UNIVERSITY OF CALIFORNIA
RIVERSIDE

Momentum-Based Balance Control For Simulated Characters

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

Adriano Patrick Macchietto

December 2008

Thesis Committee:

Dr. Victor B. Zordan, Chairperson
Dr. Christian R. Shelton
Dr. Eamonn Keogh

Acknowledgments

I would like to thank Victor Zordan and the rest of the Riverside Graphics Lab for their invaluable feedback.

ABSTRACT OF THE THESIS

Momentum-Based Balance Control For Simulated Characters

by

Adriano Patrick Macchietto

Master of Science, Graduate Program in Computer Science
University of California, Riverside, December 2008
Dr. Victor B. Zordan, Chairperson

Within the past five years the control of angular momentum for bipedal robots has gained popularity within the robotics community. Research has shown that for a wide variety of motions both linear and angular momentum must be tightly regulated to preserve stability. This work expands upon these recent developments in robotics to develop a controller for a physically-simulated character applicable to the field of computer animation. Through an efficient quadratic optimization, this research develops a robust, real-time controller capable of accurately tracking a statically-stable reference motion in the presence of external perturbations. In addition, the system is shown to be generalizable to characters with non-humanoid morphologies. The output of the system produces characters which adapt to external disturbances in a natural, life-like manner while simultaneously retaining the overall style of the tracked motion.

Contents

List of Figures

Chapter 1

Introduction

From the beginnings, character animation researchers have been interested in understanding and virtually recreating the rich variety of motions that humans and other animals exhibit. In the past 30 years a great amount of progress has been made in synthesizing human motion. Unfortunately, there still does not exist a system which can synthesize a rich variety of convincing, physically-interactive motions automatically. Our current lack of understanding of human biomechanics and neuroscience has forced character animation researchers to avoid these limitations by modelling what is understood, and compensating for missing knowledge with assumptions or data. Despite the limitations, much progress has been made in the past 10 years towards developing robust, physically-interactive character animation systems.

This thesis loosely categorizes all 3D character animation systems into one of three broad categories: physically-simulated, data-driven, or procedurally generated. Physically-simulated approaches generate motion by controlling the dynamics of the character through control of the actuator forces. Data-driven approaches achieve interactivity by splicing and editing motions together to generate new motions based upon the requirements. Lastly, procedural approaches generate motion through a set of user-specified rules. Each approach has proved useful at generating motion effectively under different requirements. Although the categories are fuzzy, as many systems can be a hybrid of all three, this distinction is important as it provides a means of assessing the potential strengths and weaknesses of a given system.

Up until recently, data-driven approaches have commanded the greatest commercial appeal due to their strength of preserving physical-realism and motion style. Any motion which is needed can be captured or keyframed offline or extrapolated from an existing motion database[20, 3, 4, 13]. Limitations of data-driven approaches lie in their inability to generalize well to a variety of interesting simulation conditions: conditions which are too dissimilar to motion database examples or too hazardous, complex or laborious for the capture crew to capture or keyframe. Realtime applications with unpredictable environments and user input suffer most dramatically from this limitation, often resorting to playback of canned responses when interaction is required. Despite the limitations of data-driven approaches, it would be difficult to find a robust full-body character animation system which utilizes no data at all.

Procedural character animation avoids some of the limitations of data-driven approaches by defining a set of rules which are used to synthesize motion automatically. As one would imagine, the rules may be potentially difficult for the animator or programmer to specify. Procedural character animation has found application for animating simple character animation tasks, or for producing animations for characters with fictitious morphologies[16, 14]. For example, in the PC videogame Spore users are allowed to create creatures of varying morphologies which are then animated automatically according to a set of developed-specified relationships between the body parameters. Procedural systems have great flexibility, but unfortunately, due to the complexity of specifying the rules, are time consuming and difficult to implement robustly.

Lastly, we have physical simulation, the character animation category of this thesis. In contrast to procedural or data-driven approaches, physical simulation systems generate motion by controlling the dynamics of a physical representation of the character[10, 27, 26]. Physical simulation guarantees physically-realistic motion, which is also the reason why physical-simulation is difficult. Character simulation requires the control of a highly non-linear dynamic system of ODEs. For a long time simulation approaches lacked commercial viability due to the limited amount of computational resources available for solving the dynamics and the difficulty of producing efficient, robust, convincing control algorithms. Up until very recently, physical simulation had only been used sparingly, mostly for passive ragdoll motions which do not require complex control. Despite the control difficulties, much progress has been made and the state of physical simulation has finally reached the point of commercial appeal, with NaturalMotion's Euphoria animation system serving as a notable example. Physical simulation offers the greatest possibility for synthesizing robust, physically-interactive motion since it attempts to simulate the physical processes which are responsible for producing the motion in the real world.

It is helpful to further subdivide the category of physical simulation into more concrete subfields by introducing the notion of bottom-up and top-down physical simulation, each of which tackle the problem of physical simulation in a much different manner.

Bottom-up approaches model the physical processes of the character anatomy accurately which give rise to emergent, desired characteristics within the motion. For instance, a bottom-up modeller would approach the problem of creating a full-body animation system by carefully placing muscles, bones, and then create a neuromuscular controller which simulates the human brain at the neural level. Of course, without the computational resources or the knowledge needed to simulate and control the complexity of the human anatomy, bottom-up approaches are typically much more limited in scope, tending to focus on simpler isolated systems that are more understood. Bottom-up approaches offer the greatest potential for generating motion since all of the processes responsible for human motion are accurately modelled. In addition, due to the extra emphasis on modeling the complexity of the human anatomy, they also have a greater degree of generalizability to unforeseen simulation conditions. In contrast, top-down approaches work in the opposite direction, by achieving specific characteristics by physically modeling only what is needed to create the behaviors of interest. In the previously presented example, a top-down modeller might instead simplify the human body with a set of 10-20 bones, approximate the entire muscle system with simple 1-DOF or 3-DOF motors, and then command the entire system using a set of simple control laws which most likely use motion data to some extent based upon the required needs of the system. Top-down approaches are typically concerned with performance and controllability, and thus sacrifice modelling accuracy when it isn't important to generate the characteristic of interest.

Top-down simulations have typically dominated the field of Computer Animation in the simulation category for full-body character animation since it is simply not computationally feasible to animate the entire character without making many modeling simplifications. Instead, bottom-up approaches, have been used for simpler processes where high-level decision-making is less of an important issue, such as swallowing or breathing; and to field of Biomechanics where the principle goal is to understand certain aspects of human motion and accurate modelling is essential.

While both physical-simulation approaches can be used to synthesize realistic, interactable motion, the problem of control still remains a very open issue for both. Control can be stated as the requirement that a character follows a set of objectives as closely as possible and as convincingly as possible (i.e. the character does not fall over when a suitable recovery strategy exists that a human would employ). The question of how it is possible to simultaneously control a character subject to all of the natural laws while preserving realism is not fully understood. Control algorithms have been devised to tackle specific categories of motion, such as locomotion, but no robust high-quality, general-purpose controllers currently exist.

One of the most fundamental problems with control is balance: how to ensure the character remains standing when its real-world equivalent should. A good controller should be able to pick strategies which are similar to those that a human would employ. In addition, it should be able to handle unexpected disturbances robustly. A great deal of work has been devoted to developing balanced motion within various contexts [7, 24, 21, 25]. The context this thesis is concerned with is with real-time, systems subject to all of the physical conditions present in the real world: friction, unilateral contact, external forces.

This thesis develops a realtime, top-down, physically-simulated system capable of balancing during a wide variety of statically- stable single-support and double-support motions. While the focus of this work is on statically-stable motions, there is reason to believe that it could be extended to statically-unstable motions such as locomotion. This system employs a controller capable of following motion capture data accurately while allowing for external disturbances on a wide variety of character morphologies. In addition, we present a robust system architecture which can be extended to accomplish secondary goals in addition to motion tracking.

The developed controller achieves its robust control by regulating two important quantities discovered to be of critical importance to balancing behaviors: linear and angular momentum. While much research has already been devoted to controlling the linear momentum of the simulated character, only until recently have researchers begun to pay attention to the importance of angular momentum regulation [18, 15, 23, 22]. As this work will show, the regulation of angular momentum is essential to statically-stable single-support motions, and, in the presence of large perturbations, double-support motions as well.

Chapter 2

Background

This section will introduces concepts important to understanding material presented in later chapters. A basic understanding of rigid body mechanics is assumed. It will begin by introducing rigid body linkages, otherwise known as articulated bodies. From there, the two different parameterizations for articulated rigid bodies will be introduced. Finally, this chapter will conclude with a presentation of the equations of motion.

2.1 Articulated Rigid Bodies

An articulated rigid body (ARB) may be defined as a system of n rigid bodies connected together by m joints[1]. A free-floating rigid body has 6-DOFs: 3 degrees of freedom for position, and another 3 degrees for rotation. Joints may be viewed as constraints which reduce the degrees of freedom of the rigid body system by $6n-j$ where j is the degrees of freedom of the joint. A 1-DOF hinge joint[2], a joint which only allows both bodies to rotate about a common axis, reduces the DOF of the system by 5. For a two body ARB connected by a revolute joint the DOFs of the system becomes $2*6-5=7$. Articulated rigid bodies are critical to virtual character simulations since humans may be modelled efficiently as a simplified system of rigid

[1]For ARBs without kinematic loops, $n = m$, as the root is connected by an invisible joint to the inertial reference frame

[2]Hinge joints are also commonly referred to as "revolute joints"

bodies connected together by joints.

Depending on the type of system being modeling certain joint may be powered or unpowered. A powered joint has the ability to internally generate forces to push the connected bodies. It may be the case that a powered joint is not able to actuate in the same space of its motion, in which case, the motion space is not equal to the actuation space. A joint which exhibits a non-empty actuation space which is a strict subspace of the motion space is referred to as an under-powered joint. For humanoid character simulation models, typically all joints which are actuable are also fully-powered.

For an ARB simulating a humanoid character, one link is connected to the inertial reference frame via a special 6-DOF, unpowered, floating joint. This link is known as the *root*. Unlike other joints, a floating joint removes no freedom from the ARB. As we will see in subsequent sections, controlling a character with 6 unpowered DOF in the presence of unilateral ground contact forces is the fundamental control problem.

A character with n bodies may be modelled with 1 floating joint for the root, and another $n - 1$ 3-DOF ball joints or 1-DOF hinge joints, depending on whether the character is 2D or 3D and whether we wish to model the knees and elbows as hinges. This work is concerned with the 3D balance problem so all non-root joints are modelled as 3-DOF ball joints. For joints which don't exhibit ball joint behavior, such as the knees and elbows, we utilize soft joint limits discussed in Chapter 6. Soft limits enable compliance that the knees and elbows exhibit which hard constraints do not model.

2.2 Maximal and Reduced Coordinate Parameterizations

The previous section discussed ARBs, but did not describe how the ARB system is parameterized. The type of parameterization is of great importance, since issues of stability, performance, and simplicity of control all depend on how the system is represented. Two basic approaches exist for parameterizing ARBs: maximal and reduced coordinate.

