Online Hierarchical Optimization for Humanoid Control

Siyuan Feng

CMU-RI-TR-16-03

February 2016

The Robotics Institute

Carnegie Mellon University

Pittsburgh, PA 15213

Thesis Committee:

Christopher G. Atkeson, Chair

Hartmut Geyer

Koushil Sreenath

Jerry Pratt

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy in Robotics

Abstract

This thesis presents an online approach for controlling humanoid robots using hierarchical optimization. While our primary focus is to develop a fast and robust walking controller that is able to follow desired foot steps, full body manipulation capability is also achieved. The proposed hierarchical system consists of three levels:

- a high level trajectory optimizer that generates nominal center of mass and swing foot trajectories, together with useful information such as a local value function approximation and a linear policy along the nominal trajectory;
- a middle level receding-horizon controller that tracks the nominal plan and handles large disturbances by rapidly replanning for a short horizon;
- a low level controller that computes joint level commands by solving full body inverse dynamics and kinematics using quadratic programming.

Using just the high level and the low level controller, we achieved rough terrain walking and close to human walking speed and stride length in simulation. Walking and manipulation controllers were also developed for the Atlas robot based on the same architecture, and performed reliably during the DARPA Robotics Challenge. The full hierarchy with the middle level controller is implemented afterwards, and dynamic walking with strong perturbations is successfully demonstrated on the Atlas robot.

Acknowledgments

I feel fortunate to be Chris Atkeson's student, and I am very grateful for all his support and guidance during my study. Chris is a great advisor in the sense that he does not "advise" much. Although I did suffer initially, I very much enjoy the independence during the later years. He can always say something intelligent (although not always correct) about even the hardest problems. To learn from him and eventually see through his "lies" has always been a great pleasure. Personally, Chris was the greatest moral support and motivator during the DRC, and he kept the stress at a manageable level. I am also very grateful for my committee members: Jerry Pratt, Hartmut Geyer and Koushil Sreenath, who have been continuously providing me valuable inputs during my study.

The DRC was an exceptional experience for me not only because I had a chance to play with one of the most advanced humanoid robots, but more importantly, I made many friends. Among all the people, I would like to thank Xinjilefu first for all the discussions and long hours we spent together. Also because without his excellent work, none of this would have happened on the robot. I am grateful for every member of Team WPI-CMU, especially Matt DeDonato and Felipe Polido for their support during the DRC. Last but not least, I am indebted to Jerry Pratt and his gang at IHMC for all the selfless help and communications.

I want to thank my parents, Jingruo Feng and Liya Wu, and my significant other, Fan Zhang, for their unconditional support and letting me do what I enjoy the most. Finally, big thanks to WARNER, our beloved walking metal monster.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humanoid robots have been promoted for being able to traverse rough terrain, suitable for human-centric environments, and facilitating human robot interactions. Despite increasing interests and attention in academia and popular culture over the past decade, especially with the recent DARPA Robotics Challenge (DRC) focusing on disaster response, state-of-the-art humanoid robots are still not ready for field deployment. In this thesis, we explore potential solutions to these control challenges that block humanoid robots from becoming a truly useful mobile platform: low reliability and tendency to fall, slow and lack of compliance. Due to the large number of degrees of freedom, humanoid robots typically have a bigger workspace with a smaller footprint comparing to conventional platforms. However, this complexity makes planning and control harder for humanoids. Unlike statically stable platforms, maintaining balance is a much harder problem for humanoids, and dynamics matter for planning as well. When operating in a complex and dynamic environment, such as any typical human occupied environment, the robot needs to replan using sensor feedback within a reasonable time window. It also needs to be fast and agile. Speed is important for usefulness as well as for certain recovery behaviors. With compliant behaviors, the robot can deal with unexpected perturbations better, and is safer for human robot interaction.

Originally targeted at rough terrain bipedal walking, we first developed a walking controller

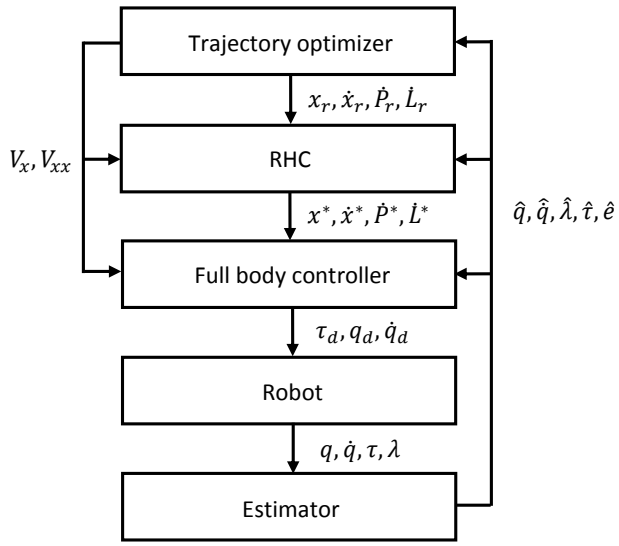


Figure 1.1: Subscript $_r$ indicates nominal values optimized by the trajectory optimizer, and $_d$ denotes commands generated by the full body controller. P and L stand for system linear and angular momentum, and e stands for modeling errors. The value function approximation, denoted by V_x and V_{xx}, is generated by the trajectory optimizer, and it guides the lower level controllers. x and q stand for Cartesian and joint space positions, τ stands for joint torques, and λ stands for all the contact wrenches.

that can follow a sequence of foot steps and walk on level ground at close to human speed and step length. The approach is rooted in model-based optimal control. A center of mass (CoM) trajectory that follows the foot steps is first optimized, which is then tracked using inverse dynamics formulated as a convex optimization problem. Trajectory optimization uses a simplified model that only reasons about the CoM to speed up computation, and the inverse dynamics modules manages the details about physics and constraints but only for one time step. This architecture separates the behavior level design process and full body control problem. It offers a versatile and powerful platform for rapidly developing behaviors especially in the context of the DRC, where we were required to solve a wide range of mobility and manipulation tasks in a very limited time frame. After the DRC, we implemented a third level that uses the high level plan as a guide to solve a short trajectory optimization problem in a receding-horizon fashion. By rapidly replanning using sensor feedback based on the original plan, it adds robustness to the overall controller.

1.1 Overview of Walking Systems

Generally speaking, existing walking controllers can be categorized as either offline or online solutions. The former usually has a few parameters that can be altered during execution, or a selection of discrete behaviors to choose from. It is often nontrivial to achieve the desired behaviors by manipulating these parameters. Online approaches are usually more flexible on the behavior level, since they are designed to recompute solutions based on different inputs. Building the policies (behaviors) by hand typically requires a great deal of insight into the specific situation and much trial and error for tuning. For simple systems, very robust and dynamic behaviors have been achieved with this approach on real hardware [71]. Inspired by limit cycle walking and running research [12, 21, 52], controllers [6, 24, 102] have been developed. Simple policy based controllers have also been applied to animated figures [96, 105] in the graphics community. Once designed, policy optimization [81, 96] or online learning [42, 48, 57] can be used for tuning. Optimization is another powerful tool for generating reference trajectories and controllers offline. Dynamic programming [5] generates a globally optimal policy for a large region of the state space, but it suffers from the "curse of dimensionality". Relaxations [3, 4, 99] can be made to enable applications to larger problems. Nominal walking patterns can be found with trajectory optimization [13, 55, 69]. Feedback controllers are then used to stabilize the system around the trajectories. Trajectories can be combined into a library [50, 101] to cover larger regions of the state space. Many of these offline solutions are impressively capable and robust at what they are designed for, but they typically have limited abilities in terms of adaptation. Most of them are incompatible with achieving desired foot steps, which limits their application for more general purposes.

A typical setup for a complete online walking solution is presented in [107], where a higher level foot step planner such as [11, 28] generates a plan using perception, and the walking controller follows it to the best of its ability. Directly planning walking motions with the full model in an online setting is unrealistic due to model complexity. A more practical approach is to plan

using simple models. For example, a center of mass trajectory is first planned using a point mass model, and full body motions are then computed with inverse kinematics. Some of the most successful humanoids such as Honda's Asimo [26], the HRP series [2, 31, 36, 37, 91], and HUBO [67] use this method for walking. This approach requires accurate joint level motion tracking, and is often achieved through stiff position control, which is more vulnerable to external perturbations and impacts. On the other hand, by taking recovery steps, they are capable of robust dynamic walking even with strong perturbations [92, 93]. Rapid advancement in inverse dynamics algorithms [8, 14, 19, 25, 29, 30, 39, 44, 47, 49, 66, 72, 74, 78] and hardware development of force controlled humanoids such as PETMAN and Atlas built by Boston Dynamics and many other research platforms [10, 40, 65, 77, 89, 90] open up a viable alternative for full body motion generation. These new platforms are attractive for inverse dynamics approaches because they can all be force controlled, and some of them have built-in compliance. The inverse dynamics algorithms are typically formulated as one step constrained convex optimization problems using the full dynamic model. They track planned trajectories and enforce constraints on ground reaction forces and actuation at the same time.

1.2 Thesis Contribution

The core of this thesis is dividing a hard planning and control problem into different levels that consider progressively shorter time horizons but use more complete dynamics and constraints so that the overall problem can be solved online. Using the proposed scheme, we have achieved human-like walking speed and stride length in simulation. We are also able to achieve robust dynamic walking on rough terrain or with strong perturbations on a real humanoid robot. A variety of mobility and manipulation tasks were successfully demonstrated throughout the entire DARPA Robotics Challenge. The main contribution is to synthesize existing concepts of trajectory optimization, receding-horizon control and full body inverse dynamics and kinematics into one versatile system and its implementation on real hardware solving real life tasks. Other

contributions include lessons learned and practical solutions to address modeling errors on the real robot.

1.3 Hardware Overview

We use the Atlas humanoid robot built by Boston Dynamics for all hardware experiments, and a large amount of experimental results are shown in the context of the DARPA Robotics Challenge (DRC), which was a DARPA sponsored program aimed at developing semi-autonomous robots for disaster response. Roughly six months prior to the DRC Finals, our Atlas robot went through a major upgrade to enable tetherless operation. Shoulder joints were also reconfigured to increase arm manipulability. Most of the experimental data in this thesis were collected after this upgrade. Four months later, the robot went through another upgrade to replace the last two joints on the forearms with three electric joints. This gave Atlas seven degrees of freedom arms for much better manipulability. On the other hand, because of the tight testing timeline, these electric forearms suffered greatly from hardware reliability issues. This upgrade brings the total number of actuators from 28 to 30: six for each leg, seven for each arm, three for the spine, and one for the neck.

For all the hydraulic joints, position and torque are measured pre-transmission on the actuator side. The back roll and pitch joints and all the leg joints are linear hydraulic actuators. For these joints, position is measured with Linear Variable Differential Transformers (LVDT), and force is measured with piston pressure sensors. Transmission information is used to compute joint position and torque. Joint velocity is generated by low-pass filtering finite differences of joint position. The arm joint position can be measured with encoders after transmission. Velocity and torque signals are generated similarly to the legs. For each foot, there is a 3-axis force torque sensor measuring the vertical force and roll and pitch torque. Two 6-axis force torque sensors are mounted at the wrists. A sensor head that includes a pair of stereo cameras and a spinning Hokuyo laser range finder is attached to the upper body through a single axis (pitch) neck joint.

A high precision 6-axis IMU is attached to the pelvis link, providing linear acceleration, angular velocity and orientation measurements. Most of the joint sensors do not directly measure joint position or torque, which makes accurate forward kinematics and torque tracking difficult.

For the hydraulic actuators, the joint level servo computes valve command i based on

$$i = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + K_f(\tau_d - \tau) + c, \tag{1.1}$$

where $q_d, \dot{q}_d, \tau_d$ are desired joint, velocity and torque values, $q, \dot{q}, \tau$ are the measured values, and c contains other auxiliary feedback and feedforward terms and a constant valve bias term. This servo runs at $1kHz$, and we can update all the gains and desired values at the same rate.

1.4 Outline

We focus on high level trajectory optimization in Chapter 2, which generates a nominal CoM trajectory and other useful information that guides the lower level controllers. The current implementation of the low level full body controller is presented in Chapter 3. This full body controller is responsible for generating joint level control signals that best track the nominal plan while managing all the physical constraints. Our walking and manipulation controllers for the DARPA Robotics Challenge are implemented based on these two modules, and experimental results are presented in Chapter 4. Chapter 5 describes recent work on developing a middle level component that replans rapidly over a short horizon in a receding-horizon fashion. Robot and simulation results for a foot placement controller and an angular momentum controller are also shown. Directions for future work and conclusion are presented in Chapter 6.

Chapter 2

Center of Mass Trajectory Generation for Walking

2.1 Simple Models for Online Planning

It is computationally prohibitive to use full dynamics models for planning walking motions online. The common approach is to plan with simplified models that approximate the overall dynamics, and reconstruct full body motion afterwards. The Linear Inverted Pendulum Model (LIPM) [33, 34] was introduced for this purpose. LIPM combined with Zero Moment Point (ZMP) [94] have been widely used in center of mass (CoM) motion generation. Preview Control [35] is one of the most successful applications. Capture point [70] can also be generalized to generate walking patterns [45, 46]. Similarly, divergent component of motion [27, 83] is introduced to encode the unstable part of the LIPM dynamics and used for walking pattern generation. For dynamic behaviors, especially balancing with limited foot placement, angular momentum plays an important role. LIPM was then extended to include angular momentum [43, 70], which is generated by applying a torque around the CoM. These simple linear models are useful for analytical solutions and fast computation, particularly when used in receding-horizon controllers [15, 18, 22, 62, 93, 100]. [64] connected both linear and angular momentum, generalized ve-

locity of the system and net external wrench by introducing the centroidal momentum matrix, which influenced many later approaches for balancing and walking.

Our high level controller for dynamic walking is similar in spirit to Preview Control in the sense that we use a CoM model, reason about ZMP, and use future information to guide the current trajectory. On the other hand, our formulation can be easily generalized to nonlinear models as opposed to LIPM used in Preview Control. Given a sequence of desired foot steps, we plan a CoM trajectory that minimizes stance foot ankle torque and deviation from the desired states with trajectory optimization. Currently, we approximate the entire robot as a point mass, without considering angular momentum or effects of the swing leg. These can be reintroduced without changing the rest of the algorithm, but computation will take longer. We explicitly add the vertical dimension into our CoM model to capture terrain heigh changes on rough terrain. The dynamics of this simple model are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \frac{(x-p_x)F_z}{mz} \\ \frac{(y-p_y)F_z}{mz} \\ \frac{F_z}{m} - g \end{bmatrix}. \tag{2.1}$$

The state, $X = (x, y, z, \dot{x}, \dot{y}, \dot{z})$, is the position and velocity of the CoM. The control $u = (p_x, p_y, F_z)$ is the commanded center of pressure (CoP) and force in the z direction. The current high level controller is not aware of step length limits, and we are relying on the foot step planner to produce a reasonable foot step sequence.

2.2 Differential Dynamic Programming

Differential Dynamic Programming [32] is an iterative trajectory optimization technique that has two passes for each iteration:

- The **backward pass** updates the current control signals based on the spatial derivatives of the value function.
- The **forward pass** uses the updated controls to generate a trajectory for the next iteration.

It can find globally optimal trajectories for problems with linear dynamics and quadratic costs, and rapidly converge to locally optimal trajectories for problems with nonlinear dynamics or nonquadratic costs. The Linear Quadratic Regulator (LQR) can be treated as a special case of this algorithm. DDP can also be extended to handle stochastic systems [87, 88]. It is also possible to use DDP in a receding-horizon fashion with the full dynamics [16, 84], but it is typically slower than real-time. Combining coordinated locally optimized trajectories as an approximation of the globally optimal solution is shown in [3]. Instead of optimality, LQR-Tree [85, 86] uses DDP-like trajectories to cover the state space and achieve asymptotic stability.

This approach modifies (and complements) existing approximate Dynamic Programming approaches in these ways:

- We approximate the value function and policy using many local models (quadratic for the value function, linear for the policy) along the trajectory.
- We use trajectory optimization to directly optimize the sequence of commands and states.
- Refined local models of the value function and policy are created as a byproduct of the trajectory optimization process.

We represent value functions and policies using Taylor series approximations at each time step along a trajectory. For a state X^t, the local quadratic model for the value function is

$$V^t(X) \approx V_0^t + V_X^t(X - X^t) + \frac{1}{2}(X - X^t)^T V_{XX}^t(X - X^t), \tag{2.2}$$

where t is the time index, X is some query state, V_0^t is the constant term, V_X^t is the first order gradient of the value function with respect to the state evaluated at X^t, and V_{XX}^t is the second

order spatial gradient evaluated at X^t. The local linear policy is

$$u^t(X) = u_0^t - K^t(X - X^t), \tag{2.3}$$

where u_0^t is a constant term, and K^t is the first derivative of the local policy with respect to state evaluated at X^t, and is also the gain matrix for a local linear controller. V_0^t, V_X^t, V_{XX}^t and K^t are stored along with the trajectory.

The one step cost function is

$$L(X, u) = 0.5(X - X^*)^T Q(X - X^*) + 0.5(u - u^*)^T R(u - u^*), \tag{2.4}$$

where R is positive definite, and Q is positive semi-definite. X^* and u^* are the desired state and control trajectories. The simplest version of X^* is a square wave that instantly switches between the desired foot steps and remains stationary for the entire stance phase. u^* is specified in a similar way for p_x^* and p_y^*, and $F_z^* = mg$.

$$Q = \begin{bmatrix} 1 \times 10^{-4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 \times 10^{-4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \times 10^{-2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \times 10^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \times 10^{-2} \end{bmatrix} \tag{2.5}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \times 10^{-6} \end{bmatrix}$$

For each iteration of DDP, we propagate the spatial derivatives of the value function V_{XX}^t and V_X^t backward in time, and use this information to compute an update to the control signal. Then

we perform a forward integration pass using the updated controls to generate a new trajectory. Although we are performing nonlinear trajectory optimization, due to analytical gradients of the dynamics of the model we use, this process is fast enough in an online setting.

Initialization: Given the last desired center of mass location and desired center of pressure, (X^*, u^*) in the foot step sequence, we first compute a Linear Quadratic Regulator (LQR) solution, and use its policy to generate an initial trajectory from the initial state X_0. V_{XX} of this LQR is also used to initialize the backward pass.

Backward pass: Given a trajectory, one can integrate the value function and its first and second spatial derivatives backwards in time to compute an improved value function and policy. We utilize the "Q function" notation from reinforcement learning: $Q^t(X, u) = L^t(X, u) + V^{t+1}(f(X, u))$. The backward pass of DDP can be expressed as

$$\begin{aligned}
Q^t_X &= L^t_X + V^t_X f^t_X \\
Q^t_u &= L^t_u + V^t_X f^t_u \\
Q^t_{XX} &= L^t_{XX} + V^t_X f^t_{XX} + f^{tT}_X V^t_{XX} f^t_X \\
Q^t_{uX} &= L^t_{uX} + V^t_X f^t_{uX} + f^{tT}_u V^t_{XX} f^t_X \\
Q^t_{uu} &= L^t_{uu} + V^t_X f^t_{uu} + f^{tT}_u V^t_{XX} f^t_u \\
K^t &= (Q^t_{uu})^{-1} Q^t_{uX} \\
\delta u^t &= (Q^t_{uu})^{-1} Q^t_u \\
V^{t-1}_X &= Q^t_X - Q^t_u K^t \\
V^{t-1}_{XX} &= Q^t_{XX} - Q^t_{Xu} K^t.
\end{aligned} \tag{2.6}$$

Derivatives are taken with respect to the subscripts, and evaluated at (X, u).

Forward pass: Once we have computed the local linear feedback policy K^t and updates for controls δu^t, we integrate forward in time using

$$u^t_{new} = (u^t - \delta u^t) - K^t(X^t_{new} - X^t) \tag{2.7}$$

with $X^{t0}_{new} = X_0$. We terminate DDP when the cost-to-go at X_0 does not change significantly across iterations. This approach can be thought of as a generalized version of Preview Control. Figure 2.1 shows trajectories of the CoM generated with LQR policy and after DDP optimization. Another example is shown in Figure 2.2. In both cases, we set the desired CoP at the support stance foot, and it can instantaneously switch to the next stance foot. A smoother desired CoP trajectory can be specified to represent double support.

2.3 Summary

Center of mass motion is arguably the most important aspect of walking. Using our formulation, a globally optimal CoM trajectory can be generated with linear models and quadratic cost functions, and a locally optimal solution can be achieved for more complex problems. In addition to a nominal trajectory, our approach also generates a quadratic approximation of the value function and a linear policy along it. The value function approximation encapsulates information about the future, and can provide useful guidance for the lower level controllers that have much shorter planning horizons in Chapter 3 and Chapter 5.

The policy produced by DDP is not suitable for handling large disturbances. When the state is far from the planned trajectory, DDP's policy can generate large invalid controls. Although it is possible for DDP to handle unilateral constraints [32], we have not implemented ground reaction force constraints. Instead, they are treated as high weight terms in the cost function during the optimization procedure. When using ankle torque alone is insufficient for balancing, the local policy performs poorly after saturating the control. The purpose of the high level controller is to

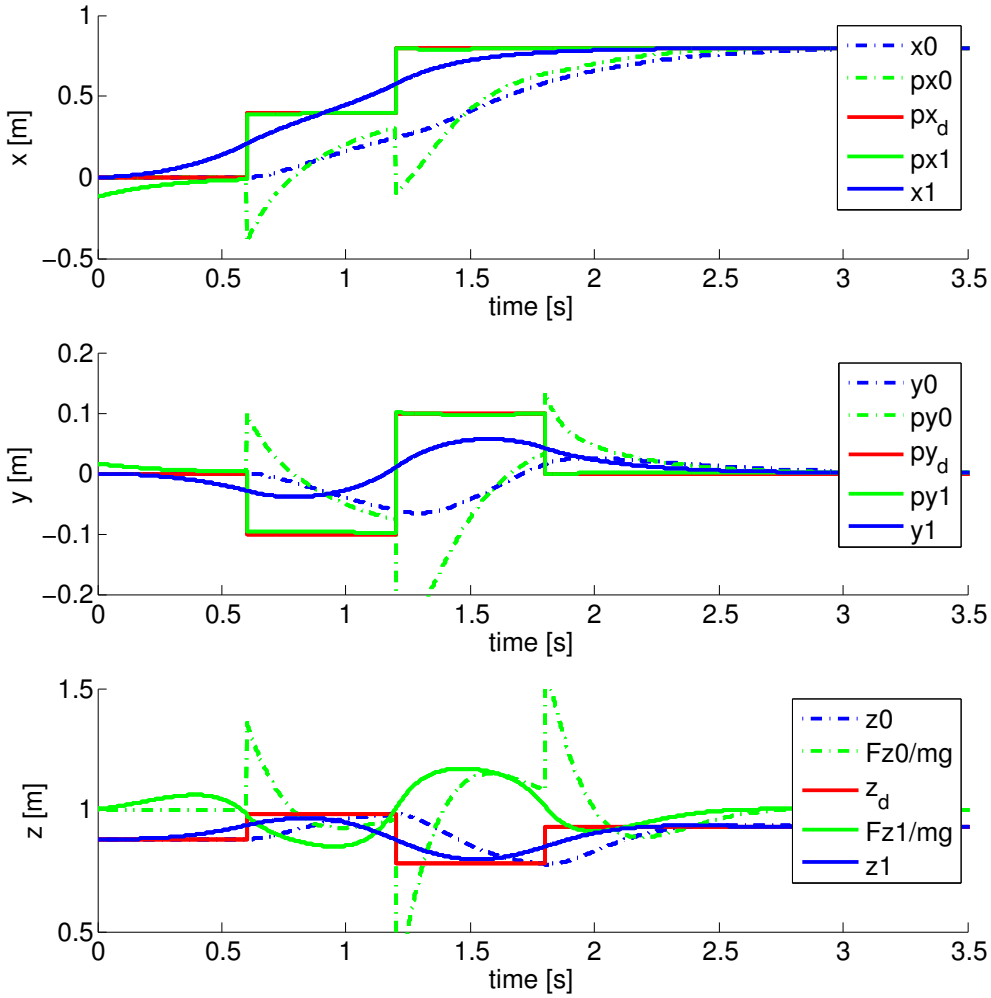
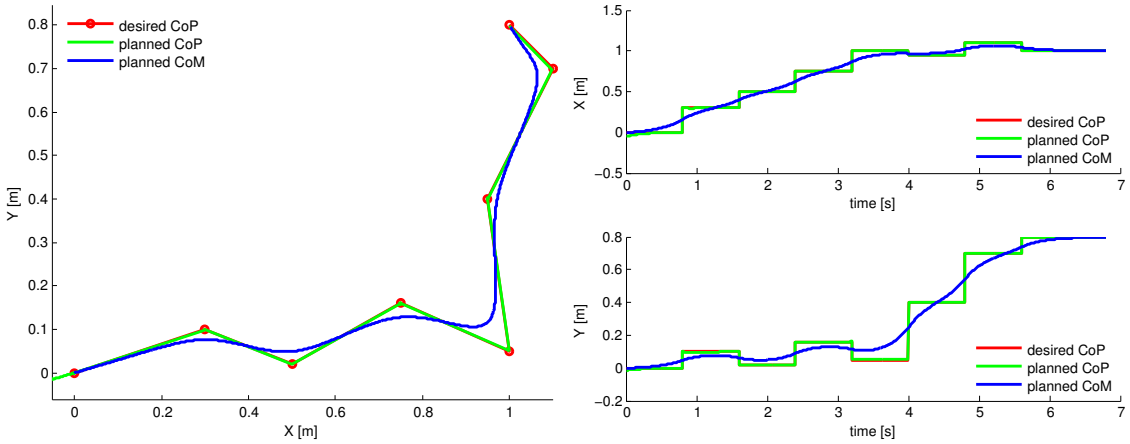


Figure 2.1: The desired CoP in the X and Y directions and CoM height are plotted with solid red lines from top to bottom, and are referred to as px_d, py_d, z_d in the legends. The desired CoP is at the middle of the stance foot, and the sharp changes are the contact switching events. Trajectories shown in dashed lines are generated by the LQR policy. The state trajectories are $x0, y0, z0$, and the control trajectories are $px0, py0, \frac{Fz0}{mg}$ respectively. These are used to initialize DDP. The optimization results are $x1, y1, z1$ and $px1, py1, \frac{Fz1}{mg}$.



(a) Planned CoM trajectory in the XY plane

(b) Planned trajectory against time

Figure 2.2: Figure 2.2(a) shows a top down view of the desired CoP, planned CoM and CoP trajectories in the XY plane. Figure 2.2(b) plots these traces against time.

generate a nominal CoM trajectory and useful information for the lower level controllers, which replan at a much higher rate, and are better at disturbance rejection. The high level controller replans per step. With this arrangement, we can afford to use more complex nonlinear models that better capture the overall dynamics.

The computation time scales roughly cubicly with respect to model dimensionality for each iteration. It is also important to mention that without analytical derivatives of the model dynamics and cost function, the process will be much slower due to numerical differentiation. This is one of the main reasons why this technique is used to optimize full body models offline [50] but is not applicable in an online setting. Although [16, 84] is a promising attempt at online trajectory optimization with the full dynamics, it is has not quite achieved real-time performance.

Chapter 3

Full Body Controller

3.1 Introduction

This chapter focuses on the full body controller, which uses quadratic programming to solve inverse dynamics and inverse kinematics. Inverse dynamics based approaches have gained popularity because they provide compliant motions and robustness to external perturbations such as impacts. On the other hand, using inverse dynamics alone performs poorly when facing inevitable modeling errors on real systems. So for robot implementation, it is still necessary to provide kinematic targets to generate accurate motions.

For the DARPA Robotics Challenge Trials, we implemented a low level controller using independent inverse kinematics (IK) and inverse dynamics (ID) modules [20], which was suitable for the quasi-static motions in the Trials, and was preferred for easy implementation. However, we were concerned about inconsistency between the two modules due to their different sets of constraints and gains. For the DRC Finals, two versions of the full body controllers are implemented. During walking, we no longer use IK for joint level kinematic targets. For the leg and spine joints, desired velocities are numerically integrated from the joint accelerations computed by ID. Desired arm motions are always specified in joint space during walking, so they are used directly in the joint level controllers. For manipulation, IK is used to generate the full

state that best tracks the desired Cartesian commands. ID is then used to track the IK's full state. Although IK does not enforce dynamic constraints, we argue that by properly positioning the CoM, running into a dynamic constraint is very unlikely during slow manipulation for a strong robot like Atlas. These design choices are motivated by the observation that the legs on Atlas are much better engineered than the arms, and the leg joints have less sensor noise and friction. For the arm joints, we were unable to use higher velocity gains and failed to achieve decent joint tracking without the position control loop. An explicit IK is also preferred as opposed to double integration for stability and accuracy reasons.

