Abstract

This thesis proposal presents an online approach that is based on hierarchical optimization for controlling humanoid robots. While our primary focus is on developing a fast and robust walking system that is able to follow ~~the~~ desired footsteps, full body manipulation capability is also achieved. The proposed hierarchical system consists of three levels~~.~~: ~~A~~ 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 trajectories~~.~~; ~~A~~ a middle level ~~R~~r eceding-horizon controller that tracks the nominal plan and handles large disturbances obeying center of pressure constraints~~.~~, and a ~~A~~ low level controller that computes joint level commands by solving full body inverse dynamics and kinematics using quadratic programming. The current implementation is capable of walking on rough terrain~~,~~ and achieves close to human walking speed and stride length in simulation. It has also been successfully applied to the Atlas robot, built by Boston Dynamics~~,~~ for the DARPA Robotics Challenge, in which static walking over rough terrain and full body manipulation ~~has~~ have been demonstrated. Future work focuses on implementing a fast robust walking algorithm on the real Atlas robot using the full hierarchy.

Chapter 1

Introduction

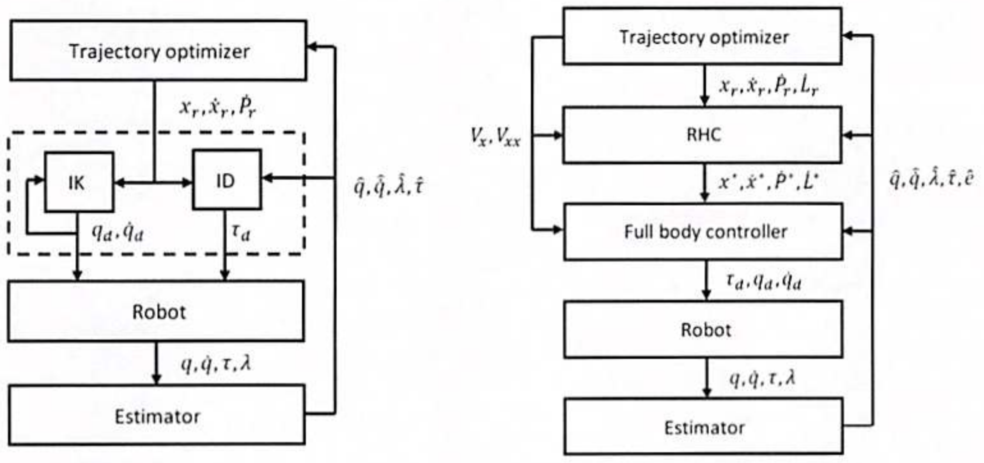
1.1 Motivation

Humanoid robots have been promoted for ~~abilities to~~ traverse ing uneven terrain, suitable lity for environment s designed for human s, and facilitate ing human robot interactions. Despite increasing ~~ly~~ ~~gained~~ interests and attention in academia and popular culture over the past decade, especially with the ongoing DARPAR Robotics Challenge (DRC) focusing on disaster response, ~~the~~ state-of-the-art humanoid robots are still not ready for deployment. ~~We think these are~~ Some of the key challenges that block humanoid robots from being a truly useful mobile platform in a human centric environment~~s~~ are :

- robustness and speed
- deliberative walking
- compliant motions
- full body manipulation

Locomotion is the foundation for any mobile system. Despite years of research, robust and fast walking is still challenging. When operating in complex and dynamic scenarios, such as any typical human occupied environment, the robot has to be capable of autonomously generating plans based on sensor feedback. In terms of walking, the robot needs to plan a reasonable path from start to goal avoiding obstacles on the way, which obviously requires the controller being able to follow the plan. With compliant behaviors, the robot can deal with unexpected external perturbations much better, and it is much safer ~~with~~ near humans ~~presence~~. Because of the large number of degrees of freedom, humanoid robots typically have bigger workspaces with smaller footprints compar~~ing~~ed to conventional platforms. On the other hand, it is much harder to maintain balance for humanoids. Motion planning is also harder for humanoids due to this complexity. On top of all these challenges, the robot needs to be robust and fast. Speed is important for usefulness as well as fast dynamic recovery motions. Th~~e~~is thesis aims at creating a control system for humanoid robots that addresses these challenges.