Maximal coordinate parameterizations treat each body in isolation, requiring 6 scalars for each body. Joints are represented using a set of dynamic constraint equations enforced during the forward dynamics solve. Maximal coordinate approaches are popular since they do not require much extra effort to integrate in pre-existing rigid-body physics systems. However, they suffer from problems of numerical drift where bodies can become detached at the joint and need to be corrected[3]. If the character is to be actively controlled (i.e. beyond a simple ragdoll), it will often be necessary to convert between Cartesian and joint space. [6] contains a thorough discussion on the advantages and disadvantages of maximal-coordinate approaches.

In contrast to maximal coordinate methods, reduced coordinate methods parameterize the system such that joint constraints are implicitly enforced in the parameterization. Instead of the parameters describing the position and orientation of the body, reduced coordinate systems describe the relative transformation between bodies. For example, the static state of a system comprised of two floating bodies connected by a revolute joint can be described using 7 scalars, 6 for the translation and orientation of the system, and 1 to describe the rotation about the axis of the revolute joint. Since reduced coordinate approaches use the joint parameters as state variables, controlling these systems is more intuitive than maximal coordinate approaches. A reduced coordinate parameterization is used in this work.

2.3 Equations of Motion

The equations of motion for an ARB can be written in matrix form as:

$$Q = H(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) \tag{2.1}$$

where θ, $\dot{\theta}$, $\ddot{\theta}$ are the generalized coordinated, accelerations and velocities; Q are the generalized forces; H is composed of the inertial coefficients; C represents centripetal

[3]This limitation may be also beneficial as it allows for the modeling of soft joints which are not easy to model with reduced-coordinate parameterizations

and Coriolis forces; and G represents the gravitational force. Eqn. 2.3 can be derived from a Lagrangian formulation. For more detail see [11].

Forward dynamics algorithms solve for $\ddot{\theta}$ in Eqn. 2.3, while inverse dynamics algorithms solve for Q. Typical approaches linearize Eqn. 2.3, and solve numerically. Non-recursive numerical approaches can solve Eqn. 2.3 in $O(n^3)$ time. In contrast, recursive forward dynamics algorithms can solve Eqn. in $O(n)$. For a more detailed overview of numeric reduced-coordinate forward dynamic algorithms see [12, 11].

Chapter 3

Balance Control Strategies

This section will provide an overview of useless balance control metrics for statically-stable controllers. From these metrics, control laws for the regulation of linear and angular momentum will be developed.

3.1 Static and Dynamic Stability

A character may be considered statically-stable if it's center of mass (CM) projection onto the support polygon is within the support polygon boundaries. Fig. 3.1a provides an example of a statically-stable character. As long as the character is able to control itself such that the center of mass projection remains within the support polygon, the character will not tip over.

Unfortunately, the static-stability requirements are too restrictive for generating a wide-variety of natural biped motions. For example, during biped locomotion the character will lose static-stability as it switches between support legs. In effect, bipeds throw themselves into a controlled fall which is then recovered by the swing leg as it transitions to the support leg as in Fig. 3.1b. This is known as dynamic stability: the ability to recover when the CM exits the support polygon.

Unlike static stability, dynamic stability has no widely agreed upon rigorous definition[1]. The lack of a widely disagreed upon definition makes producing controllers for statically-unstable motions particularly challenging. Despite this,

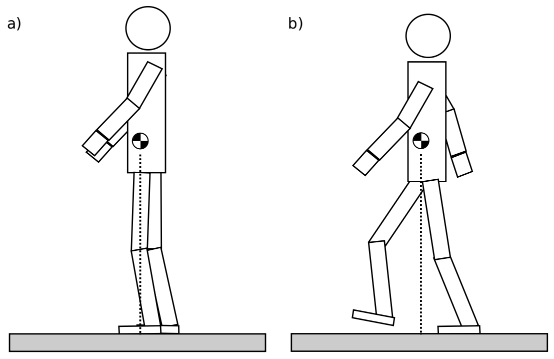

Figure 3.1: Statically-stable(a) and dynamically-stable(b) motion.

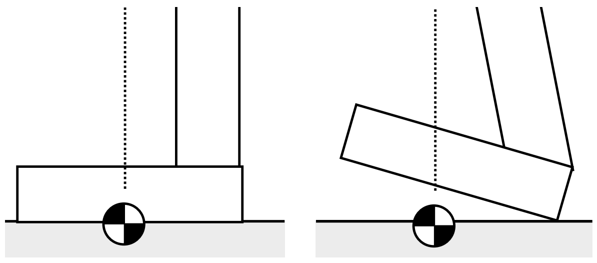

Figure 3.2: Static-instability resulting from a discontinuous change in the support polygon.

robust controllers have been developed for certain classes of dynamically-stable motion, particularly locomotion.

This work is primarily concerned with the problem of static-stability for biped motions which have no support transition phases. While it is conceivable that these strategies may be employed to build a controller which generates dynamically-stable motions, it is outside the scope of this research.

3.2 The Center of Pressure (COP)

The location of the CM within the support polygon provides some indication of the stability of the character. One control strategy might be to direct the CM away from the boundaries of the support polygon. This strategy tends to be

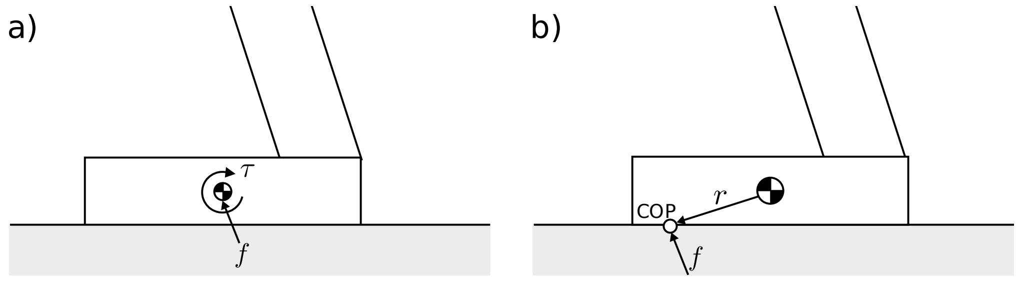

Figure 3.3: (a) The applied ground force and torque at the CM of the support. (b) The resulting location of the COP.

problematic since the location of the CM within the support polygon provides no indication of support rotation; In fact, it is possible for a CM well within the support polygon to move outside the support polygon instantaneously as in Fig. 3.2. This rotation leads to a dynamic-balance problem which is more difficult to recover from.

The center of pressure (COP) is a metric which may be used to help assess the rotational characteristics of the character. The COP can be viewed as a point between the contact surface and support where the resultant ground reaction force acts. More precisely, the COP is the location where the aggregate ground force must be applied to a net zero moment. If the COP is within the support polygon, where "in" excludes the support polygon boundaries, then the foot is not rotating.

If we assume that gravity acts along the -Z axis we can compute the COP very easily given the aggregate force and torque applied at the CM of the support by the ground: f and τ. τ and f are related by

$$\tau_x = \|r_{yz}\|\|f_{yz}\|\sin(\phi_1) = r_y f_z - r_z f_y \tag{3.1}$$

$$\tau_y = \|r_{xz}\|\|f_{xz}\|\sin(\phi_2) = -r_x f_z + r_z f_x, \tag{3.2}$$

where r is the vector from the CM to the COP, and r_{xz} and r_{yz} are the orthogonal projections of r onto the XZ and YZ planes. The location of the COP may be determined by first calculating r.

Eqn. 3.1 is a pair of two equations with three unknowns, so an additional constraint on the location of the COP is needed. For these calculations it is assumed that the ground and gravity are perpendicular, although it is possible to generalize them to situations where this is not the case. By observing that the COP must lie along the contact surface formed between the support, an additional constraint may be introduced: $r_z = -h$, where h is the height of the support CM from the ground contact surface. Plugging $r_z = -h$ into Eqn. 3.1 and rearranging:

$$\begin{aligned} r_x &= -\frac{\tau_y + hf_x}{f_z} && (3.3) \\ r_y &= \frac{\tau_x - hf_y}{f_z} && (3.4) \\ r_z &= -h. && (3.5) \end{aligned}$$

Let c_{sp} be the location of the CM of the support and p be the location of the COP. The COP may be computed from r and c_{sp}:

$$p = c_{sp} + r. \tag{3.6}$$

The COP provides a summary of the rotational characteristics of the character. Only when the COP lies on the edge of the support polygon is the foot rotating. In the absence of external forces, the total linear and angular momentum of the character about the CM, denoted L and H, is conserved. Furthermore, if no external perturbations are applied, any change in system momentum must come from the GRF and gravity.

Assuming a GRF force f is applied at the COP, the linear and angular momentum derivatives are:

$$\begin{aligned} \dot{L} &= -Mg + f && (3.7) \\ \dot{H} &= s \times f && (3.8) \end{aligned}$$

where g is the gravitational constant, M is the total mass, and c is the CM of the character, and $s = p - c$. Fig. 3.4 summarizes the external forces on the character. As Eqn. 3.8 has shown, in the absence of external perturbations the

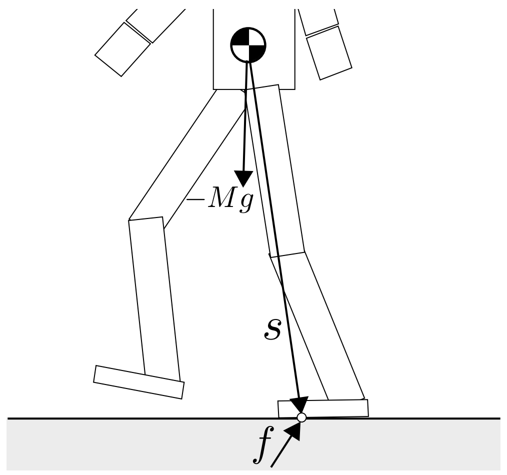

Figure 3.4: Typical external forces applied to the character.

coupling between the COP and the GRF direction determine the change in angular momentum.

In the absence of angular momentum change, the line of action of f must pass through the CM and COP of the character. It is in the preservation of angular momentum that the character can be considered to be rotationally stable. Furthermore, if the linear momentum is conserved and the CM is within the support polygon, then by Equation 3.7, the COP is directly equal to the projection of the CM onto the ground along the direction of gravity.

For the constraint-based ground contact model (zero-interpenetration), the requirement that the COP is not at the edge of the support polygon necessitates that the foot may not rotate. This is due to the zero-work requirement of the ground: the ground can resist interpenetration but may not push the support from the contact surface. Stated more concretely, the contact points of the support can either have a non-zero force, or a non-zero relative acceleration with the ground, but not both[5]. This implies that for a rigid-support with a COP not at at the edge of the convex support polygon that there are at least three contact points applying a force across a contact manifold with a non-zero area, which by the previously states zero-work requirement, requires these contact points to have a zero relative acceleration with

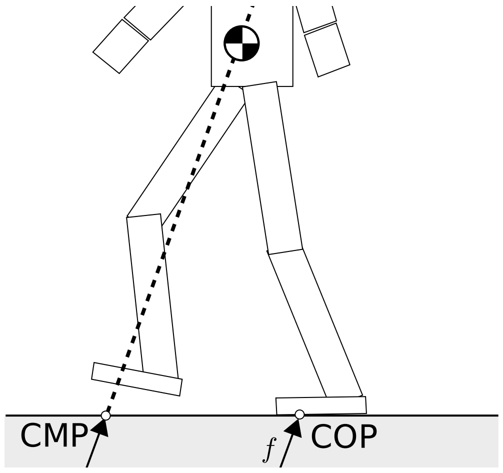

Figure 3.5: CMP relation to the CMP and the GRF

the ground. Having at least three non-collinear contact points with zero-relative acceleration sufficiently constrains the rigid body of the support to maintain a zero-relative linear and angular acceleration with respect to the ground. Therefore, it is not possible to have foot rotation unless the COP is on the boundary of the support polygon.

