Modeling standing, walking and rolling skills for physics-based character animation

by

Ernesto Torres Vidal

B.S., Instituto Tecnológico Autónomo de México, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

November 2012

Abstract

Physics-based character simulation is an important open problem with potential applications in robotics and biomechanics and computer animation for films and games. In this thesis we develop controllers for the real-time simulation of several motion skills, including standing balance, walking, forward rolling, and lateral rolling on the ground. These controllers are constructed from a common set of components. We demonstrate that the combination of a suitable vocabulary of components and optimization has the potential to model a variety of skills.

Preface

Chapter 4 describes work based on "Generalized Biped Walking Control" published in ACM Transactions on Graphics, 2010 by Stelian Coros, Philippe Beaudoin and Michiel van de Panne. Figures with the phrase "Used with permission" in the caption are used with permision of the authors of the respective papers cited in the caption.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisor Michiel Van de Panne, for his support, guidance, motivation and time. Michiel allowed me to work with encouragement to try new ideas and generate interesting results. I am also grateful to professor Robert Bridson for his valuable feedback on the writing of this thesis and motivating me to work on Weta Digital for an internship and professor Wolfgang Heidrich with whom I worked as Teaching Assistant. I also want to thank my friends: Simona, Nuray, Ozgur, Shabab, Vasanth, Marcos, Lee, Mikhail, Rahul, Anika, Shailen, Dasha, David and to all my friends at Imager lab for making my masters a fun and memorable experience. Finally, my parents Maribel and Ernesto and my sister Maribel and brother in law Francisco for their support during these last two years.

Chapter 1

Introduction

1.1 Motivation

In computer animation for film and games, digital actors play an important role. Most of them are created using either key frame animation or motion capture techniques. However it is also possible to model skills that produce physical plausible outputs by using dynamics simulation.

In robotics, there remains a persistant gap between humans and robots in the ability for robots produce a wide range of agile motions in the same way humans interact with the environment. Humans and animals are able to perform complex, skilful and fast motions. Some examples of robotics applications that could benefit from motion skills are disaster areas like earthquakes, floods and nuclear disasters, where the conditions are dangerous for humans. Other hazardous environments include subsea and space exploration. Having plausible physics based simulation is a way of shortening the existing gap that prevents robots from performing agile motions in the real world.

Dynamic simulation uses appropiate equations of motion to ensure that the generated motions are physically reasonable. The character is modeled as a set of rigid bodies connected with joints. The evolution over time of each joint is complex because unlike in kinematics-based methods it is described in terms of torques. The torques drive the motion of the system and serve to model the net effect of muscle forces. Agile human motions performed in sports are good examples of

what robots are still not able to perform and is a motivation for increasing the dictionary of actions a physics model of a character should do. A basic set of skills to model are those including locomotion like walking, jogging and running. Other actions include rolling in different ways: forward rolling, lateral rolling and roll to stand up. The development of a unified framework that integrates all of these actions is still a challenge. In this work we focus on three main actions: standing, walking and rolling on the floor.

1.2 Contributions

The main contributions of this thesis are:

- The development of controllers used for real-time 3D physics-based simulations of walking and standing skills. The standing and walking are designed to be robust to external force perturbations of moderate size.
- The development of controllers for rolling motions on the ground, which take the character from prone-to-supine, supine-to-prone and stand-to-supine poses. An optimization framework is introduced in order to help with the design these controllers.
- A software framework to model the character, the actions to be performed and controllers design using XML scripting to run tests and generate results.

1.3 Organization

This thesis has the following structure. Chapter 2 reviews related work in computer animation and robotics. Chapter 3 describes the main components used to design the controllers and presents a system overview. Chapter 4 describes the strategy followed for standing with balance using virtual forces and the transition to falling. Chapter 5 describes the integration of several components for balanced walking. Chapter 6 describes the controllers for rolling using an offline optimization process to obtain the parameters. Chapter 7 concludes and provides directions for future work.

Chapter 2

Related work

There are several approaches to creating animated character motion. Kinematic techniques resequence and interpolate existing joint trajectories that are manually created through keyframing or are obtained from motion capture data. Trajectory optimization methods create and adapt existing trajectories automatically in order to satisfy the goals of the motion as well as criteria such as the physical realism of the motion. Finally, controller-based methods use physics-based animation to model the motor control and forward dynamics simulation to synthesize the motions. The three approaches are now discussed in detail.

2.1 Kinematic methods

2.1.1 Key frame animation

Key frame animation is a commonly used technique in the film and game industry to describe the motion of a character. The animator defines key poses in time and the computer interpolates the poses in time. Afterwards motion retiming curves can be used for more precise control over the timing. In interactive applications, motion retiming is based on events, rules and scripts. One of the main disadvantages of key frame animation is that it cannot interact with the surrounding environment using the laws of physics. Since there is not a general model to produce motion, the result relies entirely on the user and therefore is not of practical use in robotics

and dynamically simulated characters.

2.1.2 Motion capture

Data driven animation consists of describing the motion of each joint in time based on motion data captured from a motion capture system. Approximately 50 markers are attached to the actor in a standard system. A skeleton can be composed of approximately 90 degrees of freedom. The process of motion capture consists of the following steps

- Cameras capture images of the capture volume at a high frame rate.
- The tracking of the sensors begins by segmenting and identifying them in the images. The correspondences between markers are determined from multiple viewpoints and a 3D cloud of marker positions is reconstructed.
- A generic skeleton model is constructed using a range of motion sequence to adjust the length of the bones and the position of the markers on the bones by a high dimensional optimization approach.
- The poses of the joints of a character model are estimated by fitting the skeleton with known dimensions and bone marker locations to the 3D marker positions of a given time instant.

In films and games, animators either use these poses directly or else use them as a start starting point to generate animated motion. Data driven animation suffers from one major drawback: the ability to generate realistic and non-repetitive responsive animations is always restricted by the contents of the motion database. The promise of physics-based methods is to produce a simulation model that produces motions adapted to a given situation and physics.

2.2 Trajectory-optimization methods

A trajectory is the set of values of all the degrees of freedom of an object, modeled as a function of time. The basic premise of this method is that an animator specifies the goals for a motion, and a physically plausible motion is automatically created. An optimization method is used to solve for the joints trajectories

and global position and orientation over time generating a resulting motion with desirable features such as follow-through and anticipation [28]. Trajectory optimization methods, commonly referred as spacetime constraint methods, have been used for a variety of scenarios, including creatures with arbitrary morphologies [24]. These approaches typically lead to very large scale optimization problems, especially when complex characters with many degrees of freedom are to be animated or when long motions are desired. The optimization criteria can influence the quality of the synthesized motions or require some degree of manual parameter tuning. These methods can get stuck in local minima, and require significant time to be computed. Therefore they are not suitable for real-time applications and are better suited for off-line animations.

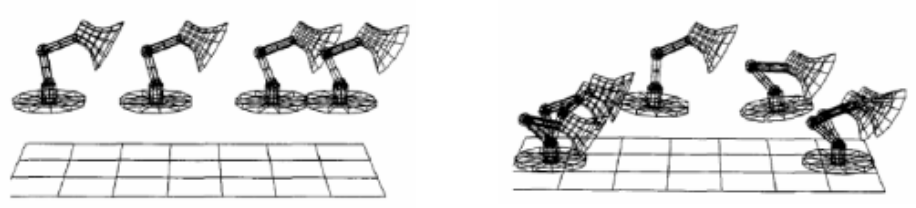

Figure 2.1: Spacetime optimization example. Initial trajectory (left) and Optimized trajectory (right). Adapted from Witkin and Kass[28].

2.3 Controller-based methods

Another method to generate animated motion is by using internal forces and torques to drive plausible motions in the simulated world. The idea of using physics simulation to animate virtual characters allows interaction with objects without the need for additional motion data. After the torques and forces are computed a forward dynamic simulation is used to obtain accelerations. The accelerations are integrated to update the positions and velocities of the rigid bodies and therefore the motion of a character. Instead of computing the accelerations for a known set of joint torques, it is also possible to do this the other way around: compute the torques and forces required for a character to perform a specific motion. This process is called inverse dynamics. It is often used in biomechanics to analyse the dynamics

of human motion, based on motion data that is augmented with external force measurements [8]. Inverse dynamics is also used in physics-based character animation to find the torques required to achieve a desired acceleration [9, 13, 15, 30]

Modeling motions in a physics environment is a hard problem because of the high dimensional nature of human motion. Some advantages offered by these methods include the interaction with the environment and the use of friction forces to represent collisions with other objects. [4]

Some of the tasks a humanoid character could be asked to perform are walking, standing, rolling, standing, jumping and balancing. The success of a physics-based animation relies entirely on the design of the controller. It is therefore important to increase the set of skills available to a character, which is our chief motivation in this work. The set of skills included in this work are walking, standing, falling and rolling.

In computer animation, in some early applications motion styles appeared stiff or robotic and required detailed modeling of the tension and relaxation simulation [16]. Considerable previous work has been focused on locomotion. Foot placement is a key element and it has been a key component to achieve balanced locomotion in animation [6, 7, 12, 18, 25, 31]. Other work has beed focused on rising up and falling strategies [19].

The main components used in this thesis are based on those proposed by Coros [5]. This solution is directly applied to characters with different proportions, mass distributions, styles and velocities. Some components such as the inverted pendulum model have also been explored elsewhere for example in a balance filter method to track a motion capture reference trajectory with stepping modified by the model [23].

