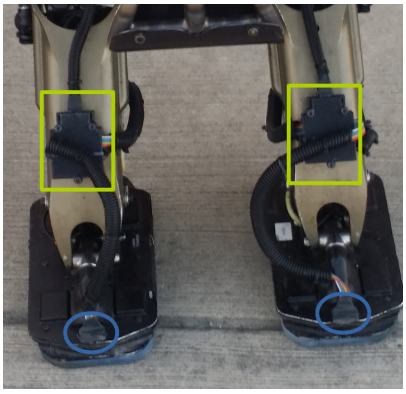
Figure 5.3: The MPU-6050 gyroscope measurement when the IMU is still.

data from the gyroscope while it was kept still. Refer to Fig. 5.3, the gyro bias offset for each axis is below $10°/s$ (the zero-rate bias offset is $\pm 20°/s$ from the data sheet).

5.2.2 System Setup

To enhance the sensing capability of the Atlas humanoid robot, we installed one sensor node at the back of each its lower leg, and attached one IMU to the shin, and one IMU to the back of the foot. The IMUs communicate with sensor node via ribbon cables. Refer to Fig. 5.4 for the setup.

The sensor node is composed of three parts: a WIZ550io auto configurable ethernet controller module, a Teensy-3.1 USB-based microcontroller board, and an adapter board that connects them. The Teensy board provides 3.3V DC power to the IMUs, and receives data through the SPI buses at a rate of 1000Hz, it then sends the data as UDP packets through the WIZ550io board to the control computer. We do not synchronize data from the IMUs to that from the robot sensors, as they run on separate clocks. This will not be



(a) Feet IMUs in the blue circle, sensor nodes in green box

(b) Shin IMUs right below the knee in the blue circle

Figure 5.4: The IMUs on the feet and shins are connected to the sensor node behind the lower leg. The sensor nodes collect data from the IMUs, and send UDP packets to the control computer via a switch.

an issue as long as the latency between both sets of data is small. We believe the round trip delay between the robot and the control computer is about 20ms, and latency between the IMU data and the control computer is below 3ms. Each sensor node is designed to handle a maximum of two IMUs at the same time.

Originally we also attached a sensor node below the pelvis and two IMUs to the thigh inside the shield (because the outside surface is concave). However due to electrical interference, they never functioned properly.

5.3 Distributed IMU Kalman Filter

Once rigidly attached to a link, the IMU can measure the angular velocity of the link in the IMU frame, this provides indirect velocity measurements for the joints that involved

in the kinematics chain. We do not use the accelerations from the IMU as measurements due to their poor performance.

We introduce the distributed IMU Kalman filter in this section. The state vector is defined as the joint velocities of the robot $x = \dot{\theta}$, the process dynamic is given by:

$$x_k^- = x_{k-1}^+ + \ddot{\theta}^d \Delta t \tag{5.1}$$

where θ, $\dot{\theta}$, and $\ddot{\theta}^d$ are the vector of joint positions, joint velocities and desired joint accelerations respectively, Δt is the time step. The desired joint accelerations $\ddot{\theta}_d$ are considered as input to the process dynamics, and are given by the controller after solving the Quadratic Program (QP). The linear state transition matrix is simply identity.

$$A_k = \mathbb{I} \tag{5.2}$$

The actual sensor measurements are composed of two parts: the derivative of the sensed joint position using finite difference, denoted as $\dot{\theta}^m$, and the angular velocities of the link in the IMU frame, denoted as $\omega_{imu}^{\star}$, where "$\star$" is a place holder for the actual link name, and "imu" stands for the IMU frame. The actual sensor measurement is given by

$$z_k = \begin{bmatrix} \dot{\theta}^m \\ \omega_{imu}^{rfoot} \\ \omega_{imu}^{rshin} \\ \omega_{imu}^{lshin} \\ \omega_{imu}^{lfoot} \end{bmatrix} \tag{5.3}$$

The measurement equation for the Kalman filter is also composed of two parts: the joint velocity measurements are simply the states, and the link angular velocities are computed using the Jacobian matrices of that link, $J_{\omega}^{\star}$. Only the angular part of the Jacobian is

required. The link angular velocities need to be transformed into the IMU frames from their respective body frame, this is achieved by the rotation matrices ${}^{imu}R_b^\star$, "b" stands for body. The measurement equation and observation matrix are given by

$$y_k = C_k x_k^- = \begin{bmatrix} \mathbb{I} \\ {}^{imu}R_b^{rfoot} J_\omega^{rfoot} \\ {}^{imu}R_b^{rshin} J_\omega^{rshin} \\ {}^{imu}R_b^{lshin} J_\omega^{lshin} \\ {}^{imu}R_b^{lfoot} J_\omega^{lfoot} \end{bmatrix} x_k^- \tag{5.4}$$

Each rotation matrix ${}^{imu}R_b^\star$ from the body frame to the IMU frame is a constant once the IMU is rigidly attached to the link. The link orientation in the world frame can be obtained by forward kinematics starting from the pelvis, we denoted these orientations by the rotation matrices ${}^{w}R_b^\star$s. There is a tactical grade IMU mounted on the pelvis of the robot. It provides pelvis orientation and angular velocity, which are used in the forward kinematics and Jacobian computation.

We implemented an online and an off-line method to compute the rotation matrices.

In our online implementation, we used a simple sensor fusion algorithm based on Sebastian Madgwick's work [78] to estimate the IMU's orientation in the world frame as a rotation matrix ${}^{w}R_{imu}^\star$. The algorithm is similar to Mahony's complementary filter algorithm [79]. We chose this algorithm over a Kalman filter because its parameter tuning was simpler, and it could compensate gyroscope bias drift without introducing additional state. The rotation matrices can be computed on each time step by

$${}^{imu}R_b^\star = ({}^{w}R_{imu}^\star)^T ({}^{w}R_b^\star) \tag{5.5}$$

Given ${}^{imu}R_b^\star$s are constants, we averaged the rotation for the first 2 seconds when the

robot is standing still. The mean rotation is solved using Singular Value Decomposition (SVD) [80].

The online algorithm is convenient because we could just attach an IMU to a link, and start using it as an indirect velocity sensor without post processing. However, this algorithm relies on accurate estimate of the IMU orientation in the world frame. This requires the sensor to be well calibrated, which is difficult for our automotive grade IMUs. We observed no drift in the IMU orientation after removing the gyro offset bias thanks to the gyro bias drift compensation. The main issue is that it is difficult to remove the accelerometer offset which has a huge impact on the IMU orientation estimate. The other issue is that the gyro frame is not necessarily well aligned with the accelerometer frame in these cheap IMUs, so there are potential systematic errors when using the online algorithm.

The off-line algorithm requires only the gyro angular velocity measurements $\omega^{\star}_{imu}$ from the IMUs, together with the link angular velocity $\omega^{\star}_{b}$ computed from forward kinematics using sensed joint positions and their time derivatives. We collect walking data over several steps, and our objective is to find the constant rotation matrices ${}^{imu}R^{\star}_{b}$s that minimize the Euclidean distances between the measured and computed link velocities in the IMU frames. Denote the aggregation of $\omega^{\star}_{imu}$ as a $3 \times n$ matrix Y, and $\omega^{\star}_{b}$ as a matrix X of the same size, where n is number of data points, we are solving the following optimization problem:

$$\underset{R \in SO(3)}{\text{minimize}} \sum_{i=1}^{n} |RX_i - Y_i|^2 \tag{5.6}$$

this is equivalent to

$$\underset{R \in SO(3)}{\text{maximize}}\ Tr\left((Y^T R)X\right) = \underset{R \in SO(3)}{\text{maximize}}\ Tr\left((XY^T)R\right) \tag{5.7}$$

This problem was first solved by Kabsch in [81][82], the solution is

$$R = Udiag(1, 1, sign(XY^T))V^T \tag{5.8}$$

where U and V are the orthonormal basis of XY^T's row and column space from Singular Value Decomposition (SVD)

$$XY^T = USV^T \tag{5.9}$$

The off-line algorithm has the advantage that it requires only the angular velocity measurements from the IMUs. This is desirable because we can reliably remove angular velocity offset bias by calibrating the gyroscopes at the beginning of each experiment. As long as the sensor location is fixed, we only have to compute the orientation offsets once.

We implemented the sequential Kalman filter algorithm introduced in Chapter 1 by using a constant diagonal measurement noise covariance R. The sequential measurement update replaces the matrix inversion with scalar multiplications. The sequential update step is summarized below, for clarity, we omit the time index in the equations.

Sequential Kalman Filter Update Step

$$x^0 = x^-, \quad P^0 = P^- \tag{5.10}$$

for $i = 1, 2, \cdots, m$

$$k^i = \frac{P^{i-1}c^i}{c^{iT}P^{i-1}c^i + r_{i,i}} \tag{5.11}$$

$$P^i = \left[\mathbb{I} - k^i c^{iT}\right] P^{i-1} \tag{5.12}$$

$$x^i = x^{i-1} + k^i \left[z^i - c^{iT}x^{i-1}\right] \tag{5.13}$$

$$x^+ = x^m, \quad P^+ = P^m \tag{5.14}$$

m is the number of measurements, k^i is a column vector, c^i is a column vector representing the i^{th} row of the observation matrix C, z^i is the i^{th} entry of vector z in Eq. (5.3, $r_{i,i}$ is the i^{th} diagonal entry of the measurement noise covariance R, $\mathbb{I}$ is the identity matrix, x^is and P^is are intermediate states and state error covariances. The Kalman Gain K is related to the vector k^i by

$$\mathbb{I} - KC = [\mathbb{I} - k^m c^{mT}] \times \cdots \times [\mathbb{I} - k^1 c^{1T}] \tag{5.15}$$

5.4 Results

The off-line algorithm computes the orientation offset using SVD. In Fig. 5.5, we plot the measured IMU angular velocity of the right shin, and the angular velocity computed using forward kinematics expressed in the IMU frame. In general, the gyro velocity measurements have lower noise level than the time derivative of the joint positions.