The COP and GRF provide a good indicator of stability for statically-balanced motion types. By attempting to control momentum change which contribute to the COP and GRF values, one could devise a control scheme which maintains stability in the presence of disturbance.

3.3 Centroidal Moment Pivot (CMP)

In the previous section the COP was introduced as the point where the resultant ground force must act to produce a zero net moment. It was shown that the location of the COP and the direction and magnitude of the ground force at this point can be used to calculate the total change of angular momentum of the character. The centroidal moment pivot point is the point which summarizes the coupling between the COP and the GRF.

The CMP is the point where the COP would need to be to for there to be no momentum change due to the ground forces. In other words, $(p - c) \times f = 0$ and $p_z = h$, where h is the height of the ground. Figure 3.5 provides an illustration of the CMP.

The CMP may be computed easily. Let g denote the location of the CMP. Given the location of the CM, and the GRF, the CMP location is

$$g_x = c_x - \frac{f_x}{f_z}(c_z - h) \tag{3.9}$$

$$g_y = c_y - \frac{f_y}{f_z}(c_z - h) \tag{3.10}$$

The distance between the CMP and COP provide a useful metric for assessing the rotational stability of the character. In the next section the CMP will be used to devise a momentum control scheme.

3.4 Angular Momentum Control

In the previous section it was shown that momentum conservation is a good approach to preserve stability as it guarantees that the support will not rotate as long as the CM projection remains within the support polygon. It turns out that for a wide variety of human motions, such as standing, walking and running, angular momentum about the CM is mostly conserved[22]. Robotics researchers have used this idea to create a walking gait by preserving angular momentum along specified axes[18]. However, it turns out that for a wide variety of other motions, such as a standing toe-touch or leg-swing motion, angular momentum is not conserved. Figure 3.6 presents the angular momentum magnitude of a leg-swing motion in the sagittal plane that is clearly non-zero. This suggests that humans may be tightly regulating angular momentum change to carry out rotationally unstable tasks.

This thesis presents a control law that allows the character to have a non-zero angular momentum. Instead of restricting the controller to always attempt to conserve angular momentum, the controller only attempts to preserve angular momentum when instability indicators are present. The instability indicator used

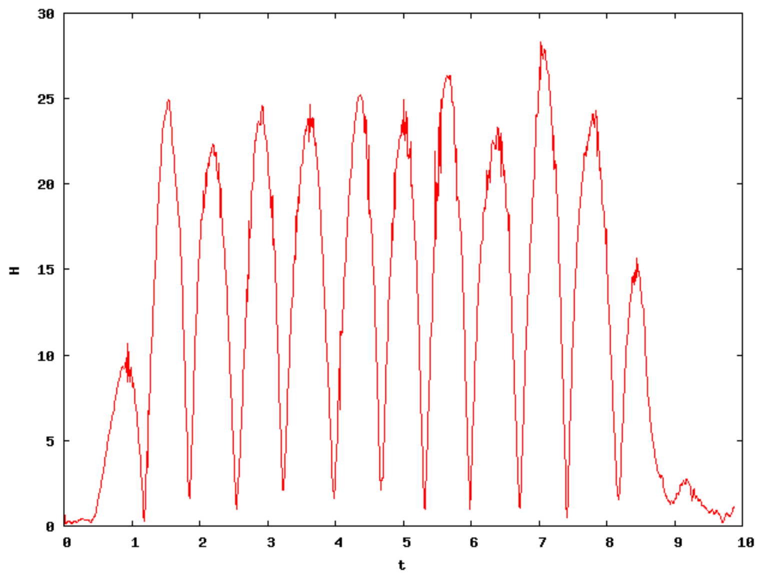

Figure 3.6: The angular momentum computed about the CM of a motion-captured leg-swing motion. Despite having a widly-varying momentum value, the actor is able to remain balanced.

in this work is the distance between the COP and the CMP. As mentioned in the previous section, angular momentum change can only come from the GRF and external perturbations. The distance between the CMP and the COP directly relates to the angular momentum change produced by the ground and it used to determine when momentum conservation is appropriate.

In Section 3.2 it was discussed that linear and angular momentum preservation ensures that the COP remains within the support polygon. Of course, it is not possible to preserve momentum indefinitely: preserving linear momentum will cause the character CM to leave the support polygon, and preserving angular momentum will eventually result in the character reaching a joint limit. Despite these limitations, it is possible for the character to control its angular momentum to aid in damping out linear momentum. As the character is forced to remove linear momentum from the system, either due to external perturbations or tracking instability, it produces frictional torque with the ground (a character can only remove momentum via friction). By preserving this angular momentum with the upper body the character can counter-act the effects of rotation on the support. After the CM stabilizes, the character can then proceed to recover its desired posture.

The control rule can now be stated. The character should attempt to dampen out linear momentum based upon the position and velocity of the CM relative to some desired CM. Damping the linear momentum will lead to angular momentum change. Based upon the distance between the CMP and the COP the character should attempt to remove any angular momentum change induced by the linear momentum damping.

Other research work has suggested that during large external perturbations humans absorb the impact by preserving the momentum for a specified period of time [1]. After the impact has been absorbed the character recovers its posture. This idea is supported by the proposed control law as any impact would result in a large instantaneous velocity change of the CM according to Eqn. 3.7. Were the character to attempt to resist the impact, the COP could possibly be pushed forward to the edge of the support and rotation would occur.

The optimization and the objectives are presented in Chapter 5.

Chapter 4

System Overview

This chapter introduces the system architecture used in this work. The goal is to provide the reader with a high-level understanding of how the various components of the system interact. More low-level implementation discussion will be deferred to later chapters.

Fig. 4.1 presents an diagram of the system components. The core of system is a least-squares optimization. This optimization is responsible for choosing $\ddot{\theta}$ which meets user specified goals. These objectives may compete, therefore the optimizer is responsible for choosing the optimal set of accelerations which mutually satisfy each objective.

$\ddot{\theta}$ is passed as input along with the penalty-based ground reaction forces, f_g, to produce the actuator torques which achieve the generalized accelerations in the presence of ground reaction forces. Lastly, the torques output from the inverse dynamics are fed into a forward dynamics algorithm along with all other external forces (not including gravitational or ground forces), f_e, to produce the final accelerations to be integrated. Errors are corrected in subsequent optimization runs by the feedback components of the tracking and balance objectives.

The optimizer runs at a frequency of 30-200 Hz to achieve greater computational efficiency and run in real-time. To maintain simulation stability the inverse-dynamics/forward-dynamics loop is run at much higher rate of 1-1 kHz. This decoupling of the desired acceleration signal from the actuation force signal is

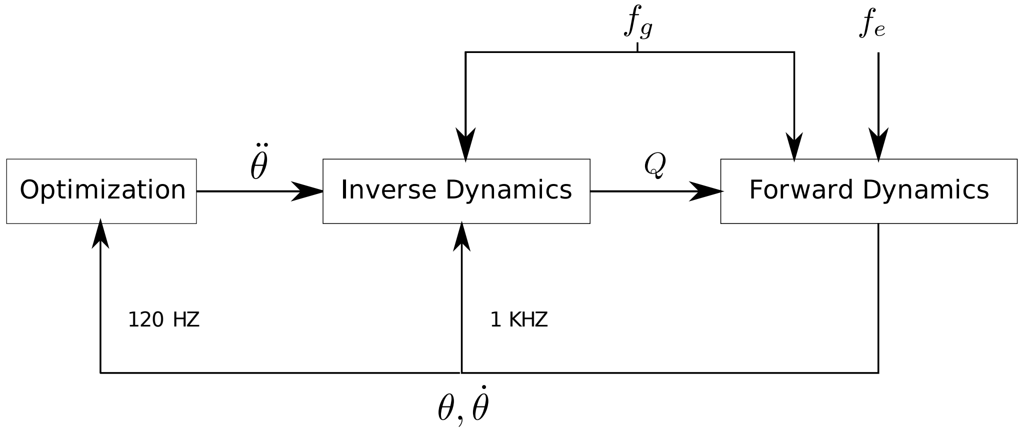

Figure 4.1: Diagram of the system architecture

analogous to separating the high-level planning frequency from the low-level control frequency. This is in contrast to other systems which compute new torques at the frequency of optimization that are held constant until the next optimization run[2, 9].

4.1 Optimizer

As stated in the previous section, the optimizer is responsible for choosing an appropriate set of accelerations necessary to carry out multiple, potentially conflicting, objectives. The optimizer solves a quadratic objective function subject to linear constraints:

$$\begin{aligned} \min_{\ddot{\theta}} \quad & \|W(b - A\ddot{\theta})\|^2 \\ \text{subject to: } & C\ddot{\theta} = d \end{aligned} \tag{4.1}$$

where W represents a diagonal weighting matrix, b is the desired value of the objectives, and A maps $\ddot{\theta}$ to the objective space. In this system, the number of constraints is low so this optimization can be solved efficiently.

The following subsections will provide a brief overview of the core objectives for simplicity. In later chapters, we will present other additional objectives such as point objectives.

4.1.1 Tracking Objective

The tracking objective attempts to follow a proscribed motion trajectory as closely as possible (In this work, motion data is generated through motion capture and keyframing). Due to the discrepancies between the physical and simulated actor and environment, postural stability is not automatically guaranteed in the simulation environment. In dynamic environments characterized by unpredictable disturbances tracking is insufficient for maintaining balance. The tracking objective is primarily used to help maintain the stylistic aspects of the desired motion.

4.1.2 Balance Objective

To alleviate the limitations of tracking, additional balance objectives are introduced. As discussed in Chapter 3, controlling the linear and angular momentum is vital to ensuring stability. Both the linear momentum and angular momentum goals are separated into two objectives. The linear momentum objective chooses momentum values which slowly guide the current CM back to the desired CM, while the angular momentum objective preserves momentum when the CMP is distant from the CM.

One might wonder how the balance objectives cooperate with the tracking objective. Since the linear and angular momentum require only 6 scalars to describe, whereas a typical simulated humanoid character has around 45 degrees of freedom, both objectives are able to work cooperatively with the tracker without disturbing the overall posture of the motion. As one would expect, motion which is stable is less effected by both objectives.

4.2 Inverse Dynamics

Once the optimization solves for the accelerations, a floating-base inverse dynamics algorithm described in [11] is used to convert the accelerations into actuator torques. Unlike Recursive Newton-Euler inverse dynamics algorithm, this algorithm assumes root is unactuated and generates consistent torques. The inverse dynamics algorithm solves the following equation:

$$Q = H(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) + J^T f \tag{4.2}$$

for Q, which is the same as Eqn. 2.3 with the addition of an extra term representing the generalized forces due to the ground, $J^T f$. Here f represents a vector of Cartesian ground forces and J^T is the transpose of the Jacobian which maps the Cartesian ground force to a set of generalized forces. For more detail on the computation of the Jacobian see Chapter A.

Inverse-dynamics is technically not needed to create the optimized motion, since the accelerations can be integrated directly. This step is performed so that effects of external impulses may also be added with the torques needed to fully compensate for the ground.