There are other methods that use learning and optimization to avoid the use of complex analytical models of the dynamics and careful manual tuning of parameters [20]. For walking and rolling it is hard to develop the right objective functions and the results are computed in an offline process or rely on a particular character model or a known initial position to start the motion. A set of motions that are interesting to look for future work are rising up motions [26] . Similar components to the ones used in this thesis have been shown to work for quadruped locomotion [3] and further use optimization techniques for parameter adjustment [25]. Since there

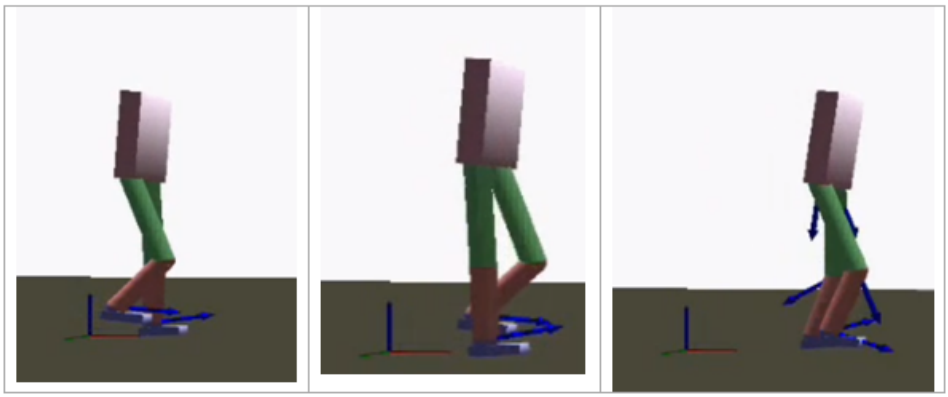

Figure 2.2: Example of controller-based locomotion [31]

Figure 2.3: Physics based character rolling on the floor. Used with permission from van de Panne [14]

are multiple configurations, this is often solved using a mix of control components and optimization.

Chapter 3

Control framework

This chapter describes the theory of the control components use for the standing, walking and rolling skills. A description of the physics of the character is provided. Finally, a description of the modules of the system framework and how they are used for each skill.

3.1 Character model

Most physics-based characters are modelled after humans, while some research papers model characters based on animals [3, 10, 22], robots [18] or creatures that do not exist in nature. Character models are often simplified to increase simulation performance and to make them easier to control. The choice of the model is usually linked to the control strategy of the character. The biped character used in this thesis is designed using a set of 15 rigid bodies. The character in total has 14 joints as is shown in Figure 3.1. Rigid body primitive geometries such as spheres, cylinders and boxes are used to generate the overall character geometry to deal with collisions.

The mass distribution of the character reassembles that of a human body with more mass distributed in the torso and head than in the limbs. This is important for designing walking and rolling controllers since it affects the balance and the overall design of a successful strategy. The joint types depend on the degrees of freedom, for example, hinge, universal and ball joints are the ones most used in

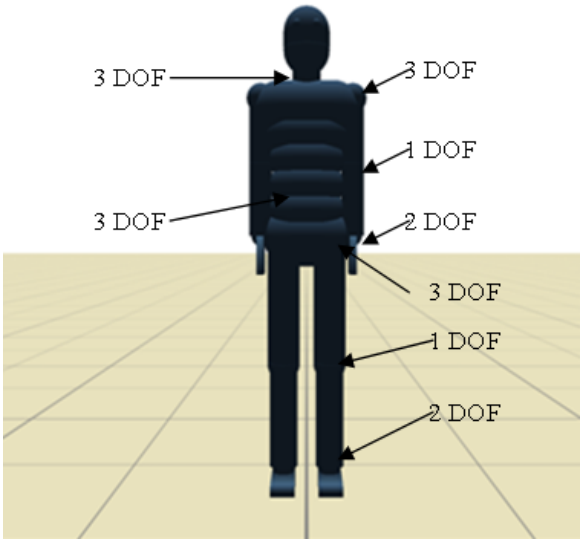

Figure 3.1: Character model using rigid bodies on its initial pose.

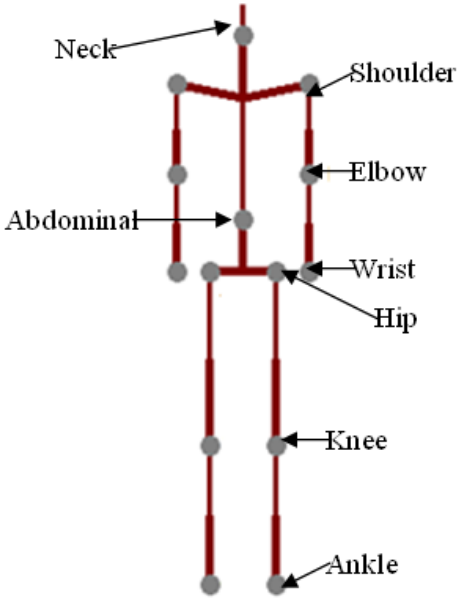

Figure 3.2: Character skeleton and joints used.

Parameter	Values
Height of the character	1.81 m
Mass of the character	70 Kg

Table 3.1: Character height and weight

this work and they have one, two and three degrees respectively. The lengths of the rigid bodies were chosen using the anatomical proportions of a real human body. The inertia tensors were set to correspond to a uniform mass density. The mass density is set to be the rigid body mass divided by its volume.

Including 6 DOF for global position and orientation the character has a total of 36 DOF. 3 DOF corresponds to ball joints, 2 DOF to universal joints and 1 DOF to hinge joints as shown in Figure 3.2. For collisions against the floor and other objects contact joints are used. The parameters for friction and restitution are set to be 0.5 and 0.1 respectively. These values are constant for all motions described in this work.

3.2 Pose tracking

The pose tracking controller takes as input the desired relative orientation of the bodies in a particular controlled joint and computes the joint torques using proportional derivative control (PD). For one degree-of-freedom (DOF) joints such as the elbow and the knee, the axis of rotation remains constant and equal to the joint normal plane of motion so the difference can be defined using the angle between the child body angle with respect the parent reference frame and the desired angle. The resulting error angle is used to compute the torque with the PD control as follows: $\tau = K_p * E_{angle} + K_d * E_{angle}$ where E_{angle} is the error angle

For two and three DOF joints such as the shoulders and the hips, the axis of rotation is computed in the plane composed by the current and the desired quaternion. The difference between the current quaternion and the desired quaternion defines an error angle and an axis of rotation. The PD control is applied then similarly to the one DOF joint. The only difference is that the axis of rotation for the desired quaternion should be specified. This process is similar to a yaw, pitch and roll ori-

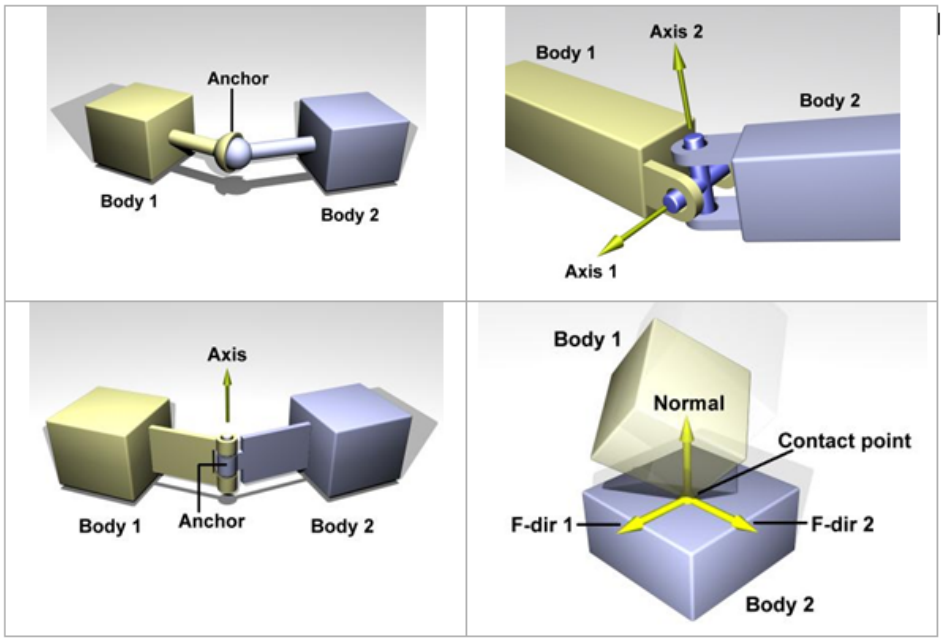

Figure 3.3: Joint types used for the articulated character and contact joints used for collisions. Adapted from Smith [21].

entation. In this case using quaternions avoids numerical instabilities. When the child orientation is controlled with respect the world frame, the desired orientation is transformed to obtain the respective error.

3.3 Inverse kinematics

Inverse kinematics (IK) is a commonly used tool for helping to pose characters for animation. IK automatically computes the joint angles that result, for example, in the character's hands and feet being placed at specified locations in the world. This allows foot and hand trajectories to be used to help define full body animations [1, 29].

In this thesis, IK is used to control the pose of two link legs by specifying the trajectory of the end effector. The main value of using the IK component is that the position of a foot can be set directly by another control component. To prevent numerical problems, the end effector target is adjusted so it lies within the limb's

reach. In three dimensions the normal of the plane is constrained to a particular direction, for example the pelvis on the leg or the torso for an arm since there is otherwise no unique solution.

3.4 Joint PD control

The simplest way of controlling the rotation of a character joint is by setting a desired quaternion and computing the corresponding torque using a PD controller. It is possible to vary the rotation angle over time. Figure 3.4 shows a Catmull-Rom spline containing four knots with different values. The trajectory indirectly controls the speed of the movement because when the desired value drifts away from the current value the error increases producing a higher torque as an output of the PD control. Working with Catmull-Rom trajectories produces smooth motions and it is easier for the user to design it, particularly when the joint is controlled in the world reference frame.

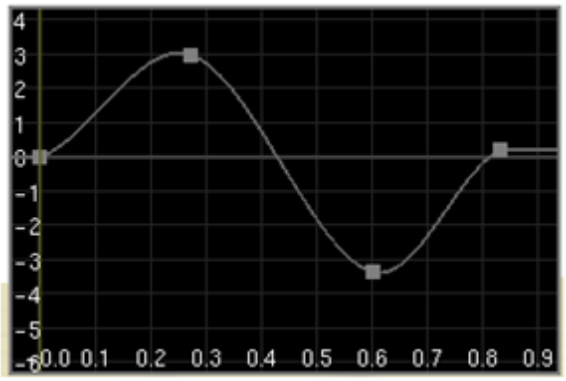

Figure 3.4: A Catmull-Rom spline containing four knots is used to define a trajectory evaluated in one second phase length to describe the rotation angle of a joint

3.5 Virtual forces

Virtual forces (VF) are a method that allows the generation of joint torques for a chain of links to counter a force applied at the end effector, similar to the effect of an external force. The base link is typically close to the root of the character.