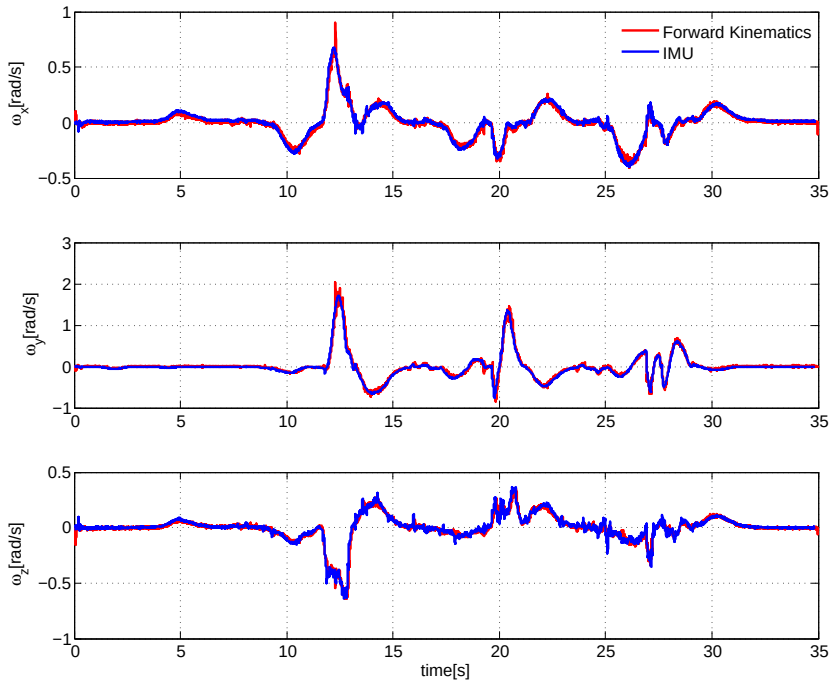


Figure 5.5: Measured and computed angular velocity of the right shin, both in the IMU frame. The transformation from the body frame to the IMU frame is computed using the off-line algorithm with SVD.

Table 5.2: Noise Parameters of the Distributed IMU Kalman filter

Process Noise Cov.	Back z	Back y	Back x	Neck y		
Q	$4.5e^{-3}$	$4.5e^{-3}$	$4.5e^{-3}$	$1.6e^{-4}$		
	Hip z	Hip x	Hip y	Knee y	Ankle y	Ak. x
	$4.5e^{-3}$	$7.2e^{-3}$	$1.1e^{-2}$	$1.1e^{-2}$	$1.1e^{-2}$	$1.1e^{-2}$
	Shoulder z	Sh. x	Elbow y	El. x	3 Electrical Wrists	
	$2.1e^{-3}$	$2.1e^{-3}$	$1e^{-3}$	$1.1e^{-3}$	$4.5e^{-3}$	
Observation Noise Cov.	Joints Velocities			IMU Gyro		
R	1			$1e^{-2}$		

Our filter parameters are summarized in Table 5.2. For joint velocities, the noise covariances are chosen such that with zero acceleration input and without the distributed IMU measurements, each joint has the same cutoff frequency as its second order Butterworth counterpart documented in Table. 5.1. This part of the Kalman filter approximates a first order low pass filter. We used significantly smaller values for the measurement noise covariance R on the IMU gyro signal, so that these measurements dominate the relevant leg joint state estimates.

The experiment result is shown in Fig .5.6. The data was collected during a fast walk in place experiment (two steps per second). We plotted the estimated pitch joint velocities on the left leg, because most of the motion was in that direction. A blown-up view of the same plot is displayed in Fig. 5.7. The Kalman filter traces are about 15-20ms faster than the low pass filtered ones, because we integrate the commanded acceleration in the process dynamics, and have indirect but faster velocity measurements.

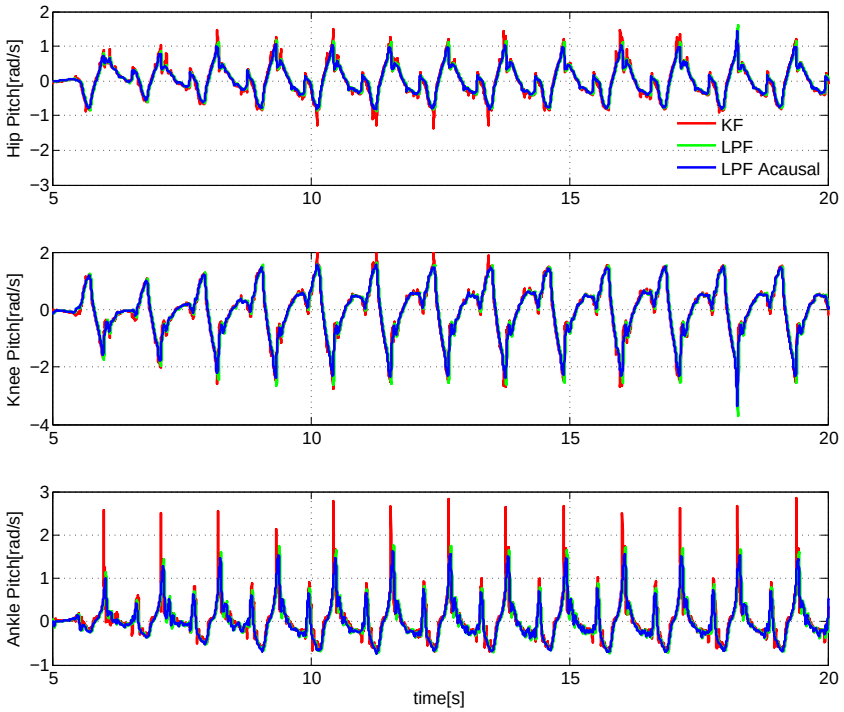


Figure 5.6: Plot of the pitch joint velocities of the left hip, knee and ankle during a fast walk. The red traces are the velocities from the Kalman filter, the blue traces are from an acaussal second order Butterworth filter (without delay), and the green traces are velocities from a second order Butterworth filter with a cutoff frequency of 15Hz.

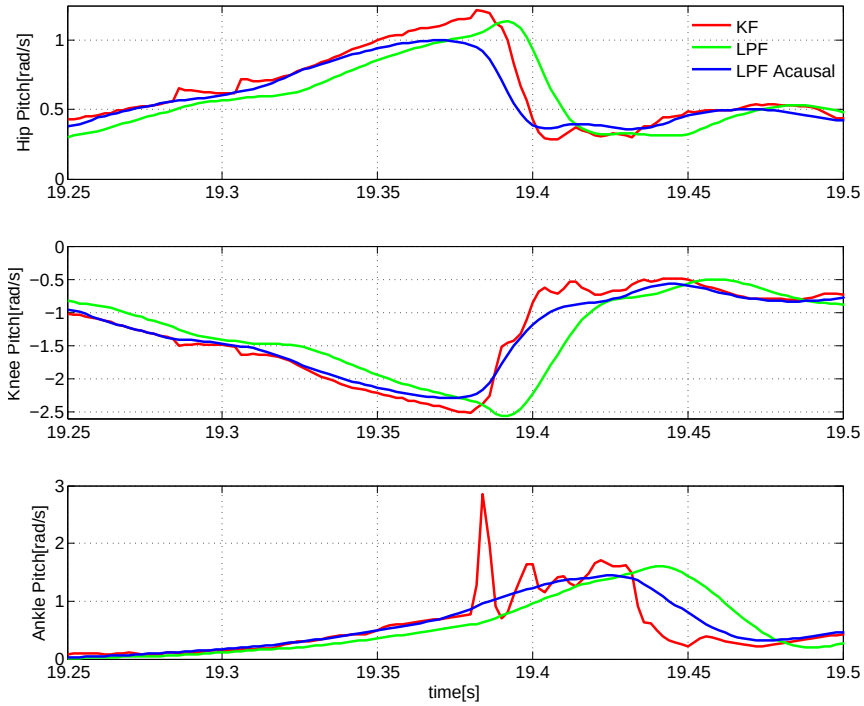


Figure 5.7: Zoomed in plot of the Fig. 5.6. There is a spike in Z direction at around 19.38s due to ground impact. The delay reduction from the Kalman filter is about 15-20ms compared to a second order Butterworth low pass filter with 15Hz cutoff (the group delay is around 15-20ms in the pass band, refer to Fig. 5.2)

5.5 Conclusion

In this chapter we introduced the distributed IMU Kalman filter. By attaching low cost MEMS IMUs to the robot, we have direct measurements of the link angular velocity. To use these measurements in joint velocity estimates, we developed online and off-line algorithms to determine the sensor orientation relative the attached body part. Choice can be made based on the application and the sensor quality. The performance of the Kalman filter is compared to that of a second order Buttworth filter, and we demonstrate its effectiveness in reducing the joint velocity delay for hydraulic robots without joint velocity sensing.