In this work a penalty-based ground contact model is used. Since the forces of a penalty method are based upon the current state of the character and not coupled with the generalized accelerations, these penalty forces can be computed and passed into the inverse dynamics algorithm and solved. This is in contrast to constraint-based contact models which enforce strict non-penetration constraints and solve for the ground forces and joint accelerations simultaneously as a linear complementary optimization[5]. The benefits of such LCP approaches is that they are more numerically stable, which allows for a larger integration step size, and also prevents interpenetration (with some numerical error); however, they are also more difficult to implement.

By performing inverse dynamics it is possible to determine the generalized forces required to achieve the optimization specified accelerations in the presence of ground contact forces. Under normal support circumstances (i.e. not landing from a jump or standing on a moving platform) humans and other animals display a remarkable ability to track well[19]. Therefore, it seems reasonable that the ground may be fully-compensated for under these circumstances. Some compliance is lost in fully-compensating for the ground and is not addressed in this work.

4.3 Forward Dynamics

As mentioned in the previous section, the inverse-dynamics/forward-dynamics loop is performed primarily to allow for the incorporation of external perturbations into the system.

The optimization algorithm has no knowledge of the external forces. Instead it relies upon tracking error feedback to correct for the disturbance using a PD-controller. By incorporating the forward-dynamics loop it is possible to accurately model the impact dynamics.

The forward dynamics algorithm used is Featherstone. Featherstone is an efficient $O(n)$, reduced-coordinate algorithm which solves the equations of motion through recursion. In contrast to the competing $O(n^3)$ Composite Rigid Body Method, Featherstone can be shown to be faster when $n > 9$ [11].

Chapter 5

Optimization

Each simulated actor is composed of n links connected together by $n-1$ 3-DOF actuated ball joints, and a 6-DOF unactuated floating joint connecting the root to the inertial reference frame. The total degrees of freedom for this system is $m = 3n + 3$. The optimization attempts to find the generalized accelerations across all degrees of freedom, $\theta \in R^m$, necessary to achieve both the tracking and balance objective.

Solving across all DOFs is not a trivial task in the presence of under-actuated joints. In a fully-powered system, where the motion space of the ARB is identical to the actuation space, every generalized acceleration vector has a corresponding set of realizable actuation forces. The term *realizable* is used to indicate that the forces required to generate the motion can be provided by the actuators; In other words, no magic external forces are required to produce the desired accelerations.

Since the humanoid character is connected to the world via a 6-DOF unactuated floating joint, the problem of control is more difficult. Were that character floating in space, the conservation of linear and angular momentum would produce a character without any control over the acceleration of its center of mass, and limited control over the angular velocity of its system (the character can speed up and slow down by either contracting or expanding its extremities, but the momentum is conserved). Any additional control over the character requires physical contact with the environment. The amount of control regained depends both on the type of character and the type of contact. Since a character may push but not pull on the ground, the character does not regain full control over all degrees of freedom; Were the character able to pull on the ground the it would be possibly to model the characters using all sphere joints, replacing the unactuated floating joint with an actuated ball joint at the foot in contact, regaining full control over all degrees of freedom. The problem is further exacerbated by the presence of static and dynamic friction. To ensure that the accelerations chosen by the optimization are realizable, the optimization must be properly constrained.

Certain approaches have tackled this problem by incorporating the dynamics and the contact friction cone of the character as an optimization constraint within a quadratic programming (QP) problem formulation[2, 8]. This provides a guarantee that the optimization will find realizable accelerations, if they exist, which achieve the objective requirements and do not result in slip. However, due to the large number of inequality and equality constraints the cost of solving the QP is high. To run in realtime the optimization is executed at around 30 Hz and requires the inclusion of an additional low-gain PD-controller to correct for latency errors between optimization runs. The effect of this additional latency on the stability of the character is not well documented.

This work attempts to devise a similar solution that is less computationally costly which does not guarantee slip-free accelerations. Instead of attempting to optimize over the accelerations, torques, and ground forces simultaneously, we perform an optimization over the accelerations only and rely upon the robustness of our balance objective to avoid slip conditions. The extra unactuated degrees of freedom are resolved by assuming that when the character is statically stable (the feet are on the ground and the center of mass is within the support polygon) it has full control over the acceleration of the root through its contact with the ground. The optimization imposes an acceleration constraint on the supports to ensure that they maintain the acceleration of the ground.

The fixed support constraint does not guarantee that the support will not slip or detach when the inverse/forward dynamics phase occurs. The responsibility of ensuring that the supports remain level with the ground is that of the balance objectives.

Let x and q denote the position and orientation of the body (q is a quaternion), v and ω represent the linear and angular velocity of a rigid body, and let a and α denote the linear and angular acceleration. For notational convenience we will combine the linear and angular values into a single vector. Let $\hat{x} = \begin{bmatrix} x \\ q \end{bmatrix}$, $\hat{v} = \begin{bmatrix} v \\ \omega \end{bmatrix}$ and $\hat{a} = \begin{bmatrix} a \\ \alpha \end{bmatrix}$.

The optimization is

$$\begin{aligned} \min_{\ddot{\theta}} \quad & \beta_t C_t(\theta, \dot{\theta}, \ddot{\theta}) + \beta_{bl} C_{bl}(\theta, \dot{\theta}, \ddot{\theta}) + \beta_{ba} C_{ba}(\theta, \dot{\theta}, \ddot{\theta}) \\ \text{subject to:} \quad & J_{sup}\ddot{\theta} + \dot{J}_{sup}\dot{\theta} = \hat{\mathbf{a}}_{sup} \end{aligned} \tag{5.1}$$

where C_t, C_{bl}, C_{ba} represent the tracking, linear and angular balance objective functions of the form $\|W(b - A\ddot{\theta})\|^2$; β represents the objective weights; and $\hat{\mathbf{a}}_{sup}$ is the linear and angular accelerations of the supports. Through the objective weights the animator may trade-off between style preservation and balance robustness depending on requirements.

The constraint expression ensures that the supports of the character maintain the linear and angular acceleration of the ground at the point of contact. Let $J(\theta) = \left[\frac{\partial \theta}{\partial \mathbf{x}}\right]$ be the $6n \times m$ Jacobian which we use to map generalized velocities to Cartesian body velocities:

$$\hat{\mathbf{v}} = J(\theta)\dot{\theta} \; . \tag{5.2}$$

Computing the derivative of Eqn. 5.2 over the rows corresponding to our support bodies we obtain the constraint expression in Eqn. 5.1. If p is the number of supports then $J_{sup}, \dot{J}_{sup}$ are $6p \times m$ matrices, and $\hat{\mathbf{a}}_{sup}$ is a size $6p$ vector. For details on the computation of the Jacobian and its derivative see Appendix A. The following sections will present the computation of the objective terms, C_t, C_{bl}, and C_{ba}.

5.1 Tracking

The tracking objective attempts to follow a prescribed motion trajectory as closely as possible. It may be the case that the character may deviate from the prescribed motion: Either from external applied forces, or due to competition between the balance objectives. Extracting the acceleration from the motion trajectory and setting it as the desired acceleration is insufficient since the tracker will cease following the motion in the presence of disturbances. A PD-controller is used to provide the feedback necessary to correct for disturbances:

$$\ddot{\theta}_{des} = k_s(\theta_{mot} - \theta) + k_d(\dot{\theta}_{mot} - \dot{\theta}) + \ddot{\theta}_{mot} \tag{5.3}$$

where θ_{mot} and $\dot{\theta}_{mot}$ are the motion coordinates and coordinate velocities, and $\ddot{\theta}_{mot}$ is a feed-forward acceleration term extracted from the motion data. Introducing this feed-forward term allows the characters feedback tracking gains to be decreased which allows for less stiff reactions in the presence of external disturbances. The feed-forward acceleration may be calculated by finite differences, or by computing the derivatives of fitted curves.

The objective may now be stated:

$$C_t = \|W_t(\ddot{\theta}_{des} - \ddot{\theta})\|^2. \tag{5.4}$$

C_t is the sum of the squared errors between the accelerations output from the PD-controller and the accelerations chosen by the optimizer. W_t is a diagonal user-specified weighting matrix which allows for additional tracking emphasis or deemphasis on particular joints. Thus, a user can create a motion which makes greater utilization of the arms during balance by simply lowering the weights of the corresponding the arm bodies.

5.2 Linear Balance

The linear balance component controls the characters linear momentum. As we will see, controlling the derivative of the linear momentum of the character is equivalent to controlling the acceleration of the center of mass (CM), $\ddot{c}$. A control scheme is devised which modifies the linear momentum of the character to guide the CM to the center of the support polygon.

Let L_i denote the linear momentum of the ith rigid body:

$$L_i = m_i v_i \tag{5.5}$$

where m_i denotes the mass of body i.

The momentum of the entire articulated body, L, can be computed from the momenta of each individual body:

$$L = \sum_{i=1}^{n} L_i. \tag{5.6}$$

Since the optimizer optimizes over the generalized accelerations and L is a function of velocity, the derivative, $\dot{L}$, will be controlled instead:

$$\dot{L} = \sum_{i=1}^{n} \dot{L}_i = \sum_{i=1}^{n} m_i a_i \tag{5.7}$$

As mentioned previously, the CM acceleration and the momentum derivative are directly related:

$$\ddot{c} = \frac{1}{m_T} \sum_{i=1}^{n} m_i \frac{dx_i}{dt} = \frac{1}{m_T} \sum_{i=1}^{n} m_i a_i = \frac{\dot{L}}{m_T} \tag{5.8}$$

where $m_T = \sum_{i=1}^{n} m_i$.

By maintaining the projection of the CM within the support polygon the character is considered statically balanced. As mentioned in Chapter 3 the foot may still rotate and the CM may leave the support area instantaneously; therefore, controlling the center of mass location is insufficient for balance alone.

Equation 5.8 shows that controlling the derivative of the linear momentum is the same as controlling the mass-scaled CM acceleration. The reason momentum is controlled instead of the CM acceleration is to maintain structural consistency with the angular momentum controller.

The controller can now be stated as follows:

$$\dot{L}_{des} = k_s m_T (c_{des} - c) - k_d (L_{des} - L) \tag{5.9}$$

where k_s and k_d represent spring and damping gains used to control the acceleration of recovery, and L_{des} and c_{des} is the desired momentum and CM position.

c_{des} is chosen to be a user-specified point within the support relative to the center of the support feet, s. Similarly, p_{des} is chosen to be the mass-scaled velocity of s, $p_{des} = m_T \dot{s}$.

Since control of the characters momentum along the gravitational axis is not important to remain within the support polygon, it is ignored. Equation 5.9 is only taken along the two orthogonal axes perpendicular to the gravitational axis while lie within the support polygon plane (in the examples presented, the gravitation axis is Z and the perpendicular orthogonal axes are the X and Y, all expressed in world-frame coordinates). By removing control over the gravitation axis the tracking and angular balance objectives gain full control over the axis without unnecessary interference.

C_{bl} can now be specified in matrix form:

$$C_{bl} = \|\dot{L}_{des} - \dot{L}\|^2 = \|\dot{L}_{des} - (R\ddot{\theta} + r_{bias})\|^2 \tag{5.10}$$

where R is an $2 \times m$ matrix, and r_{bias} and $\dot{L}$ are both size 2 vectors. The calculation of R and r_{bias} from equation 5.7 can be found in Appendix A. The loss is the sum of the squared errors between the desired momentum provided by the PD-controller and the momentum selected by the optimizer.