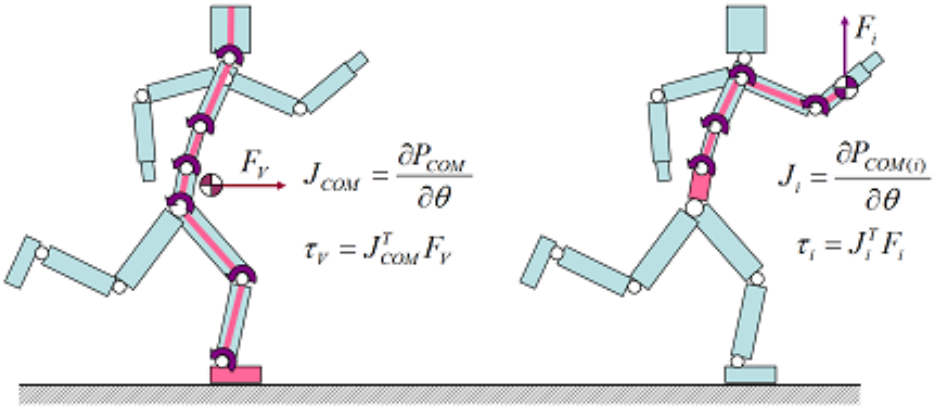

Figure 3.5: Virtual forces applied for A)Velocity tuning and B) Gravity compensation. Used with permission from van de Panne [5]

Forward kinematics relates the force with the torque applied over a joint by the cross product of the Jacobian as follows:

$$\tau = J^T F \tag{3.1}$$

In this thesis virtual forces are used in the walking controller for gravity compensation and for velocity tuning. They are also used for the standing controller to maintain balance. Figure 3.5 shows an example of the application of a virtual force to the link of joints within the character. Virtual forces help PD controllers to be more precise and work with lower gains by directly computing the torques to compensate for many of the stronger forces involved in a motion such as gravity. Virtual forces must be applied in both sagittal and coronal planes in three dimensions.

3.6 Inverted pendulum model

The inverted pendulum model (IPM) is used for walking balance control. For testing this component a one leg character was first used, the torso was pushed and the resulting foot step calculated. When using two legs it is important to consider the

distance between the hip joint of the stance leg and the hip joint of the swing leg. This distance must be added to the distance obtained to produce a correct position for the desired foot position. The position is only computed once at the beginning of the state to generate a trajectory that remains constant during the rest of the state time.

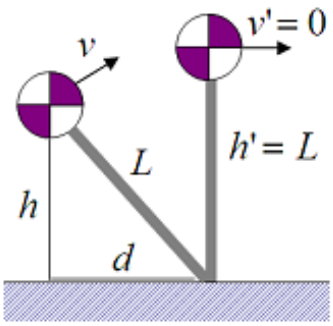

Figure 3.6: Inverted pendulum model Used with permission from van de Panne [5].

In particular, we desired to compute a desired stepping point (xd, zd), using an inverted pendulum model. The model used assumes constant leg length [17] as shown in Figure 3.6. The analysis equates the sum of the potential and kinetic energy of the IPM at the current state, described by its current velocity v, its height h, and distance, d from the future point of support with that at the balance rest state $\frac{1}{2}mv^2 + mgh = \frac{1}{2}mv'^2 + mgh'$, where $v' = 0$ and $h' = L = \sqrt{h^2 + d^2}$. Solving this relation for d gives $d = v\sqrt{h/g}$. The above model computes the desired value of d in order to reach zero velocity at the next step. Taking a shorter step will achieve a positive velocity while taking a larger step will achieve a negative velocity producing a backwards movement. Accordingly, d' is computed $d' = d - \alpha V_d$, where Vd is the magnitude of the desired velocity and alpha is a constant. A value of $\alpha = 0.05$ is used in this work, similar to Corots [5]. The IPM is applied in both sagittal and coronal planes in an analogous way. The true center of mass and velocity are computed and used in this work. The IPM prediction may be out of reach so a maximum step length of $d = 0.6L$ was used.

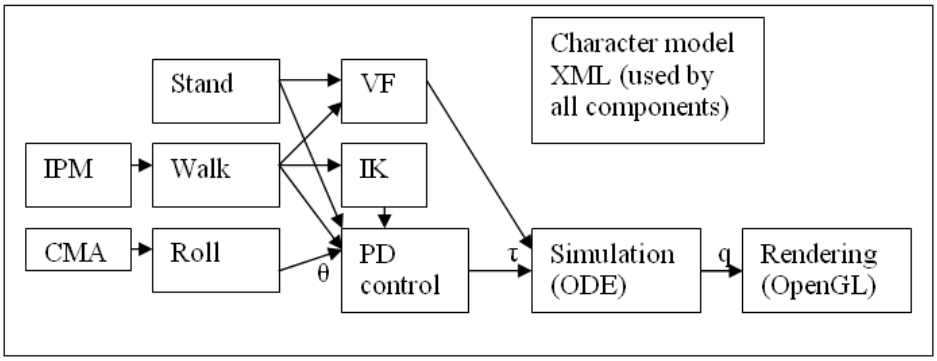

Figure 3.7: Modules of the system θ joint angles, τ torques and q positions and orientations. IPM (Inverse pendulum model) CMA (Covariance matrix adaptation optimization method) VF (Virtual forces) IK (Inverse kinematics) ODE (Open dynamics engine) PD (Proportional derivative)

3.7 System framework

Both the character model and the actions are described in XML script files. The main advantage is that the code is kept devoid of parameters and it is possible to easily test different components together and see what the output is without making changes in the code. The framework consists of a control system on top of a physics engine. The control system assembles basic control units into states and actions. These units are designed using a XML script. The controllers work on top of a layer that interacts with Open Dynamics Engine: the physics simulation engine. The system has the following overall structure:

1. Input scripts: This module reads all the parameters describing the model of the character and the initial pose. A second file includes the actions, states per action, transitions and the values for the control trajectories and parameters for the control components.

2. High level control: This module includes the implementation for the CMA optimization and the implementation of all the high level components such as the inverted pendulum model (IPM), virtual forces (VF), inverse kinematics control (IK).

3. Low level control: This modules includes the transitions between states and action using time, contact or direct transitions described in the input XML file and evaluates trajectories to control the joints of the character through PD control.

4. Rendering: This module uses OpenGL to display all the elements of the scene, mainly the floor and the rigid bodies of the character, but also the GUI, the display of the skeleton and control trajectories.

5. Physics engine ODE: This module interacts with ODE to perform the simulation steps, apply forces and torques, compute positions and velocities.

The character model has the following properties:

- Initial position and rotation of the character center.
- Body part: Group of rigid bodies, joints and PD controllers.
- Rigid body: A composite body composed of primitive geometries like sphere, cylinder and boxes, some of its properties are the mass, the friction and restitution parameters, the lengths, the color and the initial position
- Joint: A joint connects two rigid bodies, some of its properties are the hard limits, the axis of rotation and the anchor point
- PD Control: Controller parameters attached to a joint including Kp, Kd and maximum torque allowed.

Action	Optimization	Quaternion control	IK	Inverted pendulum	Virtual forces
Stand	No	Yes	No	No	Yes
Walk	No	Yes	Yes	Yes	Yes
Roll	Yes	Yes	No	No	No

Table 3.2: Framework components

Table 3.2 shows the main components included in the framework (Optimization, Quaternion control, Inverse kinematics (IK), Inverted pendulum model (IPM)

and virtual forces (VF)) and how are they used for each skill. Figure 3.7 shows all the components including in the system.

The software was developed in C++ using Microsoft Visual Studio 2010. It contains around 70 classes and approximately 15 thousand lines of code. The simulation runs between 30 and 60 frames per second including simulation and rendering time. The dynamics simulation step is 100 Hz. The project was run using a an Intel Core Quad CPU with 2.4 GHz and 3.24 GB of RAM. with an ATI Radeon HD graphics card.

Chapter 4

Standing and falling

This chapter focuses on describing the integration of control components including virtual forces and PD joint control for standing and passive control for falling.

4.1 Standing controller implementation

One of the simplest actions to perform for a character is to maintain balance without stepping. As simple as the motion is, the action requires the use of several control components. If we consider how a person keeps balance without stepping we can realize how a group of muscles in the legs and the torso are used. If a person is pushed forward, a natural reaction is to shift weight towards the toes of the feet to bring the center of mass to a vertical position.

Maintaining the character in the standing position without stepping in place is achieved by using a combination of local PD joint control and a virtual force computed according to $F = K_p * (X_d - x) - K_d * (X_d - x)'$ to apply torques into the legs and torso computed. This implements a PD control based on the error between the ideal center of mass located vertically over the ankles and the real center of mass as shown in Figure 4.1. The torques are applied in both sagittal and coronal planes. Lifting the arms forward and backward helps by moving the center of mass in the opposite direction allowing the character to remain in balance within some reasonable range.

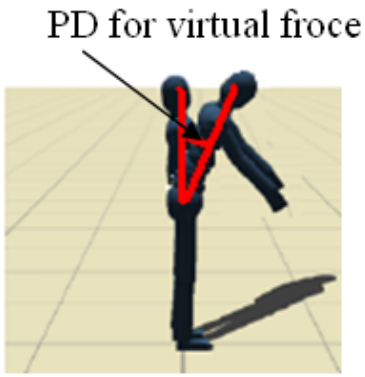

Figure 4.1: Virtual force using a PD control for standing control

4.2 Falling controller implementation

In order to stop applying the standing and walking controllers on the character it is necessary to determine when they are no longer effective. One simple way of doing this is to compute a valid range for the vector going from the center of the torso to the head and transitioning to the fall action once it goes out from this range. Once a fall is detected, the fall action disables any torques applied over the joints and the character falls passively like a ragdoll. Using a ragdoll looks more natural than attempting to walk or stand while falling. Humans tend to use their arms to protect the head and torso and cushion their fall . This would require to implement a full-fledged falling controling which was not done due to time constraints in the development of the solution.