6

CoM Estimator and Its Applications

In order to achieve reliable estimation of the humanoid robot Center of Mass (CoM) motion, we need a model that explains how the CoM moves under external forces. The Linear Inverted Pendulum Model (LIPM) model is a natural choice given its simplicity [50]. We introduce an additional offset term into the LIPM dynamics. This offset can be interpreted as a modelling error on the CoM position, or an external force exerted on the CoM of the robot, or a combination of both. This chapter introduces the CoM estimator, and its application in inverse dynamics, fall detection, and fall prevention for humanoid robots. The estimator itself is similar to [39], but the implementation and application are different.

6.1 Modeling Error

The estimation accuracy of a model-based state estimator relies on the model itself. For the Atlas robot, Boston Dynamics provides both the kinematic and the dynamic parameters. The kinematic model is useful for manipulation tasks and foot placement during walking. The dynamic model is used by the controller to generate desired joint torques. Both kinematic and dynamic models are used to estimate the full-body states of the robot (Chapter 3 and 4).

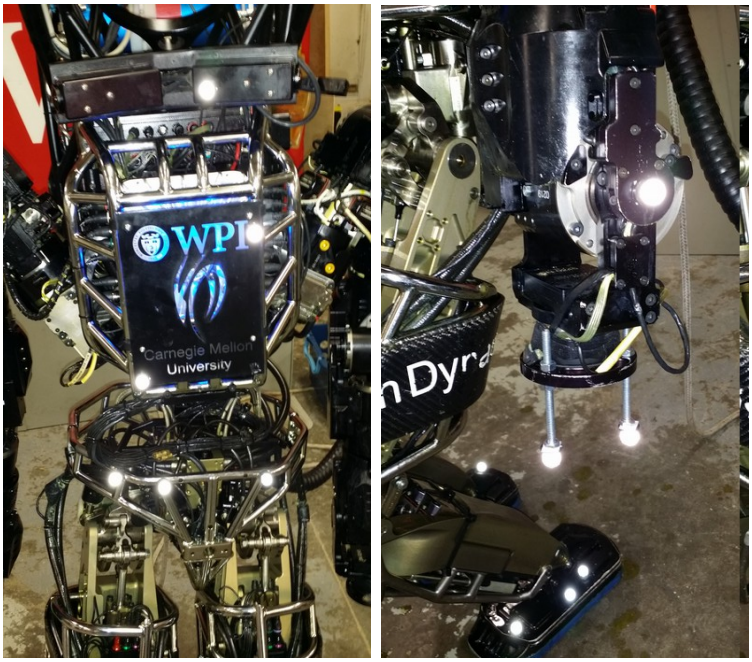


Figure 6.1: Marker locations

6.1.1 Kinematics Modeling Error

The kinematic model is rooted at the pelvis. We experimented on the hand end effector to determine the arm kinematic error. We put Atlas in double support, and controlled the hand to follow a predefined trajectory. The trajectory was a concatenation of manipulation task trajectories, because we care about the kinematic model accuracy in those situations. We tracked the motion of the hand, feet, and pelvis using an OptiTrack motion capture system with 8 cameras, and recorded the marker data and robot data simultaneously at 100Hz. The marker locations are shown in Fig. 6.1. We found that the hand end effector had on average an error of about 1cm relative to the pelvis.

We also tested the leg kinematic error during double support without the motion capture system, because we know that both feet are not moving. We control the pelvis of the robot to follow a spiral trajectory in the coronal plane. The average error for the distance

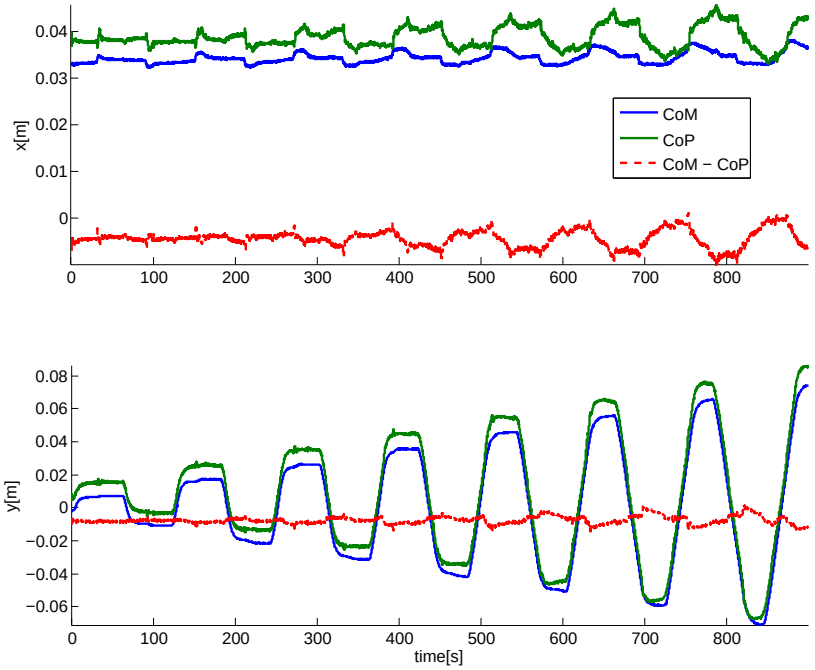


Figure 6.2: Computed CoM, measured CoP and their difference in the horizontal ground plane

between the two feet is within 1cm.

There are several factors contributing to the end effectors' kinematic error. The joint position data we collected was pre-joint transmission (the sensors are on the actuator, not the joint axis). It is computed based on the linear actuator length measured by the Linear Variable Differential Transformers (LVDT) which is affected by the loading conditions. The sole pads beneath the robot's feet are deformable up to a few millimeters. There are also backlash, play and structural deformation on the robot. These effects all contribute to the kinematic error between the two feet.

6.1.2 Dynamics Modeling Error

The dynamic model has the mass, inertia and center of mass location of each link as its parameters. They are generally not identified as accurately as the kinematics parameters. The error of these parameters is difficult to model individually because the dimension of the parameter space is high. We were not allowed to take the robot apart to measure the inertial parameters. It is hard to account for flexible elements such as oil filled hoses that cross joints in a rigid body model.

On the other hand, the dynamics of the center of mass of the robot can be modeled with a small number of parameters. A widely used control scheme for humanoids is two-level hierarchical control. The high level controller generates the center of mass motion using a simple dynamic model, such as the LIPM, and the low level controller tries to generate the full-body motion given the center of mass motion. It is therefore essential for the controllers to know accurately the motion of the center of mass. Due to modeling error of the link parameters, the center of mass position and velocity computed from forward kinematics will have modeling error as well.

To observe the modelling error, we conduct a simple quasi-static experiment using our Atlas robot. The robot pelvis follows a spiral trajectory in double support. In an ideal situation, where there is neither modeling error in the dynamic model parameters nor forward kinematics, and the foot force/torque sensors are calibrated, the CoM should coincide with the Center of Pressure (CoP) in the horizontal plane. We observe there is a nearly constant offset between the CoM position computed from kinematics, and the CoP computed from foot force/torque sensors and kinematics. Fig. 6.2 shows the computed CoM, measured CoP and their differences in the horizontal plane.

6.2 The Center of Mass Kalman Filter

We use the following parameterization of the discrete time linear Kalman filter,

$$x_{k+1} = f(x_k, u_k) = Ax_k + Bu_k \tag{6.1}$$

$$y_k = h(x_k) = Cx_k + Du_k \tag{6.2}$$

The LIPM is a very simplified model to describe the dynamics of a humanoid robot. It assumes the CoM is at a constant height, therefore the total vertical force must be equal to the body weight at all times. The dynamic equation of LIPM in the horizontal direction is given by Eq. (6.3)

$$\ddot{x}_{com} = \omega^2(x_{com} - u_{cop}) \tag{6.3}$$

where $\omega = \sqrt{g/z_{com}}$ is the LIPM eigenfrequency. x_{com}, $\dot{x}_{com}$ and $\ddot{x}_{com}$ are the position, velocity and acceleration of the CoM in the horizontal plane respectively. u_{cop} is the position of the CoP, g is gravity, and z_{com} is the CoM height relative to the CoP. In Eq. (6.3), u_{cop} represents the total CoP.