5.3 Angular Balance

Our angular balance objective regulates the change in angular momentum of the character about the CM. Let H_i represent the angular momentum of body i computed with respect to the CM, c. H_i may be computed from the inertia matrix of body i, I_i:

$$H_i = I_i \omega_i + r_i \times m_i v_i \tag{5.11}$$

where $r_i = x_i - c$, and v_i, ω_i are the linear and angular velocities of body i[17].

The angular momentum of the articulated body, H, is given by

$$H = \sum_{i=1}^{n} H_i. \tag{5.12}$$

$\dot{H}$ may be computed in the following manner:

$$\dot{H} = \sum_{i=1}^{n} \dot{H}_i = \sum_{i=1}^{n} I_i\alpha_i + \omega_i \times I_i\omega_i + \dot{r}_i \times m_i v_i + r_i \times m_i a_i \tag{5.13}$$

where

$$\dot{r}_i = \dot{x}_i - \dot{c} = \dot{x}_i - \frac{1}{m_T}\sum_{i=1}^{n} m_i v_i. \tag{5.14}$$

Building off of the control rule specified in Chapter 3, the goal is to preserve momentum when the CMP is far from the CM. The desired momentum derivative is therefore

$$\dot{H}_{des} = 0. \tag{5.15}$$

The optimization objective function can now be specified:

$$C_{ba} = \|\dot{H}_{des} - \dot{H}\|^2 = \|\dot{H}_{des} - (S\ddot{\theta} + s_{bias})\|^2 \tag{5.16}$$

where S is a $3 \times m$ matrix, and s_{bias} and $\dot{H}$ are both 3×1 vectors. The calculation of S and s_{bias} from equation 5.7 can be found in Appendix B.

Since the system should only attempt to preserve angular momentum when the CMP is far from the COP, the user specified objective gain is scaled according to the distance between the CMP and the CM. Let g and p denote the location of the CMP and COP. The objective weight can be computed:

$$\beta_{ba} = \gamma\|r\| \tag{5.17}$$

where $r = p - g$, and γ is the user-specified pre-scaled objective gain.

As with Eqn. 5.7, the loss is the sum of the squared errors between the desired momentum of the PD-controller and the momentum chosen by the optimizer.

Chapter 6

Extensions

This chapter introduces certain useful extensions to the core optimization objectives presented in Chapter 5.

6.1 Soft Point-Acceleration Constraints

It may be desirable for an animator to control specific points on a body without directly specifying the forward kinematics of the entire character. This problem is analogous to inverse kinematics(IK) where the goal is typically to direct an end-effector to a specified position without deviating too much from a desired posture. For instance, an animator may wish to have his character reach for an object; This would generally be very intensive to perform using a forward kinematic approach where the joint coordinates would have to be explicitly specified.

Point-acceleration constraints operate by constraining a point on the body to maintain a specific Cartesian acceleration. Similar to IK, a user can iteratively drive a fixed point of interest to a desired position by specifying a desired acceleration of the point and integrating the corresponding generalized accelerations. Like IK, the user does not need to explicitly specify the generalized accelerations across the entire character.

In this work the acceleration of the point of interest is controlled using a simple PD-controller of the form:

$$\ddot{p}_{des} = k_s(p_{des} - p_{cur}) - k_d(\dot{p}_{cur} - \dot{p}_{des}), \tag{6.1}$$

where p, $\dot{p}$, and $\ddot{p}$ represent the Cartesian position, velocity and acceleration of the fixed point of interest; k_s and k_d are the constant spring and damper gains; and the subscripts *cur* and *des* refer to current and desired values.

Within this proposed optimization framework, soft point acceleration constraints are used. Soft constraints allow for the optimization to handle multiple, possibly conflicting constraints as well as allow for trading optimization priority with other objectives. With hard constraints the character would be unable to sacrifice acceleration accuracy for additional balance stability provided by the linear and angular balance objectives. As such, point-constraints are modelled as an objective rather than a constraint in the optimization.

The objective for a single point constraint can be stated:

$$\begin{aligned} C_p &= \|\ddot{p}_{des} - \ddot{p}\|^2 && (6.2)\\ &= \|\ddot{p}_{des} - (Q\ddot{\theta} + q_{bias})\|^2 && (6.3)\end{aligned}$$

where Q is a $3 \times m$ matrix and q_{bias} is a size 3 vector relating $\ddot{\theta}$ to $\ddot{p}$. For details on the computation see Appendix C. Multiple point constraints can by handled by introducing additional objectives.

6.2 Soft Joint Limits

This section will show how it is possible to augment the tracking objective presented in Chapter 5.1 to handle joint limits. The section will increase the tracking gains, k_s and k_d of Eqn. 5.3 to produce a larger restoring acceleration as the joint limits are reached. As a desired consequence, other link segments within their limits will compensate so that the desired balancing objectives are still met.

Due to joints being represented as 3-DOF ball joints implemented with quaternions, it is not immediately apparent how one would specify the limits. For a simple 1-DOF hinge or slider joint the generalized coordinate values can be limited

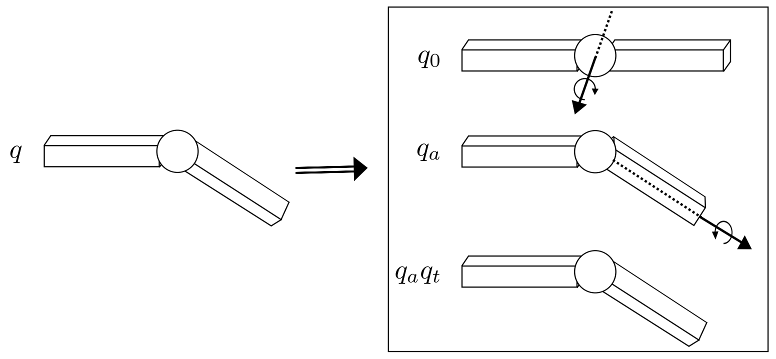

Figure 6.1: The decomposition of a quaternion q into an axis-twist representation: $q = q_a q_t$. q_0 represents the zero-configuration orientation which is the identity quaternion.

using a simple box constraint. Simple box constraint would not work for a quaternion joint since it is not apparent how those limits on the quaternion components would affect the limit space. To places limits on a quaternion-based 3-DOF ball joint a more intuitive decomposition is needed.

A common approach to implementing limits with a quaternion-based 3-DOF ball joint is to convert the quaternion rotation into an Euler angle representation. Unfortunately, a good limit space is still somewhat unintuitive to specify with Euler angles.

This work implements limits using a axis/twist decomposition, where the quaternion representing the joint transformation, q, is decomposed into an axis rotation followed by a twist. Both the axis, and the twist can then be constrained independently. This approach leads to a limit space which closely resembles that of an actual human.

The decomposition can be stated as follows:

$$q = q_a q_t \tag{6.4}$$

where q_a represents the rotation from the inboard axis to the outboard, and q_t represents the twist rotation about the resulting transformed outboard axis. Figure 6.1 illustrates the steps of the decomposition.

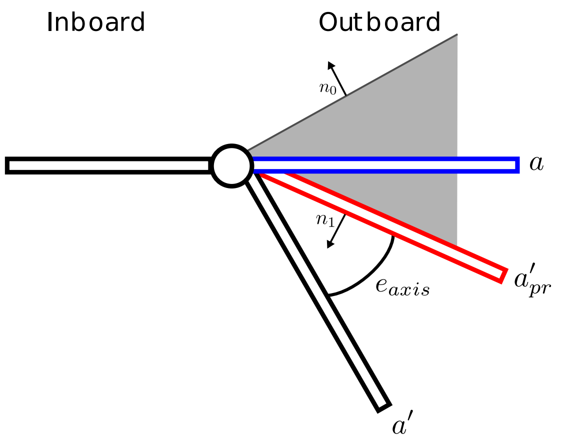

Figure 6.2: A 2D representation of the polygon cone affixed to the inboard link. Here the black, red and the blue links correspond to the transformed axis, the orthogonal projection of the transformed axis onto the limit cone, and the untransformed axis (0-configuration).

To compute q_a the direction of the untransformed bone-axis, a, and the transformed axis, $a' = qaq^{-1}$ are needed. One can then easily compute the axis-angle transform, $\langle v_a, \phi_a \rangle$, needed to rotate a to a' by first computing axis, $v_a = a \times a'$, and the angle, $\phi_a = \cos^{-1}(a \cdot a')$, and then convert back into a quaternion. Once q_a is computed q_t may be easily derived: $q_t = q_a^{-1} q$. It is then possible to intuitively specify a boundary limit for both the axis, a', and the angle, ϕ_t, which can be extracted from q_t by conversion into axis-angle representation.

The boundary limit for the axis is represented as polygon cone, with the polygon normals facing outward from the bounding region. To determine if an axis lies inside of the bounding region a projection of the outboard bone axis onto the normal of each face is performed. If the projected value is negative along the direction of each face normal, n_i, then the axis is within the polygon cone: $a' \cdot n_i < \pi$ for all i. Figure 6.2 illustrates the concept.

The bounding limit for the angle is much simpler to specify. As with simple 1-DOF joints, a box constraint suffices. The constraint requires that the outboard link twist must be within a certain range, $[\alpha_l, \alpha_u]$, where α_l and α_u represent the lower and upper angle twist limits: $\alpha_l < \alpha_u$.

The tracking values k_s and k_d are increased when either the angle or axis limits are violated. The proportion of increase is directly related to the limit violation error. For the axis, this limit violation error, e_{axis}, is the angle between the current axis and its orthogonal projection onto the limit cone boundary: $e_{axis} = cos^{-1}(a' \cdot a'_{pr})$, where a'_{pr} represents the normalized orthogonal projection of a' onto the polygon cone boundary. For the angle, the limit violation error, e_{twist}, is the difference between the twist angle, ϕ, and the bounding values, α_l and α_u: $e_{twist} = \min(|\alpha_l - \phi|, |\alpha_u - \phi|)$. The new values of k_s and k_d, k'_s and k'_d are:

$$k_s' = k_s + l_s(e_{axis} + e_{twist}) \tag{6.5}$$

$$k_d' = k_d + l_d(e_{axis} + e_{twist}), \tag{6.6}$$

where l_s and l_d represent stiffness gains.

The consequence of Eqn. 6.5 is that the character increases its tracking stiffness and damping as the joint limits are exceeded. This extra stiffness results in increased actuation torques output from the inverse dynamics stage which allow for a greater resistance towards limit violations. As an added benefit, this extra resistance increases the loss error of the optimization forcing the optimizer to utilize other unviolated joints to achieve the objective.

Unfortunately, the tracking objective makes no distinction between twist and axis stiffness, therefore any error generated, whether it be from axis or twist limit violations, will result in a stiffening of both the axis and twist. Also, it is necessary to ensure that the tracked motion always stays within the limit boundaries, otherwise the character may unnecessarily stiffen and become unreactive to outside disturbances.

Chapter 7

Results

All simulations were run in real-time at 60 Hz on a 4200+ AMD Athlon machine. Forward-Euler integration with a step size size of $1e^{-3}$ was used to update the dynamics. The optimization was run at a much lower frequency of 200 Hz, although it is possible to run the optimization at a much lower frequency of 30-60 Hz and achieve similar quality results.