Using full body inverse dynamics for force control has become a popular topic recently. This direction of research originates from [39]. Within this broad category, control designers can directly specify reference motions in task space, then rely on convex optimization to handle constraints and solve for controls that best track the reference motions. Although detailed formulations differ, most active research have converged to formulating the floating base inverse dynamics as a quadratic programming (QP) problem. [14, 17, 25, 29, 30, 75, 97] explore using a hierarchical approach to resolve redundant degrees of freedom in humanoid robots. These approaches typically ensure low priority objectives are within the null space of higher priority ones. A generic solution to hierarchical quadratic programs is presented in [17] that enforces equality and inequality constraints at all hierarchies, and is significantly faster than previous methods [14, 38]. Although the method is currently applied to solve inverse kinematics, the authors claim it is also applicable to inverse dynamics. Another hierarchical framework designed for humanoid robots to handle constraints and objectives using virtual linkages is presented in [78, 79]. Contrary to these hierarchical approaches that have hard priorities, we prefer using weighted cost terms to specify preferences among the objectives. We gain numerical stability by sacrificing a small fraction of precision. There is also much interest in formulating a smaller optimization problem to reduce computation time. Contact forces can be removed from the equations of motion using orthogonal decomposition [54, 73, 74]. Balancing is demonstrated on a torque controlled humanoid in [66], where simple PD servos are used to generate a desired net

ground reaction wrench, which is then distributed among predefined contacts using optimization. Decoupled dynamics are used to speed up inverse dynamics calculations in [51, 72]. [49] has a two stage optimization setup. The first optimizes individual ground reaction forces and center of pressure (CoP) for each contact and the resulting admissible change in centroidal momenta. A second least square problem is then solved for state acceleration. Joint torques are generated explicitly afterwards. In [44, 46], desired centroidal momenta change is generated based on the instantaneous capture point, and accelerations and contact forces are simultaneously optimized. Joint torques are then generated with explicit inverse dynamics. [47] is similar in terms of optimization variables and torque generation, but a novel QP solver is implemented to exploit the observation that inequality constraints rarely change in this context. [106] applies QP based inverse dynamics to a quadruped robot on a slippery surface. Without using constrained optimization, a novel approach to generate full body torques with a combination of gravity compensation and task dependent attractors is proposed in [58]. We continue to use the formulation previously developed in our group [82, 98, 99] that is similar to [8, 9]. We directly optimize a quadratic cost in terms of state accelerations, torques and contact forces using the full robot model. This design choice gives us the most flexibility in terms of trading off directly among physical quantities of interest. It also allows us to easily add extra terms into the dynamics equations for compensating modeling errors.

Inverse kinematics with stiff joint position tracking is another traditionally popular approach to controlling humanoid robots. Similar to inverse dynamics, we also formulated the IK problem as a quadratic program, where the unknown is the generalized velocity $\dot{q}_{ik}$. At each time step, we solve for a set of $\dot{q}_{ik}$ that obeys kinematic constraints and minimizes a combination of costs. q_{ik} is computed by integrating $\dot{q}_{ik}$. Contacts are handled as soft constraints in our inverse kinematics QP. Our approach is an extension to the damped least squares method used by [60, 95]. In a similar work [53], the generalized velocity is computed by inverting a matrix composed of end effector and contact constraint Jacobian. Computing $\dot{q}_{ik}$ and integrating it to compute q_{ik} converges to local minima, but it produces continuous results. Another advantage for this gradient

based method is that it can be very reactive to changes in the desired motions since computation is spread out to many time steps.

3.2 Floating Base Position-Velocity State Estimator

The state estimator used in this work is based on [103, 104]. For a floating base humanoid, the base has three translational and three rotational degrees of freedom. We designate the pelvis as the base link. For the Atlas robot, there is a 6-axis IMU attached to the pelvis. The IMU measures angular velocity and linear acceleration, and also provides an estimate of the pelvis orientation in the world frame. We use the orientation estimate from the IMU without modification. The pelvis state estimator estimates the global pelvis position p_x and linear velocity p_v. It is a multiple model Kalman filter with contact switching. We design a steady state Kalman filter for each possible contact state in advance. The current contact state can be specified by the controller or estimated from vertical forces measured by the foot force torque sensors. The IMU has known position and orientation offsets relative to the pelvis origin. These offsets have already been taken into account, so the following equations are offset free.

The prediction step (process dynamics) of the pelvis Kalman filter is simply

$$x_k^- = \begin{bmatrix} p_{x,k}^- \\ p_{v,k}^- \end{bmatrix} = f(x_{k-1}^+) = \begin{bmatrix} p_{x,k-1}^+ + p_{v,k-1}^+ dt \\ p_{v,k-1}^+ + a_{k-1} dt \end{bmatrix}, \tag{3.1}$$

where x_k is the state estimate. The subscript k is the step index, and the superscript "$-$" and "$+$" represent before and after the measurement update. The net linear acceleration of the IMU a is transformed from the IMU frame to the world frame using the IMU orientation. It is straightfor-

ward to linearize the process dynamics to get the state transition matrix F_k

$$F_k = \begin{bmatrix} I & dtI \\ 0 & I \end{bmatrix}. \tag{3.2}$$

The measurement update step is slightly more complicated. There is no sensor directly measuring the position and velocity of the pelvis 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 space. 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 space, and the corresponding velocities be $\dot{c}_l$ and $\dot{c}_r$. In the double support phase (DS), we assume the feet are not moving to obtain the following measurements

$$z_{k,DS} = \begin{bmatrix} c_{l,k} \\ c_{r,k} \\ \dot{c}_{l,k} \\ \dot{c}_{r,k} \end{bmatrix} = \begin{bmatrix} c_{l,\eta_l} \\ c_{r,\eta_r} \\ 0 \\ 0 \end{bmatrix}. \tag{3.3}$$

The time index η is the time step when the foot is detected to be firmly on the ground using the force sensors.

To write the measurement equations, we need the prediction to be a function of pelvis position and velocity. We use the floating base forward kinematics $FK(\cdot)$:

$$y_{k,DS} = \begin{bmatrix} FK_{c_l}(q_k) \\ FK_{c_r}(q_k) \\ FK_{\dot{c}_l}(q_k, \dot{q}_k) \\ FK_{\dot{c}_r}(q_k, \dot{q}_k) \end{bmatrix} \tag{3.4}$$

The observation matrix H_k is computed by linearizing Eq. 3.4, and it turns out to be the identity.

$$H_{k,DS} = I_{12\times 12} \tag{3.5}$$

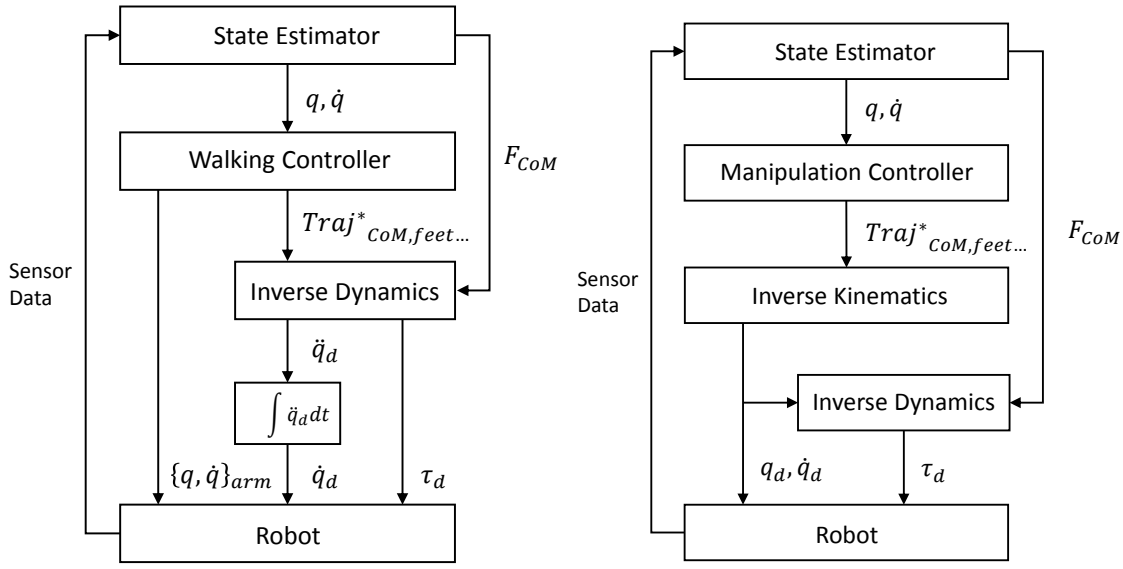
The stance foot is assumed to be fixed in the single support phase, so Eq. 3.3, Eq. 3.4, and Eq. 3.5 are modified to account for contact switching.

3.3 Full Body Controller as Quadratic Programs

For many tasks, we specify desired Cartesian motions for specific locations on the robot (e.g. foot, hand and CoM) in the high level controller. The full body controller takes these motions as inputs, and computes physical quantities for individual joints such as joint position, velocity, acceleration, and torque. Some of these outputs are directly used as targets in the joint level servos on the robot in Eq. 1.1. Figure 3.1 shows block diagrams for the walking and manipulation controllers. Both IK and ID are formulated as quadratic programming problems that share a general form of Eq. 3.6.

$$\begin{aligned} \min_{\mathcal{X}} \;& 0.5\mathcal{X}^T G\mathcal{X} + g^T\mathcal{X} \\ s.t.\;& C_E\mathcal{X} + c_E = 0 \\ & C_I\mathcal{X} + c_I \geq 0 \end{aligned} \tag{3.6}$$

The unknown $\mathcal{X}$ and constraints C_E, c_E, C_I and c_I are problem specific, and will be discussed in detail. Both QP problems are solved at each time step in a $2ms$ control loop with a standard solver. For both problems, we optimize a cost function of the form $0.5\|A\mathcal{X} - b\|^2$, and for Eq.



(a) Walking controller (b) Manipulation controller

Figure 3.1: For walking, in addition to tracking desired joint torques, the lower body joint level controllers have velocity control loops whose targets are the integrated accelerations optimized by inverse dynamics. The arm joints have position and velocity control loops, and the desired values come directly from the high level plan. For manipulation, a desired full state is first solved using inverse kinematics, which is then tracked by inverse dynamics.

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

$$A = \begin{bmatrix} w_0 A_0 \\ w_1 A_1 \\ \vdots \\ w_n A_n \end{bmatrix}, b = \begin{bmatrix} w_0 b_0 \\ w_1 b_1 \\ \vdots \\ w_n b_n \end{bmatrix}. \tag{3.7}$$

Through this cost function, we specify a set of desired behaviors according to the high level controller's goal, and penalize the robot's deviation from the desired behaviors. Each row in Eq. 3.7 emphasizes a certain desired behavior. w_i are weights that express the relative importance among often overly constrained and potentially conflicting goals. During implementation, finding reasonable weights is fairly straight forward, and takes fewer than one day of robot experiments.

3.4 Centroidal Momentum Matrix

Introduced in [63, 64], the centroidal momentum, which consists of the system's linear momentum P and angular momentum L, is linear with respect to the generalized velocity $\dot{q}$.

$$\begin{aligned} \begin{bmatrix} P \\ L \end{bmatrix} &= C\dot{q} \\ \begin{bmatrix} \dot{P} \\ \dot{L} \end{bmatrix} &= C\ddot{q} + \dot{C}\dot{q} \end{aligned} \tag{3.8}$$

C is the so called centroidal momentum matrix. C is $6 \times N$, where N is the degrees of freedom for the system. C is particularly useful for our inverse dynamics and inverse kinematics controller because it provides a linear relationship between the motion of the system centroid and the generalized state variables. Eq. 3.8 enables us to map costs on P, L, $\dot{P}$, and $\dot{L}$ to costs on $\dot{q}$ and $\ddot{q}$. Since $\dot{P}$ and $\dot{L}$ has to equal to the net external force and torque, C also relates net external wrench with $\ddot{q}$.

We can compute C with

$$C = \mathcal{P}\mathcal{J}, \tag{3.9}$$

where $\mathcal{P}$ is $6 \times 6M$, $\mathcal{J}$ is $6M \times N$, and M is the number of links.

$$\begin{aligned} \mathcal{P} &= \begin{bmatrix} T & 0_{3\times 3M} \\ U & V \end{bmatrix} \\ T &= \begin{bmatrix} m_1 I_{3\times 3} & m_2 I_{3\times 3} & \dots & m_M I_{3\times 3} \end{bmatrix} \\ U &= \begin{bmatrix} m_1 [r_1]_\times & m_2 [r_2]_\times & \dots & m_M [r_M]_\times \end{bmatrix} \\ V &= \begin{bmatrix} R_1 I_1 R_1^T & R_2 I_2 R_2^T & \dots & R_M I_M R_M^T \end{bmatrix} \end{aligned} \tag{3.10}$$

m_i is mass of the ith link, r_i is the vector from the system CoM to the ith link's center of mass,

and $[r_i]_\times$ is the cross product matrix such that

$$\begin{aligned} [r_i]_\times b &= r_i \times b \\ r_i &= x_i^{com} - x_{CoM}, \end{aligned} \tag{3.11}$$

where x_i^{com} and x_{CoM} are the center of mass for the ith link and the overall system. R_i is the rotation matrix representing the orientation of the ith link, and I_i is the moment of inertia matrix for the ith link in its body frame.

$$\mathcal{J} = \begin{bmatrix} J_1^p \\ J_2^p \\ \vdots \\ J_M^p \\ J_1^r \\ J_2^r \\ \vdots \\ J_M^r \end{bmatrix} \tag{3.12}$$

J_i^p and J_i^r are the position and rotation part of the Jacobian matrix for the ith link computed at its center of mass.

Since $P = \sum_i^M m_i \dot{x}_{CoM}$, we can define the Jacobian for CoM, J_{CoM}, as

$$J_{CoM} = \frac{1}{\sum_i^M m_i} C^p, \tag{3.13}$$

where C^p consists of the top three rows of C.

3.5 Inverse Dynamics

Let M be the inertia matrix, h be the sum of gravitational and Coriolis and centrifugal forces, τ stand for a vector of joint torques, J be the stacked Jacobian matrix for all the contacts, and λ be a vector of all contact wrenches in the world frame. As pointed out by Herzog, et al. [25], the equations of motion can be decoupled into two parts:

$$\begin{aligned} M_u\ddot{q} + h_u &= J_u^T\lambda \\ M_l\ddot{q} + h_l &= \tau + J_l^T\lambda, \end{aligned} \tag{3.14}$$

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

$$\tau = M_l\ddot{q} + h_l - J_l^T\lambda. \tag{3.15}$$

The top six rows of Eq. 3.14 can be rewritten as

$$\begin{bmatrix} M_u & -J_u^T \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \lambda \end{bmatrix} + h_u = 0. \tag{3.16}$$

Let $\mathcal{X} = \begin{bmatrix} \ddot{q} & \lambda \end{bmatrix}^T$. Given a state, the equations of motion are linear in terms of $\mathcal{X}$. In addition, Cartesian motions and centroidal momentum are also linear with respect to $\mathcal{X}$ given by

$$\ddot{x} = J\ddot{q} + \dot{J}\dot{q} \tag{3.17}$$

and Eq. 3.8. Thus, inverse dynamics can be formulated and solved efficiently as a quadratic program. We used to included τ as part of $\mathcal{X}$ as well, which incurred unnecessary computation. The current implementation replaces all occurrences of τ in the cost function and inequality constraints with Eq. 3.15. The cost function consist of mainly momentum and motion tracking

terms and various regularization terms. Eq. 3.16 is enforced as equality constraints, and the inequality constraints mostly deal with contact wrench and joint torque limits.

3.5.1 Structural Change Smoothing

In the previous implementation, the inverse dynamics QP changes dimension based on the number of contacts. Discrete changes can also happen due to constraint manifold changes, e.g. shrink foot support polygon to be at the toe during toe-off. Such changes cause large changes in the outputs that can induce undesired oscillations on the physical robot. These jumps are caused by different problem specifications, and cannot be smoothed by just adding cost terms that penalize changes. Our solution is to maintain the same QP dimension and gradually blend the constraints over a short period of time. We always assume the largest number of possible contacts, but heavily regularize the magnitude of the contact wrench and relax the acceleration constraints for the unused contacts. When a change is required in hard constraints, we gradually blend it using a low pass filter.

3.5.2 Cost Function

The cost function is essentially a weighted sum of many quadratic terms penalizing $\mathcal{X}$ for deviating from some desired $\mathcal{X}^*$. It can be rewritten in a block form as in Eq. 3.7, and we list a few examples in the following subsections.

Cartesian Space Acceleration Tracking

Given Eq. 3.17, we can penalize deviation from the desired Cartesian acceleration using

$$\begin{aligned} A_{cart} &= \begin{bmatrix} J & 0 \end{bmatrix} \\ b_{cart} &= \ddot{x}^* - \dot{J}\dot{q}. \end{aligned} \tag{3.18}$$

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

$$\ddot{x}^* = K_p(x_d^* - x) + K_d(\dot{x}_d^* - \dot{x}) + \ddot{x}_d^*, \tag{3.19}$$

where x_d^*, $\dot{x}_d^*$ and $\ddot{x}_d^*$ are specified by a higher level controller, and x and $\dot{x}$ are computed by forward kinematics based on the estimated current robot state. Many objectives such as CoM, hand, foot and torso motions are specified with this form. J_{CoM} is defined by Eq. 3.13. Depending on the tasks, we can drop rows in Eq. 3.18 that correspond to dimensions that we do not want to track.

We maintain contact acceleration constraints using high weight cost terms (soft constraints) as opposed to hard constraints. For these cost terms, we set $\ddot{x}^* = 0$ in Eq. 3.19. We think using high weight soft penalties is much more forgiving when given conflicting desired motions and numerically more stable.

Centroidal Dynamics

In the previous implementation, the desired change in centroidal momentum was expressed as a cost term using the net external wrench from the contact forces. On the other hand, it is also linear with respect to the generalized acceleration in Eq. 3.8. In the current implementation, the latter form is preferred since we can then freely add virtual forces in the generalized coordinates or in the Cartesian space that are useful for handling modeling errors.

$$\begin{aligned} A_{CM} &= \begin{bmatrix} C & 0 \end{bmatrix} \\ b_{CM} &= \begin{bmatrix} \dot{P}^* \\ \dot{L}^* \end{bmatrix} - \dot{C}\dot{q}, \end{aligned} \tag{3.20}$$

where $\dot{P}^*$ and $\dot{L}^*$ are desired change in linear and angular momentum. In the current implementation, $\dot{P}^*$ is generated by plugging in the output of DDP's linear policy into the same point mass

model, and $\dot{L}^*$ is set to damp out angular momentum. Optimization for $\dot{L}^*$ is part of the future work.

Approximated Value Function for the CoM

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

$$\begin{aligned} X'_{CoM} &= \begin{bmatrix} x + \dot{x}_{CoM}dt + 0.5(J_{CoM}\ddot{q} + \dot{J}_{CoM}\dot{q})dt^2 \\ \dot{x}_{CoM} + (J_{CoM}\ddot{q} + \dot{J}_{CoM}\dot{q})dt \end{bmatrix} \\ &= \begin{bmatrix} 0.5J_{CoM}dt^2 \\ J_{CoM}dt \end{bmatrix} \ddot{q} + \begin{bmatrix} x_{CoM} + \dot{x}_{CoM}dt + 0.5\dot{J}_{CoM}\dot{q}dt^2 \\ \dot{x}_{CoM} + \dot{J}_{CoM}\dot{q}dt \end{bmatrix}. \end{aligned} \tag{3.21}$$

We can also approximate the cost-to-go for the next time step with

$$V(X'_{CoM}) \approx 0.5(X'_{CoM} - X^*_{CoM})^T V^*_{XX}(X'_{CoM} - X^*_{CoM}), \tag{3.22}$$

where V^*_{XX} and X^*_{CoM} are the value function's second order derivative and the setpoint. These are generated by CoM trajectory planning with DDP. Finally, we can write the term for the value function as

$$\begin{aligned} A_V &= \begin{bmatrix} V' \begin{bmatrix} 0.5J_{CoM}dt^2 \\ J_{CoM}dt \end{bmatrix} & 0 \end{bmatrix} \\ b_V &= -V' \begin{bmatrix} x_{CoM} + \dot{x}_{CoM}dt + 0.5\dot{J}_{CoM}\dot{q}dt^2 - x^*_{CoM} \\ \dot{x}_{CoM} + \dot{J}_{CoM}\dot{q}dt - \dot{x}^*_{CoM} \end{bmatrix}, \end{aligned} \tag{3.23}$$

where $V^*_{XX} = V'^T V'$

Center of Pressure Tracking

Directly specifying CoP for each contact can be especially useful during transitions such as toe-off. The stacked contact wrench λ is defined as

$$\lambda = \begin{bmatrix} F_0 \\ M_0 \\ F_1 \\ M_1 \\ \vdots \end{bmatrix}, \tag{3.24}$$

where F_i and M_i are the ith contact force and torque. For the current implementation, the first two contacts correspond to the left and right foot. This term can be written as

$$\begin{aligned} A_{cop} &= \begin{bmatrix} 0 & \begin{bmatrix} \dots & \begin{bmatrix} 0 & 0 & p_x^* & 0 & 1 & 0 \\ 0 & 0 & p_y^* & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R_w^b & 0 \\ 0 & R_w^b \end{bmatrix} & \dots \end{bmatrix} \end{bmatrix} \\ b_{cop} &= 0, \end{aligned} \tag{3.25}$$

where $p_x^* x$ and p_y^* are desired CoP in the body frame, and R_w^b is the rotation from world frame to body frame.

Weight Distribution

In the current implementation, weight is only distributed between two feet, and this term is active only in double support. Desired weight distribution is specified by $w^* = F_z^L/(F_z^L + F_z^R)$, where F_z^L and F_z^R are the vertical forces at the left and right foot. We add this term to the cost function with

$$\begin{aligned} A_{weight} &= \begin{bmatrix} 0 & \begin{bmatrix} S_{weight} & 0 \end{bmatrix} \end{bmatrix} \\ b_{weight} &= 0. \end{aligned} \tag{3.26}$$

S_{weight} is a 1×12 vector with zeros, except $S_{weight}(3) = 1 - w^*$ and $S_{weight}(9) = -w^*$.

Direct Tracking and Regularization

We can also directly specify desired values for $\ddot{q}$, τ and λ:

$$\begin{aligned} A_{qdd^*} &= \begin{bmatrix} I & 0 \end{bmatrix} \\ b_{qdd^*} &= \ddot{q}^* \\ A_{\lambda^*} &= \begin{bmatrix} 0 & I \end{bmatrix} \\ b_{\lambda^*} &= \lambda^* \\ A_{\tau^*} &= \begin{bmatrix} M_l & -J_l^T \end{bmatrix} \\ b_{\tau^*} &= -h_l + \tau^*. \end{aligned} \tag{3.27}$$

Zero is used if no target value is specified except for the vertical forces, which are regularized using gravitational forces weighted by the weight distribution. Similar to Eq. 3.19, target joint acceleration can be computed based on the desired joint position, velocity and acceleration. This term is useful for directly controlling specific joints or forces. It also regularizes $\mathcal{X}$ to make the problem well conditioned.

Change in Torque and Contact Wrench

To avoid high frequency oscillations in the outputs, we penalize changes in τ and λ similarity to Eq. 3.27.

$$\begin{aligned} A_{d\lambda} &= \begin{bmatrix} 0 & I \end{bmatrix} \\ b_{d\lambda} &= \lambda_{prev} \\ A_{d\tau} &= \begin{bmatrix} M_l & -J_l^T \end{bmatrix} \\ b_{d\tau} &= -h_l + \tau_{prev} \end{aligned} \tag{3.28}$$

where τ_{prev} and λ_{prev} are torque and contact wrench computed in the last time step.

3.5.3 Constraints

F and M in Eq. 3.24 are defined in the world frame. They can be rotated into the body frame with

$$\begin{aligned} F^b &= R_w^b F \\ M^b &= R_w^b M, \end{aligned} \tag{3.29}$$

where R_b^w is the rotation matrix representing the foot's orientation in the world frame. Friction constraints are approximated by

$$\begin{aligned} |F_x^b| &\leq \mu F_z^b \\ |F_y^b| &\leq \mu F_z^b. \end{aligned} \tag{3.30}$$

The forces and torques at the feet need to obey CoP constraints,

$$\begin{aligned} d_x^- &\leq -M_y^b/F_z^b \leq d_x^+ \\ d_y^- &\leq M_x^b/F_z^b \leq d_y^+, \end{aligned} \tag{3.31}$$

where F^b and M^b denote forces and torques in the foot frame, and d^- and d^+ are the sizes of foot. The vertical force also needs to be positive

$$0 \leq F_z^b. \tag{3.32}$$

Eq. 3.16 is used as equality constraints. Torque limits can be easily added into the inequality constraints using Eq. 3.15.

3.5.4 Parameters

Weights for the walking controller are summarized in Table 3.1. w_{qdd} is for joint acceleration. w_{comdd} is for CoM acceleration. $w_{utorsowd}$ and $w_{pelviswd}$ are for upper torso and pelvis angular acceleration. w_{footdd} is for tracking foot linear and angular acceleration, and it is multiplied by 100 during stance. w_{regF} and w_{regTau} are regularization weights for contact force and joint torques. w_w is for weight distribution. w_{cop} is for center of pressure. w_{dF} and $w_{d\tau}$ penalize changes in contact force and joint torques between two consecutive time steps. w_V is for value function.

For upper body orientation tracking, K_p and K_d in Eq. 3.19 are 50 and 5.66 for roll and pitch, and 10 and 3.16 for yaw. For swing foot tracking, K_p and D_d are 100 and 8 for X and Y, 70 and 13.39 for Z, 50 and 11.31 for roll and pitch, and 30 and 8.76 for yaw. For joint tracking, K_p and K_d are 5 and 2. Parameters are task dependent, but do not change much. These parameters are for the walking controllers on the physical robot.

Table 3.1: Weights for ID cost function (walking controller)

w_{qdd}	w_{comdd}	$w_{utorsowd}$	$w_{pelviswd}$	w_{footdd}	w_{cop}
0.1	1	0.3	0.3	1	10^{-4}
w_w	w_{regF}	w_{regTau}	w_{dF}	$w_{d\tau}$	w_V
0	10^{-3}	0	10^{-2}	3×10^{-2}	0

3.6 Inverse Kinematics

We took a static approach to all the manipulation tasks for the DARPA Robotics Challenge. The motions were designed to be slow enough such that we do not need to explicitly reason about dynamics for motion planning and actual robot execution. On the other hand, the robot is more likely to run into kinematic constraints when performing manipulation tasks, So we introduced an explicit inverse kinematic module to maintain a feasible full body kinematic state that best tracks the objectives while obeying constraints. The inverse dynamics controller was used to stabilize the robot around this full state.

We use a gradient based approach to the inverse kinematics problem. A generalized velocity is solved on every control cycle using quadratic programming. Configuration is numerically integrated from the optimized velocity. For this quadratic program, $\mathcal{X} = \dot{q}_{ik}$. The numerically integrated configuration is denoted by q_{ik}, and all the internal states such as position and orientation computed by forward kinematics are denoted with subscripts $_{ik}$.

3.6.1 Contact Adaption

Since our current state estimator does not incorporate exteroceptive sensor information, it can not correct for absolute position drift. Any desired Cartesian positions for long term tracking need to be properly updated because such drift is not present in IK's internal model. In the IK controller, we use high weight and high gain tracking terms to maintain stationary contacts. These contacts are initialized based on forward kinematics when they are first established. One intuitive way to compensate for the state estimator drift is to slowly update IK's desired contact poses toward the

current estimated ones. We refer to these desired contact poses as "anchors", x^*_{anchor}.

$$x^*_{anchor} = \alpha x_{contact} + (1-\alpha)x^*_{anchor}. \tag{3.33}$$

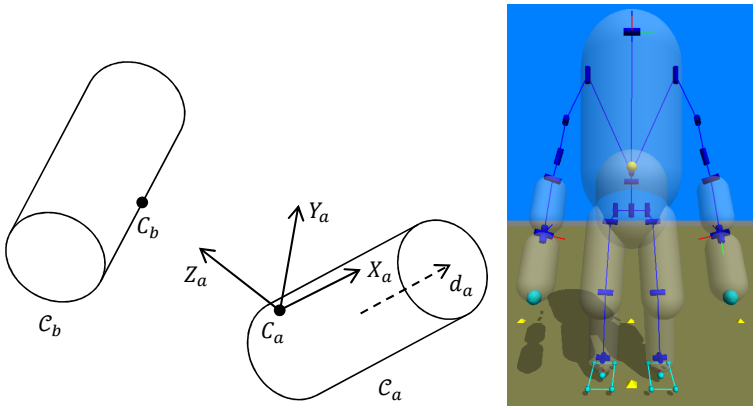
In Eq. 3.38, x^*_{anchor} replaces x^*_d, and $\dot{x}^*_d = 0$.