The error of the LIPM can be modeled by an offset. If we assume the offset is changing relatively slowly, then its first and second order time derivatives are both close to zero. We formulate the CoM estimator with offset as a Kalman filter. It predicts the robot CoM position, velocity and offset in the horizontal plane. The state vector for the CoM estimator is given by $x = [x_{com}, \dot{x}_{com}, x_{offset}]^T$. We consider the CoP as an input to our state estimator. The process model uses the LIPM dynamics to predict the CoM acceleration. The LIPM dynamics with offset is given by Eq. (6.4)

$$\ddot{x}_{com} = \omega^2(x_{com} + x_{offset} - u_{cop}) \tag{6.4}$$

x_{offset} is the LIPM offset. The states are expressed in the world coordinates, and have a dimension of two (both x and y components). z_{com} is computed every estimation time step by forward kinematics. The Kalman filter becomes time variant due to z_{com} being a parameter in the state matrix A in Eq. (6.5). The process model of the Kalman filter follows Eq. (6.1). It has the following parameters:

$$A = \begin{bmatrix} \mathbb{I} + \frac{1}{2}\omega^2\Delta t^2\mathbb{I} & \Delta t\mathbb{I} & \frac{1}{2}\omega^2\Delta t^2\mathbb{I} \\ \omega^2\Delta t\mathbb{I} & \mathbb{I} & \omega^2\Delta t\mathbb{I} \\ 0 & 0 & \mathbb{I} \end{bmatrix}, B = \begin{bmatrix} -\frac{1}{2}\omega^2\Delta t^2\mathbb{I} \\ -\omega^2\Delta t\mathbb{I} \\ 0 \end{bmatrix} \tag{6.5}$$

where Δt is the time step, $\mathbb{I}$ is a 2×2 identity matrix.

The observation model provides predicted measurement from states. We choose x_{com} and $\ddot{x}_{com}$ as predicted measurements. The measurement equation can be written as

$$\begin{aligned} y_k &= \begin{bmatrix} x_{com} \\ \ddot{x}_{com} \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{I} & 0 & 0 \\ \omega^2\mathbb{I} & 0 & \omega^2\mathbb{I} \end{bmatrix} \begin{bmatrix} x_{com} \\ \dot{x}_{com} \\ x_{offset} \end{bmatrix} + \begin{bmatrix} 0 \\ -\omega^2\mathbb{I} \end{bmatrix} u_{cop} \end{aligned} \tag{6.6}$$

where the output matrix C and feedthrough matrix D in Eq. (6.2) are given by

$$C = \begin{bmatrix} \mathbb{I} & 0 & 0 \\ \omega^2\mathbb{I} & 0 & \omega^2\mathbb{I} \end{bmatrix}, D = \begin{bmatrix} 0 \\ -\omega^2\mathbb{I} \end{bmatrix} \tag{6.7}$$

The measurement depends on sensors available to the robot. We use sensed joint position with forward kinematics to compute CoM position. The root variables are estimated using the base state estimator introduced in Chapter. 3.

Ideally, 6-axis force/torque sensors under the robot feet provide information about external forces, which can be used to calculate the CoM acceleration in the horizontal plane. However, using the 3-axis (F_z, M_x and M_y) foot force/torque sensors on the Atlas, we can only compute the CoP in the foot frames. We approximate the CoM acceleration by transforming the acceleration measured by the IMU mounted on the pelvis to the location of the CoM. This transformation assumes that the CoM is on the same rigid body (pelvis) as the IMU. The transformation is given by:

$$a_{com} = a_{imu} + \omega_{imu} \times (\omega_{imu} \times r) + \dot{\omega}_{imu} \times r \tag{6.8}$$

where a_{com} is the approximated CoM acceleration. a_{imu} and ω_{imu} are IMU measured acceleration and angular velocity expressed in the world frame. r is the vector from IMU to CoM computed using forward kinematics. $\dot{\omega}_{imu}$ is the angular acceleration of the IMU, it is computed by finite differentiating ω_{imu}. The x and y component of a_{com} are used as measurement.

6.3 Implementation and Application

We implement the CoM estimator on the Atlas robot. The filter noise parameters are summarized in Table 6.1. The measurement noise parameters for the acceleration are different for walking and for manipulation. Because the walking motion is more dynamic than manipulation, we filter the acceleration less in walking.

The estimated offset plays several roles in the control of the robot. The first application is to translate it into an external force acting on the robot CoM, and to apply the force while solving the inverse dynamics using QP [83]. During early development, the offset was also used to bias the target CoM location while solving Inverse Kinematics (IK) for walking [76], Inverse Kinematics (IK) is no longer used in our walking controller. The

Table 6.1: CoM Kalman filter noise parameters

	x_{com}	$\dot{x}_{com}$	x_{offset}
Q	$1e^{-10}$	$1e^{-4}$	$1e^{-10}$

Walking	x_{com}	$\ddot{x}_{com}$
R	$1e^{-6}$	$1e^{-3}$

Manipulation	x_{com}	$\ddot{x}_{com}$
R	$1e^{-6}$	$1e^{-2}$

second application of the offset is to detect and prevent robot falling. In this case, the offset is interpreted as a position offset of the robot CoM, and then used to compute a corrected capture point. The details will be discussed in Section. 6.4.

6.3.1 Kinematic Modelling Error Compensation

Due to pre-transmission joint position sensing and backlash, play and compliance in the mechanism, the forward kinematics model for Atlas is not very accurate. For stationary single support stance, there can be up to 3cm offsets between the model CoM from forward kinematics and measured CoP. A simple linear torque dependant heuristic to reduce the effects of joint compliance is proposed by [84], which is also employed in our controller for the leg joints.

In addition to the pre-transmission joint position sensing, the shoulder and elbow joints are also equipped with incremental encoders that measure position changes. The encoders provide more accurate joint position measurement because they are post transmission. They can only be used after calibration. Our calibration procedure is to drive all the joints to the known mechanical joint limits and fit parameters to the offsets and scale factors. A very similar procedure provided by Boston Dynamics is used to calibrate the electric

forearms.

6.3.2 Dynamic Modelling Error Compensation

The estimated CoM offset x_{offset} can be interpreted as a virtual force $f_{ext} = m\omega^2 x_{offset}$ applied at the CoM of the robot. Given the equation of motion of the full body floating base dynamics of a humanoid robot,

$$M(q)\ddot{q} + h(q, \dot{q}) = S\tau + J_c^T(q)f \tag{6.9}$$

we add the virtual force to the right-hand side of Eq. (6.9) pre-multiplied by the CoM Jacobian transpose J_{com}^T,

$$M(q)\ddot{q} + h(q, \dot{q}) = S\tau + J_c^T(q)f + J_{com}^T(q)f_{ext} \tag{6.10}$$

so that the modelling error can be handled properly by the inverse dynamics.

In this experiment, the robot walked forward a few steps.

In Fig. 6.3, we show the estimated CoM offset, and the difference between CoM and CoP considering the CoM offset. If the robot is under a static configuration, the difference between corrected CoM and CoP should be zero. When the CoM is accelerating, LIPM dynamics explains the difference. We observe near zero difference in single support, and large differences during double support when the robot is shifting its supporting foot.

Fig. 6.4 is a plot of the CoM velocity from the Kalman filter, and from forward kinematics. The CoM velocity from forward kinematics has noise injected from the measured IMU angular velocity, and occasional spikes from the left shoulder and wrist pitch joints. The estimated CoM velocity has a lower noise level compared to that computed from forward kinematics.

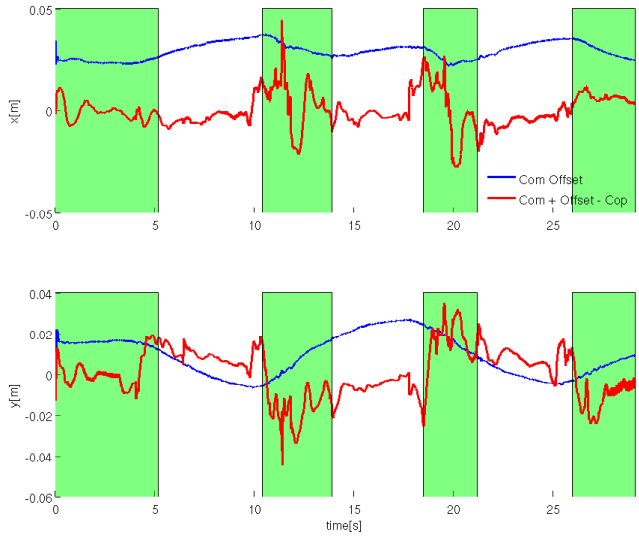


Figure 6.3: Plot of the estimated CoM offset, and the difference between CoM and CoP after applying CoM offset during static walk. The shaded green region is double support, the unshaded region is single support.

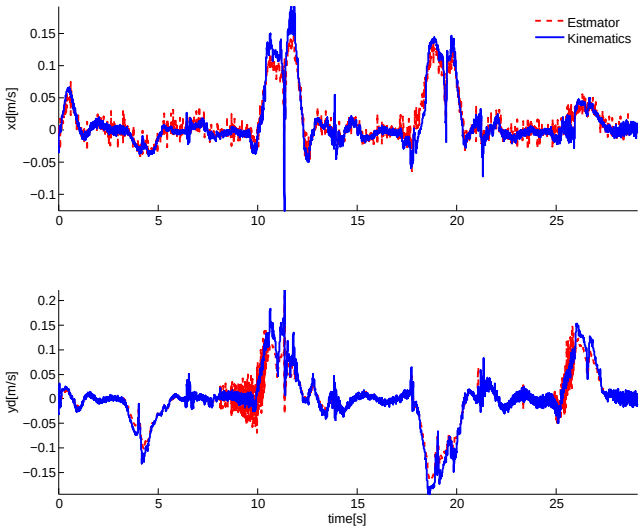


Figure 6.4: Plot of the estimated CoM velocity, from the CoM offset estimator, and from full body kinematics starting at the root

6.4 Fall Prevention and Detection

Compared to wheeled and multilegged robots, humanoids usually have higher center of mass and smaller support region. This means a higher risk of hardware damage when falling and a smaller stability margin. In a real world scenario where humanoid robots have to interact with the environment without a safety belay system, fall prevention and recovery behaviors are necessary. A good example is the DARPA Robotics Challenge, where robots have to perform disaster response related tasks involving physical contact with the environment. What we observed during the contest was that most humanoids fell down. We will discuss our implementation of fall detection and prevention algorithms using the CoM estimator.