4.3 Results

Joint limits were added to have a natural range of motion on all joints. The limits remain constant for both walking and standing.

Figure 4.2 shows a resulting simulation for standing actions while receiving pushes (red arrow) within a range not high enough to make the character lose balance. Tests were performed with slight changes of the terrain slope. Since the standing controller uses the center of mass it is possible to stand over variations of up to 20 degrees as is shown in Figure 4.3 and Figure 4.4. Finally, Figure 4.5

Parameter	Values
PD parameters	Kp 50 and Kd 10
Maximum push tolerated	250 N
Torso angle limit	1.0 *rad*

Table 4.1: Standing controller parameters

shows the simulation of the character falling down like a ragdoll after receiving a strong push.

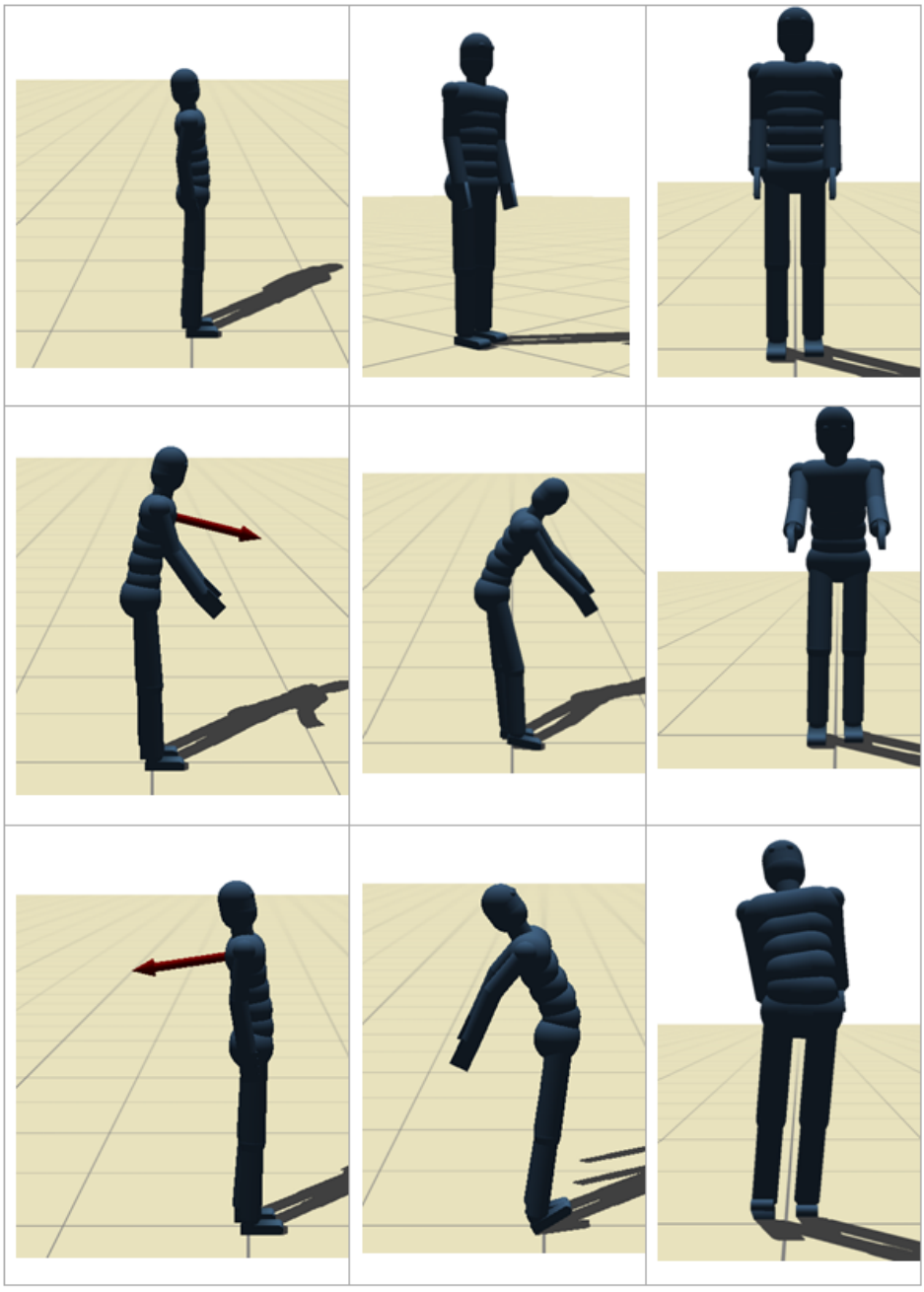

Figure 4.2: Simulation of balance standing action under moderate pushing

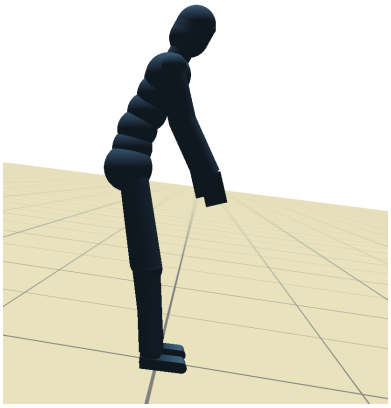

Figure 4.3: Simulation of balance standing action on a -20 degrees slope

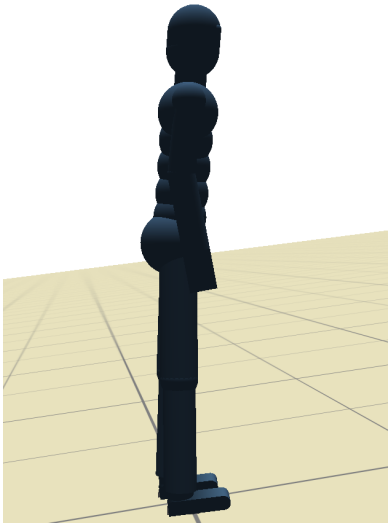

Figure 4.4: Simulation of balance standing action on a 20 degrees slope

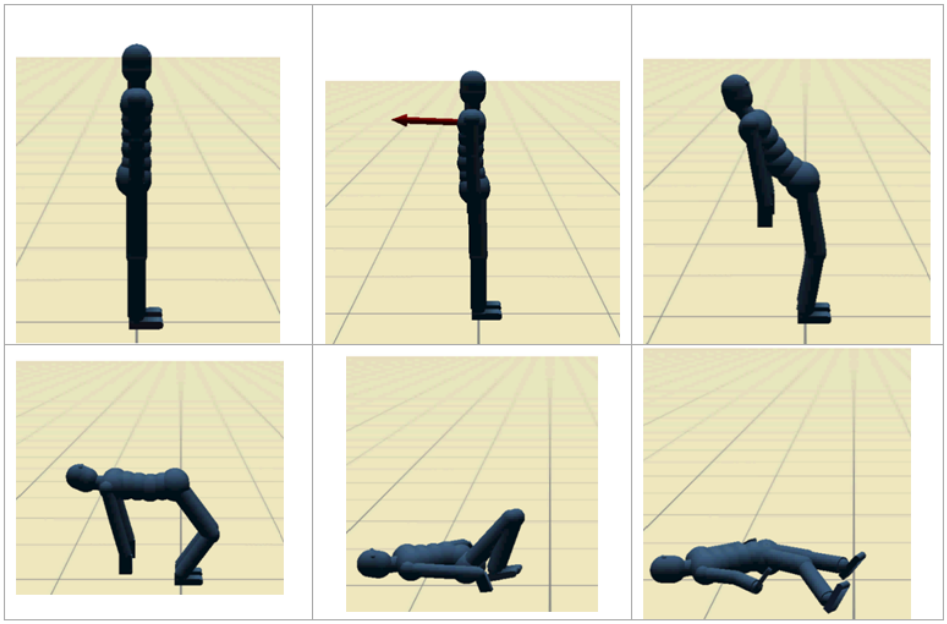

Figure 4.5: Character falling like a ragdoll after receiving a strong push

Chapter 5

Walking

The locomotion used for bipedal walking with balance is a broad field of study with many aspects to consider. This chapter focuses on describing the main components of the "Generalized Biped Walking Control" [5] used to implement the walking controller. Afterwords, we describe the process we followed to progressively implement and debug the controller.

5.1 Walking components

The control system consists of four key components that are integrated as shown in Figure 5.1.

The first component is the motion generator; it produces open loop desired joint angle trajectories, which are modeled as spline functions over time. Joint angle trajectories can be modeled relative to their parent-link coordinate frame or relative to the character coordinate frame (CCF), whose y and x axes are aligned with the world "up" vector and the character's facing direction, respectively.

The second component is a balance mechanism is added through the use of foot placement. It uses an inverted pendulum model (IPM) which helps achieve motion that is highly robust to disturbances such as pushes in any direction. The IPM does not require parameter tuning because the relevant parameters are captured by the model. This is the main component that works as feedback to maintain balance during walking as shown in Figure 3.6

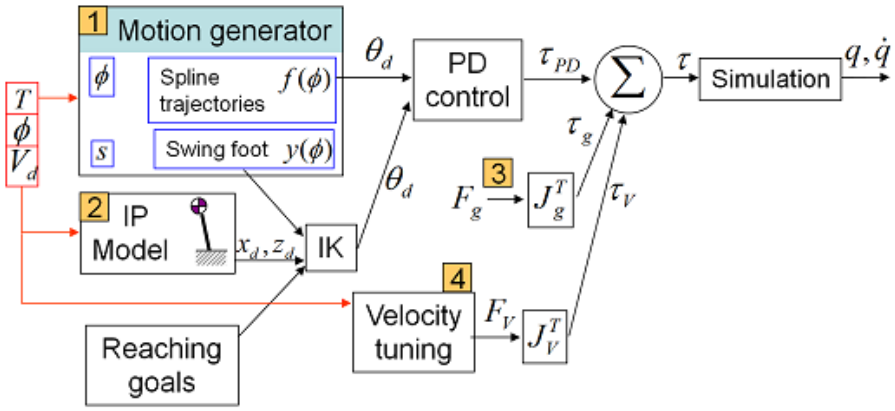

Figure 5.1: Generalized biped control modules. Used with permission from van de Panne [5]

The third component is the computed-torque gravity compensation. This compensation provides a way of reducing the torque applied. The addition of a virtual gravity compensation torques, allows for acceptable accuracy tracking to be achieved using low-gain PD tracking in joint space. The low gains, in turn allow for a more compliant, natural motion as shown in Figure 3.5.