3.6.2 Self Collision Avoidance

A lot of the fine adjustments for manipulation are done using human-in-the-loop tele-operation, where small Cartesian corrections are directly commanded. In this case, motion planning is not used because we want the system to be very responsive. So the controller has to be aware of collisions. Even with planned collision free trajectories, it is possible that the controller needs to modify the plan in unexpected situations during execution. There can be very little clearance for the planned trajectories, so small variations can invalidate the collision free property. Thus, built-in collision avoidance is beneficial. We focus on avoiding self collision because it is computationally tractable, and is still very useful for tele-operation or visual servoing.

We use capsules as collision shapes for the major limbs and torso. Figure 3.2 shows the collision shapes we use for the Atlas robot. Given the current configuration q_{ik} of the robot, for any two capsules of interest, $\mathcal{C}_a$ and $\mathcal{C}_b$, we first find the closest two points on their surfaces respectively, c_a and c_b. Assuming c_a and c_b will remain the closest points on $\mathcal{C}_a$ and $\mathcal{C}_b$ for the next time step, we can construct a frame O_a, whose origin is at c_a, the Z axis has the same direction as $c_b - c_a$, and the Y axis is perpendicular to both $c_b - c_a$ and $\mathcal{C}_a$'s principal axis d_a.

$$\begin{aligned} Z_a &= \frac{c_b - c_a}{|c_b - c_a|} \\ Y_a &= Z_a \times d_a \\ X_a &= Y_a \times Z_a \\ R_a &= \begin{bmatrix} X_a & Y_a & Z_a \end{bmatrix} \end{aligned} \tag{3.34}$$



(a) Coordinate system for setting up a collision constraint

(b) Simplified collision shapes for the Atlas robot

Figure 3.2: Figure 3.2(a) is an illustration for Eq. 3.34 and Eq. 3.35. $\mathcal{C}_a$ and $\mathcal{C}_b$ are the two collision shapes of interest. c_a and c_b are the two closest points on $\mathcal{C}_a$ and $\mathcal{C}_b$. d_a is a unit vector that represents the principle axis of $\mathcal{C}_a$. X_a, Y_a and Z_a are defined in Eq. 3.34. Figure 3.2(b) shows the collision shapes we used for the Atlas robot (represented with transparent grey capsules).

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

$$
\begin{aligned}
H'_a &= \begin{bmatrix} R_a & c_a + J_a \dot{q} dt \\ 0 & 1 \end{bmatrix} \\
c'_b &= c_b + J_b \dot{q} dt \\
\begin{bmatrix} {}^a c'_b \\ 1 \end{bmatrix} &= {H'_a}^{-1} \begin{bmatrix} c'_b \\ 1 \end{bmatrix} \\
\Rightarrow {}^a c'_b &= R_a^{-1}(c_b + J_b \dot{q} dt) - R_a^{-1}(c_a + J_a \dot{q} dt) \\
&= R_a^{-1}(J_b - J_a)\dot{q} dt + R_a^{-1}(c_b - c_a),
\end{aligned}
\tag{3.35}
$$

where J_a and J_b are Jacobian computed at c_a and c_b. Eq. 3.35 is linear with respect to $\dot{q}$, and the inequality constraint for avoiding collision is

$$ {}^{a}c_b'[Z] \geq \epsilon, \tag{3.36} $$

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

Due to the kinematics of the Atlas robot and empirical observations, we only consider these pairs for possible self collisions to speed up computation:

- left and right hand
- left and right forearm
- left (right) forearm and pelvis
- left (right) hand and right (left) forearm
- left (right) hand and torso
- left (right) hand and pelvis
- left (right) hand and left (right) thigh
- left (right) hand and right (left) thigh
- left (right) hand and left (right) shin
- left (right) hand and right (left) shin

3.6.3 Cost Function

We list a few example objectives for Eq. 3.7 for IK.

Cartesian Space Velocity Tracking

We penalize deviation from the desired Cartesian velocity $\dot{x}^*$ with

$$\begin{aligned} A_{cart} &= J(q_{ik}) \\ b_{cart} &= \dot{x}^*, \end{aligned} \tag{3.37}$$

where

$$\dot{x}^* = K_p(x_d^* - x_{ik}) + \dot{x}_d^*. \tag{3.38}$$

We use a different set of K_p here than in ID. Cartesian space tracking and contacts are both handled with Eq. 3.37 and Eq. 3.38, although the weights and desired targets, x_d^* and $\dot{x}_d^*$ are different.

Direct Tracking and Regularization

$$\begin{aligned} A_{state} &= I \\ b_{state} &= \dot{q}^*, \end{aligned} \tag{3.39}$$

where $\dot{q}^*$ can be a target joint velocity or zero for regularization. Similar to Eq. 3.38, target joint velocity can be computed based on the desired joint position and velocity.

Change in Velocity

$$\begin{aligned} A_{d\dot{q}} &= I \\ b_{d\dot{q}} &= \dot{q}_{prev}, \end{aligned} \tag{3.40}$$

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

Table 3.2: Weights for IK cost function

w_{qd}	w_{comd}	$w_{utorsow}$	$w_{pelvisw}$	w_{footd}	$w_{d\dot{q}}$
0.5	10	3	3	100	5×10^{-3}

3.6.4 Constraints

We do not impose equality constraints in the inverse kinematics QP. Inequality constraints mainly consist of joint and Cartesian limits and self collision avoidance described in Section 3.6.2.

The joint limit constraints are

$$q^- \leq q_{ik} + \dot{q}dt \leq q^+, \tag{3.41}$$

where dt is the time step, and q^- and q^+ are the upper and lower joint limit. For Cartesian space pose constraints,

$$x^- \leq x_{ik} + J(q_{ik})\dot{q}dt \leq x^+, \tag{3.42}$$

where x^- and x^+ are the upper and lower limits. Velocity constraints in joint space can be easily added, and Cartesian space velocity constraints need to be transformed by the Jacobian matrix.

Self collision avoidance is also achieved using inequality constraints, which is described in Section 3.6.2 using Eq. 3.36.

3.6.5 Parameters

Weights are summarized in Table 3.2. w_{qd} is for joint velocity tracking. w_{comd} is for CoM velocity. $w_{pelvisw}$ and $w_{utorsow}$ are for pelvis and torso angular velocity. w_{footd} is for foot linear and angular velocity. $w_{d\dot{q}}$ penalizes changes in velocity. The position gain K_p used in Eq. 3.38 is 5 along the diagonal for CoM, upper torso orientation tracking, 10 for foot tracking, and 2 for joint tracking.

3.7 Modeling Error and Disturbance Force Compensation at the CoM Level

Depending on the configuration, our Atlas has up to $3cm$ of error in the measured CoP and the CoM computed by forward kinematics when the robot is stationary. This indicates a significant modelling error on the CoM level considering the foot is only $12cm$ wide. A CoM offset estimator based on LIPM dynamics and IMU measurements is developed in [103] to estimate this modeling error x_{err}.

$$x_{err} = x_{com}^{true} - x_{com}^{fk}, \tag{3.43}$$

where x_{com}^{fk} is the model's predicted CoM position, and x_{com}^{true} is the true CoM position assuming LIPM dynamics. x_{err}'s Z coordinate is 0. The CoM offset can also be treated as a force, F_{err}, applied at the model's CoM, and vice versa. Using LIPM dynamics,

$$F_{err} = \frac{\sum_i^M m_i g}{z} x_{err}, \tag{3.44}$$

where z is the height difference between model's CoM and the support. We can lump all unplanned horizontal external contact forces into F_{err}, which is compensated in the ID controller. The sum of gravitational and Coriolis and centrifugal forces is modified as:

$$h' = h - J_{CoM}^T F_{err}. \tag{3.45}$$

x_{err} can be used to offset the desired CoM in the nominal plans. This corresponds to a behavior that leans into the external force, so that the CoP remains at the planned position.

Table 3.3: Control modes for each joint

Back z	Back y	Back x	Neck y		
$K_{qd}, K_{\tau p}, F_{qd}$	K_{ld}, K_{Fp}, F_{ld}	K_{ld}, K_{Fp}, F_{ld}	K_{qp}, K_{qd}, K_{qi}		
Hip z	Hip x	Hip y	Knee y	Ankle y	Ankle x
K_{ld}, K_{Fp}, F_{ld}	K_{ld}, K_{Fp}, F_{ld}	K_{ld}, K_{Fp}, F_{ld}	K_{ld}, K_{Fp}, F_{ld}	K_{Fp}, F_{ld}	K_{Fp}, F_{ld}
Shoulder z	Shoulder x	Elbow y	Elbow x	3 Elec. Wrists	
$K_{qp}, K_{qd}, K_{\tau p}$	$K_{qp}, K_{qd}, K_{\tau p}$	$K_{qp}, K_{qd}, K_{\tau p}$	$K_{qp}, K_{qd}, K_{\tau p}$	K_{qp}, K_{qd}, K_{qi}	

3.8 Robot Implementation

3.8.1 Joint Level Controllers

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

$$\begin{aligned} i = & K_{qp}(q_d - q) + K_{qi}\sum(q_d - q)dt \\ & + K_{qd}(\dot{q}_d - \dot{q}) + F_{qd}\dot{q} + F_{qd_d}\dot{q}_d \\ & + K_{\tau p}(\tau_d - \tau) + F_{\tau_d}\tau_d \\ & + F_{const}. \end{aligned} \tag{3.46}$$

In Eq. 3.46, K_* stands for feedback gains, F_* stands for feedforward gains, F_{const} is a constant bias term, and subscript $_d$ denotes the desired values. We use K_{qp}, K_{qd}, $K_{\tau p}$, F_{qd} and F_{const} in our implementation. The F_{qd} term is used to implement the velocity compensation scheme in [7]. The electric forearms share the same interface, but we only use the K_{qp}, K_{qd} and K_{qi} terms. During implementation, we found that for the linear actuators (all the leg joints and back pitch and roll joint), specifying gains in actuator coordinates improved joint tracking performance. Table 3.3 summarizes the set of gains we use for each joint on Atlas. For the subscripts, q denotes joint space values, and l denotes actuator space values. The actual gains are listed in Table 3.4, and the last three columns are gains specified in actuator space.

The actuator gains need to be transformed to fit the interface's joint space specification. This

is done using the transmission information disclosed by Boston Dynamics. We use actuator gains for all the linearly actuated joints, which are all the leg joints and spine pitch and roll joints. Let l denote piston length, and e_* for tracking error for the following equations. Valve commands computed in either joint or actuator space should be the same. For the K_{qd} term, since $\dot{q} = \frac{dq}{dl}\dot{l}$,

$$\begin{aligned} K_{qd}e_{\dot{q}} &= K_{ld}e_{\dot{l}} \\ K_{qd}\frac{dq}{dl}e_{\dot{l}} &= K_{ld}e_{\dot{l}} \\ K_{qd} &= \frac{dl}{dq}K_{ld}. \end{aligned} \tag{3.47}$$

Similarly, $F_{qd} = \frac{dl}{dq}F_{ld}$.

For the torque feedback term, the virtual work produced in both spaces should be the same, so $\tau dq = F dl$. For $K_{\tau p}$,

$$\begin{aligned} K_{\tau p}e_{\tau} &= K_{Fp}e_F \\ K_{\tau p}\frac{dl}{dq}e_F &= K_{Fp}e_F \\ K_{\tau p} &= \frac{dq}{dl}K_{Fp}. \end{aligned} \tag{3.48}$$

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

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

$$\begin{bmatrix} \dot{l_l} \\ \dot{l_r} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \dot{q}_y \\ \dot{q}_x \end{bmatrix}, \mathbf{J} = \begin{bmatrix} \frac{dl_l}{dq_y} & \frac{dl_l}{dq_x} \\ \frac{dl_r}{dq_y} & \frac{dl_r}{dq_x} \end{bmatrix}. \tag{3.49}$$

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

$$\begin{bmatrix} i_l \\ i_r \end{bmatrix} = \mathbf{J} \begin{bmatrix} i'_{q_y} \\ i'_{q_x} \end{bmatrix} + \begin{bmatrix} F_{q_y} \\ F_{q_x} \end{bmatrix}_{const}, \tag{3.50}$$

where i'_{q_y} and i'_{q_x} are computed with Eq. 3.46 without the F_{const} term.

Thus for the velocity term,

$$\begin{aligned} \mathbf{K}_{ld}\mathbf{e}_{\dot{l}} &= \mathbf{J}\mathbf{K}_{qd}\mathbf{e}_{\dot{q}} = \mathbf{J}\mathbf{K}_{qd}\mathbf{J}^{-1}\mathbf{e}_{\dot{l}} \\ \Rightarrow \mathbf{K}_{qd} &= \mathbf{J}^{-1}\mathbf{K}_{ld}\mathbf{J} \end{aligned} \tag{3.51}$$

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

$$\begin{aligned} \begin{bmatrix} i_l \\ i_r \end{bmatrix}_{qd} &= \mathbf{J}\mathbf{K}_{qd}\mathbf{e}_{\dot{q}} \\ &= \mathbf{J}\left(\begin{bmatrix} k_{yy} & 0 \\ 0 & k_{xx} \end{bmatrix} \begin{bmatrix} e_{\dot{q}_y} \\ e_{\dot{q}_x} \end{bmatrix} + \begin{bmatrix} k_{yx}e_{\dot{q}_x} \\ k_{xy}e_{\dot{q}_y} \end{bmatrix} \right) \\ \Rightarrow K^y_{qd} &= k_{yy} \\ K^x_{qd} &= k_{xx} \\ F^y_{const} &+= \frac{dl_l}{dq_y}k_{yx}e_{\dot{q}_x} + \frac{dl_l}{dq_x}k_{xy}e_{\dot{q}_y} \\ F^x_{const} &+= \frac{dl_r}{dq_y}k_{yx}e_{\dot{q}_x} + \frac{dl_r}{dq_x}k_{xy}e_{\dot{q}_y}. \end{aligned} \tag{3.52}$$

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

Joint Name	K_{qp}	K_{qi}	K_{qd}	$K_{\tau p}$	F_{qd}	K_{ld}	K_{Fp}	F_{ld}
Back z	0	0	2.375	0.02	0.3	X	X	X
Back y	0	0	X	X	X	100	0.0005	20
Back x	0	0	X	X	X	100	0.000465	20
Neck y	35	0.05	0.6	0	0	X	X	X
Hip z	0	0	X	X	X	80	0.00025	8
Hip x	0	0	X	X	X	100	0.001	20
Hip y	0	0	X	X	X	100	0.002	40
Knee	0	0	X	X	X	100	0.002	40
Ankle y	0	0	X	X	X	0	0.0025	30
Ankle x	0	0	X	X	X	0	0.0025	30
Shoulder z	30	0	0.3	0.02	0	X	X	X
Shoulder x	30	0	0.3	0.02	0	X	X	X
Elbow y	30	0	0.3	0.02	0	X	X	X
Elbow x	30	0	0.3	0.02	0	X	X	X
Elec. Wrists	15	0.05	0.3	0	0	X	X	X

Table 3.4: Joint level gains. The last three columns are gains specified in the actuator side.

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

$$\boldsymbol{\tau} = \mathbf{J}^T\mathbf{F}, \tag{3.53}$$

where $\boldsymbol{\tau}$ and $\mathbf{F}$ are vectors. The derivation is similar to Eq. 3.51,

$$\begin{aligned} \mathbf{K}_{Fp}\mathbf{e}_F = \mathbf{J}\mathbf{K}_{\tau p}\mathbf{e}_\tau = \mathbf{J}\mathbf{K}_{\tau p}\mathbf{J}^T\mathbf{e}_F \\ \mathbf{K}_{\tau p} = \mathbf{J}^{-1}\mathbf{K}_{Fp}\mathbf{J}^{-T}, \end{aligned} \tag{3.54}$$

and the individual terms are computed same as in Eq. 3.52.

For the ankle joints, velocity tracking is achieved by adding a velocity term in the desired torque as:

$$\tau_d \mathrel{+}= k(\dot{q}_d - \dot{q}), k = \begin{cases} 20, & \text{if swing} \\ 10, & \text{if stance} \end{cases} \tag{3.55}$$

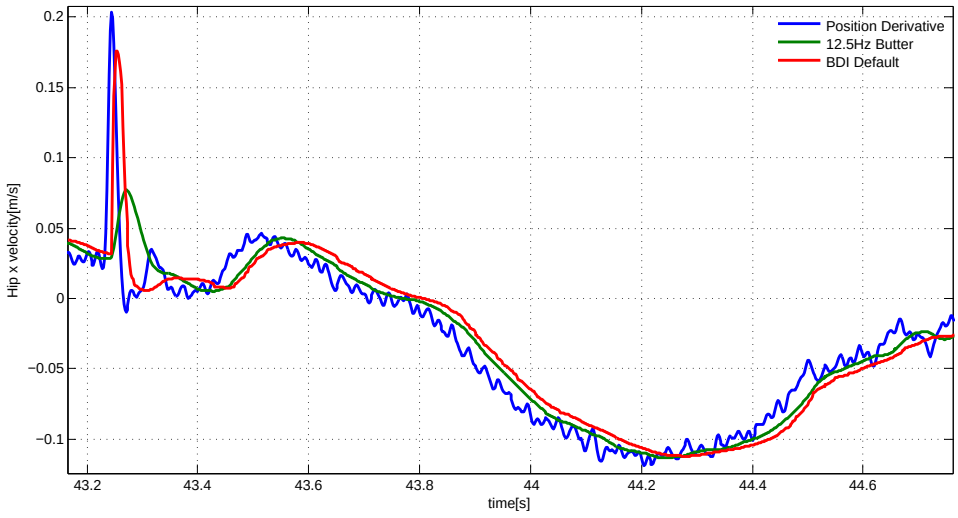


Figure 3.3: Comparison of the velocity traces of the left hip roll joint. Blue: position derivative filtered by an acausal low pass filter (50Hz cutoff) with no phase shift. Green: second order Butterworth filter with a cutoff frequency of 12.5Hz. Red: Boston Dynamics' default filter.

3.8.2 Onboard Joint Velocity Filtering

Independent of user side state estimation, the Atlas robot generates joint velocity signals by filtering the finite differences of the measured joint positions. These are also used in the onboard joint level controllers in Eq. 3.46. In addition to Boston Dynamics' default filters for joint position, velocity, torque and error in torque, second and third order low pass filters with custom parameters are also provided. We have the option to completely replace the default filters with the custom ones in the joint level servos. We used second order Butterworth filters for joint velocities and kept the default filters for the rest. We found that Boston Dynamics' default velocity filter was sensitive to sharp velocity changes, which easily resulted in high frequency oscillations when we tried to use higher velocity gains. On the other hand, it filtered slow signals more and introduced a larger delay. Figure 3.3 shows a comparison between the default velocity filter and a second order Butterworth filter for the left hip roll joint. We were able to drastically increase the velocity gains after we replaced the onboard velocity filters. The cutoff frequencies are summarized in Table 3.5.

Table 3.5: 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	Ankle x
10	12.5	15	15	15	15
Shoulder z	Shoulder x	Elbow y	Elbow x	3 Elec. Wrists	
7	7	5	5	10	

Table 3.6: Elasticity compensation parameters for the leg joints

Hip z	Hip x	Hip y	Knee y	Ankle y	Ankle x
$1/7000$	$1/6000$	0.0001	0.0001	0.0001	0.0001

3.8.3 Joint Elasticity Compensation

Due to pre-transmission joint position sensing and compliance in the mechanism, forward kinematics is not very accurate on Atlas. For stationary single stance, there can be up to $3cm$ offsets between the model CoM from forward kinematics and the measured CoP. We implemented a simple linear torque dependent heuristic to reduce the effects of joint compliance, which is proposed in [46]. The corrected joint angle q_c is computed with

$$q_c = q - K_e \tau \tag{3.56}$$

where q and τ are the measured joint position and torque. This correction is only applied to the leg joints, and stiffness parameters are summarized in Table 3.6.

3.8.4 Control Loop Frequency

The joint level controllers run at $1kHz$, and we can receive states and issue commands at the same frequency. On the other hand, one control cycle can take up to $1.5ms$. In the current implementation, the state estimator runs within the robot communication loop run at $1kHz$, and the controller runs on a separate thread at $500Hz$. We did not pursue a real-time operating system in favor of easier development. Stock Ubuntu 12.04 was used on the control computers, and we

observed consistent and uniform cycle time after disabling the X server (for graphical interface). It is possible that the nonuniform cycle time had a negative impact on earlier attempts to integrate ID accelerations to desired velocities [20].

3.9 Summary

The inverse dynamics controller was first developed and successfully applied on the simulated Atlas robot. It greatly simplified high level controller design since the details of the physical system was abstracted away. Different from other hierarchical formulations of inverse dynamics, ours is build around a weighted sum of cost terms. We can easily accommodate physically impossible desired targets or conflicting terms. This can be useful for tele-operation when feasibility checks are hard to perform. During early robot implementation, we discovered that using inverse dynamics alone worked reasonably well for the stance leg but not for the swing leg. In particular, it lacked the necessary kinematic tracking accuracy for swing foot placement and arm motion control. A main difficulty for controlling these lightly loaded joints using pure torque control is that the unmodeled and unmeasured friction forces on such joints are comparable with the torques generated by inverse dynamics. This necessitates additional compensatory torques or some form of position or velocity control. For practical reasons, adding kinematic feedback is more viable than directly compensating for modeling errors on the force level. We developed two separate full body control schemes for walking and manipulation that generated joint level kinematic targets differently. In addition to the torque feedback loops, the walking controller had velocity loops whose targets were integrated from the accelerations computed by inverse dynamics. For manipulation, inverse kinematics was used to generate the full state first, and inverse dynamics was used for stabilization. These design choices were motivated by hardware limitations and different task requirements for walking and manipulation. We successfully applied these controllers to many tasks in the DARPA Robotics Challenge, and the details can be found in Chapter 4. The same walking controller was later used for the push recovery and dynamic

walking experiments in Chapter 5.

Chapter 4

Two Level Controllers for Walking and Manipulation

In this chapter, we will briefly present results for simulated walking, and then explain the walking and manipulation controllers that we developed for the DARPA Robotics Challenge Finals in detail. Pictures of Atlas performing various DRC mobility and manipulation tasks are shown in Figure 4.4 and Figure 4.7. All of these controllers consist of a high level CoM planner and a full body controller. After the DRC Trials at the end of 2013, we focused on implementing a reliable walking controller on the Atlas robot. A large amount of energy was spent on tuning gains and filter parameters on the hardware to improve joint level tracking performance. The low level full body controller was also redesigned to address an inconsistency issue we encountered during the DRC Trials [20]. With these changes, we gained significant improvements in terms of performance and reliability. At top speed, our Atlas is able to walk over 20 times faster than in 2013. We also completed two one hour long missions at the DRC Finals without any physical human intervention.

All hardware experiments were conducted with the Atlas robot built by Boston Dynamics. It has 30 actuated degrees of freedom, six for each leg, seven for each arm, three for the spine, and one for neck pitch. Most joints have hydraulic actuators with the exception of the neck and the

last three joints on each arm that are electric.

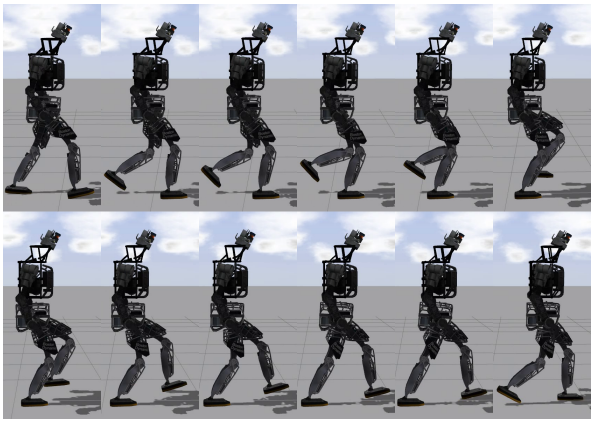
4.1 Simulated Walking

We first present results from the simulation phase of the DRC. The simulation environment was based on Gazebo by the Open Source Robotics Foundation. The simulated Atlas robot had 28 joints that were pure torque sources: six for each leg and arm, three for the back joints, and one for neck pitch. We could run simulation close to real-time on a computer with an Intel Xeon(R) CPU E5-2687W CPU with 32G memory when simulating for the full task scenario including environmental physics and sensors.

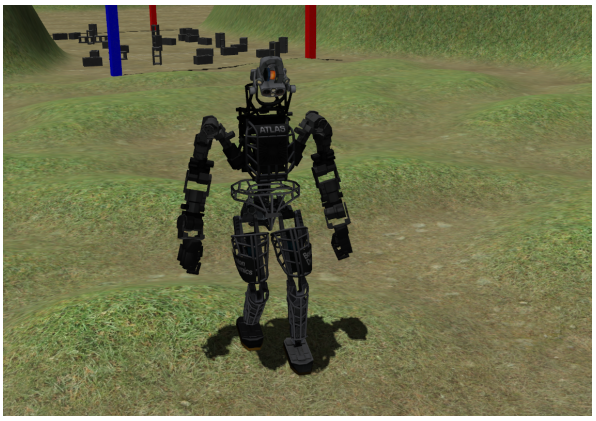
The walking controller had two threads: one thread was dedicated to the full body controller solving inverse dynamics at $1kHz$, and the other optimized the CoM trajectory for the next three foot steps. For runtime, trajectory optimization usually completed within $50ms$, and the full body controller ran at roughly $1kHz$. This walking controller replanned CoM trajectory per foot step starting with the last planned rather than the measured CoM states. This design choice was made so that we could preplan CoM trajectories in a separate thread before touchdown. Although we were able to offset the computational cost of the nonlinear trajectory optimization, it required high CoM tracking accuracy. It also required accurate swing foot tracking and terrain sensing to meet the preplanned foot placement and touchdown timing. Due to these limitations, this walking controller only worked well in simulation, which had few modelling errors and little sensing uncertainty.

4.1.1 Fast Flat Ground Walking

We were able to demonstrate toe-off and heel-strike behaviors during flat floor walking by using simple heuristics to guide the full body controller. For heel-strike, a desired touchdown pitch angle was specified. For toe-off, we first changed the reference point (where the contact Jacobian



(a) Snapshots for simulated fast walking



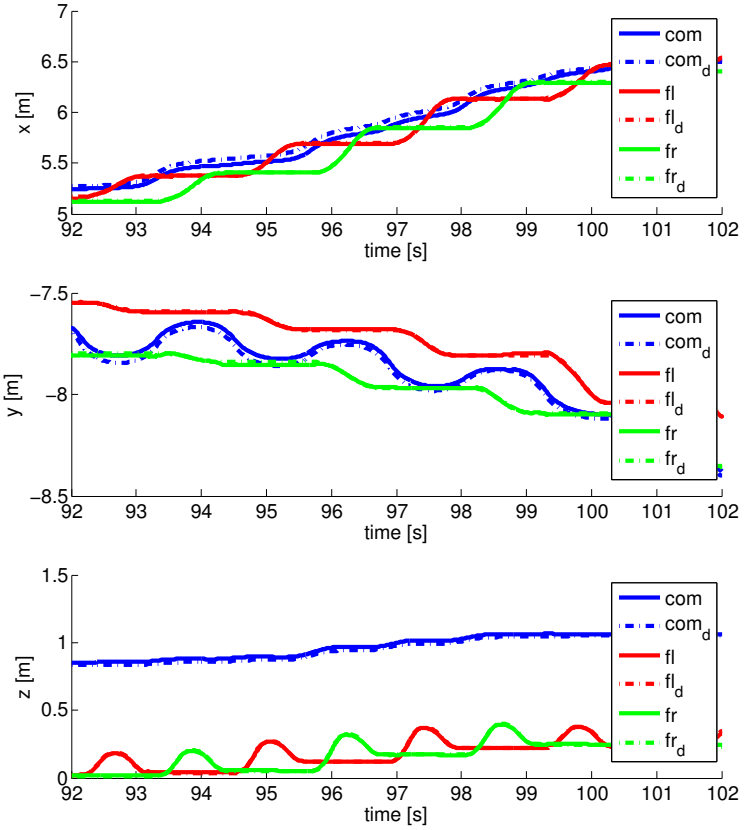
(b) Simulated walking on rough terrain

Figure 4.1: Simulated Atlas walking on flat ground with $0.8m$ step length and $0.7s$ per step, and the same controller tested in the Rough Terrain Task in DARPA's Virtual Robotics Challenge.