6.4.1 Background

Fujiwara's group in JAIST have worked on fall control of humanoid robots [85][86][87][88][89][90], their work focused on minimizing impact during fall. Another line of work deals with fall detection. The general idea is to detect abnormality in sensor measurements or computed quantities such as CoM and CoP. The methods differ in what features to use and how they are parameterized, whether the model is built online or offline, and what algorithms are used to detect abnormality. Sensor measurements are compared with sensor models in humanoid walking to determine the dynamic instability of the gait. The sensor model is built offline in [91], and a heuristic is used online in [92]. Ogata et al. detect abnormality through discriminant analysis and learning [93][94], similarly, pattern classification was used in [95]. Yun et al. introduced the concept of "Fall trigger boundary" which encloses a region in the robots feature space where the balance controller is able to stabilize the robot [96].

6.4.2 Capture Point

The Capture Point (CP), introduced in [97], is the point on the ground where a humanoid must step to in order to come to a complete stop. If the humanoid dynamics is approximated using the LIPM, the Capture Point (CP) has a very simple expression as a linear combination of the CoM position and velocity in the horizontal plane. We can derive the CP by analysing the LIPM orbital energy [97], or by solving the LIPM ODE Eq. (6.3) and posing boundary conditions [98]. We briefly derive the CP expression by solving Eq. (6.3). The CoM has the following solution

$$\begin{aligned} x_{com}(t) = &\frac{1}{2}\left[x_{com}(0) - u_{cop} + \frac{\dot{x}_{com}(0)}{\omega}\right] e^{\omega t} \\ &+ \frac{1}{2}\left[x_{com}(0) - u_{cop} - \frac{\dot{x}_{com}(0)}{\omega}\right] e^{-\omega t} + u_{cop} \end{aligned} \tag{6.11}$$

The CP x_{cp} is defined as the point when the CoM is to a complete stop, which can be written as

$$x_{cp} = \lim_{t\to\infty} x_{com}(t) = \lim_{t\to\infty} u_{cop} \tag{6.12}$$

In order for Eq. (6.12) to be finite, the divergent component of Eq. (6.11) $e^{\omega t}$ has to have zero coefficient, which translates to

$$x_{com}(0) + \frac{\dot{x}_{com}(0)}{\omega} = x_{cp} \tag{6.13}$$

Given that Eq. (6.13) has to be satisfied all the time, the time index can be dropped and the CP is defined as

$$x_{cp} = x_{com} + \frac{\dot{x}_{com}}{\omega} \tag{6.14}$$

Because the simple CP expression Eq. (6.14) is derived from the LIPM dynamics, it is only a good approximation of the real CP when the robot motion can be well explained by the LIPM dynamics. More complicated approximations such as the Angular Momentum Pendulum Model lead to more complicated expressions of the CP. Refer to [97] for a detailed discussion. We use the simplest CP on the Atlas robot because LIPM is a good enough approximation for tasks our robot performs.

6.4.3 Fall detection and prevention

Our fall detection algorithm is based on whether the CP is within the robot's polygon of support. The CoM estimator is used to compute the CP, and the key of our algorithm is to use a "Corrected Capture Point (CCP)", where the LIPM offset is used as a CoM position offset. The Corrected Capture Point (CCP) $\hat{x}_{cp}$ is defined by

$$\hat{x}_{cp} = x_{com} + \frac{\dot{x}_{com}}{\omega} + x_{offset} \tag{6.15}$$

The $\hat{x}_{cp}$ is readily available since every quantity in Eq. (6.15) is a state estimated by the CoM filter. On the one hand, it can handle modelling error and relatively small dynamics forces, such as drag from the tether. On the other hand, it is especially helpful in estimating the unplanned quasi-static external forces applied at unknown locations on the robot, which happens quite often during manipulation tasks. We are making the assumption that any external force is caused by the robot's action, and that is why we freeze/stop, to stop the force from increasing, moving the CP outside the polygon of support, and forcing a step response.

Consider the following example: the robot is standing still and pushing on the wall slowly with one hand until it tips over. Even though the modelled CP x_{cp} is kept at the center of the support polygon by the balance controller, the CCP $\hat{x}_{cp}$ is pushed back and

out of the support polygon once the robot is tipped over. Without estimating the LIPM offset, the modelled CP can not predict falling.

Even though we assume the total external force applied to the robot is slow and quasi-static, so that x_{offset} has zero derivative which is consistent with Eq. (6.5), the CoM estimator formulation is not limited by this assumption. By increasing Q (reducing expected measurement noise) on the x_{offset}, and decreasing R (increasing the expected process noise) on the CoM acceleration, we can essentially tune how fast the x_{offset} is estimating the total external force. The adaptive Kalman filter is a good candidate to handle different situations automatically. The implementation of the fall detection and prevention algorithm on the robot is controller dependent. There are two full body controllers implemented on the robot, one for manipulation and one for walking. Details of these controllers can be found in [83].

Manipulation

For the manipulation controller, the robot is always assumed to be in double support, and the support polygon is computed by finding the convex hull of the foot corner points using Andrew's monotone chain 2D convex hull algorithm [99]. The foot corner points are computed using forward kinematics. To prevent the robot from falling during manipulation, we require the CCP to lie within a subset of the support polygon called the safe region, see Fig. 6.5. The moment the CCP escapes the safe region, a freeze signal is sent to the manipulation controller, and it clears all running joint trajectories and freezes the robot at the current pose (the balance controller is still running).

During our second run in the DRC finals, the right electric forearm mechanically failed when the cutting motion was initiated during the drill task. The uncontrolled forearm wedged the drill into the drywall and pushed the robot backwards. The controller froze and saved the robot from falling. The operator was able to recover from an otherwise

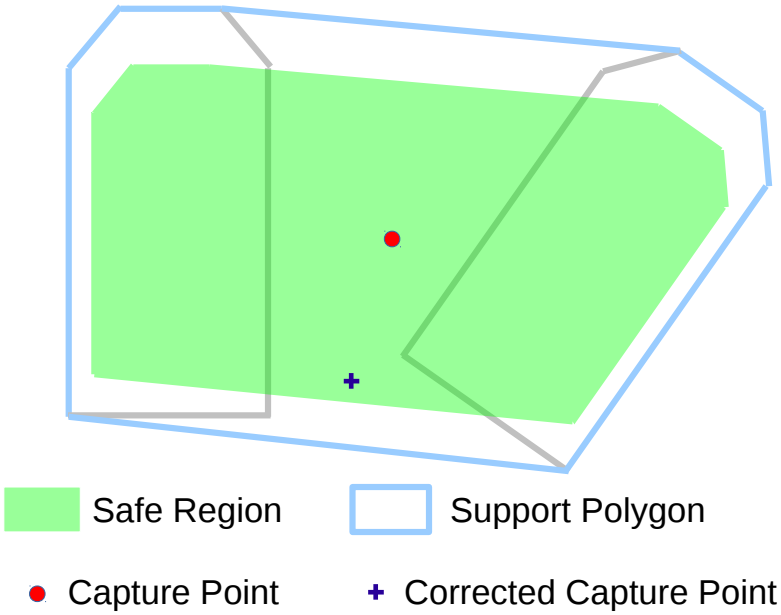


Figure 6.5: For the manipulation controller, the robot is in double support and pushed back by an external force. The modelled CP is maintained at the center of support polygon by the controller, the CCP is pushed back and close to the boundary of the safe region. The controller will freeze the robot as soon as the CCP is outside of the safe region. In our implementation, the safe region is the convex hull of the shrunk foot corners, where the shrunk foot has the following dimension: 4.5cm to the front and side, 5cm to the back of the physical foot.