Originally targeted at rough terrain bipedal walking, we developed a walking controller that can achieve a sequence of footstep targets, as well as walk fast on level ground with no obstacles. Our approach is rooted in model-based optimal control, as it takes a desired sequence of foot steps, uses optimization to generate a trajectory for the center of mass (CoM), and then tracks this trajectory using inverse dynamics (ID) and inverse kinematics (IK). Figure 1.1 shows a diagram of our approach. Given a desired footstep sequence, the high level controller performs online trajectory optimization using Differential Dynamic Programming [28] with a simplified model that only reasons about the CoM of the robot. A quintic spline in Cartesian space is used to smoothly connect two consecutive footsteps for the swing foot. The low level controller was originally designed to use ID alone, and we added an IK component to cope with modeling errors when controlling a physical robot. The hierarchical optimization based architecture separates the behavior level design process and full body control problem cleanly, and offers us a versatile and powerful platform for rapidly developing multiple applications for the DRC. We propose to add a third level, which we call the middle level controller, that uses the high level controller's nominal plan as a guide to solve a short trajectory optimization problem in a receding-horizon fashion. This new layer can handle large disturbances better while maintaining critical constraints. Emergency recover~~ing~~y behaviors, such as taking an extra step to regain


Trajectory optimizer
$x_r, \dot{x}_r, \dot{P}_r$
IK
ID
$q_d, \dot{q}_d$
τ_d
$\hat{q}, \hat{\dot{q}}, \hat{\lambda}, \hat{\tau}$
Robot
$q, \dot{q}, \tau, \lambda$
Estimator
Trajectory optimizer
$x_r, \dot{x}_r, \dot{P}_r, \dot{L}_r$
V_x, V_{xx}
RHC
$x^*, \dot{x}^*, \dot{P}^*, \dot{L}^*$
$\hat{q}, \hat{\dot{q}}, \hat{\lambda}, \hat{\tau}, \hat{e}$
Full body controller
$\tau_d, q_d, \dot{q}_d$
Robot
$q, \dot{q}, \tau, \lambda$
Estimator

(a) Old system diagram (b) New system diagram

Figure 1.1: Comparison of the implemented and proposed hierarchies. Subscript $_r$ indicates the nominal values optimized by the trajectory optimizer, and $_d$ denotes values generated by the full body controller. P and L stand for linear and angular momentum, and e stands for modeling error. In the current implementation shown in Figure 1.1(a), the full body controller (enclosed with the dashed box) has separate ID and IK controllers.

balance, can also be implemented on this level.

1.2 Overview of Walking Systems

Generally speaking, the 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. Building these policies by hand typically requires a great deal of insight into the specific problem and ~~many~~ much trial~~s~~ and error~~s~~ for ~~tunning~~ tuning the policies. For simple systems, very robust and dynamic behaviors were achieved with this approach on real hardware [61]. Inspired by limit cycle walking and running research [10, 18, 45], controllers [5, 21, 90] have been developed based on those principles. Simple policy based controllers have also been applied to animated figures [84, 93] in the graphics community. Once designed, policy optimization [70, 84] or online learning [38, 42, 49] can be used for tuning. Optimization is another powerful tool for generating reference trajectories and controllers offline. Dynamic keep

programming [4] generates a globally optimal policy for a large region of the state space, but it suffers from the "curse of dimensionality". Relaxations [2, 3, 87] can be made to enable applications to larger problems. Nominal walking patterns can be found with trajectory optimization [11, 48, 59]. Feedback controllers are then used to stabilize the system around the trajectories. Trajectories can be combined into a library [44, 89] 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 fundamentally incompatible with achieving desired footsteps, which severely limits their application for more general purposes.