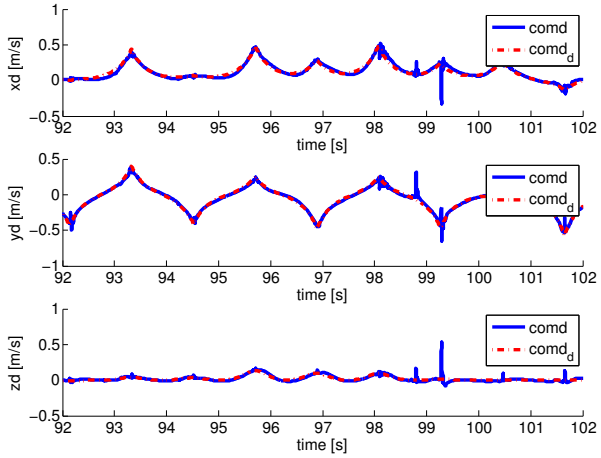
is computed) to the toe, constrained the CoP to be at the toe, and then specified a short but large pitch angular acceleration in the foot frame. With these heuristics, we achieved a maximum step length of $0.8m$ and $0.7s$ per step. Figure 4.1(a) shows a sequence of snapshots of simulated fast walking on flat ground.

4.1.2 Rough Terrain

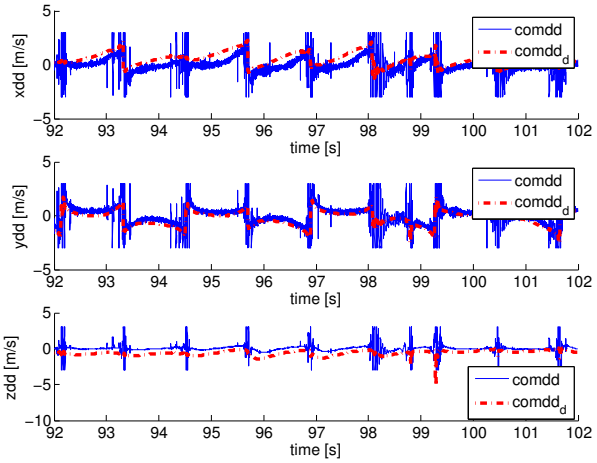
The controller could handle up to 0.4 rad inclined slopes, and continuously climb steps that were $0.2m$ high and $0.4m$ apart. We were able to successfully walk on the rough terrain environment provided in the Virtual Robotics Challenge as well. In order to traverse rough terrain, an A* planner modified from [28] was used to provide sequences of foot steps, which were given as inputs to the walking controller. The CoM trajectory was optimized based on these foot steps, and the desired swing foot trajectory was generated using quintic splines. CoM and swing foot tracking is shown in Figure 4.2.



(a) CoM and swing foot tracking



(b) CoM velocity tracking



(c) CoM acceleration tracking

Figure 4.2: fl and fr denote left foot and right foot respectively. Desire CoM and foot trajectories are denoted with subscript $_d$, and plotted in dashed lines. Actual CoM acceleration in the last plot is computed by finite differencing velocity and truncated at $\pm 3m/s^2$.

4.2 Walking Controller for the DARPA Robotics Challenge

For the DRC Finals, our approach to walking is "slow and steady", and we aim for maximizing reliability and avoiding falls. In general, going slower makes state estimation easier, and estimating total external force possible [103], with which we can achieve accurate CoM state tracking and good CoP control. Walking slowly also enables us to design safety features that allows pausing the walking cycle in unexpected scenarios. Figure 3.1(a) summarizes the overall structure of the walking controller, which mainly consists of the CoM trajectory planner (Chapter 2) and the full body controller based on ID (Section 3.5 and Section 3.8). A nonlinear point mass model that includes the Z dimension is used for CoM trajectory optimization to take height changes into account. The CoM trajectory is replanned during every single support phase for the next two foot steps. The swing foot trajectory is generated by a quintic spline from the liftoff pose to the desired touchdown pose.

Since ID alone can not generate accurate swing motions on the real robot due to various modelling errors, joint level kinematic targets are necessary. On the leg and spine joints, we add a velocity feedback loop in addition to the torque loop. The desired velocity $\dot{q}_d$ is numerically integrated from the output acceleration $\ddot{q}_d$ from ID:

$$\dot{q}_d = \alpha(\dot{q}_d + \ddot{q}_d dt) + (1 - \alpha)\dot{q}, \tag{4.1}$$

where $\alpha = 0.95$, $\dot{q}$ is the current estimated joint velocity. Since desired motions for the arms are always specified in joint space during walking, we can directly use those as kinematic targets. More implementation details about the joint level controller can be found in Section 3.8.

4.2.1 Contact Switching

Matching actual and planned contact states is important, since tracking a desired CoP outside of the actual region of support is infeasible. In the current implementation, the planned CoM

trajectory is not modified if the controller misses the planned touchdown event. Instead, we introduce a large negative Z acceleration for the swing foot to bring the foot down as fast as possible. If we miss touchdown by a significant amount of time, the robot will fall. The controller only makes the transition from double support to single support when stability is achieved for a fixed duration. Otherwise, it throws an exception and asks for the human operator's intervention. In the context of the DRC, this is typically caused by the robot pushing against the environment. On the dress rehearsal day, the robot was caught on the door frame when sidestepping through, and the walking controller stopped promptly for manual recovery. Figure 4.3 shows data from this event. After the controller detected a large external force, it paused liftoff and remained in double support.

4.2.2 Toe-off

For static walking, the CoM needs to be completely shifted to the next stance foot during double support. When taking longer strides or stepping to a greater height, extending the rear knee alone is often insufficient to move the CoM all the way. Toe-off is one solution to this problem. It used to be triggered by detecting the stance leg approaching knee straight. However, this is not a sufficient condition for toe-off. For some challenging terrain or more dynamic walking, blindly switching to toe-off mode without considering dynamic feasibility can cause the robot to fall backwards. Instead, we first prepare for toe-off when a near straight knee configuration is detected. This is achieved by adding a high weight cost term in ID that moves the CoP to the toe without changing the constraint. Once the optimized CoP is close to the toe, we then start shrinking the CoP constraint to be exactly at the toe. During toe-off, we shift the rear foot reference point (where the Jacobian is computed) to the toe. The contact cost term in Eq. 3.18 for the rear foot is also modified. Position terms remain the same, and for the rotational terms,

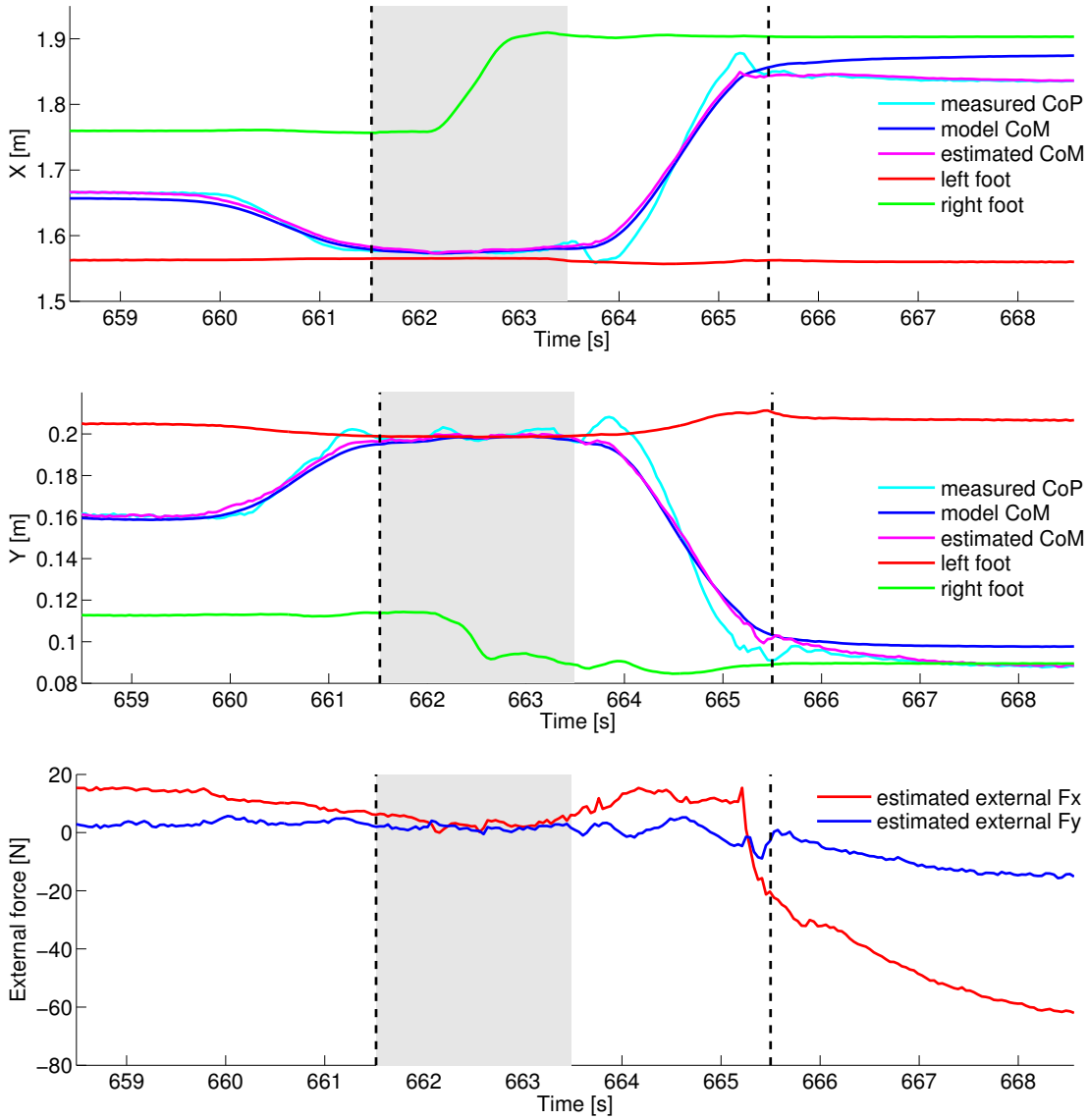


Figure 4.3: Atlas was caught on the door frame when sidestepping through it during the dress rehearsal at the DRC Finals. The walking controller properly delayed liftoff and remained in double support when it detected an anomaly. Single support phase is shown by the shaded area, and the black dashed lines indicate the planned liftoff time. Estimated CoM is the sum of the model CoM and the estimated CoM offset.

we first compute

$$
\begin{aligned}
A_{Rot} &= \begin{bmatrix} R^{-1} J_{Rot} & 0 & 0 \end{bmatrix} \\
b_{Rot} &= R^{-1}(\ddot{x}^*_{Rot} - \dot{J}_{Rot}\dot{q}),
\end{aligned}
\tag{4.2}
$$

where R is the foot's rotation matrix. Only the first and third rows in Eq. 4.2, that correspond to roll and yaw motion are added to the cost functions. $\ddot{x}^*_{Rot}$ is set to zero. We also set $d_y^- = d_y^+ = 0$ in Eq. 3.31. This effectively turns the rear foot contact into an unactuated pin joint around its pitch axis. A slightly bent knee angle is given in Eq. 3.27 to avoid singularity.

4.2.3 Offset on the Planned CoM Trajectory

In the current implementation, the CoM trajectory cannot be replanned within one control tick due to computational costs. So it is not possible to simply replan a CoM trajectory at each touchdown event to account for the inevitable foot placement errors. We choose to replan at the middle of each single support phase when CoM motion is relatively small for low walking speed. In order to reuse the outdated CoM trajectory after touchdown, we introduce a simple offset when computing the desired CoM position and CoP for the low level controller.

$$
\begin{aligned}
x_{off} &= x_{foot}^{actual} - x_{foot}^{planned} \\
x^*_{com} &= x_{com}^{planned} + \alpha x_{off} \\
x^*_{cop} &= x_{cop}^{planned} + \alpha x_{off},
\end{aligned}
\tag{4.3}
$$

where x_{foot}^{actual} is the actual location for the touchdown foot, and $x_{foot}^{planned}$ is the planned. x^*_{com} and x^*_{cop} are the modified CoM and CoP targets. α ramps from zero to one gradually after touchdown.

4.2.4 Compensating for CoM Offset and Small External Forces

On the real robot, we have observed a significant (up to $3cm$) configuration dependent CoM offset. In order to develop a reliable static walking controller, we design an estimator that explains this CoM level modeling error using LIPM dynamics [103]. In the LIPM context, this discrepancy can be treated as a combination of an external force and a true CoM offset. We treat it as an external force and compensate for it in the ID controller. When a substantial external force is detected, the walking controller pauses and waits for operator intervention.

4.2.5 Mobility for the DARPA Robotics Challenge

Mobility is a central focus in the DARPA Robotics Challenge. Aside from transitioning between tasks, the robot needs to perform several locomotion tasks such as traversing piled cinder blocks and climbing a standard staircase. We also need to walk off a platform after egressing from the vehicle. Pictures of our Atlas performing these tasks are in Figure 4.4. We also successfully walked over $30m$ several times in an outdoor parking garage in case we needed to skip the driving and egress tasks. Figure 4.5 shows data of our Atlas walking over cinder blocks with onboard battery power for the first time. To our surprise, we did not notice any significant differences between using the battery pack or the usual power tether. CoM position and velocity tracking is reasonably well during walking. CoP tracking is less accurate during the initial weight transfer phase. This is caused by a mismatch between the actual weight distribution and the one computed by the full body controller.

The biggest challenges we encountered during the DRC Finals for all the mobility tasks were actually caused by kinematic constraints. Walking down certain arrangements of cinder blocks was hard due to limited ankle pitch range. When stepping down, the rear ankle sometimes hit joint limit during double support and caused a large CoP tracking error and the robot falling. We worked around this issue by deliberately stepping over the cinder block, so that the foot could physically roll over the edge. For climbing the staircase, if we put the foot fully on the step,

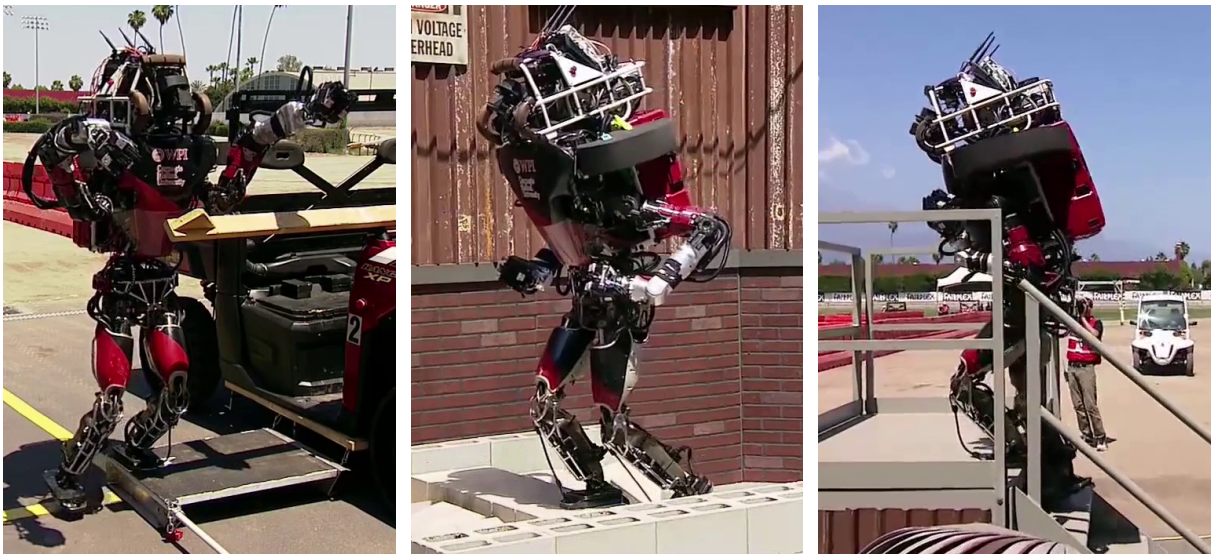


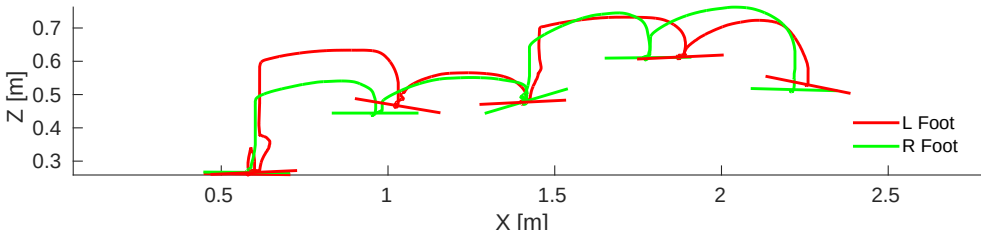
Figure 4.4: Atlas stepping off the platform after egress, walking over terrain and climbing stairs during the DARPA Robotics Challenge Finals

the shank would collide with the next step when stepping up and tip the robot backwards. Our solution was to place the foot splayed and the back of the foot slightly off the step.

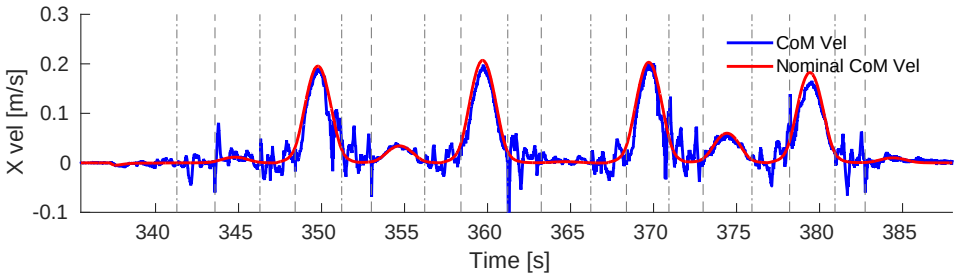
After the DRC Finals, we also achieved dynamic walking with the same walking controller. By disabling the safety features (Section 4.2.1) and simply increasing the cadence, our Atlas could walk at $0.4m/s$ with $0.8s$ per step. Data for this experiment is shown in Figure 4.6.

4.3 Manipulation Controller

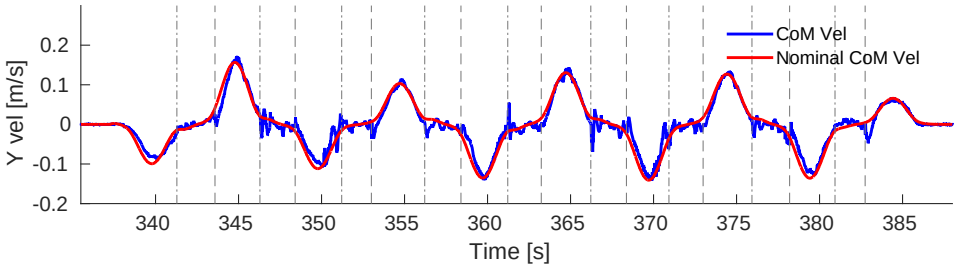
The manipulation controller is used for a variety of tasks such as traditional pick-and-place, valve and door handle turning, and cutting with power tools (Figure 4.7) for the DRC. For the manipulation controller, the desired full state is solved by IK and stabilized using ID. A block diagram for the manipulation controller is shown in Figure 3.1(b). This design decision is mainly motivated by the observation that during manipulation, the robot often runs into kinematic related constraints such as joint limits or collisions. The controller needs to consider kinematic constraints properly to generate feasible motions. We could have used a similar approach as for the



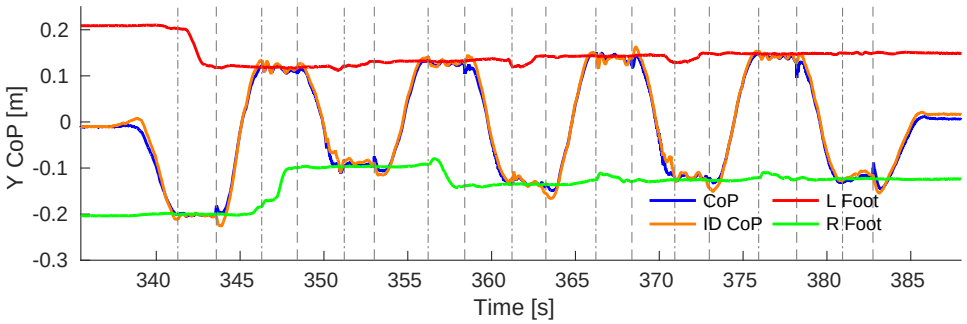
(a) Swing foot trajectories in the sagittal plane



(b) Estimated vs nominal X CoM velocity

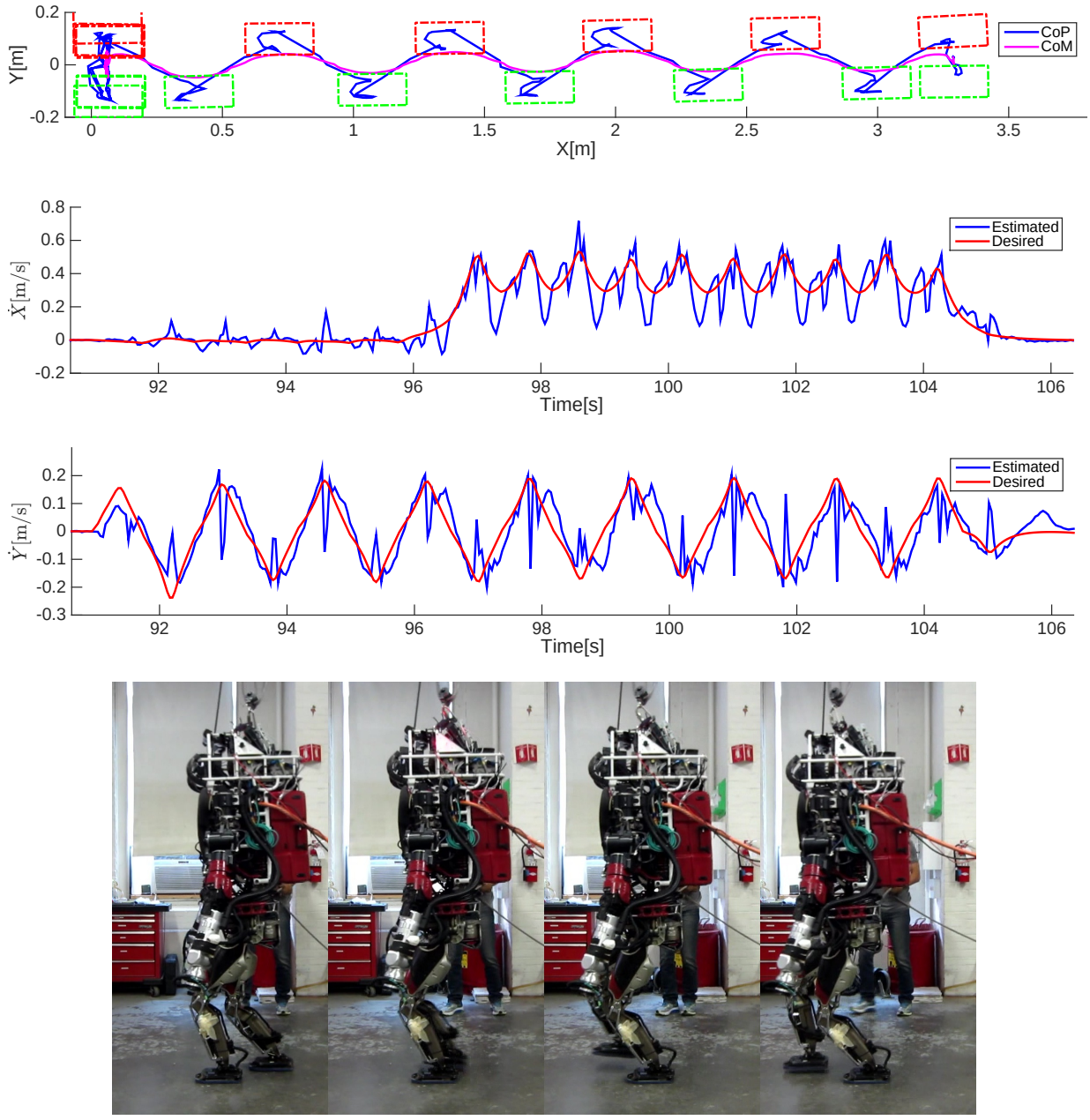


(c) Estimated vs nominal Y CoM velocity



(d) Y CoP tracking

Figure 4.5: These plots show the Atlas robot traversing part of the terrain task for the DRC Finals. X axis is the forward direction, Y points to the robot's left, and Z points upward. Straight lines in the top plot indicate the pitch angle at touchdown. The estimated CoM velocities are shown in the middle plots against the desired velocities. The dashed lines indicate touchdown, and the dot-dashed lines indicate liftoff events.



(d) Atlas walking at $0.4m/s$ with $0.8s$ per step. These snapshots are taken every $0.2s$.

Figure 4.6: In the top figure, CoP and foot positions are plotted in the XY plane. Left and right foot are shown in red and green respectively. The estimated CoM velocities are shown against the desired velocities in the next two plots.

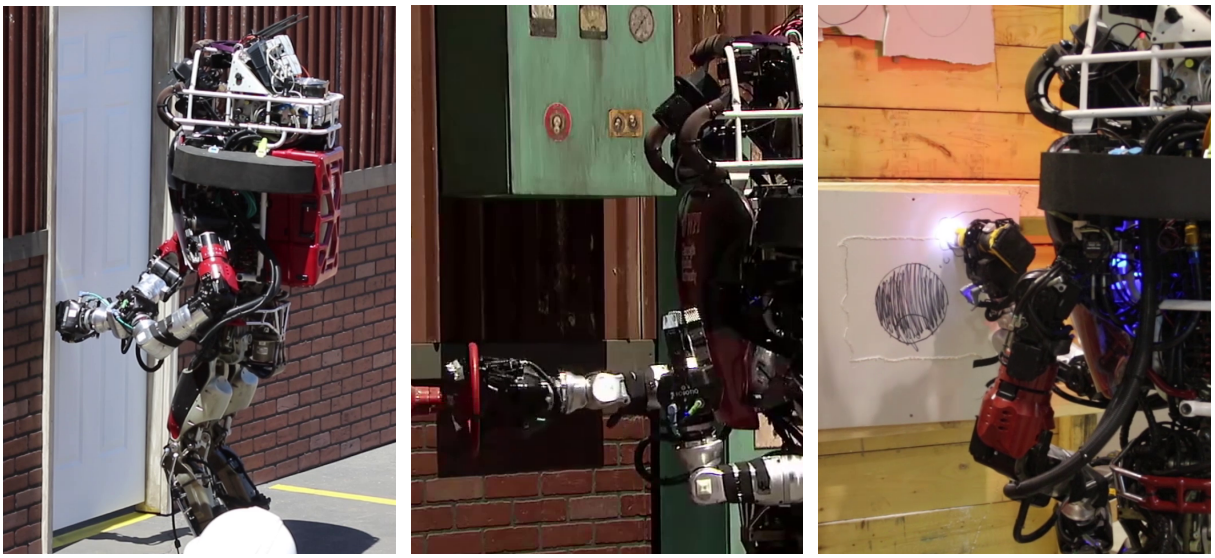


Figure 4.7: Atlas opening a door, turning a valve, and cutting with a power tool during the DARPA Robotics Challenge Finals and when practicing.

walking controller: specify all the kinematic constraints in the ID controller on the acceleration level, and integrate the resulting accelerations for position and velocity targets. However, this has some practical drawbacks. First of all, it is slower to solve due to the additional inequality constraints. When the real state penetrated some kinematic constraints, ID will generate large accelerations to bring the state back to the constraint surface, which can cause oscillation and instability around the constraint surface. Because of hardware limitations, velocity tracking is much worse on the arm joints than the leg joints. Position feedback is necessary for accurate motion tracking on the arms. On the other hand, because the robot moves so slowly during manipulation, it seldom runs into dynamic constraints such as CoP constraints or torque limits given reasonable goals. So it is reasonable to decouple the problem into solving IK first then stabilizing with ID. Eq. 3.19 is modified in the ID controller to:

$$\ddot{x}^* = K_p(x_{ik} - x) + K_d(\dot{x}_{ik} - \dot{x}), \tag{4.4}$$

where x and $\dot{x}$ are measured real robot states, and x_{ik} and $\dot{x}_{ik}$ are IK's internal states. For joint acceleration tracking,

$$\ddot{q}^* = K_p(q_{ik} - q) + K_d(\dot{q}_{ik} - \dot{q}), \tag{4.5}$$

where q and $\dot{q}$ are measured configuration and velocity, and q_{ik} and $\dot{q}_{ik}$ are from the IK controller.

4.3.1 Nudging and Interpolated Desired End Effector Motion

Our manipulation strategy for the DRC relies heavily on visual servoing, which outputs adjustments for the desired Cartesian targets. This strategy works well with our gradient based IK controller. When a big Cartesian target change is specified, we use spline interpolation for smooth motions. Position and orientation are interpolated with quintic splines and slerp respectively. However, this functionality was mostly replaced by full body motion planning towards the end of the DRC Finals.