The fourth component is the use of virtual forces on the center of mass to accelerate or decelerate the character towards the desired velocity. The torques are computing using the Jacobian transpose similar to the gravity compensation torques $\tau = J^T F$ as shown in Figure 3.5.

The walking action consists of two states: left stance and right stance. The trajectory of the endpoint of the swing leg is described using three Catmull-Rom trajectories one for each of x,y and z. The trajectories are modeled as a function of the phase step $\Phi \in [0,1]$. This trajectory is a reference of how we desire the leg to move. These trajectories do not necessarily coincide with natural trajectories as would happen in the data footage of a person walking since they are just a guide on how the torques should be applied. A feedback component is added to the target location to achieve step-to-step balance using the inverted pendulum $d = v\sqrt{h/g}$. The resulting distance is computed in both sagittal and coronal planes. The output overwrites the desired trajectory. Finally the two virtual forces components are

added. The gravity compensation is added to the swing leg and the velocity tuning to the stand leg. Another sensor feedback component is required to maintain the internal torques of the hips equal to zero. The direction of the walking should also be specified and is described by the standing leg. The transition between both states occurs when the standing leg collides with the floor or when the state reaches the time limit.

There are some components of the original generalized walking control paper [5] solution that were not included in this work. Testing was performed with a single character model and no GUI was implemented to edit and modify the proportions of the character body. Interaction with objects was not included, except for the collisions with balls thrown at the character. Finally, navigation over and under obstacles requires a higher level control outside of the scope of this work.

5.2 Walking controller implementation

In order to implement the walking action in our framework the design was slightly modified from the generalized biped walking to split the functionalities into modular control functions and extend it later on into other actions like standing and rolling. The walking key components where separated into orientation control, inverse kinematics, gravity compensation, velocity tuning, inverted pendulum and stance leg direction. We now describe some specific strategies useful for progresively implementing and debugging the walking control.

The first step in the design of the walking controller is to integrate the components from the SIMBICON framework described previously. During the development it allowed testing the PD control between two quaternions in 3D. Maintaining balance using a PD controller for the torso in the world reference frame is a good first approximation towards a more complex solution using the inverted pendulum model. Some of the challenges in this first approximation is that quaternions are hard to visualize and the resulting torque direction must coincide with the shortest direction of rotation. In this solution there were not virtual forces used. A way of testing the walking in two dimensions is by adding a plane joint to one of the rigid bodies. This automatically constrains the movement in the physics engine. Once everything is adjusted to walk forward and backwards, lateral control is added. The

main problem of moving laterally is that depending on the character configuration it is likely that when the foot is lifted the standing leg is not strong enough to maintain the torso straight and this one starts going into one side. To avoid this, the PD control parameters were not modified but the desired hip angle can be adjusted to go slightly in the opposite direction than the swing leg.Once the walking in site works in three dimensions the parameters of the trajectories are adjusted to achieve a forward direction. It also happens that it takes the first two or three steps for the character to get the necessary forward push. After that, the character keeps moving at a constant velocity.

The second step during the design of the walking controller is to add a trajectory for foot placement and inverse kinematics for the swing leg instead of using desired values for the quaternions. The inverse kinematics module represents another challenge to implement correctly. To do this it was better to test it on a two link arm connected to a static body without touching the ground. Some of the aspects taken into consideration are the following. It is necessary to define a plane, in this case normal to the knee axis of rotation. After that, by computing the solution analytically using a standard inverse kinematics approach it is possible to obtain two desired quaternions that were specified manually previously. These quaternions will define the motion of the swing leg by using a trajectory for how the foot is lifted.

The third step is adjusting a desired walk style. This is hard to do without using virtual forces. The precision required by the PD controller is not that high because the weight of the torso and swing leg is higher than half the total weight of the character. A first intuition step is to increase the PD parameters. However this does not add precision and in turn makes the walking style harder to tune since it responds too fast at the beginning but the real position never reaches the desired position. The solution for this problem is to add gravity compensation. The total weight of the torso and swing leg is countered by adding extra torques into each one of the joints in the chain link. The second virtual force used helps controlling the desired speed at which the character should walk. The center of mass velocity, V, of the biped is computed. Then using a PD controller an error with a defined desired velocity Vd is used to compute a force value as follows. Forces are computed in both sagittal and coronal planes. The lateral forces help the character following a

straight line in addition to the turning control of the stance leg.

$$F_v = K_v(V_d - V) \tag{5.1}$$

There is only a limited range of values in which the velocity tuning works. Having a larger value than this range makes the standing leg push the body forward too much causing it to lose balance. Having a lower value does not generate any difference in the walking velocity. It is important to say that this component is not crucial to the success of the controller. It should be considered as a component that helps tune a walking style once it has been designed since the foot placement strongly affects the desired speed.

Finally, the stance leg direction is added to allow the character to turn from its initial direction. The character can turn at a specific rate. This controller just adds a PD control to the hip joint of the stance leg. Adding this component also helps lateral balance control since after a rough push it brings back the character to point to the correct desired direction with respect the world frame. The direction of the character is measured at the pelvis.

5.3 Results

For the walking gait, several style variations on the height of the foot where tested. Another aspect is how the ankle touches the ground. In the walking controller it was hard to design a head-to-toe surface contact. Instead the simulation is more likely to bring the whole foot down to the ground at once. The turning behavior is very sensitive and fast turns are hard to achieve. However smooth transitions applied on the standing leg worked correctly. When moving from using only one rigid body in the torso to two rigid bodies as it is required for the rolling motions presents new challenges for the walking controller.

It is important to notice that the character is more tolerant to pushes in forward or backward directions than in sideways since it already has some momentum in those directions. On the contrary when pushed sideways it must compensate using the IPM while maintaining the desired speed. Some problems in the design are the initial few steps. A way of solving this is to use different parameters for these steps and then transition to the parameters that work better for a smooth constant

Parameter	Values
Average velocity	-0.5 to 1.6 m/s
Height use for swing leg	0.4 m at 0.225 s
Minimum transition time between states	0.3 s
Maximum transition time between states	0.45 s
Gravity compensation force	10 m/s
Velocity compensation parameters	Kp 100 and Kd 10
Turning rate	2 rad/s
Maximum push tolerated	500 N
Inverted pendulum distance limit	0.6 m
Minimum velocity margin	0.3 m/s forward and 0.2 lateral

Table 5.1: Walking controller parameters

velocity walking. When the character is pushed sideways it is possible that the swing leg passes through the stand leg. In both actions joint limits were added to have a natural range of motion on all joints. The limits remain constant for both walking and standing.

Figure 5.2 shows a simulation of how the walking action is executed while receiving pushes (red arrow) within a reasonalbe range not high enough to make the character lose balance. Table 5.1 shows the parameters and resulting values for the walking control.