A typical setup for a complete online walking solution is presented in [95], where a higher level footstep planner such as [9, 24] generates a plan using perception, and the walking controller follows it to the best of its ability. For the controller, directly generating walking motions with the full model in an online setting is unrealistic due to the model's complexity. A more sensible two level approach is to abstract away the physical system with a simple model, for which it plans a trajectory. The complete motions can then be recovered through inverse kinematics. Some of the most successful humanoids such as Honda's Asimo [23], the HRP series [1, 27, 33, 34, 79], and HUBO [57] use this method for walking. This style of walking requires accurate joint level motion tracking, which is often achieved through stiff position control. Traditionally, these walking robots are vulnerable to external perturbations. On the other hand, they have precise foot placement, and combined with fast online footstep replanning [80, 81], it is possible to acquire robust dynamic walking even under strong perturbations. Rapid advancement in inverse dynamics algorithms [6, 12, 17, 22, 25, 26, 36, 40, 41, 43, 56, 62, 64, 68] and hardware development of force controlled humanoids such as PETMAN and Atlas built by Boston Dynamics and many other research platforms [8, 37, 55, 67, 77, 78] attract a large amount of attention in recent years. These robots either have built-in passive compliance or can be force controlled, which makes them better at adjusting to perturbation and more suitable for complaint behaviors. The inverse dynamics algorithms are used to partially replace the inverse kinematics pass. These algorithms are typically formulated as one step constrained convex optimization problems using the full dynamic model. They track ~~the~~ planned trajectories and enforce constraints ~~in~~ on ground reaction forces and actuation at the same time.

1.3 Thesis Contribution and Proposed Work

The core of this thesis is decoupling a complex system into different levels with progressively shorter optimization horizons but more complete dynamics and constraints so that the overall problem can be solved in an online setting to provide the most flexibility. Our controller can walk dynamically following desired footsteps on rough terrain and achieve human like walking speed and stride length in simulation. Our ~~P~~preliminary hardware implementation is capable of rough terrain traversal, full body manipulation and walking with mild disturbances.

The main contribution is to synthesize existing concepts of trajectory optimization, receding-horizon control and full body inverse kinematics and dynamics into one versatile system and its implementation on real hardware solving real life tasks. Other~~s~~ contributions include lessons learned and attempts to address modeling errors on the real robot. In addition to work presented here, we plan to include the following in the final thesis.

- Present a~~n~~ unified ~~the~~ full body controller that is primarily based on inverse dynamics.
- Compensate for estimated modeling errors, either on the center of mass level or as a generalized force.
- Produce a robust fast walking controller for rough terrain traversal using the full hierarchy.

1.4 Hardware Overview

We use the Atlas humanoid robot (Figure 1.2) built by Boston Dynamics for hardware experiments. The robot has 28 hydraulic actuated joints: six for each leg and arm, three for the spine, and one for the neck. For all the leg and back roll and pitch joints, position and torque are measured pre-transmission. Actuator length is measured with Linear Variable Differential Transformers (LVDT), and piston pressure is measured with pressure sensors. Transmission information is used to compute joint position and torque. Joint velocity is generated by low-pass filtering the finite difference of joint position. The arm joint position can be measured with encoders. Velocity and torque signals are generated ~~same as the~~ in a similar way as for the legs. For each foot, there is a three-axis force torque sensor measuring the vertical force and roll and pitch torque. Two six-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~~, which can only pitch~~.

For each joint, its joint level servo computes valve commands 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 and torque values, $q, \dot{q}, \tau$ are the measured values, and c contains the constant valve bias term plus some other auxiliary feedback and feedforward terms. This servo runs at $1kHz$, and we can update the desired values and the constant term at the same rate.

1.5 Outline

We focus on top level trajectory optimization in Chapter 2, which generates a nominal trajectory and useful information that guides the lower level controllers. The current implementation of the bottom level full body controller is presented in Chapter 3 with proposed modifications detailed in Chapter 6. This full body controller is responsible for generating joint level control signals that best track the higher level controllers' results while managing all kinematic and dynamic constraints of the physical robot. Preliminary results of walking and manipulation on the simulated and real Atlas robot are presented in Chapter 4. Chapter 5 describes ongoing work ~~of~~ on developing a flexible middle layer that improves the nominal plan in a ~~R~~ receding-horizon fashion. Finally, a

Chapter 2

Center of Mass Reference 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) [29, 31] was introduced as a simple model for this purpose. LIPM combined with Zero Moment Point (ZMP) [82] have been widely used in center of mass (CoM) motion generation. Preview Control [32] is one of the most successful applications. For dynamic behaviors, especially balancing ~~behaviors~~, angular momentum plays an important role. LIPM was then extended to include angular momentum [39, 60], 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 r~~R~~eceding-horizon controllers [13, 16, 19, 81, 88]. [54] connected both linear and angular momentum, generalized velocity of the system and net external wrench by introducing the centroidal momentum matrix, which inspired many of the later high level controllers for balancing and walking.