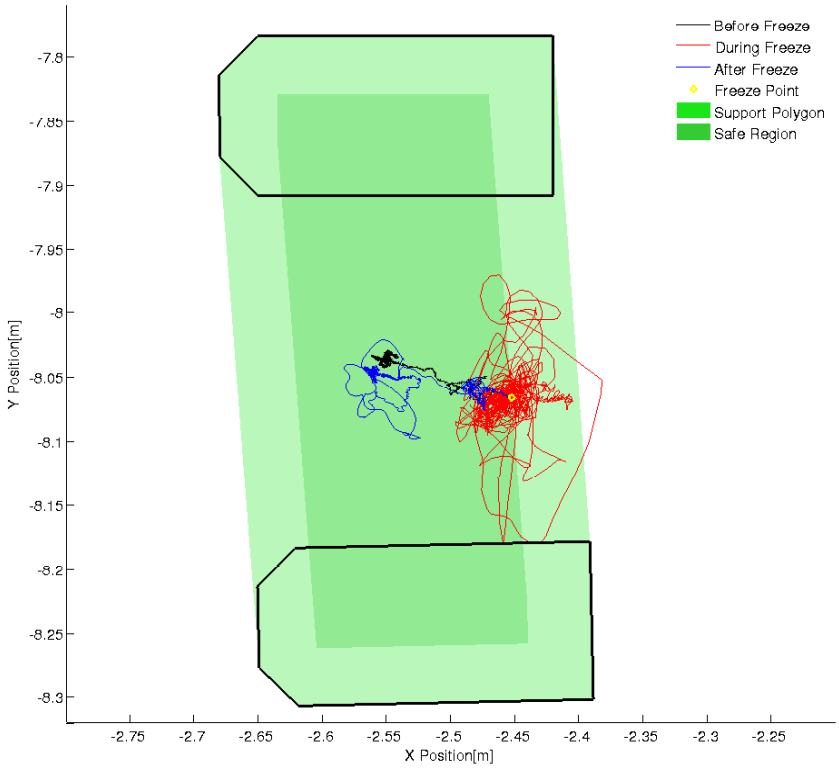


Figure 6.6: The CCP motion during the drill task on the second day of the DRC Finals. The black trace started from the center of support polygon and moved towards the back, corresponding to the robot pushing itself backwards. Once the CCP exceeds the boundary of the safe region (the yellow circle), the arm motion stopped and the robot went into freeze mode. The red trace showed the corrected CP motion during the freeze, it went outside of support polygon briefly and had oscillations. The blue trace was when the operator unfroze the robot and it released the drill.

catastrophic scenario. Fig. 6.6 is a plot of the CCP in the foot ground plane.

The time plot in Fig. 6.7 shows candidate fall predictors during this event. We can eliminate some candidate fall predictors easily. The CoM (and a âĂIJcorrectedâĂİ CoM (not shown)) usually provide a fall warning too late, because the CoM velocity is not included. The CP does not include information about external forces. The CoP can warn of foot tipping, but it is less reliable about warning about robot falling, which is not the same thing as foot tipping in a force controlled robot or if there are non-foot contacts and external forces. In this plot, we see that the CoP moves away from the safe region during recovery, predicting that the robot is going to fall, while the CCP moves towards the interior of the safe region.

The fall detection algorithm for the manipulation controller is the same as the walking controller and will be discussed next.

Walking

Compared to the manipulation tasks where the robot is always in double support, the support polygon undergoes constant change during walking, and is determined by contact state. The contact state is computed by thresholding the 3-axis foot force/torque sensor readings. The strain gages on the robot are not perfect even after calibration. Often a nonzero value up to ±50N is measured on the swing foot, sometimes a much bigger value can be observed during swing. This value can change from step to step (see Fig. 6.8), which makes robust contact state detection difficult. We implemented a simple Schmitt trigger for contact state detection. Instead of a single threshold, a Schmitt trigger has both low and high thresholds. If the foot is in contact and the force/torque sensor reading drops below the low threshold, the contact state of that foot switches to swing. If the foot is in swing and the sensed force is above the high threshold, the contact state changes to stance. We choose the low and high thresholds to be 80N and 100N respectively. The Schmitt trigger

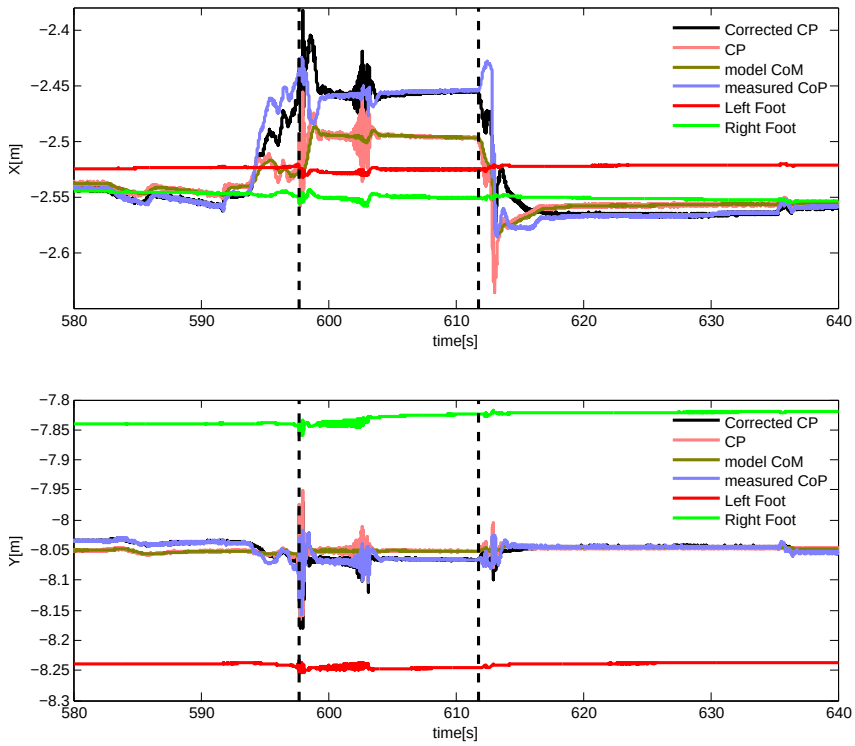


Figure 6.7: Plots of candidate fall predictors in the drill task. The black vertical dashed lines mark the freeze time and the start of manual recovery.

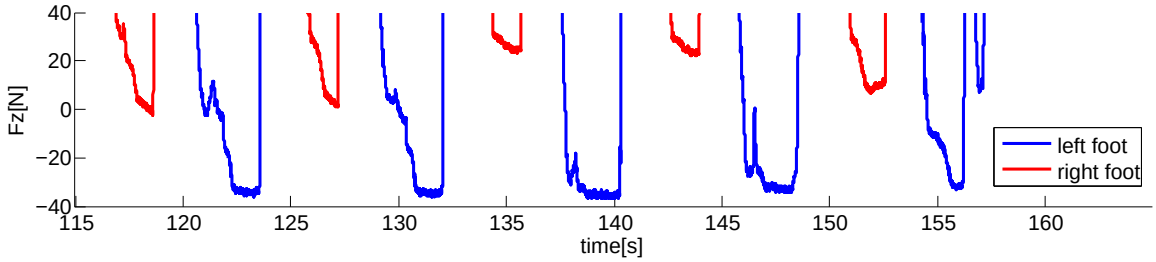


Figure 6.8: The force/torque sensor readings during swing for several steps, the sensors are calibrated in the air before taking steps. Ideally the flat part should read near zero and equal for both feet, but we observe measurements ranging from -40N to 40N and they are not constant or repeatable.

eliminates a lot of noise compared to a single threshold. Even with the Schmitt trigger, the contact detection could go wrong sometimes due to bad sensor readings. We suspect the strain gage fluctuation is due to thermal effects, and a possible change in the strain gage bonding with the substrate.

For dynamic walking, the CP naturally goes beyond the supporting polygon during swing phase and is captured by the touchdown foot. One way to predict a fall is to use the swing foot time to touchdown and predict if the current CP is within the support polygon of possible or desired footholds. This approach is complicated because it depends on the controller. Our simpler solution is to use a heuristic that detects a fall only if the CP is outside of the support polygon for a continuous period of time. The time is set to 0.6 seconds after extensive testing. As soon as the fall is detected, there are several things the humanoid can do, such as using angular moment to maintain balance, or taking a step as presented in [98]. We have implemented a simple step recovery controller that works in single support where the robot puts down the swing foot right away given there is no self collision.

If none of the aforementioned approaches could stop the robot from falling down, then a safe fall behavior should be triggered. In our case, Boston Dynamics provided us with a safe fall behavior. The robot swiftly goes to a pose where the arms are folded in front of the robot, the knee joints are bent to lower the CoM and the ankle joints are pitched down to allow the robot to roll backwards and hopefully land on its backpack. In reality we never used this behavior with our fall detection, because the hands will collide with the chest plates. The behavior was designed without considering the hands. Instead of using the provided safe fall behavior, we control the neck joint to tilt up all the way to its limit when a fall signal is received as a sanity check. There happened to be a false positive due to false contact detection during one of our terrain task practice runs two days before the glsdrc Finals, so we turned off the fall detection in the walking controller in the final contest.