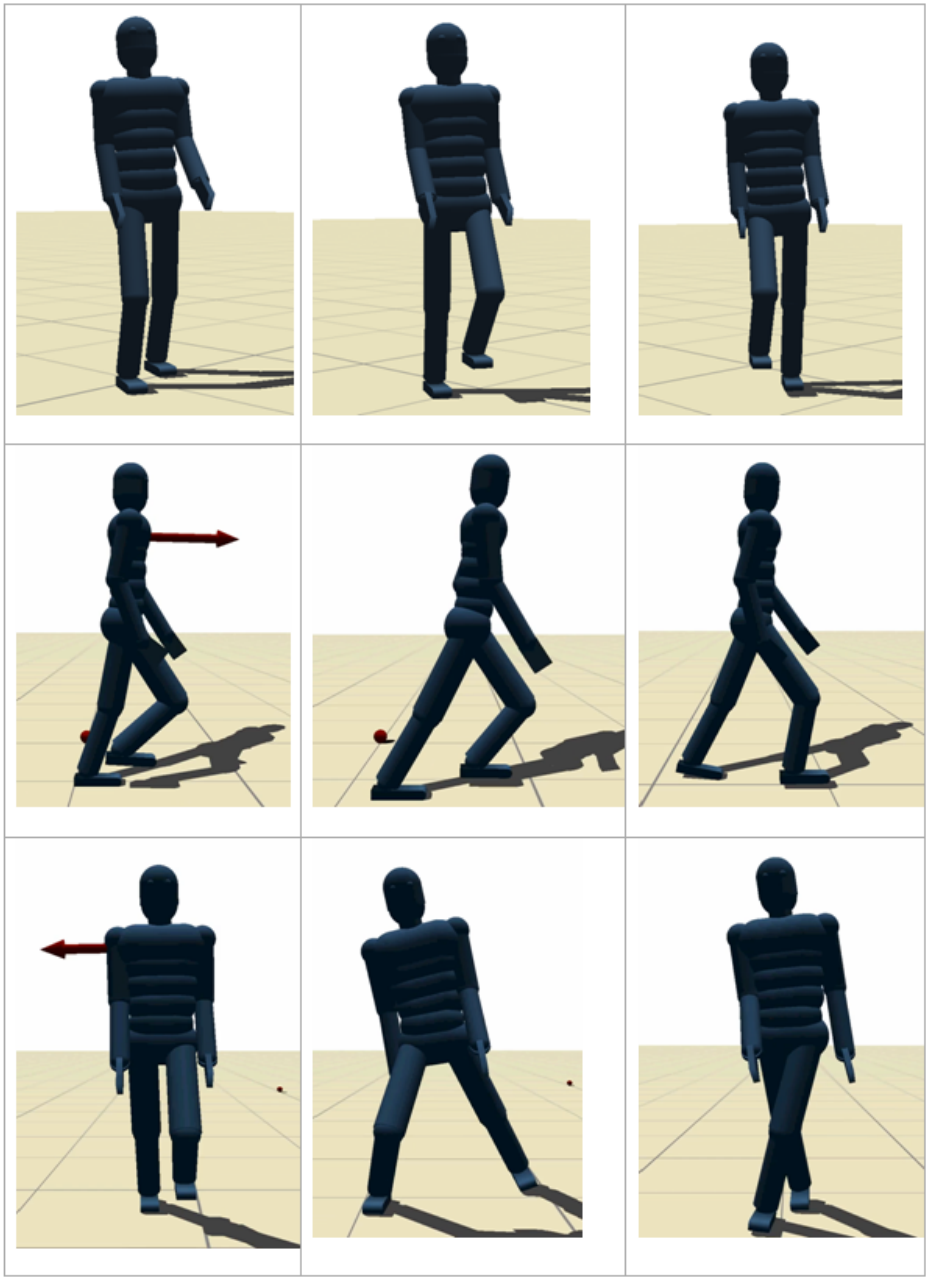

Figure 5.2: Simulation of walking action

Chapter 6

Rolling

The motion of rolling in the floor is interesting to study because it is a full body motion as shown in Figure 6.1. Hands, arms, legs and abdomen all play a part in achieving a roll. The contact with the floor makes it harder to plan transitions between states as it happens with walking motions. A challenge is that it is not clear how joints like the shoulders and hips should rotate. It is hard to manually design desired axis and angles of rotation for this joints. Finally, the momentum gained after one successfull roll, for example supine to prone, should be used in the consecutive prone to supine roll in order to achieve a repeated motion.

6.1 CMA Optimization

A stochastic global optimization technique is used in this work. Covariance Matrix Adaptation (CMA) [27] to optimize a control policy for the rolling motion. This method is effective in circumventing local minima. CMA is an evolutionary strategy. It starts with a uniform Gaussian distribution and gradually directs the search to approach the global minimum [11]. On each iteration, a number of candidates are sampled from a previously fitted Gaussian distribution over the space of parameter values. These samples are evaluated and a small number of best candidates are picked to fit a new Gaussian distribution. The process is performed iteratively until the optimization converges to the expected cost or the maximal number of evaluations is reached as shown in Figure 6.2.

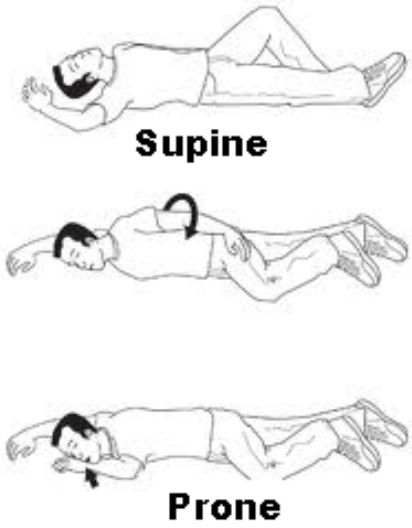

Figure 6.1: Supine to prone rolling of a person. Adapted from Carleton[2]

6.2 Rolling controller

To design the rolling controller the problem was decomposed in two phases: rolling from a start position of the torso in a supine position and rolling from a start position of the torso in a prone position. This facilitates the definition of appropriate objective functions. Different parts of the body are used to achieve each one because some joints like the knees and elbows can be bent only in one direction. The total amount of torque must be minimum to avoid unnatural motions. To design the motions the following steps were followed.

1. Choose what joints are required. Since the rolling can occur in two directions one side of the character is chosen. Only the joints of that side are used in the optimization process. Using fewer joints helps reducing the dimensionality on the search. Joints with two or three degrees or freedom need to explore not only the angle for the quaternion but the three components of the axis using a total of four variables per joint instead of one. The parameters start using initial values.

$$P = P_{initial} \tag{6.1}$$

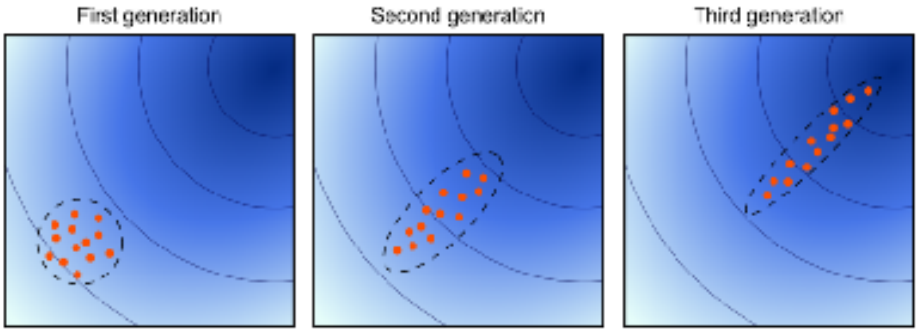

Figure 6.2: Three iterations of CMA optimization process. Adapted from Wikipedia [27].

2. Define the evaluation functions. This is the most important part to get successful optimization results. The evaluation function can be evaluated multiple times during the execution or the action or may be applied just once at the end. The evaluation functions measure the height of the head and hands, the angle between the torso front vector and the world up vector and the total torque used. The constants Ka, Kb and Kc are used to bring the different unit methods into the same scale.

$$E_{height} = hand_y + head_y \tag{6.2}$$

$$E_{torso} = angle(torso_{\vec{x}}, \vec{y}) \tag{6.3}$$

$$E_{torque} = \Sigma torque_t \tag{6.4}$$

$$E_v = Ka * E_{height} + Kb * E_{torso} + Kc * E_{torque} \tag{6.5}$$

3. Run the optimization. The optimization must select parameters, run the animation during a period of time. This is normally done faster than in real time since it is not require to render the animation during this process. Finally, the

results for each iteration are evaluated.

$$E_v = Simulation(P) \tag{6.6}$$

4. Evaluate resulting parameters. After several iterations the best set of parameters are taken into the script to be used to execute the action. This parameters work correctly only when the initial pose of the character is close enough to the one used on each iteration during the optimization process.

$$P = Generate_{CMA}(P_{last}) \tag{6.7}$$

For the supine to prone rolling optimization uses the right arm and leg to get an impulse on the floor and then using the pelvis to twist and the arm and leg to bring the center of mass forward. After analyzing this motion it is very hard to have defined it manually since the coordination of the multiple joints and the timming of the motion is not trivial. The result is a very natural motion similar to what kids do when they are learning to roll.

A second experiment consisted on adding a constraint to the height of the pelvis and the height of the hand in order to improve and make the motion look more natural and avoid the character from using the head as a pivot for rotation and instead use the shoulder. The results are shown in Figure 6.4 For prone to supine rolling the character moves his left arm close to his torso and gives an impulse with the hand while the left leg brings some of the balance to the opposite side. In this case the coordination between the shoulder and the elbow of the left arm is essential and the right arm is slightly use to bring it closer to the body while turning. The results are shown in Figure 6.5

6.3 Rolling repeatedly

For the repeated rolling it was enough to connect both actions one after the other. Tests where performed to include several rollings and evaluate the direction of the torso in intermediate times in order to make the rolling more agile. To achieve this an optimization process was run to find the best values to connect both precomputed trajectories. Instead of having two trajectories of four knots each the

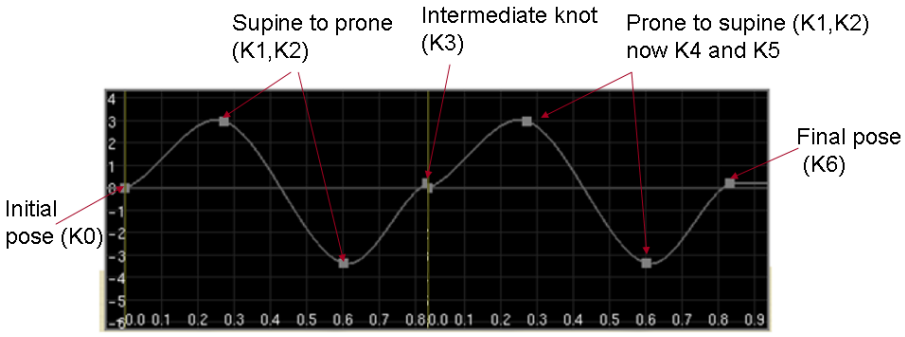

Figure 6.3: Connection of individual rolling to obtain the repeated roll trajectory.

state has one trajectory of seven knots where the initial and the last one correspond to the initial and final poses, the second and third are knots K1 and K2 for the rolling from supine to prone, fifth and sixth are the knots K1 and K2 of the prone to supine rolling as shown in Figure 6.3. The fourth knot value is optimized with the resulting values shown in the following table. The evaluation function used in this process ensures that the direction of the torso periodically shifts from pointing side up to side down every two seconds.

6.4 Forward roll

The forward rolling is a motion designed using a similar approach to the lateral rolling. The objective function consists on a combination of the torso orientation and the height of body parts that are key to the resulting motion. Starting on a standing position the character needs to bend down and crouch into a ball position with some forward momentum to finish laying on the floor in supine position. For the optimized trajectories only the initial pose plus two intermediates knots each one separated by one second of difference were used without returning the character to its initial position. Knots two and three are pluged into an offline CMA optimization process. The objective function has a total energy $E_v$that evaluates that at the end of the trajectory the up vector initially aligned with the world up vector is aligned with the world down vector E_{torso}. The height of the foot, head

and torso minus the height of the pelvis are added E_{height}. The last factor is the total torque used E_{torque}.

$$E_v = \Sigma E_{torso} + E_{height} + E_{torque} \tag{6.8}$$

Using the symmetry of the motion allows reducing the number of variables to a half by copying the values from the right side of the body into the left side. Once a solution is found the final knot K3 is added a second after the K2 to bring the body back to a rest pose. The total motion takes three seconds. The initial values are set intuitively to bend the knees, push the body forward with the ankles, and bend down the torso. It is important to notice that the joint limits play an important part in this motion. Using very constrained limits makes it harder for the optimizer to find a suitable solution though using them to relaxed produces unnatural motions. The results are shown in Figure 6.6

6.5 Results

The optimization method used to compute the lateral rolling is an off line method. In order to compute the resulting motions at least 1000 iterations were computed before converging to a suitable solution. The resulting parameters are plugged in into the trajectory knots and the resulting motion is replicated using the same initial position. The computation of the solution takes from 10 to 20 minutes overall using around 3 seconds for each evaluation using a single thread.