(typically slower than real time) ?

the

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 costs. Linear Quadratic Regulator (LQR) can be treated as a special case of this algorithm. DDP can also be extended to handle stochastic systems [75, 76]. It is also possible to use DDP in a receding-horizon fashion in sub real time [14, 72]. Combining coordinated locally optimized trajectories as an approximation of the globally optimal solution is shown in [2]. Instead of optimality, LQR-Tree [73, 74] 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 $u_{0,N-1}$ and states $X_{0,N}$.
- Refined local models of the value function and policy are created as a byproduct of our 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

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

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 to be condensed at the support stance foot, and can instantaneously switch to the next supporting 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 ~~for~~ 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 models and costs. In addition to a nominal trajectory, our approach also generates quadratic approximation of the value function and 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 policies produced by DDP are ~~not suitable~~ for handling large disturbances. When the state is far from the planned trajectory, DDP's policy can generate controls that destabilize the system since it is a local method. The other more important reason is that although it is possible for DDP to handle unilateral constraints [28], we have not implemented ground reaction force constraints such as ZMP constraints. Instead, we treat it as a high weight term in the cost function during the optimization procedure. For large disturbances when the ankle ~~strategy~~ torques alone ~~is~~ are insufficient for regaining balance, the local policy performs poorly after saturating ankle torque. ~~Our~~ The purpose of the high level controller is to generate a nominal CoM trajectory and useful information for the lower level controllers, which replan at a much higher rate, and are ~~more capable~~ better at disturbance rejection. The high level controller replans per step or ~~by~~ when requested ~~if necessary~~. With this ~~setup~~ approach, we can afford to use more complex nonlinear models that better capture~~s~~ the overall dynamics.

The computation time scales roughly cubicly with respect to model complexity for each iteration. It is also important to mention that without analytical derivatives of the model dynamics

Chapter 3

Full Body Controller

3.1 Introduction

This chapter focuses on our full body controller, which uses quadratic programming to solve inverse dynamics and inverse kinematics. Using full body inverse dynamics for force control has become a popular topic in recent humanoid research. This direction of research originates from [36]. Within this broad category, control designers can directly specify reference motions in task space, then rely on using convex optimization to handle constraints and solve for controls that best track the reference motions. Although detailed formulations differ, most active research has converged to formulating the floating base inverse dynamics as a quadratic programming (QP) problem. [12, 15, 22, 25, 26, 65, 85] 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 solution to resolv~~e~~ing hierarchical quadratic programs is presented in [15] that is more general and significantly faster than previous methods [35]. Although the method is currently applied to solve inverse kinematics, the authors claim it is also applicable to inverse dynamics. A hierarchical framework designed for humanoid robots to handle constraints and objectives is presented in [68, 69]. Contrary to these hierarchical approaches that have hard constraints, we prefer using soft constraints by adding corresponding terms in the cost function with high penalties. 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 [47, 63, 64]. [56] demonstrates a balancing controller on a torque controlled humanoid, in which simple PD servos were used to generate a desired net ground reaction wrench, which is then distributed among predefined contacts using optimization. [62] describes a recent effort using floating base inverse dynamics and ZMP-based pattern generation for dynamic walking. Their inverse dynamics formulation solves a smaller QP with decoupled dynamics. [43] 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. Then another least square problem is solved for the state acceleration. Joint torques are generated explicitly. [40] generates desired centroidal momenta change based on instantaneous capture points, and use QP to optimize for acceleration and contact forces. Joint torques are then generated with explicit inverse dynamics. [41] 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. [94] 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 [50]. We continue to use the formulation previously developed in our group [71, 86, 87] that is similar to [6, 7, 12]. We directly optimize a quadratic cost in terms of state accelerations, torques and contact forces on the full robot model. This design choice gives us the most flexibility in terms of trading off directly among physical quantities of interest. We are ~~also~~ able to directly reason about inequality constraints such as center of pressure within the support polygon, friction, and torque limits. It also allows use to easily add extra terms into the dynamics equations for compensating modeling errors. Although this formulation results in a bigger QP problem, we are still able to solve it in real time.

One traditionally popular approach to controlling humanoid robots is through inverse kine-