4.3.2 Full Body Trajectory Planning

For better quality and collision free trajectories, we use TrajOpt [76] to plan full body motions, which returns a sequence of key frames. It is also useful for planning a constrained motion such as turning a door handle. A typical planned trajectory has between 15 to 30 key frames. For each key frame, Cartesian targets for important body parts such as CoM, pelvis, torso, and hands are generated with forward kinematics. Linear interpolation is used to fill in the gap between two key frames for all configuration and Cartesian targets. During execution, the IK controller tracks all these targets, but has higher weights on the Cartesian targets. These key frames also need timing information for interpolation. We can approximate the generalized velocity between two consecutive key frames by taking their finite difference. Since the centroidal momentum is linear with respect to this velocity in Eq. 3.8, we can compute the minimum duration between these two frames by bounding the maximum magnitude of the centroidal momentum. Intuitively, the trajectory slows down when it generates large whole body motions and speeds up otherwise.

Figure 4.8 shows an example of planning a full body trajectory with TrajOpt to grab an object and executing it using the manipulation controller. The IK controller is used to connect between key frames while maintaining internal kinematic constraints. It also modifies the original trajectory using information that is unavailable to the motion planner, such as the online estimated CoM offset. Laser point clouds are used for object recognition and building collision models for the environment. In this experiment, visual servoing with a wrist-mount depth camera was used for the final approach, and the left arm was used for balancing during visual servoing.

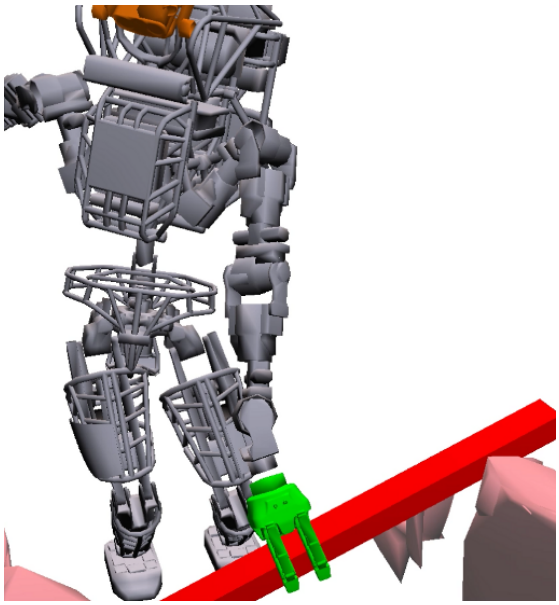
4.3.3 Hand Force Torque Servoing

In order to make a reliable cut for the drill task in the DRC Finals, the robot needs to press the drill firmly against the dry wall with a small amount of normal force. Since the arms are mostly position controlled, this is achieved by changing the desired hand kinematic target for IK.

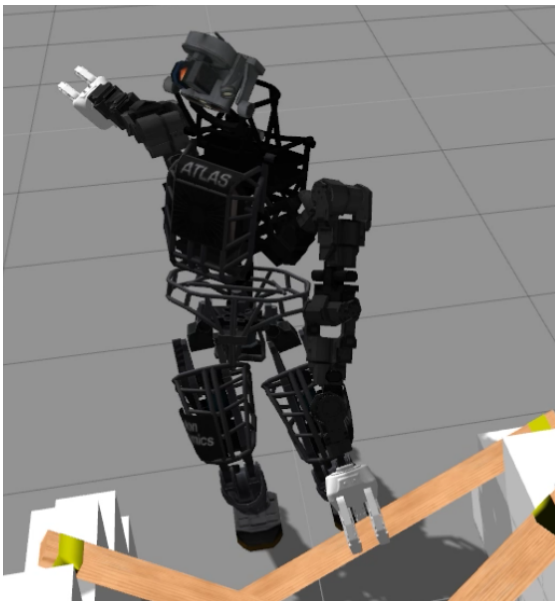
4.3.4 Fall Prevention

Fall detection is based on the Corrected Capture Point (CCP) [103]. It is essentially the capture point combined with an estimated offset that represents the net effect of an external force and the CoM modelling error using LIPM dynamics. The CCP is checked with the support region computed by the convex hull of the foot corners. When the manipulation controller detects a near falling scenario, it terminates all ongoing trajectories, maintains its current posture, and blocks future commands until the operator attempts manual recovery.

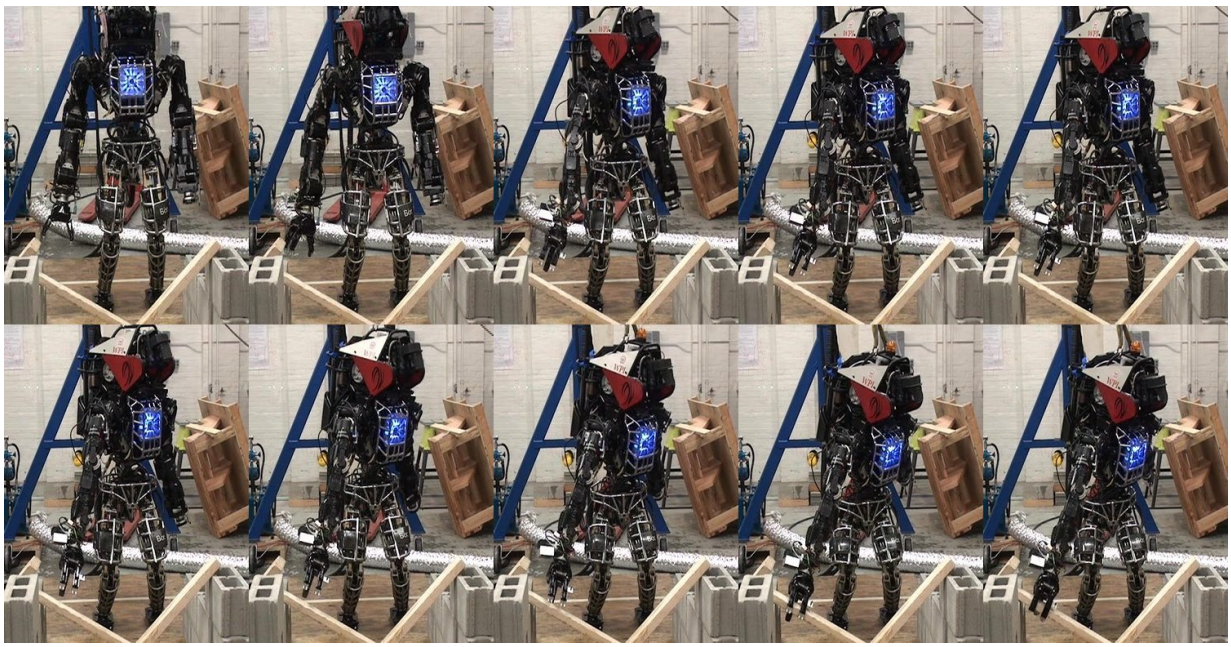
During our second mission in the DARPA Robotics Challenge Finals, the right electric forearm mechanically failed when the cutting motion was initiated for the drill task. The uncontrolled forearm wedged the drill into the dry wall and applied a large backward force. The manipulation controller quickly froze and saved the robot from falling, and the human operator was able to recover and move on to the remaining tasks.



(a) TrajOpt generated end configuration for picking up a 2 by 4.



(b) Simulated Atlas executing the planned trajectory.



(c) Snapshots of Atlas picking up a 2 by 4.

Figure 4.8: The given task is to pick up the closest wooden piece. The last key frame in the planned trajectory generated by TrajOpt is shown in Figure 4.8(a). The pink regions are convex decompositions of the environment based on laser point clouds. The red bar represents the object of interest segmented using template matching. After executing this trajectory in simulation, the robot ends up like Figure 4.8(b). Figure 4.8(c) shows the Atlas robot picking up a 2 by 4 using TrajOpt and visual servoing. The snapshots are taken every 7 seconds. Visual servoing starts from the seventh picture, and visual feedback is provided by a depth camera mounted at the right wrist.

4.4 Summary

Both of the DRC walking and manipulation controllers have two components: a high level controller that optimizes for task space trajectories into the future and a low level full body controller that generates instantaneous joint commands to track those trajectories while satisfying physical constraints. For the walking controller, a CoM trajectory is planned using a simple point mass model to traverse the given foot steps. Long term behaviors are either planned using motion planning or specified by a human operator for the manipulation tasks. This approach was suitable for our "slow and steady" strategy for the DRC. We had the capability to perform all eight tasks in the DRC Finals, and scored 7 out 8 points twice during the two hour long missions. In terms of reliability, our safety code promptly stopped the robot several times when it was close to falling so that the human operators could successfully recover.

Besides being slow, the biggest disadvantage for these controllers is that they rely on good tracking accuracy in general, and they are unable to deal with situations that are significantly different from the plan. For walking, a typical failure case happens when the full body controller can no longer track the original CoM plan either due to external disturbances such as unplanned contacts, or the original plan becomes invalid, e.g. insufficient CoM acceleration due to constrained ankle torque. We try to address some of these issues in Chapter 5 by introducing a receding-horizon layer that replans much more rapidly to account for such unexpected situations.

Although the overall walking dynamics can be well captured and handled by this hierarchical approach, kinematics can still be tricky. For more challenging steps, both the swing and stance leg can run into configuration limits or singularity. For the DRC Finals, we had to handcraft a set of special rules for the foot steps to avoid such situations. Although planned by the CoM planner, height was more often constrained by leg configurations. All these indicate configuration space trajectory planning is still necessary for more complicated problems. In retrospect, the control benefit of planning for the Z dimension was marginal especially at our walking speed. But it is an useful example of using nonlinear trajectory optimization to generate nominal CoM references.

A more interesting model involving angular momentum should be more suitable for the high level planner, which we will investigate it in the near future.

Chapter 5

Receding Horizon Based Middle Level Controllers

5.1 Introduction

For a typical hierarchical system that consists of a planner and a controller, handling tracking errors has always been a key challenge. When large tracking errors occur, the plan can become invalid, and the controller starts tracking an infeasible plan. Replanning is an intuitive and practical solution for this issue, but for unstable systems like walking robots, the time budget for replanning is very limited. In this chapter, we will develop simple controllers that can rapidly replan for a short horizon based on the long term information from the high level planner. For the sake of computation time, linear models are used, and we only consider a limited preview horizon.

There is a rich literature on using Receding Horizon Control [59] (also known as Model Predictive Control) for online walking pattern generation and push recovery. Preview Control [35] is a very popular walking pattern generation method based on LIPM. Capture point [70] is another useful conceptual tool for balancing, and it can also be generalized to generate walking patterns [45, 46]. Similarly, divergent component of motion [27, 83] is introduced to encode the

unstable part of the LIPM dynamics and used for walking pattern generation. These methods are typically used to compute reference CoM / ZMP trajectories, which are then open loop tracked by some low level controller. To improve robustness, an online approach proposed by Nishiwaki et al. rapidly adjusts the reference ZMP trajectory based on the measured robot state to account for external perturbations [61]. It is further extended to vary foot placement or step timing [62] using characteristics of Preview Control, in which timing can be resolved by a line search. Another foot placement strategy based on Preview Control is proposed by Urata et. al [92, 93]. A special form of Preview Control's cost function is used to enable fast computation of the CoM / ZMP trajectory, which is used as an evaluation function to optimize for the next foot step.

Foot placement and CoM trajectory can also be generated simultaneously by solving a linear trajectory optimization problem using LIPM dynamics [18, 22, 23, 68, 100]. These approaches are typically formulated as quadratic programs that optimize for foot placement and the time derivatives of the ZMP. Piecewise linear acceleration is assumed to reduce the number of necessary samples (optimization variables). Instead of following the desired foot steps, these approaches can track overall behavioral goals such as a desired average speed or reaching some long term position [80]. It can also be extended to combine the ankle, hip and stepping strategies [1] by using a linear model with angular momentum. Linear receding-horizon controllers are also extensively used in [82] for push recovery with strong disturbances. Our receding-horizon component is formulated similarly to this group, but the objective is to follow the nominal high level plan.

In the following sections, we will present two receding-horizon controllers that use foot placement and angular momentum respectively. The foot placement controller is implemented on the real Atlas robot, with which we achieved faster and much more robust dynamic walking.

5.2 Foot Placement Controller

During walking, when the CoM state tracking error can not be corrected by just using ankle torque (controlling CoP) alone, it is necessary to either use angular momentum or take a recovery step. In this section, we focus on using constrained optimization to generate new foot steps to regain balance. At any time during the swing phase, we assume the nominal CoM trajectory has already been planned for the next few foot steps. The basic idea is to optimize the next foot placement so that the CoM state will track the nominal CoM trajectory as closely as possible during the next swing phase. In order to reoptimize foot placement fast in a receding-horizon fashion, several assumptions and simplifications are made:

- Linear Inverted Pendulum Model (LIPM)
- Fixed step timing
- Point foot
- Short double support phase and zero CoM acceleration during double support
- Foot orientation is not optimized

Linear dynamics and fixed timing are necessary to make the system dynamics linear with respect to the optimization variable, so the problem becomes convex and fast to solve. The point foot assumption forces the CoP to coincide with foot placement, so that we do not need to sample in time to take into account variable CoP. With these assumptions, the time evolution of the CoM state can be expressed as a linear function of the initial CoM state and the CoP (foot placement) based on LIPM dynamics.

5.2.1 Foot Step Optimization with Quadratic Programming

The CoM state $\mathbf{X}$ is defined as $\begin{bmatrix} x & \dot{x} \end{bmatrix}^T$, and the foot placement is denoted by p. Given known timing t, future CoM state can be expressed as:

$$\begin{aligned} \mathbf{X} &= \mathbf{A}(t)\mathbf{X}_0 + \mathbf{B}(t)p \\ \mathbf{A}(t) &= 0.5 \begin{bmatrix} e^{\omega t} + e^{-\omega t} & \frac{e^{\omega t} - e^{-\omega t}}{\omega} \\ \omega(e^{\omega t} - e^{-\omega t}) & e^{\omega t} + e^{-\omega t} \end{bmatrix} \\ \mathbf{B}(t) &= \begin{bmatrix} 1 - 0.5(e^{\omega t} + e^{-\omega t}) \\ 0.5\omega(e^{-\omega t} - e^{\omega t}) \end{bmatrix}, \end{aligned} \tag{5.1}$$

where $\omega = \sqrt{g/z}$, and $\mathbf{X}_0$ is the initial state.

During swing, let t_{TD} be the remaining duration of the current swing phase, p_{cur} be the current stance foot position, and $\mathbf{X}$ be the estimated current CoM state. Given by Eq. 5.1, the CoM state at planned touchdown can be computed as $\mathbf{X}_{TD} = \mathbf{A}(t_{TD})\mathbf{X} + \mathbf{B}(t_{TD})p_{cur}$. Assuming zero CoM acceleration during double support, the CoM state at liftoff is $\mathbf{X}_{LO} = \mathbf{X}_{TD} + \begin{bmatrix} \dot{x}_{TD} T_{DS} & 0 \end{bmatrix}^T$, where T_{DS} is the duration of double support. For foot placement p and any time t during the next swing phase, the CoM state can then be computed as

$$\mathbf{X}_t = \mathbf{A}(t)\mathbf{X}_{LO} + \mathbf{B}(t)p. \tag{5.2}$$

The cost function consists of two terms: one for foot placement deviation from the planned location p^*, and another for CoM state tracking error during the next swing phase.

$$\min_p \sum_t (\mathbf{X}_t - \mathbf{X}_t^*)^T V_t (\mathbf{X}_t - \mathbf{X}_t^*) + w(p - p^*)^2, \tag{5.3}$$

where w is a weight, and $\mathbf{X}_t^*$ and V_t are the nominal CoM state and the second order derivative of the value function computed by DDP sampled at time t after liftoff. In the current implemen-

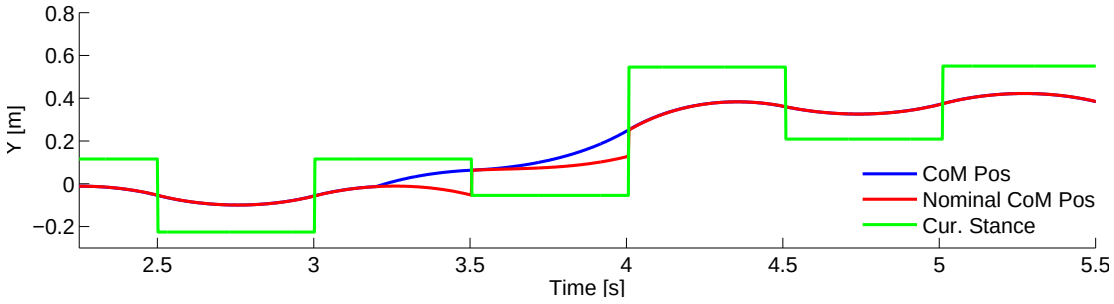
tation, five equally timed $\mathbf{X}^*$ and V are sampled in Eq. 5.3.

A set of linear inequality constraints are used to approximate the allowed stepping region. The simplest box constraint relative to the current stance foot is used in this implementation. The swing foot has to be placed within $\pm 0.5m$ in the X direction and between $[0.17, 0.6]m$ away from the current stance foot in the Y direction. Ideally, the foot step planner will generate these constraints in addition to the nominal foot step taking sensor inputs into account. The foot step planner can also produce a cost map which can substitute the first term in Eq. 5.3. The orientation of the foot step is not optimized, and the desired orientation is used for the swing foot.

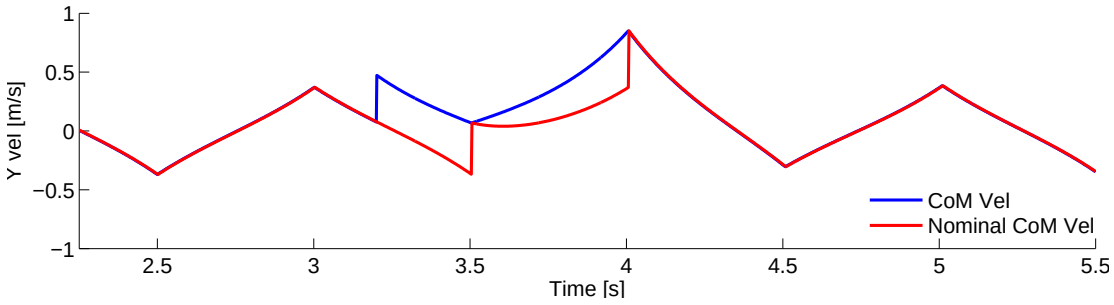
5.2.2 Simple Example

In this simple example, we use the same LIPM for both planning and simulation. There is no double support phase, and the leg swings perfectly. All the parameters are the same as for robot experiments. CoM height is set to $0.88m$. The overall task is to walk in place, and the desired foot steps are $0.34m$ apart laterally. Upon touchdown, the desired next step is updated based on the current stance foot location, and a new nominal CoM trajectory is planned with DDP. During simulation, we allow some control over CoP, which is generated by the DDP policy and bounded by the size of the stance foot ($\pm 0.12m$ for X, and $\pm 0.04m$ for Y).

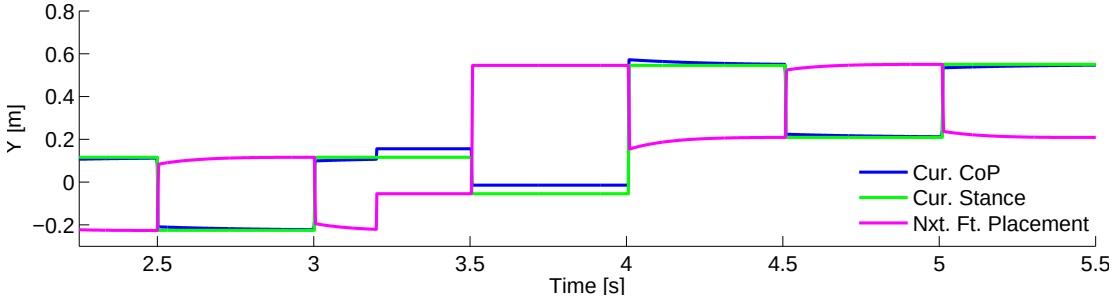
Figure 5.1 shows plots of a push recovery scenario in the coronal plane. At $3.2s$, the CoM velocity is increased by $0.4m/s$, which is equivalent to an impulse of $72Ns$. In this case, the robot is pushed towards the left during left single support, and it needs a two step strategy to recover: put down the right foot as close as possible to the left, then take a second large left side step to recover. The stance CoP control immediately saturates at the boundary of the foot. The next foot placement is set to put down as closely as possible to the current stance foot, followed by a large left side step in the next swing phase.



(a) CoM position



(b) CoM velocity



(c) CoP and optimized foot step

Figure 5.1: In this plot, the lateral CoM velocity is instantaneously increased by $0.4m/s$, which is equivalent to an impulse of $72Ns$ during left single stance at $3.2s$. The controller takes two steps to recover. Nominal CoM trajectory is replanned at every touchdown. Stance CoP control can be varied within the foot.

5.2.3 Robot Implementation

With little parameter tuning, we could push the two part walking controller (CoM trajectory optimization and full body controller) that we used in the DARPA Robotics Challenge to walk at $0.4m/s$ with $0.8s$ per step. However, we cannot achieve a higher cadence with just parameter tuning. The dominant failure mode was due to the actual CoM state diverging from the planned trajectory, which eventually resulted in loss of balance in the coronal plane. Although it is possible to achieve better CoM tracking by controlling angular momentum more intelligently, taking recovery steps gains a much larger stability margin in general. We also feel more comfortable about aggressive leg swings than large torso or arm motions. Reoptimizing foot placement becomes crucial for achieving faster and more robust dynamic walking on our Atlas.

For all the fast walking and push recovery experiments in this section, the walking controller consists of three levels: the CoM trajectory planner, the foot placement controller and the full body controller. We focus on level walking, so LIPM is used for CoM trajectory planning instead of the 3D nonlinear model. The biggest advantage for using a linear model is fast computation, which only requires one DDP iteration to converge. Consequently, desired CoM trajectory can be recomputed within a few milliseconds after touchdown using the estimated states, which also simplifies the actual software implementation. For the full body controller, the estimated CoM force compensation is disabled, because the estimation for external force on CoM is designed only for slow motions. The rest of the full body controller and state estimation remain unchanged.

Cadence

For LIPM, the CoM is always falling, and as indicated by Eq. 5.1, it falls exponentially fast with respect to time. Intuitively, higher cadence is always preferred, because the robot falls for a shorter period of time. There are also more chances to correct tracking errors since control is only intermittent. On the other hand, shorter swing phase requires higher acceleration for the swing leg that can introduce undesired CoM velocity variations. Change in CoM velocity greatly

affects the foot placement, which in turn results in more swing leg motions and causes instability. The effect of CoM velocity on optimized foot placement is apparent in Figure 5.4. For all the experiments in this section, the walking cycle is set to be at $0.5s$ per step with a $0.05s$ double support phase. Faster cadence can be achieved, but empirically, $0.5s$ seems to work best for our robot experiments.

Swing Foot Motion Generation

For every control cycle ($500Hz$) during the swing phase, a new foot placement is computed based on the current estimated CoM state and the remaining time to touchdown. Only the position part of the swing trajectory is modified, the rotational part remains the same. Let p^* be the original foot step given by the foot step planner, and p be the optimized foot placement. For the DRC walking controller in Section 4.2, the liftoff pose and p^* is used as the end knot points of a spline for generating the nominal swing foot pose x_d^*, velocity $\dot{x}_d^*$ and acceleration $\ddot{x}_d^*$ during swing, which are then used to compute the input target acceleration $\ddot{x}^*$ for the full body controller with Eq. 3.19. We have experimented with updating the knot point directly with p, but the resulting $\ddot{x}^*$ changes too drastically between time steps, and causes wild swing foot motions that can sometimes destabilize the walking cycle. Instead, we keep the spline interpolation the same, and compute $\ddot{x}^*$ with the following heuristic:

$$\begin{aligned} \ddot{x}^* &= K_p(x_d^* + \alpha(p - p^*) - x) + K_d(\dot{x}_d^* - \dot{x}) + \ddot{x}_d^* \\ \alpha &= \begin{cases} \frac{t - t_{LO}}{T_{SS}}, & \text{if } t < t_{LO} + T_{SS} \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{5.4}$$

where T_{SS} is the duration of the swing phase, t_{LO} is the liftoff time, and t is the current time. This scheme produces a much smoother $\ddot{x}^*$ since it only incorporates a portion of the new foot placement position at any time, and regulates the velocity towards the original interpolated one.

5.2.4 Robot Dynamic Walking Results

All the walking experiments in this section are done with the three level controller that has the CoM trajectory planner, the foot placement controller and the full body controller. To simplify the experimental setup, the next desired foot step is always updated with respect to the current stance foot location, so that the robot only tries to maintain balance rather than an absolute position after disturbances. This is also the simplest receding-horizon foot step planner.

The overall goal is to keep walking under strong disturbances. The first set of experiments are conducted with external kicks applied by humans around Atlas' pelvis, and the second set requires Atlas to walk over a strip of unstructured terrain made of loose rubble. Snapshots of these experiments are shown in Figure 5.2 and Figure 5.5(a). Unfortunately, we did not have instruments to measure the magnitude or the duration of these perturbations, and we are unable to estimate the net impulse due to noisy estimates of the CoM velocity. Fast walking is also attempted, and the fastest walking speed we have achieved is around $0.6m/s$. The walking controller is the same for all these experiments, and the high level CoM planner and the full body controller have few differences from Section 4.2.

Push Recovery

Figure 5.3 and Figure 5.4 show plots of the CoM states and the optimized foot steps when recovering from pushes in the sagittal and coronal plane respectively. The first thing worth noticing is that we are not controlling the CoM velocity very well, which is quite oscillatory during the single support phases. These oscillations also have a direct impact on the optimized foot placement shown with the orange lines in Figure 5.3(c) and Figure 5.4(c). The oscillations in foot placement require a large amount of smoothing and damping, otherwise they will cause large swing foot motions that induce further CoM oscillations. This motivates for only partially updating the position part of the swing foot acceleration computation in Eq. 5.4. However, this heuristic can introduce a large tracking error when the swing foot needs to move fast. Obviously, there is room

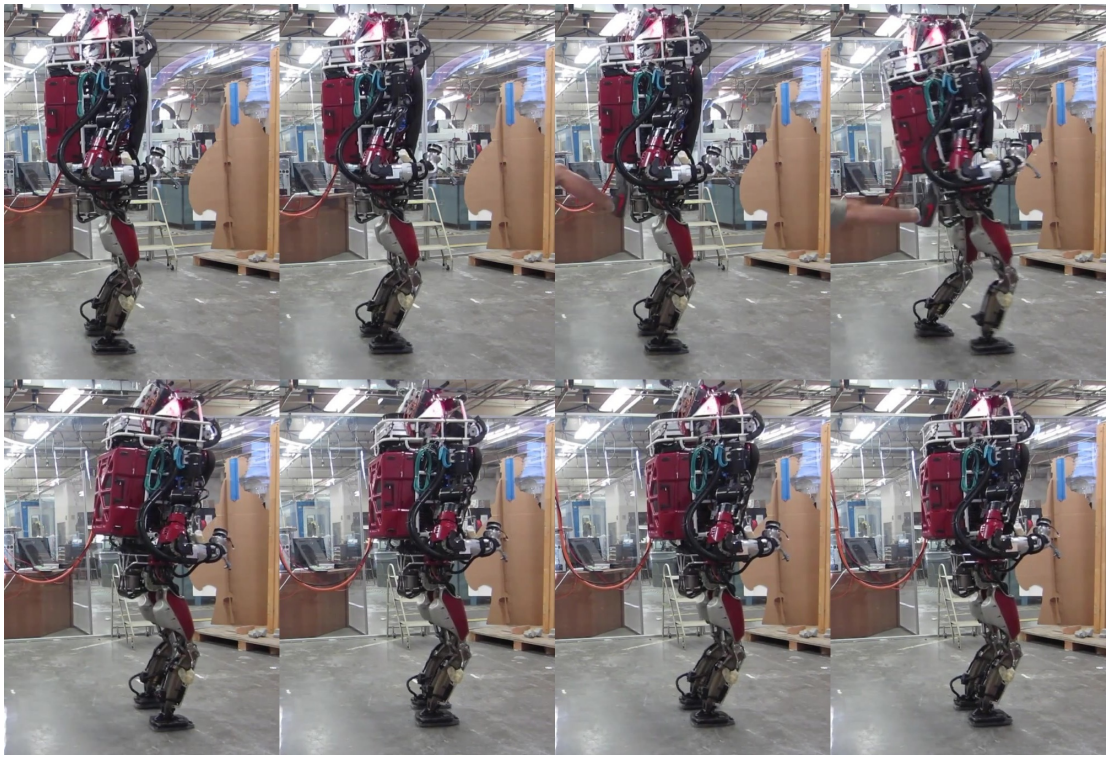
for improvements for our CoM and swing foot tracking, but this also shows that during high cadence dynamic walking, precise control is marginal, the robot can maintain dynamic balance as long as it can put down the swing foot somewhere reasonable at the right time. The effectiveness of CoP control is minimal comparing to taking steps.