Tests were performed across various humanoid and non-humanoid actors for both single-support and double-support motions. Motion was acquired either from keyframing in Blender, or from motion-capture software. All motion was filtered using IK to ensure flat and level support conditions throughout the clip.

A morphologically-realistic model was constructed to match the approximate masses of the 65-kg motion-captured actor. From the measured total mass of the actor, the total mass of each individual body part were approximated using statistics from Nasa on the relative mass distributions between the various body parts[1]. The inertia matrix of each part were then computed to have the desired total mass, assuming an axis-aligned box shape with the uniform density of water. Motion was captured using a 12-camera Vicon motion capture system. Fig. 7.1 shows the skeleton and mass properties of the actor.

For the other models, the skeletons, mass properties, and keyframe motion were roughly crafted. As the results will show, this system may handle poorly

[1]See http://msis.jsc.nasa.gov/Volume1.htm

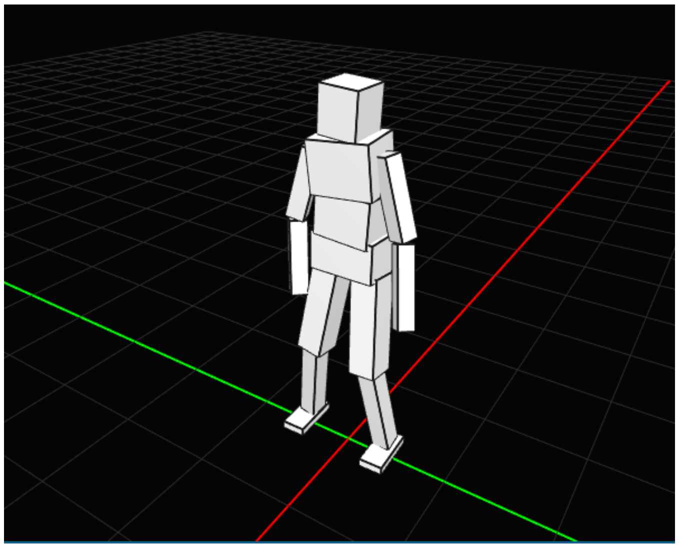

Figure 7.1: The morphologically accurate humanoid model used.

balanced motion from keyframed sources. As with the humanoid, water density and box geometry were assumed.

7.1 Impact

Two tests were performed to measure the robustness of the control to impacts. The first test subjected the character to a series of hand-generated timed perturbations across the upper body. Each perturbation was 150 N applied for 100 ms. For the second test, a single large impact of 220 N was applied for 100 ms to the head. Both tests were performed with various combinations of the objective components enabled: tracking, tracking with linear momentum control, tracking with linear and angular momentum control.

For the sequential impulse test, the character was able to maintain balance with the tracking objective only, however, the supports teetered quite a bit. With the linear momentum and tracking objectives enabled, the character fell after the second impulse. This may possibly be due to the linear momentum component attempting to drive the current CM to the desired CM too quickly, pushing the

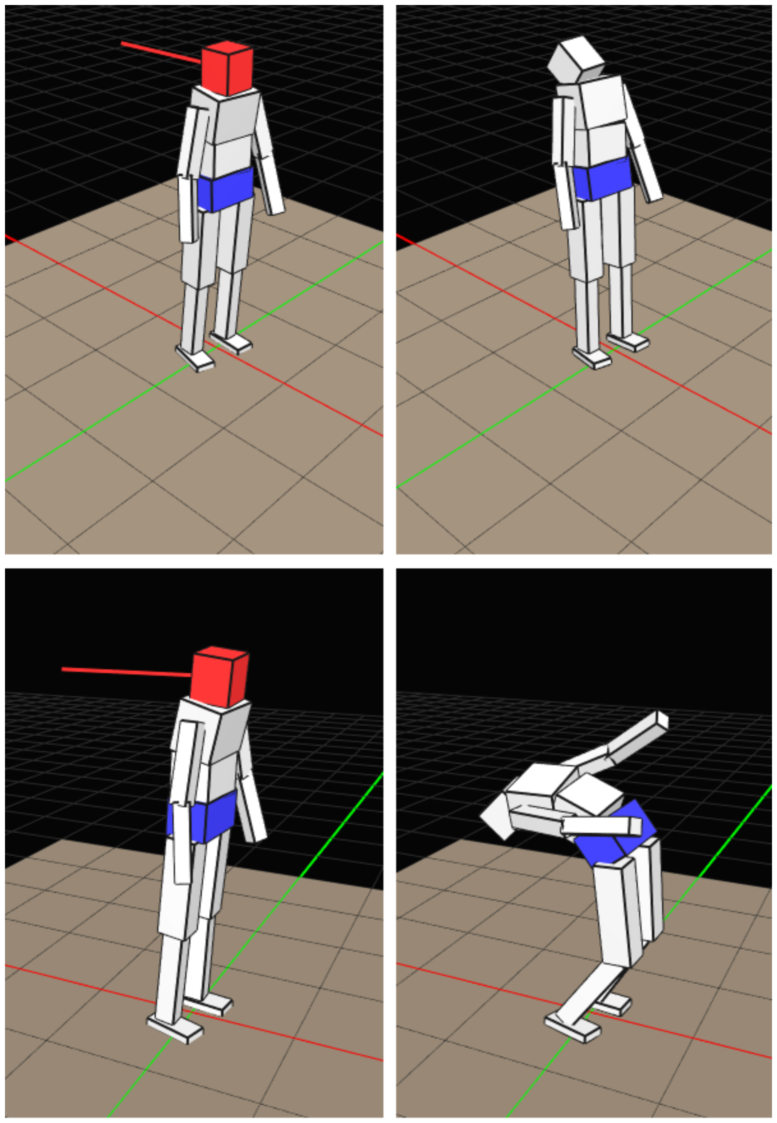

Figure 7.2: A character subjected to a 220 N impact for .1 s. Top row: Resulting impact with linear and angular momentum objectives disabled (note that the supports are rotating). Bottom row: All objectives enabled.

CMP far from the CM. Without the angular objective, the character was unable to redirect the CMP back to the CM quickly enough to avoid foot rotation. With all objectives enabled the character fared much better in terms of stability of the supports and the aesthetics of the resulting motion.

The large impact test resulted in a character unable to balance without the angular momentum objective. The addition of angular momentum objective mitigated the effects of the frictional torque due to linear momentum damping allowing the CMP to return to the CM. Figure 7.2 contrasts the motion from tracking only versus all objectives enabled.

7.2 Tracking

The tracking test attempted to assess the ability of the balance controller to adapt motion well to preserve both stability and style. Due to physical discrepancies between the simulated and real actor, assuming a motion captured source, stability of the physically-captured actor did not necessarily translate to stability of the virtual actor. For some single-support motions tracking was insufficient to maintain balance. In general, strict tracking provided stable results for double-support motion.

For the keyframed motions, the motion was even less balanced as it was not captured from a real actor and placed on a virtual character with approximate mass properties. In some instances the CM trajectory of the keyframed motion would exit the support polygon; Tracking alone would be insufficient for these motions. Humanoid and non-humanoid characters with roughly approximated mass properties and keyframed motion were used to evaluate the systems ability to cope with poorly balanced motion.

Tracking tests across a series of single-support and double-support motions were performed with various combinations of the objectives enabled. Double-support motion included butterflies and standing toe-touch exercises, a fast jab punch, and squats. The double-support motions had little difficulty tracking accurately with all objectives enabled, with a minuscule perceptual difference between

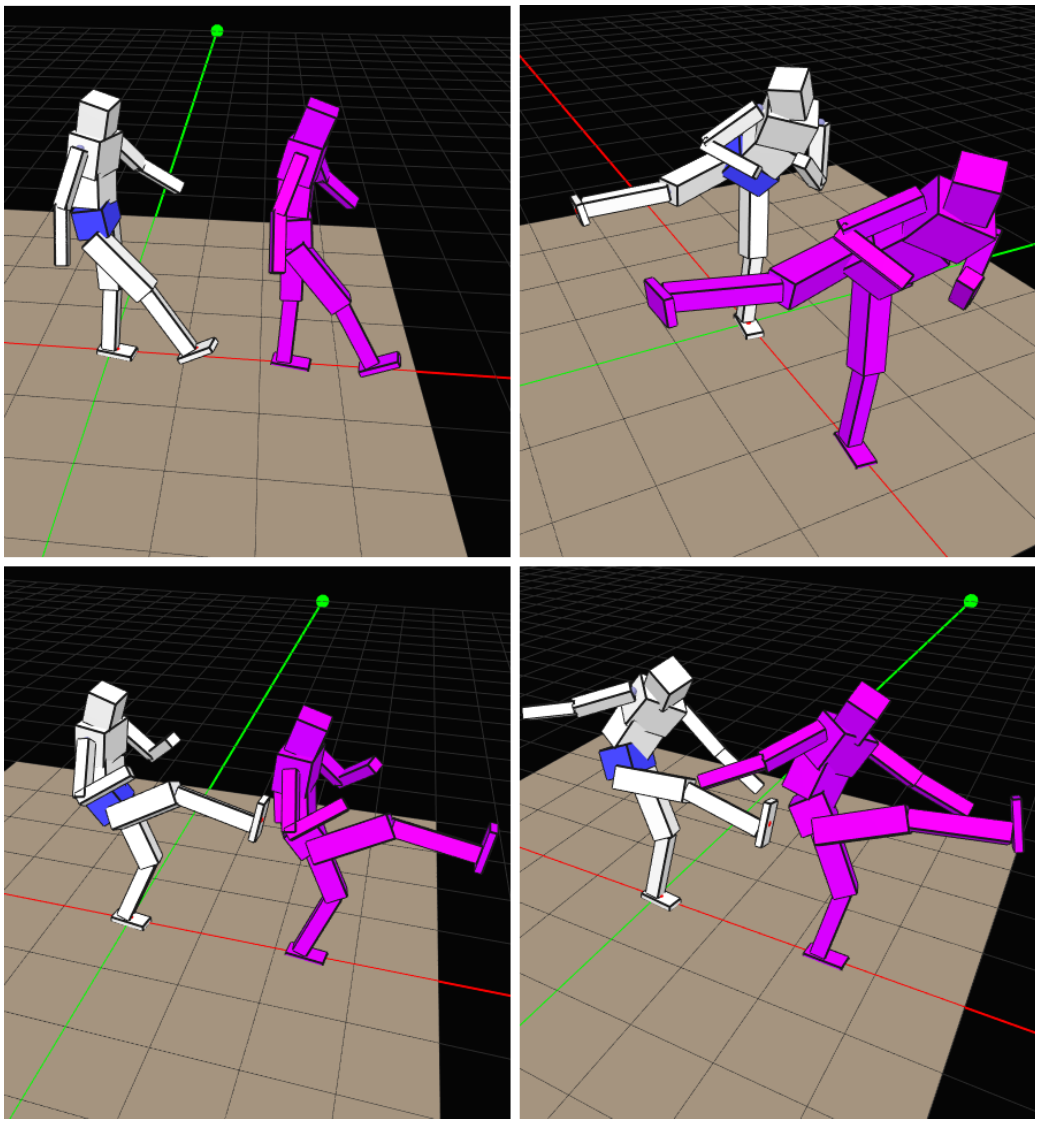

Figure 7.3: Single-support motions tracked. The white character represents the simulation while the purple represents the raw motion data. From top-left to bottom-right: leg-swing, side-kick, forward-kick, axe-kick

the motion data and the simulated motion. The stability could be attributed to the relatively large size of the support polygon and relatively little movement between the COP and CMP. The single-support motion tests were a bit more difficult, especially with some of the high energy kicking motions. Figure 7.3 presents the single-support tracking motions used. While there were parameter combinations which led to balanced motion for all single-support motions, even for those which were unstable during tracking alone, the tracking accuracy of the simulation motion suffered. The system had particular difficulty with the axe kick and swing-leg motion, most notably towards the end of both motions. This error is probably attributed to the increase of tracking error as the motion progressed due to the character striving to maintain balance under unrealistic tracking expectations. Improved style preservation might be achieved by modifying the playback speed of the tracked motion so that the character has more time to resolve the tracking error.