The joint limits used are the same of the one used for walking and standing and allow the character to use a natural range of motion for all the joints. In order to force the rolling to happen in a desired sense (clockwise or counter clockwise) only half the character's body joints on the outer side were controlled.

The following tables show the parameters obtained after the offline optimization process is run. Since each joint has four knots, the two intermediates knots (K1 and K2) are the main variables and the initial and final knots (K0 and K3) describing the initial and final pose remain constant for supine to prone, prone to supine and stand to supine. For repeated rolling the trajecotry is described in Figure 6.3 and the table shows the values for the intermediate knots.

Figure 6.4, Figure 6.5 and Figure 6.6 show a sequence of images for the rollings

Joint	Initials	Values
Abdominal K1	1.57	2.379724
Abdominal K2	-1.57	-2.09765
HipRight K1	1.57	1.338274
HipRight K2	-1.57	-1.3329
KneeRight K1	1.57	0.649201
KneeRight K2	-1.57	-1.9048
shoulderRight K1	1.57	0.906889
shoulderRight K2	-1.57	-4.1882
HipRight X	0	0.964817
HipRight Y	0	-1.21717
HipRight Z	1	0.615165
shoulderRight X	0	1.185683
shoulderRight Y	0	0.934679
shoulderRight Z	1	1.285935

Table 6.1: Supine to prone rolling using height of pelvis and of hands

Joint	Initials	Values
Abdominal K1	1.57	1.112171
Abdominal K2	-1.57	-0.67442
HipLeft K1	1.57	0.380127
HipLeft K2	-1.57	-1.51984
ShoulderLeft K1	-1.57	-2.02989
ShoulderLeft K2	1.57	1.344907
ElbowLeft K1	1.57	1.192023
ElbowLeft K2	1.57	3.38349
ShoulderLeft X	0	-1.64444
ShoulderLeft Y	0	-1.11174
ShoulderLeft Z	1	-0.09597

Table 6.2: Prone to supine rolling using pelvis height

motion: supine to prone, prone to supine and forward rolling.

Joint	Initials	Values
Abdominal K1	-1.57	-0.1326
Abdominal K2	-1.57	-3.2339
Hip K1	2.1	4.0491
Hip K2	2.1	2.4862
shoulder K1	0	4.5812
shoulder K2	0	0.0632
Knee K1	-2.1	-1.37
Knee K2	-2.1	1.13
Ankle K1	1.57	1.5921
Ankle K2	1.57	2.9834

Table 6.3: Forward rolling trajectories results

Joint	Initials	Values
Abdominal K3	0	-1.0523
HipRight K3	0	0.3428
HipLeft K3	0	-0.9478
shoulderRight K3	0	1.0856
shoulderLeft K3	0	0.5031
kneeRight K3	0	-1.1704
elbowLeft K3	0	0.0082