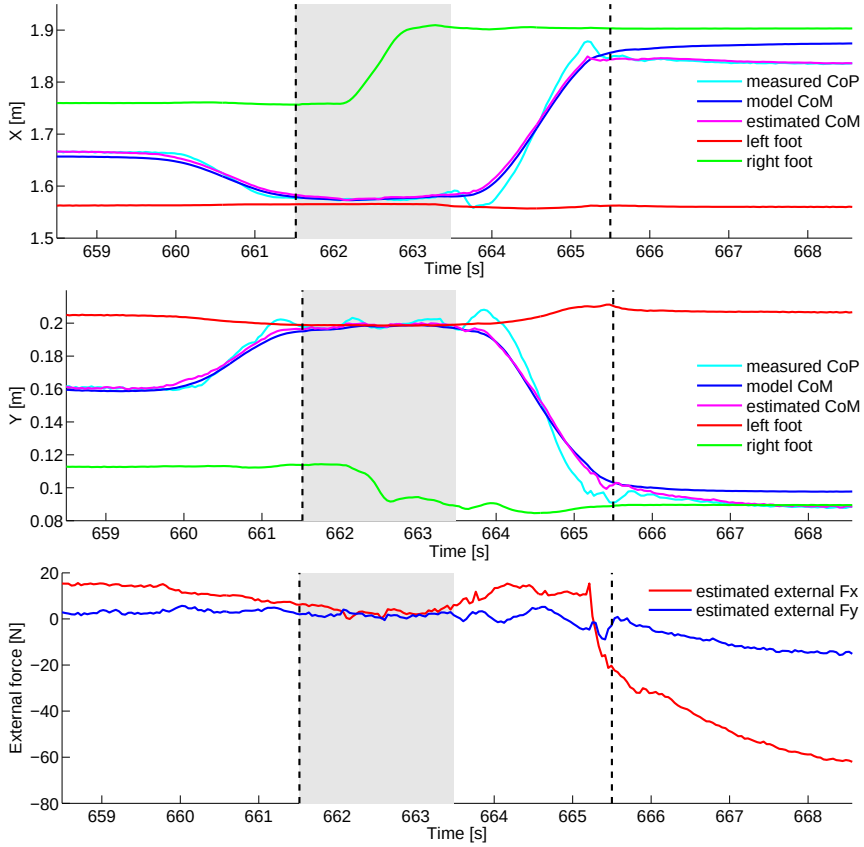


Figure 6.9: Atlas was caught on the door frame when sidestepping through it during the rehearsal at the DRC Finals. The walking controller delayed liftoff and remained in double support when the CoM estimator detected a large offset. Single support phase is shown by the shaded area, and the black dashed lines indicate the planned liftoff time. The estimated CoM is the sum of the model CoM and the estimated CoM offset.

On the DRC rehearsal day, the robot was caught on the door frame when sidestepping though. The walking controller detected that the CoP had not moved as expected, delayed liftoff and remained in double support, and stopped the current behavior for manual recovery. Data from this experiment is plotted in Fig 6.9.

6.5 Conclusions

In the chapter, we introduced the CoM estimator for humanoid robots and its applications in modeling error compensation and fall detection and prevention. A better estimate of the CoM motion improves the controller performance and stability of the robot. Essentially, it uses the LIPM dynamics with variable height to approximate the dynamics of the CoM of the robot. The modeling error on each link is lumped as the CoM offset. It makes state estimation simple and robust, and saves us from tuning mass models of the robot. More importantly, it serves as an observer of unexpected external forces. This enables fall being detected and sometimes prevented. During the DRC Finals, the CoM estimator successfully saved our robot from falling in two cases, and made us the only competitive team that attempted all tasks, and did not fall or require human physical assistance. Its limitations come from the limitation of the LIPM model, which decouples the sagittal and coronal motion by assuming constant height, and does not take into account angular momentum.

7 Conclusions

7.1 Contributions

This thesis presents applications of dynamic models for use in hydraulic humanoid state estimation. The thesis has several contributions, some of which have been published, and some have been demonstrated in real world scenarios during the DARPA Robotics Challenge (DRC) Finals. The following is a list of contributions:

- We have proposed three methods to estimate the joint velocities of hydraulic humanoids. Two of the approaches use the full body dynamics of the robot directly: the decoupled approach uses steady state Kalman filter to speed up computation, and the Quadratic Program approach does not require state space form as the Kalman filter does and handles constraints naturally. The third method requires additional sensing with low cost MEMS IMUs. It is robust to dynamic modeling error of the robot.

- We have developed accurate odometry for legged locomotion by fusing inertial sensor information with forward kinematics. The odometry accuracy is over 98% on our Atlas robot. The odometry was sufficient for the Atlas to walk between task locations registered by LIDAR and stereo vision in the DRC Finals.

- We have developed a state estimator to estimate the Center of Mass states by fusing inertial, force and kinematic information. The modeling error is estimated in real time such that very little tuning is required for our mass model. The modeling error is compensated as an external force by the full-body controller. The estimator also computes the Corrected Capture Point (CCP) used in fall detection and prevention. In the DRC Finals, potential falls were detected and prevented in two different occasions. Thanks to this estimator, we were the only team that tried all tasks, did not require physical human intervention (a reset), and did not fall in the two missions during the two days of tests.

7.2 Future Research Directions

There are still much work to be done in the field of state estimation for humanoids. Several directions of potential future research are listed:

- **Early fall predictions.**
  We think we can predict falls earlier using a variety of predictors. In manipulation, we tracked the CCP. In walking the CCP moves around quite a bit. Its deviation from its expected motion can be used to predict falling. Vertical foot forces (F_z) too high or not high enough, and other foot forces and torques being large are warning signs of large external forces. Joint torque sensors can be used to attempt to locate a single external force to either a hand (manipulation), a foot (tripping), or another part of the body (collision). Robot skin-based sensing will be extremely useful as part of an early warning system. Horizontal foot forces (F_x , F_y), yaw torque (torque around a vertical axis: M_z) going to zero, foot motion due to deflection of the sole or small slips measured using optical sensors, and vibration measured in force/torque sensors or IMUs in the foot are early warning signs of possible slipping. The center

of pressure going to a foot edge and foot tipping measured by a foot IMU are early warning signs of individual foot tipping and resultant slipping or possible collapse of support due to ankle torque saturation. Any part of the body such as the foot or hand having a large tracking error is a useful trigger for freezing the robot and operator intervention.

- **More sensing.**

  Sensing is cheap. We strongly believe that humanoids should be outfitted with as much sensing as possible. The absence of horizontal force and yaw torque sensing in the Atlas feet limited our ability to avoid foot slip, reduce the risk of falling, and optimize gait using learning. We commissioned Optoforce to build true six axis foot force/torque sensors for the Atlas feet, but they were not ready in time for the DRC. We will explore how much they improve performance in future work. Redundant sensor systems make calibration and detection of hardware failure much easier.

- **Super-human sensing is useful.**

  Super-human sensing (whole body vision systems, for example) is a useful research area which could greatly improve humanoid robustness and performance. We used vision systems on the wrists to guide manipulation and on the knees to guide locomotion. Time prevented us from implementing planned vision systems located on the feet. We plan to build super-human feet with cameras viewing in all directions and IMUs (accelerometers and gyros) to measure foot translational acceleration (accelerometers), linear and angular velocity (optical flow and gyros), and track translation and orientation (IMU orientation tracking and image matching). Longer term, we intend to build robust high resolution tactile sensing for the soles of the feet, as well as similar robust high resolution sensing skin. We intend to build many types of optical sensing into the robot's skin, to do obstacle and collision detection at a variety of distances and resolutions.

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*. IEEE, 2014, pp. 195–201.

[2] J. Rawlings and D. Mayne, *Model predictive control: theory and design*. Nob Hill Publishing, 2013.

[3] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[4] W. Sun and Y.-X. Yuan, *Optimization theory and methods: nonlinear programming*. springer, 2006, vol. 98.

[5] A. P. Ruszczyński, *Nonlinear optimization*. Princeton university press, 2006, vol. 13.

[6] R. E. Kalman and Others, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[7] D. Simon, *Optimal state estimation: Kalman, H [infinity] and nonlinear approaches*. John Wiley and Sons, 2006.

[8] J. Bellantoni and K. Dodge, "A square root formulation of the kalman-schmidt filter." *AIAA journal*, vol. 5, no. 7, pp. 1309–1314, 1967.

[9] P. Dyer and S. McReynolds, "Extension of square-root filtering to include process noise," *Journal of Optimization Theory and Applications*, vol. 3, no. 6, pp. 444–458, 1969.

[10] A. Andrews, "A square root formulation of the kalman covariance equations." *AIAA Journal*, vol. 6, no. 6, pp. 1165–1166, 1968.

[11] P. Kaminski, A. E. Bryson, and S. Schmidt, "Discrete square root filtering: A survey of current techniques," *Automatic Control, IEEE Transactions on*, vol. 16, no. 6, pp. 727–736, 1971.

[12] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*. Academic Press, 1977.

[13] H. Michalska and D. Q. Mayne, "Moving horizon observers and observer-based control," *Automatic Control, IEEE Transactions on*, vol. 40, no. 6, pp. 995–1006, 1995.

[14] D. G. Robertson, J. H. Lee, and J. B. Rawlings, "A moving horizon-based approach for least-squares estimation," *AIChE Journal*, vol. 42, no. 8, pp. 2209–2224, 1996.

[15] K. Muske, J. Rawlings, and J. H. Lee, "Receding horizon recursive state estimation," in *American Control Conference, 1993*, June 1993, pp. 900–904.

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

[17] D. Simon and T. L. Chia, "Kalman filtering with state equality constraints," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 38, no. 1, pp. 128–136, 2002.

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

[19] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Dover Publications, 2007.

[20] P. S. Maybeck, *Stochastic models, estimation, and control*. Academic press, 1982, vol. 2.

[21] M. Athans, R. P. Wishner, and A. Bertolini, "Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements," *Automatic Control, IEEE Transactions on*, vol. 13, no. 5, pp. 504–514, 1968.

[22] A. Gelb, *Applied optimal estimation*. MIT press, 1974.

[23] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference, Proceedings of the 1995*, vol. 3. IEEE, 1995, pp. 1628–1632.

[24] 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.

[25] ——, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[26] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE, 2000, pp. 153–158.

[27] R. Van Der Merwe, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," Ph.D. dissertation, University of Stellenbosch, 2004.