Walking on Rubble

Figure 5.5 shows foot steps taken by Atlas when walking over a loose rubble field that is roughly $3m$ by $0.9m$. This experiment is particularly challenging, especially for the state estimator, because the fundamental assumption of stationary contacts is often violated. Physically, the actual support region for the stance foot is shrunk significantly, and the robot is effectively walking on stilts. Estimating the actual support is close to impossible in this case. It will very difficult to walk over this kind of terrain statically since precise CoP control will be incredibly hard. Many of the position controlled humanoids achieve compliance with feedbacks on the measured ground reaction wrench. We also speculate that these approaches will not work well on this terrain because of noisy measurements and limited bandwidth for their force feedback.

Fast Walking

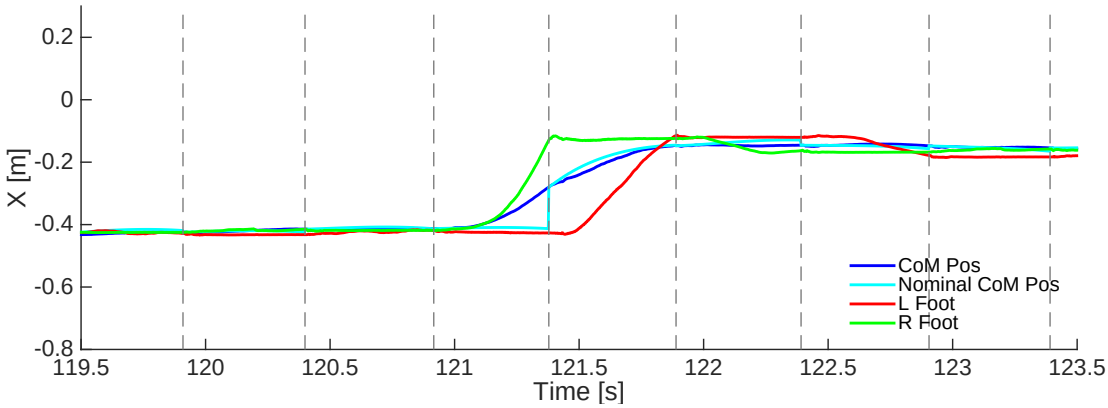
We want to explore how fast our Atlas can walk in the last set of experiments, and the fastest walking speed we achieved is roughly at $0.6m/s$. Data for this experiment are plotted in Figure 5.6. This speed is computed by averaging the estimated CoM velocity over a couple cycles once the robot starts walking forward. We think a hardware limit stopped us from going faster. The onboard pump is unable to deliver the amount of flow at the desired pressure when the robot is walking that fast. Our Atlas starts going down as it walks, which is also evident from large torque tracking errors for the stance knee. We are not able to walk for a longer distance due to limited lab space. We do not think this is the absolute limit for Atlas' walking speed, since a more powerful pump (potentially offboard) can easily solve this issue. On the other hand, our



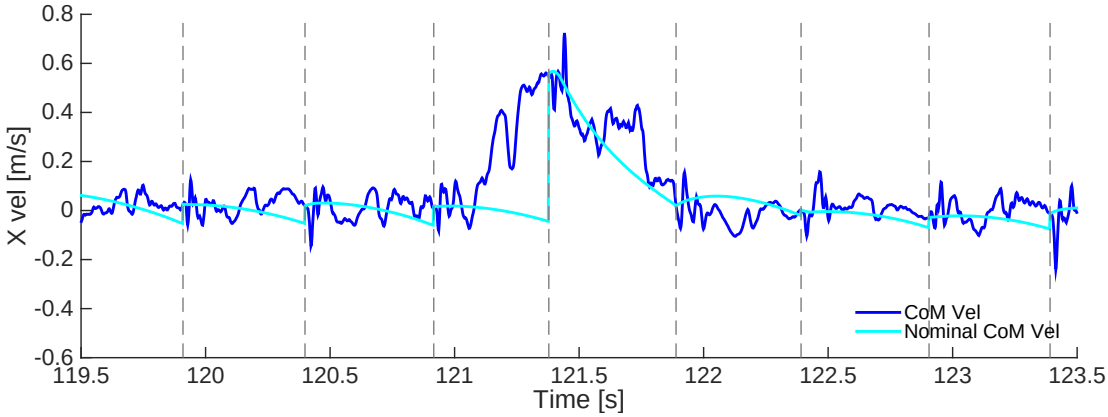
(a) Push in the sagittal plane

(b) Push in the coronal plane

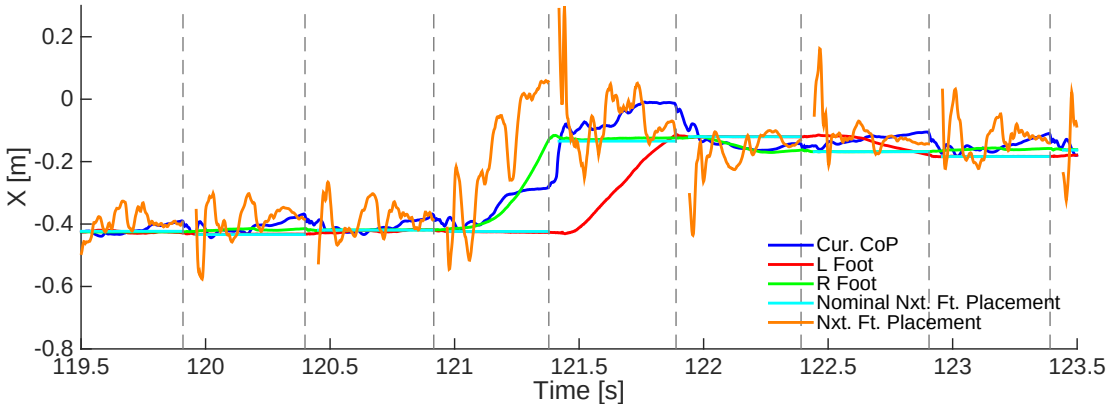
Figure 5.2: Snapshots of Atlas recover from external pushes by stepping. The snap shots are taken every $0.5s$. Data for these experiments are shown in Figure 5.3 and Figure 5.4.



(a) CoM position

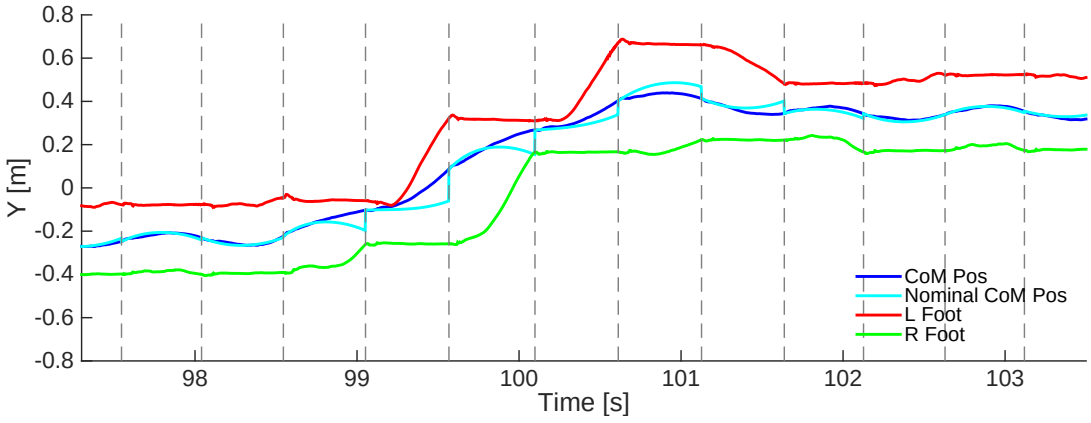


(b) CoM velocity

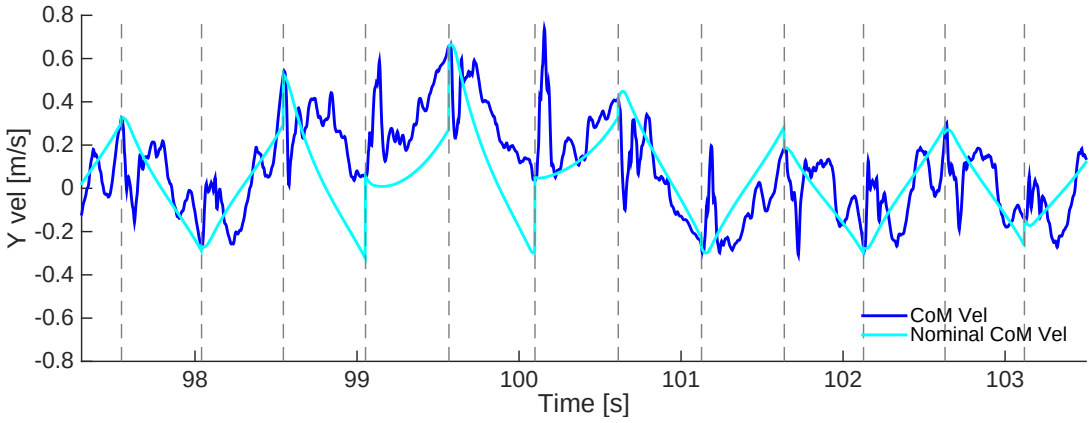


(c) CoP and optimized foot step

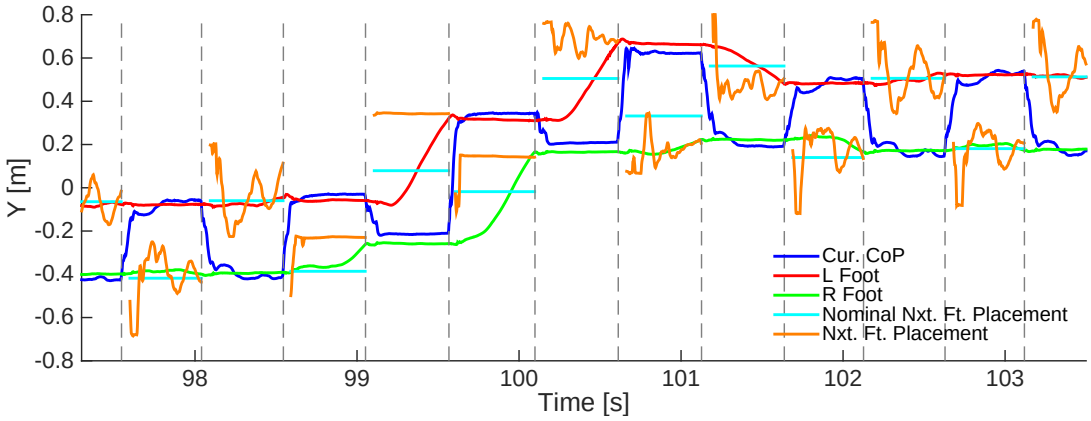
Figure 5.3: A forward push at Atlas's pelvis starts around $120.9s$ (illustrated by the third snapshot in Figure 5.2(a)), which is during the late right single stance and the following double support. Atlas is able to regain balance by taking just one forward step. The grey dashed lines indicate the touchdown events.



(a) CoM position

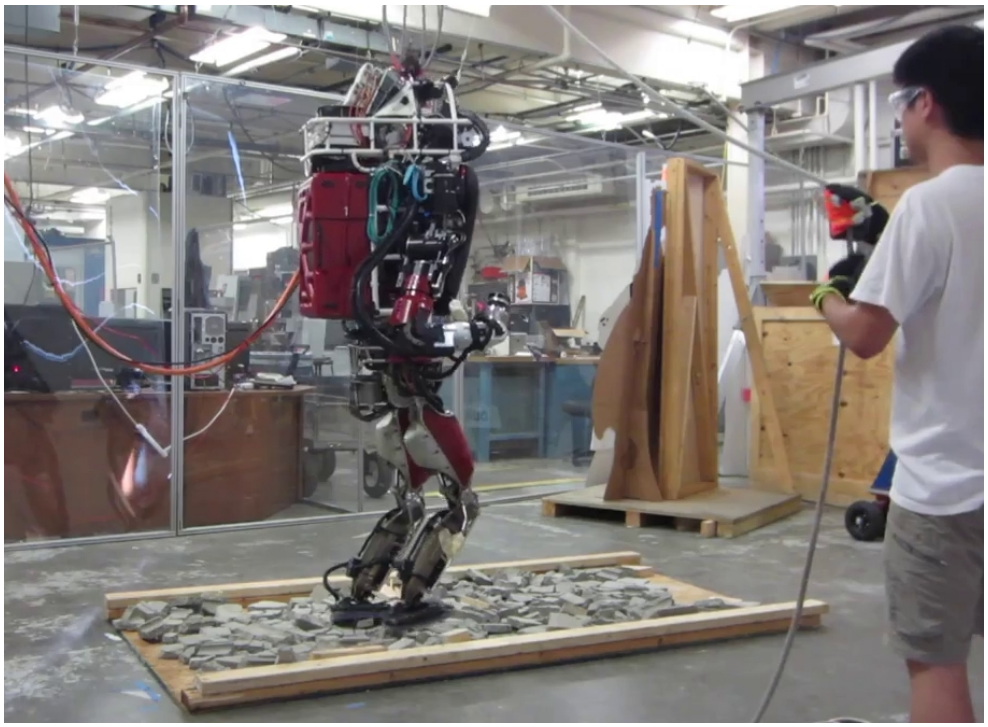


(b) CoM velocity

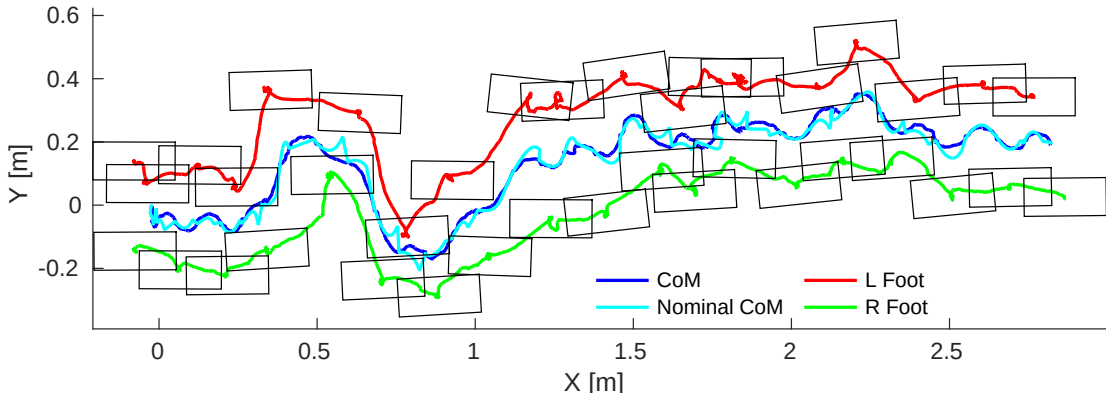


(c) CoP and optimized foot steps

Figure 5.4: A left kick at the right elbow starts around $98.4s$ (shown by the first snapshot in Figure 5.2(b)), which is during the late right single support, and it is too late to recover by extending the left swing leg. For the subsequent steps, the foot step optimizer tries to put the right foot close to the left, and extend the left foot as mush as possible to regain balance. The grey dashed lines indicate the touchdown events.

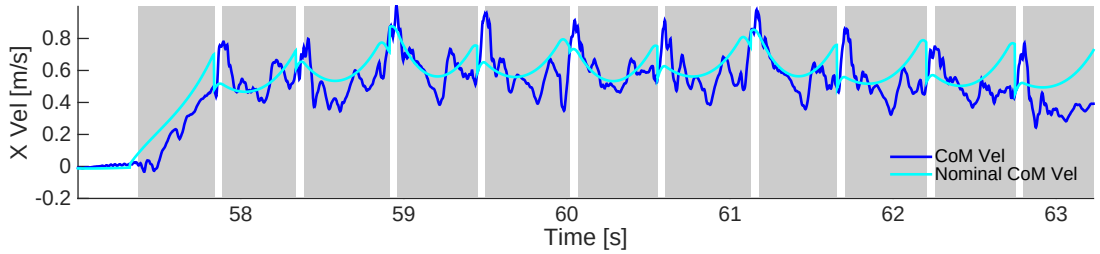


(a) Atlas walking over a rubble field

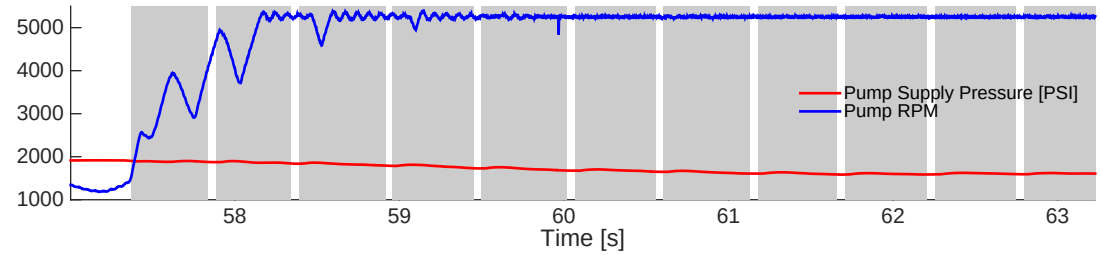


(b) Actual foot steps

Figure 5.5: The top picture shows Atlas walking over unstructured terrain made by pieces of cinder blocks and wooden blocks, which are not fixed to the ground. The rubble field is about $3m$ long and $0.9m$ wide. The bottom figure plots the foot steps Atlas takes and the actual CoM and foot trajectories through the rubble field.



(a) CoM velocity



(b) Pump data

Figure 5.6: Single support phase is shown with the shaded area. The onboard pump runs out of power once the robot starts walking fast, which is indicated by the supply pressure dropping and the pump motor saturating at top speed. The pump is unable to deliver the amount of flow at the desired pressure, and the robot's knee starts collapsing during this experiment. Atlas is walking at roughly $0.6m/s$ on average, which is the top speed we are able to achieve at the moment.

current implementation requires large knee torques since it maintains a bent stance knee to avoid singularities. Walking with straighter knees will reduce the power requirement, and we might be able to push the speed limit further.

5.3 Angular Momentum Controller

In this section, we use the Linear Inverted Pendulum with Flywheel Model (LIPFM) [82], illustrated in Figure 5.7, to capture center of mass level dynamics of the robot. This model is an extension to the well known LIPM with an extra flywheel at the CoM. Like LIPM, this model also decouples sagittal and coronal plane dynamics. For mass m, gravity g, height of CoM z,

inertia of the flywheel I and time step dt, the discrete time dynamics can be written as follows:

$$\begin{aligned}
\mathbf{X}_{t+1} &= A\mathbf{X}_t + B\mathbf{U}_t \\
\mathbf{X} &= \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix}, \mathbf{U} = \begin{bmatrix} p \\ \tau \end{bmatrix} \\
A &= \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ gdt/z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -gdt/z & dt/(mz) \\ 0 & dt/I \end{bmatrix},
\end{aligned} \tag{5.5}$$

where x is the horizontal position of CoM, θ is the angle of the flywheel with respect to vertical, p is the location of CoP, and τ is the torque applied at CoM. With a desired trajectory, Q and R, we first use DDP to plan a nominal trajectory with its value function and linear policy. The same model is used for both DDP and the receding-horizon controller.

5.3.1 Formulation

With a N time step horizon, the terminal state, $\mathbf{X}_{N+1}$, is linear with respect to $\mathbf{X}_1$ and the stacked controls, $\mathcal{U}$, shown in Eq. 5.6.

$$\begin{aligned}
\mathbf{X}_{N+1} &= \mathcal{A}\mathbf{X}_1 + \mathcal{B}\mathcal{U} \\
\mathcal{U} &= \begin{bmatrix} \mathbf{U}_1 \\ \vdots \\ \mathbf{U}_N \end{bmatrix} \\
\mathcal{A} &= A^N, \mathcal{B} = \begin{bmatrix} A^{N-1}B & \dots & AB & B \end{bmatrix}
\end{aligned} \tag{5.6}$$

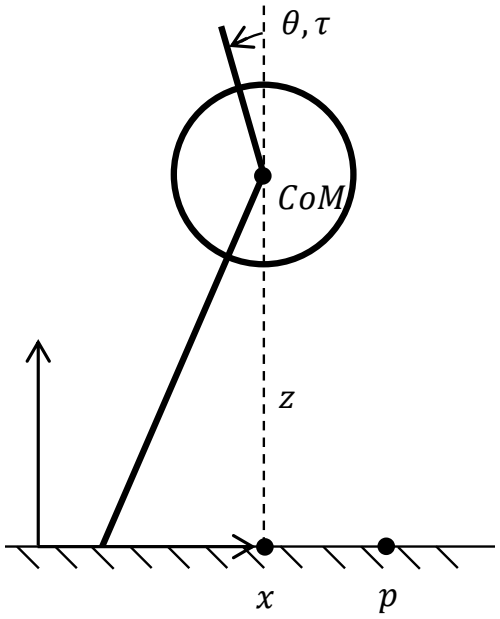


Figure 5.7: Illustration for the Linear Inverted Pendulum with Flywheel Model (LIPFM). CoM and CoP are denoted by x and p. The model assumes no vertical motion, and the height is z. The flywheel is at the CoM, and it can apply torque τ. θ is the angle of the flywheel with respect to veritical. The state is $[x, \theta, \dot{x}, \dot{\theta}]$, and the control is $[p, \tau]$.

The angular momentum controller is also formulated as a quadratic programming problem. The cost function has two components. The first one is a penalty on the terminal state using the value function approximation given by DDP, and the second part regularizes $\mathcal{U}$. For the first part,

$$\begin{aligned} c_{state} &= (\mathbf{X}_{N+1} - \mathbf{X}^*_{N+1})^T V_{N+1} (\mathbf{X}_{N+1} - \mathbf{X}^*_{N+1}) \\ &= (\mathcal{A}\mathbf{X}_1 + \mathcal{B}\mathcal{U} - X^*_{N+1})^T V'^T V' (\mathcal{A}\mathbf{X}_1 + \mathcal{B}\mathcal{U} - X^*_{N+1}) \\ &= (A_{state}\mathcal{U} - b_{state})^T (A_{state}\mathcal{U} - b_{state}) \\ A_{state} &= V'\mathcal{B}, b_{state} = V'(\mathbf{X}^*_{N+1} - \mathcal{A}\mathbf{X}_1), \end{aligned} \tag{5.7}$$

where $V'^T V' = V_{N+1}$. For regularizing $\mathcal{U}$,

$$
\begin{aligned}
c_{reg} &= \sum_{t=1}^{N} (\mathbf{U}_t - \mathbf{U}_t^*)^T R (\mathbf{U}_t - \mathbf{U}_t^*) \\
&= (A_{reg}\mathcal{U} - b_{reg})^T (A_{reg}\mathcal{U} - b_{reg}) \\
A_{reg} &= \begin{bmatrix} R'I_{2\times 2} & 0 & \dots & 0 \\ 0 & R'I_{2\times 2} & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R'I_{2\times 2} \end{bmatrix}, b_{reg} = \begin{bmatrix} R'\mathbf{U}_1^* \\ \vdots \\ R'\mathbf{U}_N^* \end{bmatrix},
\end{aligned}
\tag{5.8}
$$

where $R'^T R' = R$. A_{state}, b_{state}, A_{reg} and b_{reg} can be plugged into Eq. 3.7 together with $w_{state} = w_{reg} = 1$ to generate the cost function for the QP solver.

There are no equality constraints for this problem. p is bounded by the size of the support polygon, and we set some arbitrary torque limits on the flywheel. In this implementation, we do not constrain θ. For a more practical implementation, this is however necessary. We then need to include the inequality constraints for θ on every time step. This can be achieved similarly to Eq. 5.6.

5.3.2 Simulated Push Recovery Results

For this simulated experiment, the overall goal is to maintain balance at the center of the left foot in single support. We use LIPFM both for the high level planner and the angular momentum controller. Due to linear dynamics and fixed-point tracking, the high level controller reduces to a LQR controller. Two separate pairs of controllers are used for the sagittal and coronal plane: $m = 97kg$, $z = 0.85$, $I_s = 11$ and $I_c = 23$, where subscripts $_s$ and $_c$ stand for the sagittal and coronal plane respectively. $dt = 0.01s$ and $N = 60$. Q is a diagonal matrix with $[10, 0.01, 1, 0.01]$, and R is also a diagonal matrix with $[1000, 0.01]$. Only the ID module is used

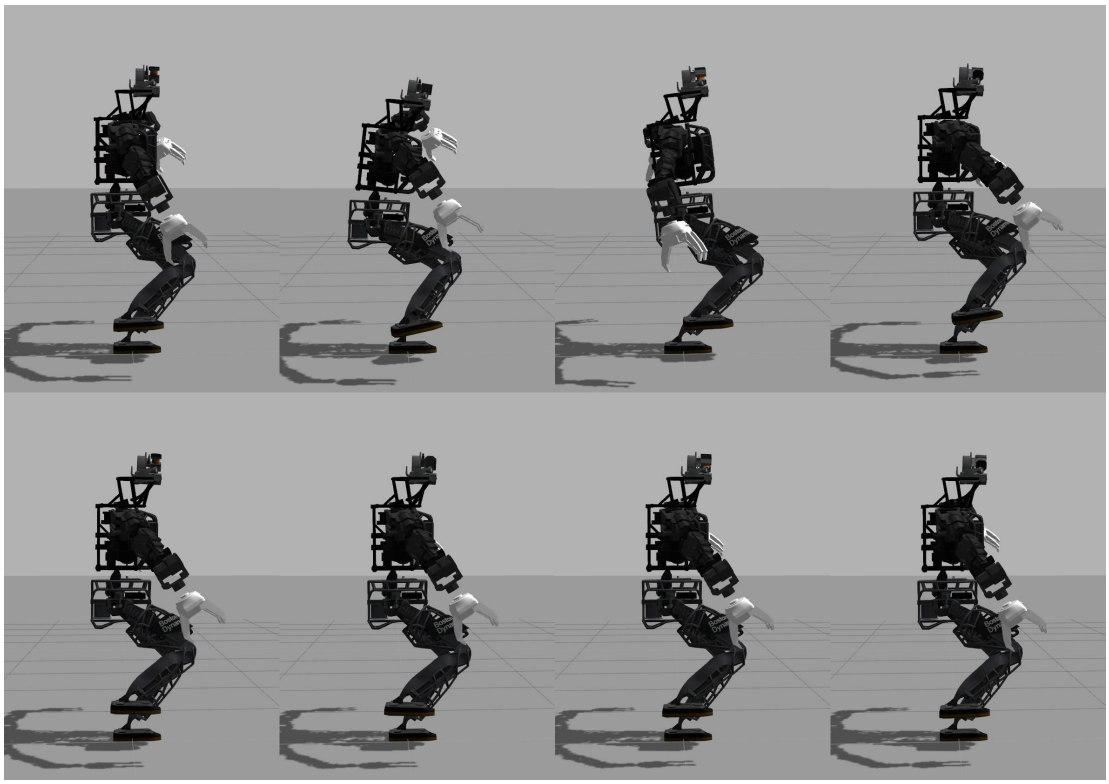
in the full body controller. The combined controller runs at $500Hz$.