Table 6.4: Intermediate values for rolling repeatedly

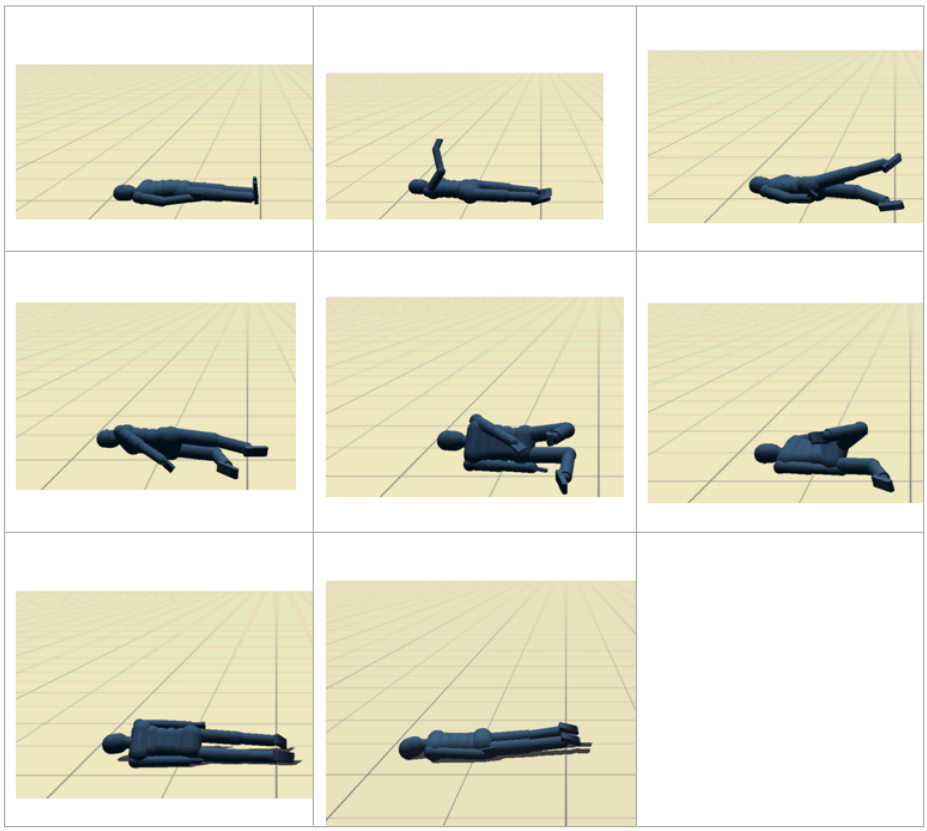

Figure 6.4: Simulation of rolling action supine to prone

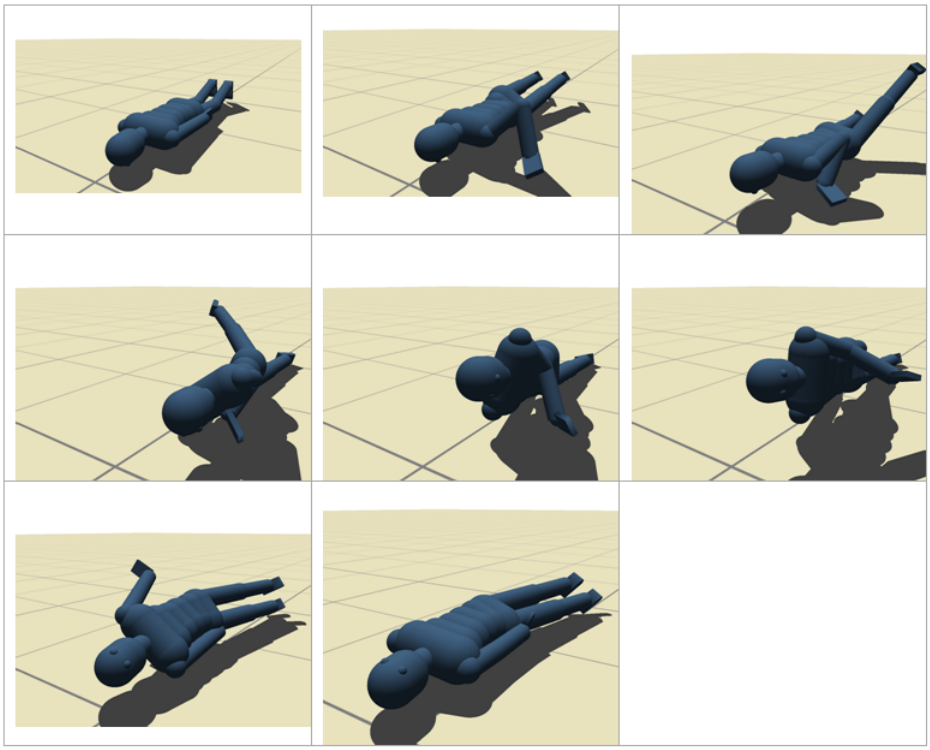

Figure 6.5: Simulation of rolling action prone to supine

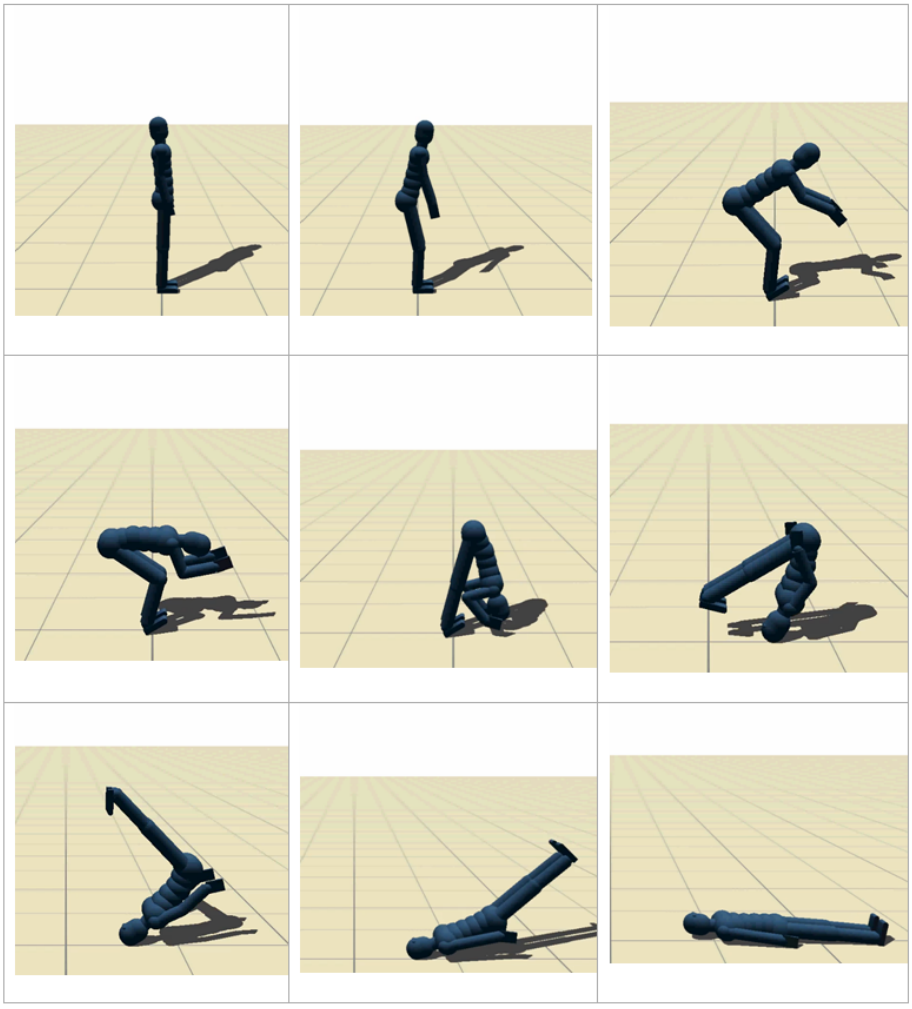

Figure 6.6: Simulation of forward roll stand to supine

Chapter 7

Conclusions

The control strategies developed in this thesis included the implementation of standing skills that works using the center of mass to compute virtual forces(COM) and pose tracking and is robust under variations on the terrain slopes and moderate pushes. Falling is developed using the character as a ragdoll without applying any torque to the joints. Walking was implemented using the Generalized biped solution implementation [5] including all the main components such as the inverted pendulum (IPM), inverse kinematics (IK), pose tracking and virtual forces (VF) for gravity compensation and velocity tuning. Variations on styles and balance control after receiving moderate pushes. Finally, a forward roll (stand to supine) and lateral repeated rollings (supine to prone and vice versa) were achieved using CMA optimization to compute the trajectories and axis of rotation to control the joints.

7.1 Discussion

The motions described in this work are split in two categories. Walking and standing were solved by designing controllers using quaternion orientation control, virtual forces and inverse kinematics. Rolling motions were produced using an offline optimization process. In both cases there is a high dimensional, non linear space to design the control policies given the unstable nature of humanoid characters.

One of the main challenges for rolling motions is that it can not be easily described as a chain of statically stable poses. To discover and perform these actions

requires an exploration of the space of motions and a well chosen evaluation function to state if the motion is correct or not. In three dimensions the number of variables increases because not only does the rotation angle need to be considered but also axis of rotation on some of the joints. A way of simplifying this problem was to use only half the body to do the rolling motions. Some of the rolling motions obtained through the optimization were physically plausible although not natural. An example is the use of a head to roll on the floor. This motion is present in breakdancing but it is not the traditional way of rolling. For this reason, objectives for the height of pelvis and hand were added.

Another issue to be considered is the use of a falling strategy to connect both walking and rolling motions. Since there are multiple ways of landing in the floor and the connection between these states and the initial pose required for the rolling is not always consistent. Finally, modeling the character using rigid bodies is a hard problem. Since all the positions and rotations are precomputed manually without using modeling software, fitting all the parts together and make them physically plausible to start testing motions requires a lot of time.

7.2 Future work

Besides walking and rolling, a general solution for standing up from a prone or supine position is a topic that has received some attention in robotics and would definitely fit connecting walking and rolling motions. Falling strategies are another area that can be explored to complete a walk-fall-roll-stand-walk sequence.

A significant limitation found during the design of the walking controller is to bring all the components together to produce a desired style of walking. It is still hard for the user to clearly specify some general attributes. Another component to be looked at is the generalization over initial pose used in the optimization process. Even when the solution is computed correctly and can be replicated several times afterwards, there is no garantee that the character will start at the exact same pose before executing an action. This is an important limitation in order to connect falling with a rolling action.

Another line of work is to generalize the forward rolling to be periodical. After having an initial roll from the standing position a second optimization process can

be used to push the character forward using the arms and legs having an evaluation function cyclically measuring the orientation of the torso and the height of key body parts similar to the repeated lateral roll.

Bibliography

[1] R. Boulic, R. Mas, and D. Thalmann. A robust approach for the control of the center of mass with inverse kinetics. *(Computer and graphics)*, 20: 693–701, 1996. → pages 11

[2] U. Carleton. recovery http://www1.carleton.ca/studentaffairs/alcohol-awareness/concerns-emergencies/recovery-position., 2012. → pages viii, 32

[3] Coros, Karpathy, Jones, Reveret, and van de Panne. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics (SIGGRAPH)*, 30(4), 2011. → pages 6, 8

[4] S. Coros, P. Beaudoin, and M. van de Panne. Robust task-based control policies for physics-based characters. *ACM Transactions on Graphics SIGGRAPH Asia*, 28(5):Article 170, 2009. → pages 6

[5] S. Coros, P. Beaudoin, and M. van de Panne. Generalized biped walking control. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):Article 130, 2010. → pages vii, viii, 6, 13, 14, 24, 25, 26, 42

[6] M. da Silva and J. Abe, Y. Popovic. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics (SIGGRAPH)*, 27(3), 2008. → pages 6

[7] M. de Lasa, I. Mordatch, and A. Hertzmann. Feature-based locomotion controllers. *ACM Transactions on Graphics (SIGGRAPH)*, 29(3):TBD, 2010. → pages 6

[8] A. Erdemir, S. Mclean, W. Herzog, and A. van den Bogert. Model-based estimation of muscle forces exerted during movements. *(Clinical Biomechanics)*, 22(2):131–154, 2007. → pages 6

[9] F. Faure, G. Debunne, M. Cani, and F. Multon. Dynamic analysis of human walking. *(Computer animation and simultaion)*, pages 53–65, 1997. → pages 6

[10] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. *(ACM Transaction on Graphics (SIGGRAPH))*, pages 63–70, 1995. → pages 8

[11] N. Hansen and S. . Kern. Evaluating the cma evlotion strategy on multimodal test functions. *Parallel Program Solving from Nature-PPSN VIII*, 3242:282–291, 2004. → pages 31

[12] J. F. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. *ACM Transactions on Graphics (SIGGRAPH)*, 27(5):155–162, 1996. → pages 6

[13] Y. Lee, S. Kim, and J. Lee. Data-driven biped control. *(ACM Transaction on Graphics (SIGGRAPH))*, 29(4):129, 2010. → pages 6

[14] L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu. Sampling-based contact-rich motion control. *ACM Transactions on Graphics (SIGGRAPH)*, 29(4):128, 2010. → pages vii, 7

[15] A. Macchietto, V. Zordan, and C. Shelton. Momentum control for balance. *(ACM Transaction on Graphics (SIGGRAPH))*, 28(3), 2009. → pages 6

[16] M. Neff and E. Fiume. Modeling tension and relaxation for computer animation. *(ACM Symposium on Computer animation (SIGGRAPH/Eurographics))*, pages 81–88, 2002. → pages 6

[17] J. E. Pratt and R. Tedrake. Velocity based stability margins for fast bipedal walking. *In Fast Motions in Biomechanics and Robots*, SPRINGER(TBD): 299–324, 2006. → pages 14

[18] M. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *ACM Transactions on Graphics (SIGGRAPH)*, 25(4):349–358, 1991. → pages 6, 8

[19] H. Sehoon, Y. Yuting, and K. Liu. Falling and landing motion control for character animation. *(SIGGRAPH ASIA)*, 2012. → pages 6

[20] D. Sharon and M. van de Panne. Synthesis of controllers for stylized planar bipedal walking. *In Proc. IEEE Int'l Conf. on Robotics and Automation*, 2005. → pages 6

[21] R. Smith. Open dynamics engine user guide v0.5, 2006. → pages vii, 11

[22] J. Tan, Y. Gu, G. Turk, and C. Liu. Articulated swimming creatures. *(ACM Transaction on Graphics (SIGGRAPH))*, 30(4):58, 2011. → pages 8

[23] W. Tsai, Y.and Lin, J. Cheng, K. B.and Lee, and T. Lee. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE Trans. on Visualization and Computer Graphics*, 26(2): 325–337, 2010. → pages 6

[24] K. Wampler and Z. Popovic. Optimal gate and form for animal locomotion. *ACM Transactions on Graphics (SIGGRAPH)*, 28(3):60, 2009. → pages 5

[25] Wang, Fleet, and Hertzmann. Optimizing walking controllers. *ACM Transactions on Graphics SIGGRAPH Asia*, 28(5):Article 168, 2009. → pages 6

[26] Wen, Chieh, Li, Yi, and J. Huang. Animating rising up from various lying postures and environments. *Computer Graphics International*, 28(4): 413–424, 2012. → pages 6

[27] Wikipedia. Cma-es http://en.wikipedia.org/wiki/cma-es., 2012. → pages viii, 31, 33

[28] A. Witkin and M. Kass. Spacetime constraints. *ACM Transactions on Graphics (SIGGRAPH)*, 22(4):159–168, 1988. → pages vii, 5

[29] K. Yamane and Y. Nakamura. Natural motion animation through constraining and deconstraining at will. *(IEEE Transactions on Visualization and Computer graphics)*, 9:352–360, 2003. → pages 11

[30] K. Yin, M. Cline, and D. Pai, B. Motion perturbation based on simple neuromotor models. *(Proceedings of the pacific Conf. on Computer Graphics and Applications)*, Canmore, Alberta Canada:445–449, 2003. → pages 6

[31] K. Yin, K. Loken, and M. van de Panne. Simbicon: simple biped locomotion control. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):105, 2007. → pages vii, 6, 7