[28] S. J. Julier and J. K. Uhlmann, "Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 2. IEEE, 2002, pp. 887–892.

[29] S. J. Julier, "The spherical simplex unscented transformation," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3. IEEE, 2003, pp. 2430–2434.

[30] D. Tenne and T. Singh, "The higher order unscented filter," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3. IEEE, 2003, pp. 2441–2446.

[31] S. Kolås, B. Foss, and T. Schei, "Constrained nonlinear state estimation based on the ukf approach," *Computers & Chemical Engineering*, vol. 33, no. 8, pp. 1386–1401, 2009.

[32] P. Vachhani, S. Narasimhan, and R. Rengaswamy, "Robust and reliable estimation via unscented recursive nonlinear dynamic data reconciliation," *Journal of process control*, vol. 16, no. 10, pp. 1075–1086, 2006.

[33] I. Tjoa and L. Biegler, "Simultaneous strategies for data reconciliation and gross error detection of nonlinear systems," *Computers & chemical engineering*, vol. 15, no. 10, pp. 679–690, 1991.

[34] I.-W. Kim, M. Liebman, and T. Edgar, "A sequential error-in-variables method for nonlinear dynamic systems," *Computers & chemical engineering*, vol. 15, no. 9, pp. 663–670, 1991.

[35] B. W. Bequette, "Nonlinear predictive control using multi-rate sampling," *The Canadian Journal of Chemical Engineering*, vol. 69, no. 1, pp. 136–143, 1991.

[36] J. S. Albuquerque and L. T. Biegler, "Data reconciliation and gross-error detection for dynamic systems," *AIChE Journal*, vol. 42, no. 10, pp. 2841–2856, 1996.

[37] T. Binder, L. Blank, W. Dahmen, and W. Marquardt, "On the regularization of dynamic data reconciliation problems," *Journal of Process Control*, vol. 12, no. 4, pp. 557–567, 2002.

[38] M. A. Brubaker, D. J. Fleet, and A. Hertzmann, "Physics-based person tracking using simplified lower-body dynamics," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on.* IEEE, 2007, pp. 1–8.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] ——, "Absolute orientation estimation for observer-based control of a five-link walking biped robot," in *Robot Motion and Control*. Springer, 2006, pp. 181–199.

[46] V. Lebastard, Y. Aoustin, F. Plestan, and L. Fridman, "An alternative to the measurement of five-links biped robot absolute orientation: estimation based on high order sliding mode," in *Modern Sliding Mode Control Theory*. Springer, 2008, pp. 363–380.

[47] 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.

[48] O. Gür and U. Saranlı, "Model-based proprioceptive state estimation for spring-mass running," *Robotics: Science and Systems VII*, p. 105, 2012.

[49] 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.

[50] 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.

[51] P. D. Hanlon and P. S. Maybeck, "Multiple-model adaptive estimation using a residual correlation kalman filter bank," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 36, no. 2, pp. 393–406, 2000.

[52] 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.

[53] W. Wen and H. Durrant-Whyte, "Model-based multi-sensor data fusion," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 1720–1726.

[54] L. Wang, Y.-T. Chiang, and F. Chang, "Filtering method for nonlinear systems with constraints," in *Control Theory and Applications, IEEE Proceedings-*, vol. 149. IET, 2002, pp. 525–531.

[55] A. Alouani and W. Blair, "Use of a kinematic constraint in tracking constant speed, maneuvering targets," *Automatic Control, IEEE Transactions on*, vol. 38, no. 7, pp. 1107–1111, 1993.

[56] J. Porrill, "Optimal combination and constraints for geometrical sensor data," *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 66–77, 1988.

[57] V. Sircoulomb, G. Hoblos, H. Chafouk, and J. Ragot, "State estimation under nonlinear state inequality constraints. a tracking application," in *Control and Automation, 2008 16th Mediterranean Conference on*. IEEE, 2008, pp. 1669–1674.

[58] S. Ko and R. R. Bitmead, "State estimation for linear systems with state equality constraints," *Automatica*, vol. 43, no. 8, pp. 1363–1368, 2007.

[59] J. De Geeter, H. Van Brussel, J. De Schutter, and M. Decréton, "A smoothly constrained kalman filter," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 10, pp. 1171–1177, 1997.

[60] C. Chevallereau, A. Gabriel, Y. Aoustin, F. Plestan, E. Westervelt, C. C. De Wit, J. Grizzle, *et al.*, "Rabbit: A testbed for advanced control theory," *IEEE Control Systems Magazine*, vol. 23, no. 5, pp. 57–79, 2003.

[61] 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, 1987.

[62] 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.

[63] K. Kaneko, F. Kanehiro, S. Kajita, M. Morisawa, K. Fujiwara, K. Harada, and H. Hirukawa, "Slip observer for walking on a low friction floor," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 634–640.

[64] N. Okita and H. Sommer, "A novel foot slip detection algorithm using unscented kalman filter innovation," in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 5163–5168.

[65] 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.

[66] 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.

[67] W. Huang, X. Xinjilefu, S. Feng, and C. Atkeson, "Autonomous Path Planning of a Humanoid Robot on Unknown Rough Terrain," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, submitted.

[68] M. F. Fallon, M. Antone, N. Roy, and S. Teller, "Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 112–119.

[69] A. Stelzer, H. Hirschmüller, and M. Görner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 381–402, 2012.

[70] J.-P. Tardif, M. George, M. Laverne, A. Kelly, and A. Stentz, "A new approach to vision-aided inertial navigation," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4161–4168.

[71] S. Roumeliotis, J. W. Burdick, *et al.*, "Stochastic cloning: A generalized framework for processing relative state measurements," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1788–1795.

[72] A. Gopalakrishnan, N. S. Kaisare, and S. Narasimhan, "Incorporating delayed and infrequent measurements in extended kalman filter based nonlinear state estimation," *Journal of Process Control*, vol. 21, no. 1, pp. 119–129, 2011.

[73] X. Xinjilefu, S. Feng, and C. Atkeson, "Dynamic state estimation using quadratic programming," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 989–994.

[74] 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.

[75] 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.

[76] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 120–127.

[77] [Online]. Available: http://www.vectornav.com/support/library/calibration

[78] S. O. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Report x-io and University of Bristol (UK)*, 2010.

[79] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *Automatic Control, IEEE Transactions on*, vol. 53, no. 5, pp. 1203–1218, 2008.

[80] C. Gramkow, "On averaging rotations," *Journal of Mathematical Imaging and Vision*, vol. 15, no. 1-2, pp. 7–16, 2001.

[81] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, pp. 922–923, 1976.

[82] ——, "A discussion of the solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 34, no. 5, pp. 827–828, 1978.

[83] S. Feng, X. Xinjilefu, C. Atkeson, and J. Kim, "Optimization based controller design and implementation for the Atlas robot in the DARPA Robotics Challenge Finals," in *Humanoid Robots (Humanoids), 2015 15th IEEE-RAS International Conference on*. IEEE, 2015, p. submitted.

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

[85] K. Fujiwara, F. Kanehiro, S. Kajita, K. Kaneko, K. Yokoi, and H. Hirukawa, "Ukemi: falling motion control to minimize damage to biped humanoid robot." in *IROS*, 2002, pp. 2521–2526.

[86] K. Fujiwara, F. Kanehiro, S. Kajita, K. Yokoi, H. Saito, K. Harada, K. Kaneko, and H. Hirukawa, "The first human-size humanoid that can fall over safely and stand-up again," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2003, pp. 1920–1926.

[87] K. Fujiwara, F. Kanehiro, H. Saito, S. Kajita, K. Harada, and H. Hirukawa, "Falling motion control of a humanoid robot trained by virtual supplementary tests." in *IEEE International Conference on Robotics and Automation*, vol. 2. IEEE; 1999, 2004, pp. 1077–1082.

[88] K. Fujiwara, F. Kanehiro, S. Kajita, and H. Hirukawa, "Safe knee landing of a human-size humanoid robot while falling forward," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2004, pp. 503–508.

[89] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, and H. Hirukawa, "Towards an optimal falling motion for a humanoid robot," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 524–529.

[90] ——, "An optimal planning of falling motions of a humanoid robot," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 456–462.

[91] R. Renner and S. Behnke, "Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes," in *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2967–2973.

[92] J. Ruiz-del Solar, J. Moya, and I. Parra-Tsunekawa, "Fall detection and management in biped humanoid robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3323–3328.

[93] K. Ogata, K. Terada, and Y. Kuniyoshi, "Falling motion control for humanoid robots while walking," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*. IEEE, 2007, pp. 306–311.

[94] ——, "Real-time selection and generation of fall damage reduction actions for humanoid robots," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008.

[95] O. Höhn and W. Gerth, "Probabilistic balance monitoring for bipedal robots," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 245–256, 2009.

[96] S.-k. Yun, A. Goswami, and Y. Sakagami, "Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 781–787.

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

[98] O. E. Ramos, N. Mansard, and P. Soueres, "Whole-body motion integrating the capture point in the operational space inverse dynamics control," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 707–712.

[99] A. M. Andrew, "Another efficient algorithm for convex hulls in two dimensions," *Information Processing Letters*, vol. 9, no. 5, pp. 216–219, 1979.