Given the estimated angular momentum L and moment of inertia around the CoM I_{sys}, we compute a overall angular velocity $\omega_{sys} = I_{sys}^{-1}L$. Since there is no equivalent position term for angular momentum, we use the angle of the torso as a substitute. Input states for the simple model are computed by

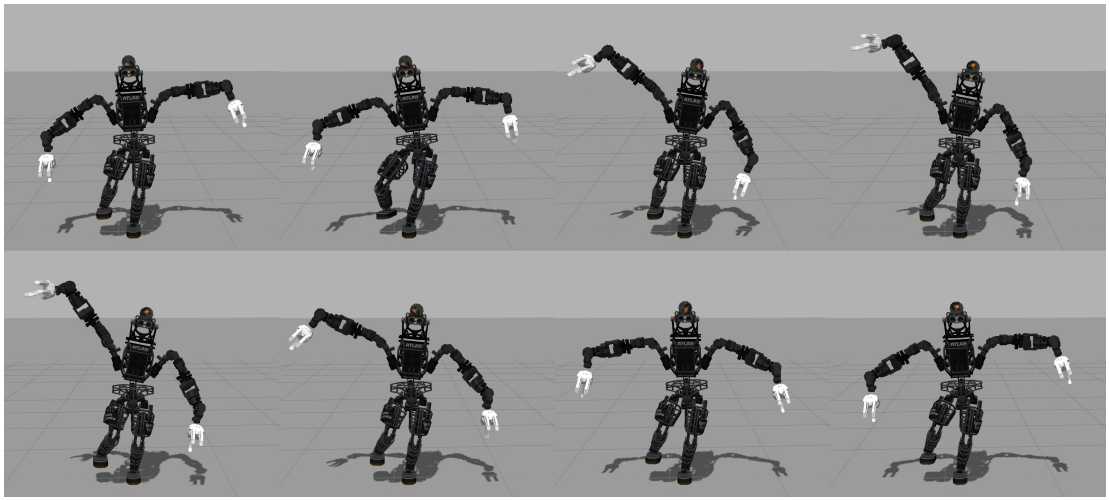
$$\mathbf{X}_s = \begin{bmatrix} x_{CoM}[X] \\ -\theta_{torso}[Y] \\ \dot{x}_{CoM}[X] \\ -\omega_{sys}[Y] \end{bmatrix}, \mathbf{X}_c = \begin{bmatrix} x_{CoM}[Y] \\ \theta_{torso}[X] \\ \dot{x}_{CoM}[Y] \\ \omega_{sys}[X] \end{bmatrix}. \tag{5.9}$$

The output CoP p is directly used as desired CoP in the ID controller. τ is treated as desired change in total angular momentum for the centroidal dynamics term in Section 3.5.2, and τ_s needs to be negated first. Desired CoM accelerations in the XY plane are also computed using LIPFM dynamics. CoM height is controlled with a PD servo as in Eq. 3.19, and desired change in angular momentum around the Z axis is set as a pure damping term.

For testing, we apply a constant force at the pelvis for $0.1s$ in the horizontal plane. Snapshots of two experiments are shown in Figure 5.8: one with $350N$ push in the negative X direction (backwards); and the other with $200N$ push in the positive Y direction (left). We also show a comparison between using only DDP and the proposed angular momentum controller in Figure 5.9. The force is applied at $19s$, and lasts for $0.1s$, which is indicated by the black vertical lines. The spikes at the end of the push in all the plots are due to a sudden jump in the pelvis acceleration, which is an artifact of the Gazebo simulator. In the DDP only case, control constraints are enforced after applying the linear policy in Eq. 2.3. Using DDP alone is not able to recover from the $200N$ push. It performs poorly when up against the CoP constraint. In contrast, the angular momentum controller uses the fly wheel to generate much larger CoM acceleration when CoP control is limited, and the ID controller is able to track the desired commands well.

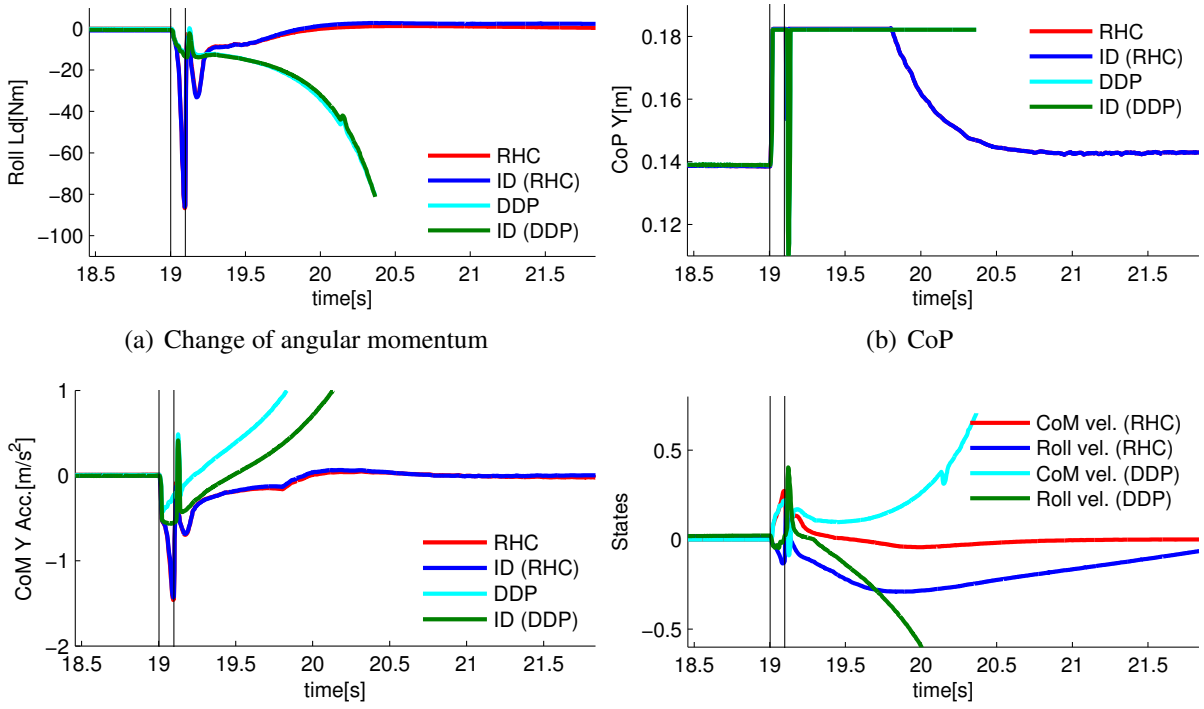


(a) Push in -X direction



(b) Push in Y direction

Figure 5.8: Snapshots of the simulated Atlas recovering from external pushes. In Figure 5.8(a), we push the robot backwards with $350N$ applied at pelvis for $0.1s$, and the snapshots are taken every second. In Figure 5.8(b), the robot is pushed towards its left with $200N$ at pelvis for $0.1s$, and the snapshots are taken every $1.75s$. In both case, the push is applied slightly before the second snapshot.



(a) Change of angular momentum

(b) CoP

(c) CoM acceleration

(d) Linear and angular velocity

Figure 5.9: This figure shows a comparison between using DDP alone and the angular momentum controller against the same $20Ns$ push at the pelvis to the robot's left. Using DDP alone is not able to recover from the push. For Figure 5.9(a), Figure 5.9(b) and Figure 5.9(c), inputs and outputs of the inverse dynamics controller are shown, where the inputs are generated by either DDP or the angular momentum controller. ID stands for the output of the inverse dynamics controller. RHC and DDP stand for the angular momentum controller and just using DDP's policy. For both experiments, the velocity traces are shown in Figure 5.9(d).

5.4 Summary

In this chapter, we add a receding-horizon layer between the long term CoM planner and instantaneous full body controller. Two versions of the new component are implemented: one focuses on optimizing swing foot placement, and the other utilizes angular momentum. Although only the foot placement controller is implemented on the real robot, both enlarge the stability region and are much more effective against external perturbations when compared with the two level setup in Chapter 4. In particular, the foot placement controller enabled our Atlas to recover from large pushes and walk over loose rubble fields.

The changes to both the CoM planner and the full body controller are minimal when implementing the foot placement controller after the DRC Finals. Once implemented, little tunning is necessary for successful robot deployment. From past experiences during the DRC, precise control of the CoM position and velocity is critical for a good static walking controller. This requires accurate and low delay state estimation for the CoM state, especially for the velocity, as well as good CoP and force control at the contacts. Correctly estimating the CoM modelling error also plays an important role. In contrast, foot placement is much more important for dynamic walking. In some sense, the robot is constantly falling down, and it just need to put the foot in the right place at the right time to start "falling" according to the plan. The control authority through foot placement is magnitudes bigger than controlling the CoP within the support region. We no longer require fine control over the CoM states, which greatly reduces performance requirements for the low level controllers and state estimators. When planned in a receding-horizon fashion, foot placement does not have to be very precise as long as the robot can roughly capture itself in the next step. On the other hand, timing is important. Dynamic walking is much easier in the sense that its error tolerance for all the individual components is much larger than for static walking.

Chapter 6

Future Work and Conclusion

6.1 Planning with Simple Models

Simple models are easy to understand and provide us with important intuitions. They are also powerful tools for analysis. Using simple models imposes structures on the underlying problem and reduces the search space. Simpler optimization problems are also less likely to encounter numerical and local minima issues. We think planning with simple models can still be beneficial even without computational limits. We want to extend our simple planners in the following ways to improve the current walking controller:

- add angular momentum to the high level trajectory planner
- combine the foot placement and the angular momentum controller
- optimize step timing

We think these modifications can be achieved in a relatively short period of time, and they will improve the current implementation in terms of robustness and potentially agility as well. Beyond these, we also want to address the question of when do we need more of a receding-horizon component. We think replanning using sensed states and environment is important, but the dividing line between a "planner" and a "receding-horizon controller" is rather vague. We imposed a

distinction based on different computation budgets, but it might not be necessary in the future. We are also interested in determining the right complexity for the models used for planning. Aside from high computational costs, it is possible for the optimizer to fall into unsatisfactory local minima or fail to find an solution due to the complexity. Contact dynamics are hard to model correctly, and taking derivatives around contact events can be difficult as well. Receding-horizon control with full body dynamics has been experimented with in [16], where special contact models are used to speed up and stabilize the trajectory optimization. However, the motions generated by this controller can be counter-intuitive at times, and it is much harder to decipher the outputs with such high dimensional models. Inverted pendulum based models capture typical walking well, and yet they are extremely simple. Finding similar simple models for manipulation or multi-contact locomotion will be very helpful. Although more complex, planning with centroidal momentum [13] seems to be a likely candidate. Demonstrating these dynamic locomotion skills on real hardware is another very interesting challenge.

6.1.1 Step Timing

Touchdown timing is not optimized in the foot placement controller described in Section 5.2 due to nonlinearity of the optimization. On the other hand, we think timing is another crucial aspect for the stepping strategy. In order to test this hypothesis, another simple stepping controller is developed and tested in simulation without constraints on computation. The problem setup is almost identical to the simple example presented in Section 5.2.2, except we optimize for the time to touchdown t as well. This controller treats X and Y independently, although they need to be coupled for a real implementation because of timing [99]. Let $f(\mathbf{X}, t, p)$ be the nonlinear function that describes the CoM evolution starting from $\mathbf{X}$ based on LIPM. We can write the

touchdown state as $\mathbf{X}_{TD} = f(\mathbf{X}, t, p_{cur})$. The nonlinear optimization problem is formulated as:

$$\begin{aligned} &\min_{t,p} \sum_i (f(\mathbf{X_{TD}}, T_i, p) - \mathbf{X}^*_{T_i})^T V_{T_i} (f(\mathbf{X_{TD}}, T_i, p) - \mathbf{X}^*_{T_i}) + w(p - p^*)^2 \\ &\text{s.t. } t_{min} \leq t \leq t_{max}, p_{min} \leq p \leq p_{max}, \end{aligned} \tag{6.1}$$

where $\mathbf{X}^*_{T_i}$ and V_{T_i} are from the nominal trajectory sampled at T_i after touchdown. For the X direction, $p_{min} = -1 + p_{cur}$, and $p_{max} = 1 + p_{cur}$. In the lateral direction, $p_{min} = -0.6 + p_{cur}$, and $p_{max} = -0.17 + p_{cur}$ for single support left, and $p_{min} = 0.17 + p_{cur}$, and $p_{max} = 0.6 + p_{cur}$ for the right side. Timing's inequality constraints are set based on the optimized touchdown time t_{prev} from the last step: $t_{min} = \max(t_{prev} - 0.05, 0)$, $t_{max} = \min(t_{prev} + 0.05, 2)$. For the beginning of the swing phase, $t_{prev} = 0.5s$, which is the nominal cycle time. This is to very crudely simulate motion constraints on the swing foot.

Two sets of push recovery tests are conducted in the X and Y directions. The desired behavior is to walk forward at $1.2m/s$ on average in the X direction, and step in place for the Y direction. The nominal walking cycle is $0.5s$, and the desired foot steps are $0.6m$ and $0.34m$ apart for X and Y respectively. CoM height is set to $0.88m$, and the simulation time step is $0.002s$. Although not planned for, CoP control is available during simulation. It is generated by the DDP policy and bounded by the size of the stance foot ($\pm 0.12m$ for X, and $\pm 0.04m$ for Y). For both X and Y, three strategies are tested for robustness: foot placement only, step timing only, and using both. The basins of attraction are plotted in Figure 6.1, and two example trajectories are shown in Figure 6.2. Perturbations are given as instantaneous CoM velocity changes, and they are plotted in the Y axis. The X axis shows the time of perturbation during the walking cycle. The color represents number of steps taken before converging into a steady state cycle, and the white spaces indicate falling. With LIPM dynamics, the CoM acceleration is determined by the relative position of the CoM and the current stance foot, and the CoM state evolves exponentially with respect to time. Being able to adjust the stance duration is very important for speeding up or slowing down the CoM especially with limited swing foot reachability. For the lateral direction,

the CoM can be accelerating towards the wrong side half of the time during push recovery, so minimizing those stance phases should help. For the timing only strategy, we can examine it from a capture point of view. Essentially, the system can not be stabilized when the CoM velocity (energy) exceeds a certain threshold given the current state and the sequence of stances (capture region). The effectiveness of changing step timing is limited by future foot steps. Combining both strategies greatly enlarges the margin of stability as shown in Figure 6.1. We think this is a really promising direction for maximizing robustness during walking. Although the optimization problem becomes more complex, we are still confident about a real-time implementation soon.

6.1.2 Gaps Between Hierarchies

To be computationally feasible for real-time control, we used a cascade of smaller optimizations with different models. However, this naturally introduces gaps between the hierarchies because the models have different expressive powers. Although practical and effective, our approach suffers from the common problem that plagues hierarchical systems: how to generate a different high level plan when something goes wrong at a lower level, which is not even modeled or represented by the high level planner? Currently, we do not have the bottom-up channel to propagate information up the hierarchy. A possible solution is to introduce state dependant cost biases in the low dimensional state space, e.g. lower CoM height at the beginning and end of the swing phase to increase swing leg reachability. This information can be gathered by running the simulated full controller offline for many different scenarios. Using policies is another practical way to introduce these biases.

In terms of the top-down channel, we pass the value function that encodes the high level planner's future preferences down to the lower level modules. This gives them limited "foresight" and guides them towards the high level goal in their limited horizon [47]. We have experimented with providing the approximated local value function computed by DDP to the inverse dynamics controller, but we have yet to observe a significant increase in either performance or stability.

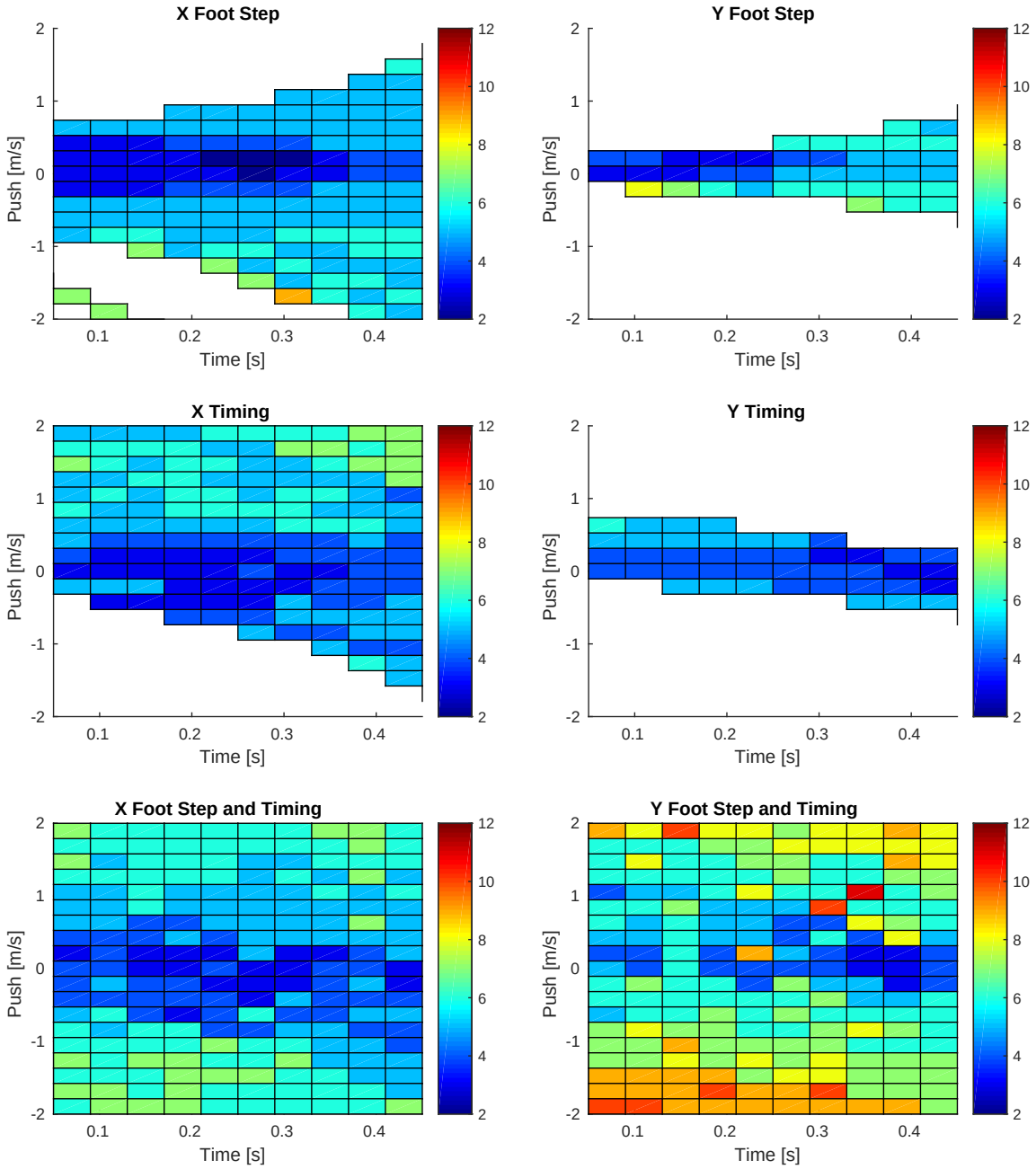


Figure 6.1: These plots show the number of steps taken after perturbation using three stepping strategies. Blue blocks correspond to quick recoveries, and slow recoveries are marked by red. Falls are indicated by the white blocks. The desired behavior is to walk forward at $1.2m/s$ on average in the X direction, and step in place for the Y direction. Perturbation is given as a CoM velocity change, and it is plotted in the Y axis. The X axis shows the time of perturbation after touchdown. The lateral pushes happens during single support right.

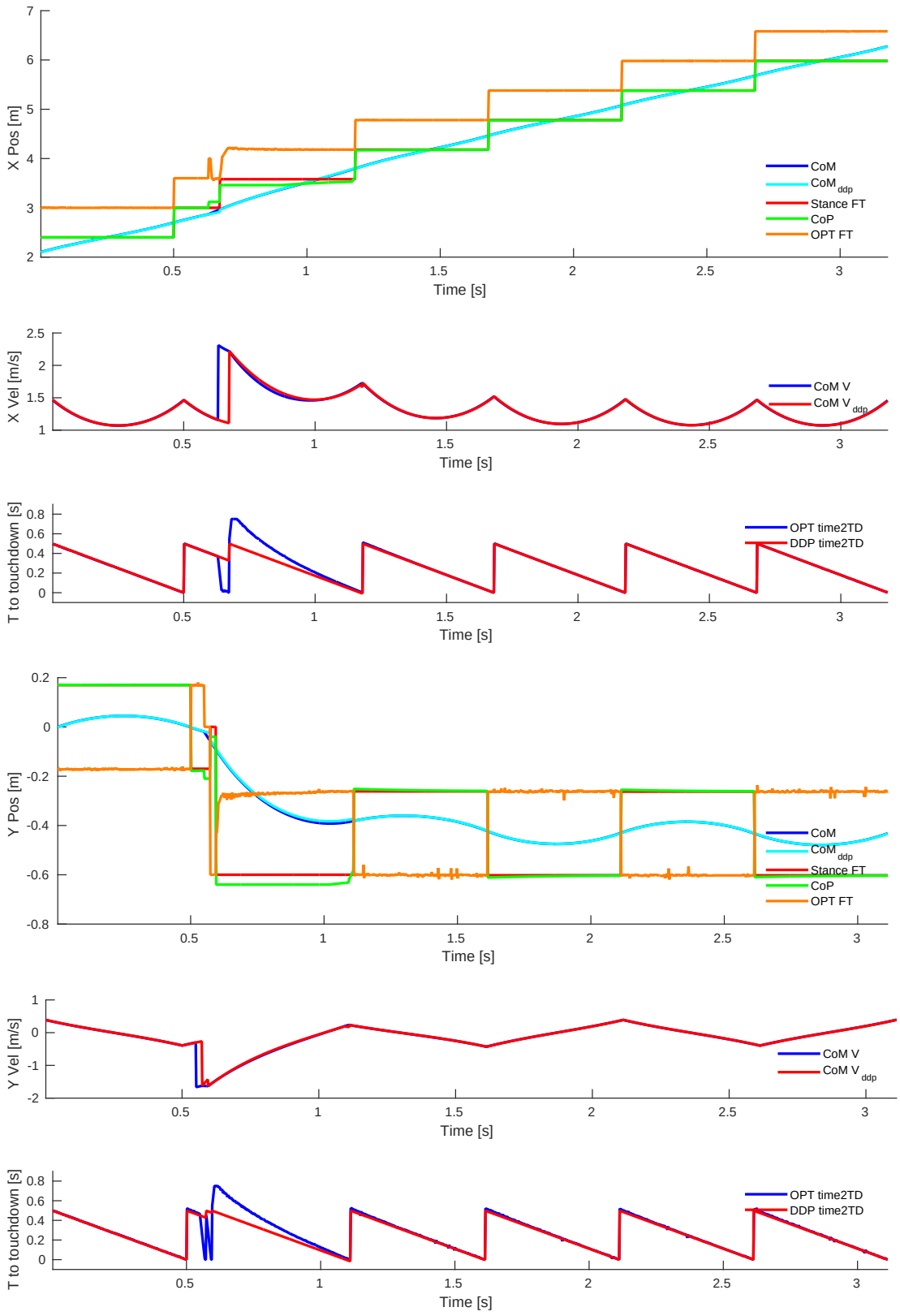


Figure 6.2: Trajectories of push recovery by optimizing both foot placement and touchdown timing.

The input desired CoM acceleration to the ID controller is generated by the DDP policy, which is derived together with the value function approximation. So adding this particular value function approximation in ID's cost function does not provide much additional information. However, with a longer horizon or a different value function approximation, e.g. from a trajectory library or derived with full dynamics, the difference should be more significant. Investigating this issue can also provide useful insights to whether planning with full dynamics is useful or not.

In general, we think the ideal solution is to include as complete a model as possible in the high level planner and replan as often as possible. On the other hand, this is harder for analysis and acquiring physical intuition as opposed to using simple models. Aside from the obvious computational costs, using more complex models can potentially make the optimization fragile and have more local minima issues. We still need to find the right balance between model complexity and performance.

6.1.3 Utilizing Offline Planning

For fixed behaviors, generating the optimal control is trivial when given a value function. We can represent the value function using a library of local approximations [50] either derived with full dynamics or simple models. Offline computation can also be used to spawn a large range of nominal trajectories, and with proper low dimensional embeddings [41], the online search space can be reduced. The cached nominal trajectories can also guide subsequent joint level motion generation. During the DRC Finals, most of the walking related issues were caused by kinematic constraints such as ankle joint limit and knee singularity. We worked around them with special purpose joint space heuristics, which are not generalizable. Trajectory optimization is useful for automatically generating these complex behaviors [13, 56], which can then be stabilized using the full body controller during execution.

6.2 Full Body Control

6.2.1 Singularity

Since most of the desired behaviors are specified in Cartesian space, computing joint space accelerations requires a matrix inversion at some level. This is susceptible to singularity, which is why most of the humanoid robots walk with bent knees. In the current implementation, we mitigate this issue by regulating joint acceleration when it is close to singularity. One way to fundamentally address this issue is to directly specify joint space references. For regular walking, it might be sufficient to control the swing knee and hip using simple feedbacks on the desired foot position and touchdown timing. Establishing contact at the right place at the right time is important, but tracking some arbitrary swing foot trajectory precisely is entirely artificial. To avoid obstacles properly, joint space trajectories should be planned in any case. Another possible direction is to significantly reduce the Cartesian gains in the singular directions. We should be able to rotate the desired Cartesian motions and the corresponding gains into the space spanned by the manipulability ellipsoid, and then reduce the gains in the singular direction.

6.2.2 Adding Kinematic Constraints

Inverse kinematics can be embedded in inverse dynamics. Specifying position and velocity constraints in terms of acceleration can be easily done with Taylor expansion. For constraints in generalized coordinates, we have

$$\begin{aligned} q^- &\leq q_t + \dot{q}_t dt + 0.5\ddot{q}dt^2 \leq q^+ \\ \dot{q}^- &\leq \dot{q}_t + \ddot{q}dt \leq \dot{q}^+, \end{aligned} \tag{6.2}$$

and for constraints in Cartesian space,

$$\begin{aligned} x^- &\leq x_t + J\dot{q}_t dt + 0.5(J\ddot{q} + \dot{J}\dot{q}_t)dt^2 \leq x^+ \\ \dot{x}^- &\leq \dot{x}_t + (J\ddot{q} + \dot{J}\dot{q}_t)dt \leq \dot{x}^+. \end{aligned} \tag{6.3}$$

We briefly experimented with adding joint limits when preparing for the DRC Finals, but decided not to use them in the end. We ran into an issue about the ID controller generating large accelerations when some kinematic constraints were violated. The inequality constraint forces an acceleration that snaps the position back to the constraint surface in the next time step, which can cause stability issues. Adding these inequality constraints also slows down the QP solver. So for the DRC Finals, we added potential wells in the cost function to force the joints away from their limits. We need to investigate this issue more in the future.

6.2.3 Compensating for Modeling Errors

We introduce a generalized force τ_{err} in the equations of motion to explain the difference between the model's prediction and physical reality. For the preliminary results presented here,

$$\begin{aligned} r &= -M\ddot{q}_m - h + S\tau_m + J^T\lambda_m \\ \tau_{err} &= \alpha\tau_{err} + (1-\alpha)r. \end{aligned} \tag{6.4}$$

M, h and J^T are computed with the model, and $\ddot{q}_m$, τ_m and λ_m are measured by the sensors. $\ddot{q}_m$ is measured by finite differencing the velocity signals. S is a selection matrix, where the first six rows are zeros, and the rest form an identity matrix. r represents the residual of the equations of motion when the measured signals are plugged in. We hope to use the estimator in [103] as a replacement for Eq. 6.4 in the future. $\alpha = 0.99$ for the experiments here. In the ID controller, this term can be directly added to h, which is the sum of gravitational and Coriolis and centrifugal forces.

In the following simulated example, a different robot model is used in the controller, which

has heavier hands. We also add a constant $10Nm$ offset to the left knee torque command given to the simulator, and a $-10Nm$ offset in the measured left knee torque. With τ_{err} enabled around $7.2s$ in Figure 6.3, the controller is able to compensate for these modeling errors and bring the robot back to the nominal pose. The model used in the controller is about $3.5kg$ heavier than in simulation. The weight difference is roughly $34N$ in terms of gravitational force, and matches the estimated generalized force in the Z axis shown with the red trace in Figure 6.3(g). The estimated compensation torque for the left knee is roughly $10Nm$ smaller than the right knee, which correlates with the added fictional torque well.

6.2.4 Compensating for Fixed Delays

Assuming a fixed T time step delay in the torque commands, instead of using the current measured state $(q_t, \dot{q}_t)$ in the inverse dynamics calculation, we use a forward simulated state $(q_{t+T}, \dot{q}_{t+T})$, which is generated from the current state and the past commands using approximated forward dynamics.