7.3 Low-Friction Tracking

This test determined the ability of the system to track accurately while coping with low frictional forces. The friction coefficient between the character support and the ground was chosen to be 0.03 (ice-on-ice is 0.1). The character performed a single-support leg-circle exercise with all objectives enabled.

The character modified the tracked posture to deal with the lower friction of the surface. As the CMP and COP moved apart the character leaned about the axis of rotation to preserve the angular momentum and regain stability. Figure 7.4 is a snapshot of the character sacrificing tracking accuracy to preserve stability.

7.4 Point Constraints

A set of point constraints were applied to a character to test the systems ability to balance in the presence of competing objectives. Figure 7.5 is an example of a character subjected to three point constraints. The character is able to balance as these constraints are created and dragged around.

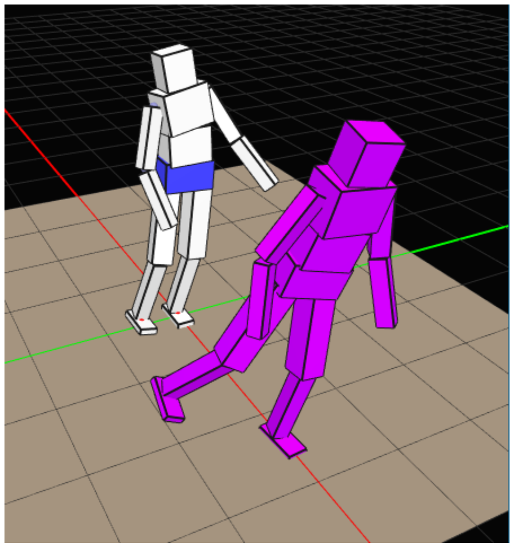

Figure 7.4: A character on a low-frictional surface ($\mu = 0.03$) adjusting its posture to maintain stability. The purple and white character represent the data and simulation posture.

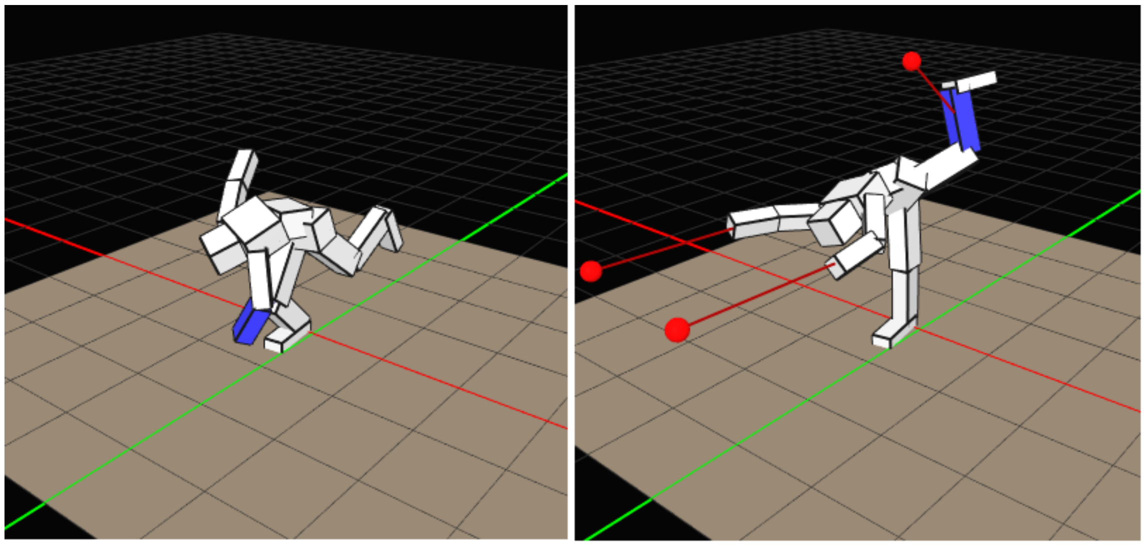

Figure 7.5: Left: Desired motion. Right: A character subjected to three point constraints while simultaneously staying balanced and close to the original reference motion.

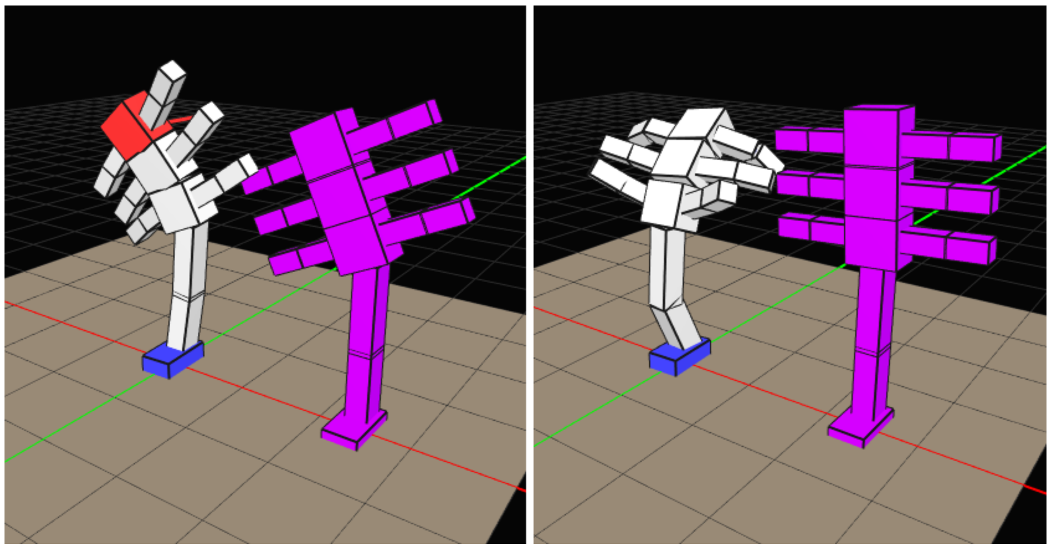

Figure 7.6: A character with a non-humanoid morphology tracking motion while subjected to external perturbations.

7.5 Varying Morphologies

This tests show that it is possible that momentum control is important for characters that do not have humanoid morphologies. Figure 7.6 shows the character tracking a keyframed reference motion with all objectives enabled while perturbed. Interesting arm motions result from the impact.

Chapter 8

Conclusion

This thesis has presented an approach towards balance maintenance which regulates a characters aggregate linear and angular momentum via a fast quadratic optimization. It has been shown that the system is capable of tracking single and double-support motions, even motions which are poorly balanced in the data, while subjected to to moderate and large disturbances and other objective goals.

Certain challenges still remain. Due to the stiff penalty-based ground contact model small integration steps were required. Future work may involve revising the architecture to incorporate a constraint-based ground contact model. This would require solving a new inverse dynamics problem in which the ground force and actuator torques would need to be solved simultaneously. In addition, resolving the problem of compliance resulting from the inverse dynamics approach to ground compensation. Without compliance the motions were susceptible to teetering and unresponsive to large ground force reactions. Lastly, possibly adjusting the playback speed of the tracked motion so that the character can catch up to the desired posture and better preserve overall style.

Bibliography

[1] M. Abdallah and A. Goswami. A biomechanically motivated two-phase strategy for biped upright balance control. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1996–2001, April 2005.

[2] Y. Abe, M. DaSilva, and J. Popović. Multiobjective control with frictional contacts. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 249–258, 2007.

[3] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490, New York, NY, USA, 2002. ACM.

[4] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 402–408, New York, NY, USA, 2003. ACM.

[5] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 223–232, New York, NY, USA, 1989. ACM.

[6] David Baraff. Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA, 1996. ACM.

[7] Ronan Boulic, Ramon Mas, and Daniel Thalmann. A robust approach for the control of the center of mass with inverse kinetics. *Computers & Graphics*, 20(5):693–701, September 1996.

[8] Marco da Silva, Yeuhi Abe, and Jovan Popović. Interactive simulation of stylized human locomotion. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.

[9] M. DaSilva, Y. Abe, and J. Popović. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics 27(3)*, 2008.

[10] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 251–260, New York, NY, USA, 2001. ACM Press.

[11] Roy Featherstone. *Robot Dynamics Algorithm.* Kluwer Academic Publishers, Norwell, MA, USA, 1987. Manufactured By-Kluwer Academic Publishers.

[12] Roy Featherstone. Robot dynamics: Equations and algorithms. In *IEEE Int. Conf. Robotics and Automation*, pages 826–834, 2000.

[13] Michael Gleicher. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff., New York, NY, USA, 1997. ACM.

[14] Michael Gleicher. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff., New York, NY, USA, 1997. ACM.

[15] A. Goswami and V. Kallem. Rate of change of angular momentum and balance maintenance of biped robots. *Robotics and Automation, 2004. Proceedings. ICRA'04*, 2004.

[16] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. Graph.*, 27(3):1–11, 2008.

[17] R. C. Hibbeler. *Engineering Mechanics Statics and Dynamics.*

[18] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Resolved momentum control: humanoid motion planning based on the linear and angular momentum. *Intelligent Robots and Systems, 2003.(IROS 2003)*, 2003.

[19] Mitsuo Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9(6):718–727, December 1999.

[20] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM.

[21] Michael Neff and Eugene Fiume. Methods for exploring expressive stance. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 49–58, New York, NY, USA, 2004. ACM Press.

[22] M. Popovic, A. Hofmann, and H. Herr. Angular momentum regulation during human walking: biomechanics and control. *Robotics and Automation, 2004. Proceedings. ICRA'04*, 3, 2004.

[23] M. Popovic, A. Hofmann, and H. Herr. Zero spin angular momentum control: definition and applicability. *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, 1, 2004.

[24] Seyoon Tak, Oh young Song, and Hyeong-Seok Ko. Motion balance filtering. *Computer Graphics Forum*, 19(3):437–446, August 2000.

[25] K. Yin, D.K. Pai, and M. van de Panne. Data-driven interactive balancing behaviors. *Pacific Graphics*, 2005.

[26] Kangkang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):105:1–105:10, July 2007.

[27] Victor B. Zordan and Jessica K. Hodgins. Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 89–96, July 2002.

Appendix A

Computation of the Jacobian

This section will show how the Jacobian and its derivative, J and $\dot{J}$, may be constructed iteratively from the kinematic equations of motion for an n-link serial articulated body indexed from 1 to n, beginning at the root and ending at the end-effector. While the calculations will be provided for a serial articulated body for simplicity of notation, it can be generalized to branched articulated bodies easily by introducing an indexing function to match a body to its parent.