Let N_c be the number of contacts, and N is the dimension for the generalized acceleration. Assuming the $6N_c$ contact constraints are linearly independent, we can perform QR decomposition on J^T [54],

$$J^T = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{6.5}$$

where Q is orthogonal, and R is an upper triangular matrix of rank $6N_c$. Multiplying the equations of motion by Q^T, we have

$$\begin{aligned} S_c Q^T (M\ddot{q} + h) &= S_c Q^T S\tau + R\lambda \\ S_u Q^T (M\ddot{q} + h) &= S_u Q^T S\tau \\ S_c &= \begin{bmatrix} I_{6N_c \times 6N_c} & 0_{6N_c \times (N-6N_c)} \end{bmatrix} \\ S_u &= \begin{bmatrix} 0_{(N-6N_c)\times 6N_c} & I_{(N-6N_c)\times(N-6N_c)} \end{bmatrix}. \end{aligned} \tag{6.6}$$

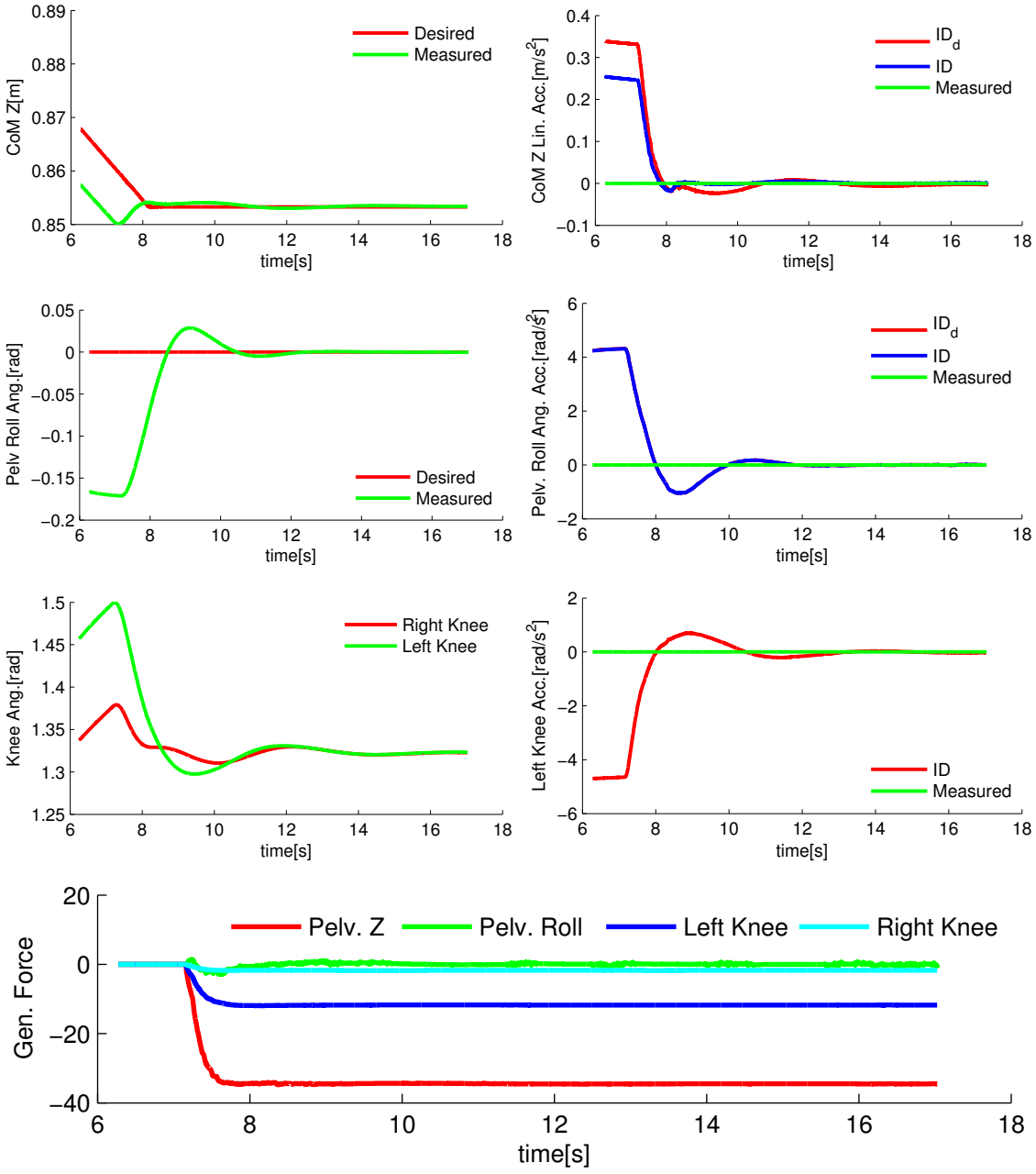


Figure 6.3: The first three rows correspond to CoM, pelvis roll and knee. Position and acceleration data are shown in the left and right column respectively. The bottom plot shows the estimated generalized force τ_{err}. Since there is no desired knee joint angle, we use the right knee as a reference.

Assuming no contact force constraints and the zero contact acceleration, $\ddot{q}$ can be computed by

$$\begin{aligned} \ddot{q} &= M_{QR}^{-1}\beta \\ M_{QR} &= \begin{bmatrix} S_u Q^T M \\ J \end{bmatrix} \\ \beta &= \begin{bmatrix} S_u Q^T (S\tau - h) \\ -\dot{J}\dot{q} \end{bmatrix}. \end{aligned} \tag{6.7}$$

For each time step $t+i$, $\ddot{q}_{t+i}$ is computed using Eq. 6.7, and $(q_{t+i+1}, \dot{q}_{t+i+1})$ is generated with numerical integration.

6.3 Conclusion

We focus on using online optimization to control a full size humanoid robot in this thesis. In order to enable real-time application, a hierarchical scheme is implemented. It has a top level center of mass trajectory planner, a receding-horizon layer that uses stepping or angular momentum for balance, and an inverse dynamics based full body controller for generating joint level commands. The top level planner reasons about long term goals using a simple model, and the lower level modules use progressively more complete models and constraints but optimize for shorter horizons into the future. In addition to a nominal trajectory, a value function approximation generated by the top level planner is passed down the hierarchy to encode preferences about the future states and guide the lower level optimizations. The receding-horizon component closes the loop by rapidly replanning a short trajectory that tracks the long term goal considering critical physical constraints. The low level full body controller abstracts away the robot details and provides a behavior level interface, which greatly facilitates application design and implementation.

With just the CoM trajectory planner and the full body controller, we have achieved relatively

reliable walking and static manipulation capabilities. These controllers were successfully developed for the Atlas robot and demonstrated in the DARPA Robotics Challenge. We completed two one-hour long missions at the DRC Finals without falling or any physical human intervention, and we scored 7 out of 8 points for both missions. A receding-horizon component that optimizes foot placement was later introduced to enable faster and more robust dynamic walking on the robot. With the complete controller, our Atlas can withstand large external pushes, walk over unstructured terrain made by loose rubble, and achieve a top speed of $0.6m/s$ for flat ground walking.

To summarize, we list a few lessons that we learned from this work and important directions for future work:

- An open loop sense-plan-act approach does not work well in the real world. Using feedback and replanning in real-time is critical for developing robust systems. This is achieved by the receding-horizon layer in this thesis.
- Simple models are powerful tools for analysis and optimization. Even with unlimited computation, they can still be beneficial because of simplicity and the added structure, which simplifies controllers and may make them more robust. On the other hand, due to their limited expressive powers, planners using simple models either generate potentially infeasible plans or become overly conservative. We still need to find a better balance between model complexity and performance.
- In terms of motion tracking, torque control alone performs poorly for lightly-loaded joints, e.g. ones on the swing leg, due to modeling errors. We used a hybrid torque and velocity control scheme to trade off compliance during swing for motion accuracy. Another potential solution is to directly estimate and compensate for the modeling errors.
- Foot placement and timing are both important for dynamic walking, and we should start optimizing both for bigger stability margins. Angular momentum provides more control authority during single stance, and should be incorporated for planning and control.

- Although sensing and state estimation are not discussed in depth here, they are critical for model-based full state feedback approaches. Comparing to actuators, sensors are becoming cheaper, smaller and much more accurate. Adding redundant sensing on the robot improves state estimation and increases the overall performance of the system [103]. With additional touch sensors, we can estimate the contact locations and control the contact wrenches properly in the full body controller, so that we can start designing contact rich behaviors.

Bibliography

[1] Z. Aftab, T. Robert, and P.-B. Wieber. Ankle, hip and stepping strategies for humanoid balance recovery with a single model predictive control scheme. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 159–164, Nov 2012. doi: 10.1109/HUMANOIDS.2012.6651514. 5.1

[2] K. Akachi, K. Kaneko, N. Kanehira, S. Ota, G. Miyamori, M. Hirata, S. Kajita, and F. Kanehiro. Development of humanoid robot HRP-3P. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 50–55, Dec 2005. doi: 10.1109/ICHR.2005.1573544. 1.1

[3] Christopher G. Atkeson. Using local trajectory optimizers to speed up global optimization in dynamic programming. In *Advances in Neural Information Processing Systems*, pages 663–670. Morgan Kaufmann, 1994. 1.1, 2.2

[4] Christopher G. Atkeson and Benjamin J. Stephens. Random sampling of states in dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, pages 38(4), 924–929, 2008. 1.1

[5] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957. 1.1

[6] Pranav Audhut Bhounsule. *A controller design framework for bipedal robots: Trajectory optimization and event-based stabilization*. PhD thesis, Cornell University, Ithaca, NY, USA, May 2012. 1.1

[7] Thiago Boaventura. *Hydraulic Compliance Control of the Quadruped Robot HyQ*. PhD

thesis, Istituto Italiano di Tecnologia (IIT) and University of Genova, 2013. 3.8.1

[8] K. Bouyarmane and A Kheddar. Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4414–4419, San Francisco, CA, USA, Sept 2011. doi: 10.1109/IROS.2011.6094483. 1.1, 3.1

[9] K. Bouyarmane, J. Vaillant, F. Keith, and A Kheddar. Exploring humanoid robots locomotion capabilities in virtual disaster response scenarios. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 337–342, Osaka, Japan, Nov 2012. doi: 10.1109/HUMANOIDS.2012.6651541. 3.1

[10] G. Cheng, S. Hyon, Jun Morimoto, A. Ude, G. Colvin, W. Scroggin, and S.C. Jacobsen. CB: A humanoid research platform for exploring neuroscience. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 182–187, Dec 2006. doi: 10.1109/ICHR.2006.321382. 1.1

[11] Joel Chestnutt. *Navigation Planning for Legged Robots*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, December 2007. 1.1

[12] Steven H. Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001. doi: 10.1177/02783640122067561. 1.1

[13] Hongkai Dai, A. Valenzuela, and R. Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 295–302, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041375. 1.1, 6.1, 6.1.3

[14] Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. Feature-based locomotion controllers. *ACM Trans. Graph.*, 29(4):131:1–131:10, July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1781157. 1.1, 3.1

[15] Dimitar Dimitrov, Alexander Sherikov, and P. B Wieber. A sparse model predictive control formulation for walking motion generation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2292–2299, San Francisco, CA, USA, Sept 2011. doi: 10.1109/IROS.2011.6095035. 2.1

[16] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov. An integrated system for real-time model-predictive control of humanoid robots. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, Atlanta, GA, USA, 2013. 2.2, 2.3, 6.1

[17] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014. doi: 10.1177/0278364914521306. 3.1

[18] S. Faraji, S. Pouya, and A. Ijspeert. Robust and agile 3D biped walking with steering capability using a footstep predictive approach. In *Robotics: Science and Systems (RSS)*, Berkeley, CA, USA, July 2014. 2.1, 5.1

[19] S. Feng, X. Xinjilefu, W. Huang, and C. Atkeson. 3D walking based on online optimization. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, Atlanta, GA, USA, 2013. 1.1

[20] Siyuan Feng, Eric Whitman, X. Xinjilefu, and Christopher G. Atkeson. Optimization-based full body control for the DARPA Robotics Challenge. *Journal of Field Robotics*, 32 (2):293–312, 2015. ISSN 1556-4967. doi: 10.1002/rob.21559. 3.1, 3.8.4, 4

[21] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Spring-mass running: simple approximate solution and application to gait stability. *Journal of Theoretical Biology*, 232 (3):315 – 328, 2005. ISSN 0022-5193. doi: http://dx.doi.org/10.1016/j.jtbi.2004.08.015. 1.1

[22] A. Herdt, N. Perrin, and P.-B. Wieber. Walking without thinking about it. In *Intelligent*

Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 190–195, Oct 2010. doi: 10.1109/IROS.2010.5654429. 2.1, 5.1

[23] Andrei Herdt, Holger Diedam, Pierre-Brice Wieber, Dimitar Dimitrov, Katja Mombaur, and Moritz Diehl. Online Walking Motion Generation with Automatic Foot Step Placement. *Advanced Robotics*, 24(5-6):719–737, 2010. doi: 10.1163/016918610X493552. 5.1

[24] Ayonga Hereid, Shishir Kolathaya, Mikhail S. Jones, Johnathan Van Why, Jonathan W. Hurst, and Aaron D. Ames. Dynamic multi-domain bipedal walking with atrias through slip based human-inspired control. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 263–272, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2732-9. doi: 10.1145/2562059.2562143. 1.1

[25] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal. Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*, Chicago, IL, USA, Sept 2014. 1.1, 3.1, 3.5

[26] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of Honda humanoid robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1321–1326 vol.2, Leuven, Belgium, 1998. doi: 10.1109/ROBOT.1998.677288. 1.1

[27] M.A. Hopkins, D.W. Hong, and A. Leonessa. Humanoid locomotion on uneven terrain using the time-varying divergent component of motion. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 266–272, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041371. 2.1, 5.1

[28] Weiwei Huang, Junggon Kim, and C.G. Atkeson. Energy-based optimal step planning for humanoids. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3124–3129, Karlsruhe, Germany, May 2013. doi: 10.1109/ICRA.2013.6631011.

1.1, 4.1.2

[29] M. Hutter, M. A. Hoepflinger, C. Gehring, M. Bloesch, C. D. Remy, and R. Siegwart. Hybrid operational space control for compliant legged systems. In *Robotics: Science and Systems (RSS)*, Sydney, NSW, Australia, July 2012. 1.1, 3.1

[30] Marco Hutter, Hannes Sommer, Christian Gehring, Mark Hoepflinger, Michael Bloesch, and Roland Siegwart. Quadrupedal locomotion using hierarchical operational space control. *The International Journal of Robotics Research*, 33(8):1047–1062, 2014. doi: 10.1177/0278364913519834. 1.1, 3.1

[31] Y. Ito, S. Nozawa, J. Urata, T. Nakaoka, K. Kobayashi, Y. Nakanishi, K. Okada, and M. Inaba. Development and verification of life-size humanoid with high-output actuation system. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3433–3438, May 2014. doi: 10.1109/ICRA.2014.6907353. 1.1

[32] D. H. Jacobson and David Q. Mayne. *Differential Dynamic Programming*. Elsevier, 1970. 2.2, 2.3

[33] 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*, pages 1405–1411 vol.2, Apr 1991. doi: 10.1109/ROBOT.1991.131811. 2.1

[34] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 239–246 vol.1, 2001. doi: 10.1109/IROS.2001.973365. 2.1

[35] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference*

on, volume 2, pages 1620–1626 vol.2, Taipei, China, Sept 2003. doi: 10.1109/ROBOT.2003.1241826. 2.1, 5.1

[36] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi. Design of prototype humanoid robotics platform for HRP. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2431–2436 vol.3, 2002. doi: 10.1109/IRDS.2002.1041632. 1.1

[37] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot hrp-2. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1083–1090 Vol.2, Barcelona, Spain, April 2004. doi: 10.1109/ROBOT.2004.1307969. 1.1

[38] O. Kanoun, F. Lamiraux, and P. B Wieber. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality tasks. *Robotics, IEEE Transactions on*, 27(4):785–792, Aug 2011. ISSN 1552-3098. doi: 10.1109/TRO.2011.2142450. 3.1

[39] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43–53, February 1987. ISSN 0882-4967. doi: 10.1109/JRA.1987.1087068. 1.1, 3.1

[40] Joohyung Kim, Hoseong Kwak, Heekuk Lee, Keehong Seo, Bokman Lim, Minhyung Lee, Jusuk Lee, and Kyungsik Roh. Balancing control of a biped robot. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 2756–2761, Oct 2012. doi: 10.1109/ICSMC.2012.6378165. 1.1

[41] Junggon Kim, N.S. Pollard, and C.G. Atkeson. Quadratic encoding of optimized humanoid walking. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, pages 300–306, Oct 2013. doi: 10.1109/HUMANOIDS.2013.7029991. 6.1.3

[42] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal lo-

comotion. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624 Vol.3, April 2004. doi: 10.1109/ROBOT.2004.1307456. 1.1

[43] T. Komura, H. Leung, S. Kudoh, and J. Kuffner. A feedback controller for biped humanoids that can counteract large perturbations during gait. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1989–1995, April 2005. doi: 10.1109/ROBOT.2005.1570405. 2.1

[44] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. McCrory, J. van Egmond, M. Griffioen, M. Floyd, S. Kobus, N. Manor, S. Alsheikh, D. Duran, L. Bunch, E. Morphis, L. Colasanto, K.-L. H. Hoang, B. Layton, P. Neuhaus, M. Johnson, , and J. Pratt. Summary of Team IHMCs Virtual Robotics Challenge entry. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, Atlanta, GA, USA, 2013. 1.1, 3.1

[45] Twan Koolen, Tomas de Boer, John Rebula, Ambarish Goswami, and Jerry Pratt. Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models. *The International Journal of Robotics Research*, 2012. doi: 10.1177/0278364912452673. URL `http://ijr.sagepub.com/content/early/2012/06/29/0278364912452673.abstract`. 2.1, 5.1

[46] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas de Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt. Design of a momentum-based control framework and application to the humanoid robot Atlas. *in preparation for International Journal of Humanoid Robotics*, 2015. 2.1, 3.1, 3.8.3, 5.1

[47] S. Kuindersma, F. Permenter, and R. Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *Robotics and Automation, 2014. ICRA '14. IEEE International Conference on*, Hong Kong, China, 2014. 1.1, 3.1, 6.1.2

[48] J. Kulk and J.S. Welsh. Evaluation of walk optimisation techniques for the NAO robot.

In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 306–311, Oct 2011. doi: 10.1109/Humanoids.2011.6100827. 1.1

[49] Sung-Hee Lee and A Goswami. Ground reaction force control at each foot: A momentum-based humanoid balance controller for non-level and non-stationary ground. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3157–3162, Taipei, China, Oct 2010. doi: 10.1109/IROS.2010.5650416. 1.1, 3.1

[50] Chenggang Liu, Christopher G. Atkeson, and Jianbo Su. Biped walking control using a trajectory library. *Robotica*, 31:311–322, 3 2013. ISSN 1469-8668. doi: 10.1017/S0263574712000203. 1.1, 2.3, 6.1.3

[51] N. Mansard. A dedicated solver for fast operational-space inverse dynamics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4943–4949, May 2012. doi: 10.1109/ICRA.2012.6224851. 3.1

[52] Tad McGeer. Passive dynamic walking. *Int. J. Rob. Res.*, 9(2):62–82, March 1990. ISSN 0278-3649. doi: 10.1177/027836499000900206. 1.1

[53] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal. Inverse kinematics with floating base and constraints for full body humanoid robot control. In *Humanoid Robots (Humanoids), 2008 8th IEEE-RAS International Conference on*, pages 22–27, Daejung, Korea, 2008. doi: 10.1109/ICHR.2008.4755926. 3.1

[54] 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*, pages 3406–3412, Anchorage, AK, USA, 2010. doi: 10.1109/ROBOT.2010.5509646. 3.1, 6.2.4

[55] K. Mombaur, J.-P. Laumond, and E. Yoshida. An optimal control model unifying holonomic and nonholonomic walking. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 646–653, Dec 2008. doi: 10.1109/ICHR.

2008.4756020. 1.1

[56] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.*, 31(4):43:1–43:8, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185539. URL `http://doi.acm.org/10.1145/2185520.2185539`. 6.1.3

[57] Jun Morimoto and ChristopherG. Atkeson. Nonparametric representation of an approximated poincar map for learning biped locomotion. *Autonomous Robots*, 27(2):131–144, 2009. ISSN 0929-5593. doi: 10.1007/s10514-009-9133-z. 1.1

[58] Federico L. Moro, Michael Gienger, Ambarish Goswami, Nikos G. Tsagarakis, and Darwin G. Caldwell. An attractor-based whole-body motion control (WBMC) system for humanoid robots. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, Atlanta, GA, USA, October 2013. 3.1

[59] Kenneth R. Muske and James B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993. ISSN 1547-5905. doi: 10.1002/aic.690390208. 5.1

[60] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Controls*, 108:163–171, 1986. 3.1

[61] K. Nishiwaki and S. Kagami. Frequent walking pattern generation that uses estimated actual posture for robust walking control. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 535–541, Dec 2009. doi: 10.1109/ICHR.2009.5379519. 5.1

[62] K. Nishiwaki and S. Kagami. Strategies for adjusting the zmp reference trajectory for maintaining balance in humanoid walking. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4230–4236, May 2010. doi: 10.1109/ROBOT.2010.5510002. 2.1, 5.1

[63] DavidE. Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2-3):161–176, 2013. ISSN 0929-5593. doi: 10.1007/s10514-013-9341-4. 3.4

[64] D.E. Orin and A Goswami. Centroidal momentum matrix of a humanoid robot: Structure and properties. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 653–659, Nice, France, Sept 2008. doi: 10.1109/IROS.2008.4650772. 2.1, 3.4

[65] C. Ott, C. Baumgartner, J. Mayr, M. Fuchs, R. Burger, Dongheui Lee, O. Eiberger, A. Albu-Schaffer, M. Grebenstein, and G. Hirzinger. Development of a biped robot with torque controlled joints. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 167–173, Dec 2010. doi: 10.1109/ICHR.2010.5686340. 1.1

[66] C. Ott, M.A Roa, and G. Hirzinger. Posture and balance control for biped robots based on contact force optimization. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 26–33, Bled, Slovenia, Oct 2011. doi: 10.1109/Humanoids.2011.6100882. 1.1, 3.1

[67] Ill-Woo Park, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. Mechanical design of humanoid robot platform KHR-3 (KAIST Humanoid Robot 3: HUBO). In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 321–326, Dec 2005. doi: 10.1109/ICHR.2005.1573587. 1.1

[68] S. Piperakis, E. Orfanoudakis, and M.G. Lagoudakis. Predictive control for dynamic locomotion of real humanoid robots. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4036–4043, Sept 2014. doi: 10.1109/IROS.2014.6943130. 5.1

[69] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33

(1):69–81, 2014. doi: 10.1177/0278364913506757. 1.1

[70] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *Humanoid Robots (Humanoids), 2006 6th IEEE-RAS International Conference on*, pages 200–207, Genoa, Italy, Dec. 2006. doi: 10.1109/ICHR.2006.321385. 2.1, 5.1

[71] Marc H. Raibert. *Legged Robots That Balance*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1986. ISBN 0-262-18117-7. 1.1

[72] O.E. Ramos, N. Mansard, O. Stasse, and P. Soueres. Walking on non-planar surfaces using an inverse dynamic stack of tasks. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 829–834, Osaka, Japan, Nov 2012. doi: 10.1109/HUMANOIDS.2012.6651616. 1.1, 3.1

[73] 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*, pages 1085–1090, Shanghai, China, 2011. doi: 10.1109/ICRA.2011.5980156. 3.1

[74] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research*, 32(3):280–298, 2013. doi: 10.1177/0278364912469821. 1.1, 3.1

[75] L. Saab, O.E. Ramos, F. Keith, N. Mansard, P. Soueres, and J. Fourquet. Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *Robotics, IEEE Transactions on*, 29(2):346–362, April 2013. ISSN 1552-3098. doi: 10.1109/TRO.2012.2234351. 3.1

[76] J. Schulman, A. Lee, I. Awwal, H Bradlow, and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: Science and*

Systems (RSS), Berlin, Germany, June 2013. 4.3.2

[77] C. Semini. *HyQ Design and Development of a Hydraulically Actuated Quadruped Robot*. PhD thesis, Istituto Italiano di Tecnologia and University of Genoa, Italy, 2010. 1.1

[78] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2641–2648, Orlando, FL, USA, May 2006. doi: 10.1109/ROBOT.2006.1642100. 1.1, 3.1

[79] L. Sentis, Jaeheung Park, and O. Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *Robotics, IEEE Transactions on*, 26(3):483–501, June 2010. ISSN 1552-3098. doi: 10.1109/TRO.2010.2043757. 3.1

[80] A. Sherikov, D. Dimitrov, and P.-B. Wieber. Whole body motion controller with long-term balance constraints. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 444–450, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041399. 5.1

[81] Seungmoon Song and H. Geyer. Generalization of a muscle-reflex control model to 3D walking. In *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pages 7463–7466, July 2013. doi: 10.1109/EMBC.2013.6611284. 1.1

[82] Benjamin Stephens. *Push Recovery Control for Force-Controlled Humanoid Robots*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2011. 3.1, 5.1, 5.3

[83] T. Takenaka, T. Matsumoto, and T. Yoshiike. Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1084–1091, Oct 2009. doi: 10.1109/IROS.2009.5354662. 2.1, 5.1

[84] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *To appear in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012. 2.2, 2.3

[85] Russ Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. In *Robotics: Science and Systems (RSS)*, page 8, 2009. 2.2

[86] Russ Tedrake, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010. doi: 10.1177/0278364910369189. 2.2

[87] E. Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, pages 300–306 vol. 1, June 2005. doi: 10.1109/ACC.2005.1469949. 2.2

[88] E. Todorov and Y. Tassa. Iterative local dynamic programming. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL '09. IEEE Symposium on*, pages 90–95, March 2009. doi: 10.1109/ADPRL.2009.4927530. 2.2

[89] N.G. Tsagarakis, Zhibin Li, J. Saglia, and D.G. Caldwell. The design of the lower body of the compliant humanoid robot "cCub". In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2035–2040, May 2011. doi: 10.1109/ICRA.2011.5980130. 1.1

[90] N.G. Tsagarakis, S. Morfey, G.M. Cerda, Li Zhibin, and D.G. Caldwell. COMpliant huMANoid COMAN: Optimal joint stiffness tuning for modal frequency control. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 673–678, May 2013. doi: 10.1109/ICRA.2013.6630645. 1.1

[91] J. Urata, Y. Nakanishi, K. Okada, and M. Inaba. Design of high torque and high speed leg module for high power humanoid. In *Intelligent Robots and Systems (IROS), 2010*

IEEE/RSJ International Conference on, pages 4497–4502, Oct 2010. doi: 10.1109/IROS.2010.5649683. 1.1

[92] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba. Online decision of foot placement using singular lq preview regulation. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 13–18, Oct 2011. doi: 10.1109/Humanoids.2011.6100894. 1.1, 5.1

[93] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba. Online walking pattern generation for push recovery and minimum delay to commanded change of direction and speed. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3411–3416, Oct 2012. doi: 10.1109/IROS.2012.6385840. 1.1, 2.1, 5.1

[94] Miomir Vukobratović and Branislav Borovac. Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173, 2004. doi: 10.1142/S0219843604000083. 2.1

[95] C.W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *Systems, Man and Cybernetics, IEEE Transactions on*, 16 (1):93–101, Jan 1986. ISSN 0018-9472. doi: 10.1109/TSMC.1986.289285. 3.1

[96] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Optimizing walking controllers for uncertain inputs and environments. *ACM Trans. Graph.*, 29(4):73:1–73:8, July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1778810. 1.1

[97] P.M. Wensing and D.E. Orin. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3103–3109, Karlsruhe, Germany, May 2013. doi: 10.1109/ICRA.2013.6631008. 3.1

[98] E.C. Whitman and C.G. Atkeson. Control of instantaneously coupled systems ap-

plied to humanoid walking. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 210–217, Nashville, TN, USA, Dec 2010. doi: 10.1109/ICHR.2010.5686339. 3.1

[99] Eric Whitman. *Coordination of Multiple Dynamic Programming Policies for Control of Bipedal Walking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, September 2013. 1.1, 3.1, 6.1.1

[100] P.-B. Wieber. Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 137–142, Dec 2006. doi: 10.1109/ICHR.2006.321375. 2.1, 5.1

[101] Pierre-Brice Wieber and Christine Chevallereau. Online adaptation of reference trajectories for the control of walking systems. *Robotics and Autonomous Systems*, 54(7):559 – 566, 2006. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2006.04.007. 1.1

[102] A Wu and H. Geyer. The 3-D spring-mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments. *Robotics, IEEE Transactions on*, 29(5):1114–1124, Oct 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2263718. 1.1

[103] X Xinjilefu. *State Estimation for Humanoid Robots*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2015. 3.2, 3.7, 4.2, 4.2.4, 4.3.4, 6.2.3, 6.3

[104] X. Xinjilefu, S. Feng, W. Huang, and C. Atkeson. Decoupled state estimation for humanoids using full-body dynamics. In *Robotics and Automation, 2014. ICRA '14. IEEE International Conference on*, Hong Kong, China, 2014. 3.2

[105] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. *ACM Trans. Graph.*, 26(3), July 2007. ISSN 0730-0301. doi: 10.1145/1276377.1276509. 1.1

[106] Samuel Zapolsky, Evan Drumwright, Ioannis Havoutis, Jonas Buchli, and Claudio Semini. Inverse dynamics for a quadruped robot locomoting along slippery surfaces. In *International Conference on Climbing and Walking Robots (CLAWAR)*, Sydney, Australia, July 2013. 3.1

[107] M. Zucker, J.A. Bagnell, C.G. Atkeson, and J. Kuffner. An optimization approach to rough terrain locomotion. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3589–3595, May 2010. doi: 10.1109/ROBOT.2010.5509176. 1.1