The calculations will assume that all joints are holonomic: joint motion space is described as an equality constraint between the two bodies independent of generalized velocity that may or may not be a function of time. Let $S(\theta, t)$ denote the matrix which maps the generalized velocity of a joint to a relative Cartesian linear and angular velocity between the bodies. S will be further separated into two submatrices, S^L and S^A denoting the components which contribute to the relative linear and angular velocity. If the joint has m degrees of the freedom, S is a $6 \times m$ matrix. The relative Cartesian velocity between two bodies can be determined from the generalized velocity:

$$\begin{bmatrix} v^R \\ \omega^R \end{bmatrix} = S\dot{\theta} = \begin{bmatrix} S^L \\ S^A \end{bmatrix} \dot{\theta} \, . \tag{A.1}$$

The linear velocity of link i as a function of the linear and angular velocity link $i - 1$ is

$$v_i = v_{i-1} + \omega_{i-1} \times r_i + v_i^R \ , \tag{A.2}$$

where $r_i = x_i - x_{i-1}$. The term $\omega_{i-1} \times r_i$ is the velocity created by the rotation of body $i-1$.

The angular complement to Equation $A.2$ follows similarly:

$$\omega_i = \omega_{i-1} + w_i^R \tag{A.3}$$

Before we express Eqn. $A.2$ and $A.3$ in matrix form as a function of the generalized coordinates and derivatives, the cross-product operator will be introduced. Let $[.]_\times$ be the operator which maps a vector u to a matrix of the form:

$$[u]_\times = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} . \tag{A.4}$$

$[.]_\times$ has the effect performing a cross product with the r.h.s vector (e.g. $[u]_\times w = u \times w$)

To reiterate the definition of the Jacobian,

$$\hat{v} = J\dot{\theta} = \begin{bmatrix} J^L \\ J^A \end{bmatrix} \dot{\theta} \tag{A.5}$$

where J^L and J^A are the rows corresponding to the linear and angular velocities: i.e. $v = J^L\dot{\theta}$ and $\omega = J^A\dot{\theta}$.

Equation A.3 can be expressed as a function of the generalized values in matrix form as:

$$\omega_i = J_{i-1}^A \dot{\theta} + S_i^A \dot{\theta}_i \tag{A.6}$$

where the subscript $i-1$ denotes the $3 \times m$ block-row corresponding to body $i-1$. Put into matrix form:

$$J_i^A = \begin{bmatrix} J_{i-1,1}^A & \dots & J_{i-1,i-1}^A & S_i^A & 0 & \dots \end{bmatrix} . \tag{A.7}$$

Here the second subscript denotes the $3 \times m_i$ column-block for joint i, where m_i is the degrees of freedom for joint i.

J_i^L follows similarly:

$$J_i^L = \left[\begin{array}{cccccc} J_{i-1,1}^L + [-r_i]_\times J_{i-1,1}^A & \dots & J_{i-1,i-1}^L + [-r_i]_\times J_{i-1,i-1}^A & S_i^L & 0 \end{array} \right]. \tag{A.8}$$

Since both J_{i-1}^L and J_{i-1}^A depends only on joint 1 through $i-1$, columns corresponding to i through n are 0.

The derivative, $\dot{J}$, may be computed by taking the derivative of both Eqn. A.7 and A.8:

$$\dot{J}_i^A = \begin{bmatrix} \left[\dot{J}_{i-1,1}^A\right]^T \\ \vdots \\ \left[\dot{J}_{i-1,i-1}^A\right]^T \\ \left[\dot{S}_i^A\right]^T \\ 0 \\ \vdots \end{bmatrix}^T \tag{A.9}$$

and

$$\dot{J}_i^L = \begin{bmatrix} \left[\dot{J}_{i-1,1}^L + [-\dot{r}_i]_\times J_{i-1,1}^A + [-r_i]_\times \dot{J}_{i-1,1}^A\right]^T \\ \vdots \\ \left[\dot{J}_{i-1,i-1}^L + [-\dot{r}_i]_\times J_{i-1,i-1}^A + [-r_i]_\times \dot{J}_{i-1,i-1}^A\right]^T \\ \left[\dot{S}_i^L\right]^T \\ 0 \\ \vdots \end{bmatrix}^T. \tag{A.10}$$

Appendix B

Computation of the Momentum Derivative Matrices

B.1 Linear Momentum Derivative Matrix

The momentum derivative matrix and vector, R and r_{bias}, presented in Eqn. 5.10 are used to map the generalized accelerations to the linear momentum derivative. In this section, it will be shown exactly how R and r_{bias} are constructed.

Eqn. 5.7 will first be placed into matrix form. Let M denote an $3 \times 3n$ block matrix with n, 3×3 mass matrices as it's entries:

$$M = \begin{bmatrix} M_1 & M_2 & \dots & M_n \end{bmatrix} \tag{B.1}$$

where

$$M_i = \begin{bmatrix} m_i & 0 & 0 \\ 0 & m_i & 0 \\ 0 & 0 & m_i \end{bmatrix} \tag{B.2}$$

corresponds to the mass matrix of rigid body i.

Multiplying M by the $3n$ linear acceleration vector, a, we obtain

$$\dot{L} = Ma. \tag{B.3}$$

Eqn. B.3 can be made a function of $\ddot{\theta}$ by observing

$$a = J_{lin}\ddot{\theta} + \dot{J}_{lin}\dot{\theta} \tag{B.4}$$

where J_{lin} and $\dot{J}_{lin}$ are $3n \times m$ submatrices of J and $\dot{J}$ corresponding to the linear entries.

Plugging equation B.4 into equation B.3 we obtain

$$\begin{aligned} \dot{L} &= MJ_{lin}\ddot{\theta} + M\dot{J}_{lin}\dot{\theta} & \text{(B.5)} \\ &= R\ddot{\theta} + r_{bias} & \text{(B.6)} \end{aligned}$$

where $R = MJ_{lin}$ and $r_{bias} = M\dot{J}_{lin}\dot{\theta}$.

B.2 Angular Momentum Derivative Matrix

The angular momentum derivatives matrices, S and s_{bias}, can be computed in a similar manner as the linear matrices.

Let

$$A = \begin{bmatrix} M_1 [r_1]_x & \dots & M_n [r_n]_x & I_1 & \dots & I_n \end{bmatrix} \tag{B.7}$$

and

$$b = \sum_{i=1}^{n} [w_i]_\times I_i w_i + M_i [\dot{r_i}]_\times v_i \tag{B.8}$$

where $[.]_\times$ is the cross-product matrix introduced in Appendix A.

Equation 5.13 can be rewritten in matrix form as:

$$\dot{H} = A\hat{a} + b \tag{B.9}$$

As in the previous section, equation B.9 can be made a function of $\ddot{\theta}$ by plugging in the expression, $\hat{a} = J\ddot{\theta} + \dot{J}\dot{\theta}$:

$$\dot{H} = AJ\ddot{\theta} + A\dot{J}\dot{\theta} + b = S\ddot{\theta} + s_{bias} \tag{B.10}$$

where $S = AJ$ and $s_{bias} = A\dot{J}\dot{\theta} + b$.

Appendix C

Computation of the Point-Acceleration Constraint Objective Matrices

This section shows how it is possible to compute Q and q_{bias} presented in Chapter 6.

Let x denote the position of a body B, v and ω denote the linear and angular velocity of B, and a and α denote the linear and angular acceleration of body B. Also, let r be the vector from body B center of mass to some point p fixed to body B. The acceleration of p, a_p, can be computed as follows from the rigid body motion as follows:

$$\begin{aligned} a_p &= \frac{dv_p}{dt} && \text{(C.1)} \\ &= \frac{d}{dt}\left[v + \omega \times r\right] && \text{(C.2)} \\ &= a + \alpha \times r + \omega \times \dot{r}\ . && \text{(C.3)} \end{aligned}$$

If x_p is the position of point p, then $r = x_p - x$ and $\dot{r} = w \times r$.

Let J_B be the block row of the Jacobian corresponding to body B. The linear and angular velocity of B, $\hat{v}$, is related to the Jacobian by

$$\hat{v} = J_B\dot{\theta} = \begin{bmatrix} J_B^L \\ J_B^A \end{bmatrix} \dot{\theta} \ . \tag{C.4}$$

Eqn. C.3 may be placed into matrix form as a function of the Cartesian acceleration $\hat{a}$:

$$a_p = \begin{bmatrix} I & [-r]_\times \end{bmatrix} \begin{bmatrix} a \\ \alpha \end{bmatrix} + [w]_\times [w]_\times r \ , \tag{C.5}$$

where I is the identity matrix.

The acceleration of B may be determined from the Jacobian, J, and it's derivative, $\dot{J}$, as follows (For details on the computation of J and $\dot{J}$ see Appendix A):

$$\begin{bmatrix} a \\ \alpha \end{bmatrix} = \frac{d}{dt}\left[J_B\dot{\theta}\right] = J_B\ddot{\theta} + \dot{J}_B\dot{\theta} \ , \tag{C.6}$$

where $\dot{J}_B$ is the block-row of the Jacobian derivative corresponding to body B.

Let $Z = \begin{bmatrix} I & [-r]_\times \end{bmatrix}$. Eqn. C.5 can be expressed as a function of the generalized coordinates and derivatives by plugging in Eqn. C.6 into Eqn. C.5:

$$\begin{aligned} a_p &= ZJ\ddot{\theta} + Z\dot{J}\dot{\theta} + \left[J^A\dot{\theta}\right]_\times \left[J^A\dot{\theta}\right]_\times r && \text{(C.7)} \\ &= Q\ddot{\theta} + q_{bias} \ , && \text{(C.8)} \end{aligned}$$

where $Q = ZJ_B$ and $q_{bias} = Z\dot{J}_B\dot{\theta} + \left[J_B^A\dot{\theta}\right]_\times \left[J_B^A\dot{\theta}\right]_\times r$.

Appendix D

Implementation of the Optimization Solver

This work implements a quadratic solver for weighted least-squares problems with linear equality constraints. This optimizer solves problems of the form:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}\|W(b - Ax)\|^2 \\ \text{subject to:} \quad & Cx = d \end{aligned} \tag{D.1}$$

Eqn. D.1 may be solved using the method of Langrange Multipliers. The Lagrangian function of Eqn. D.1 may be stated as:

$$\begin{aligned} F &= \frac{1}{2}\|W(b - Ax)\|^2 - \lambda^T(Cx - d) & \text{(D.2)} \\ &= \frac{1}{2}(Wb - WAx)^T(Wb - WAx) - \lambda^T(Cx - d) & \text{(D.3)} \\ &= \frac{1}{2}x^T A^T W^T W A x - 2b^T W^T W A x + b^T W^T W b - \lambda^T(Cx - d)\ . & \text{(D.4)} \end{aligned}$$

Solutions to Eqn. D.1 occur when $\frac{\partial F}{\partial x} = 0$ and $\frac{\partial F}{\partial \lambda} = 0$. Computing the gradient of F with respect to x and λ we obtain:

$$\frac{\partial F}{\partial x} = A^T W^T W A x - A^T W^T W b - C^T \lambda \tag{D.5}$$

$$\frac{\partial F}{\partial \lambda} = C x - d \, . \tag{D.6}$$

Letting $M = A^T W^T W A$ and $u = A^T W^T W b$, Eqn. D.5 can be rearranged and solved for λ first and then x:

$$C M^{-1} C^T \lambda = d - C M^{-1} u \tag{D.7}$$

$$M x = u + C^T \lambda \, . \tag{D.8}$$