SKILL LEARNING BASED CATCHING MOTION CONTROL

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Gökçen Çimen

July, 2014

ABSTRACT

SKILL LEARNING BASED CATCHING MOTION CONTROL

Gökçen Çimen
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Tolga Kurtuluş Çapın
July, 2014

In real world, it is crucial to learn biomechanical strategies that prepare the body in kinematics and kinetics terms during the interception tasks, such as kicking, throwing and catching. Based on this, we presents a real-time physics-based approach that generate natural and physically plausible motions for a highly complex task- ball catching. We showed that ball catching behavior as many other complex tasks, can be achieved with the proper combination of rather simple motor skills, such as standing, walking, reaching. Since learned biomechanical strategies can increase the conscious in motor control, we concerned several issues that needs to be planned. Among them, we intensively focus on the concept of timing. The character learns some policies to know how and when to react by using reinforcement learning in order to use time accurately. We demonstrate the effectiveness of our method by presenting some of the catching animation results executed in different catching strategies.In each simulation, the balls were projected randomly, but within a interval of limits, in order to obtain different arrival flight time and height conditions.

Keywords: Ball Catching Simulation, Physics-Based Character Animation, Reinforcement Learning.

ÖZET

BECERİ ÖĞRENME BAZLI YAKALAMA HAREKET KONTROLÜ

Gökçen Çimen
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Tolga Kurtuluş Çapın
Temmuz, 2014

Gerçek hayatta, tekmeleme, atış ve yakalama gibi çarpışma gerektiren activiteler sırasında vücudu kinematik ve kinetik açıdan hazırlayan biyomekanik stratejiler öğrenmek oldukça önemlidir. Buna dayanarak, son derece karmaşık bir görev olan top yakalama görevi için doğal ve fiziksel açıdan makul hareketler oluşturabilen gerçek zamanlı ve fizik tabanlı bir yaklaşım sunuyoruz. Biz diğer birçok karmaşık görevler gibi yakalama activitesinin, ayakta durma, yürüme, uzanmak gibi oldukça basit motor becerilerinin doğru bir şekilde birleşimi ile elde edilebilir olduğunu gösterdik. Öğrenilen biyomekanik stratejilerin motor kontrolünde bilinçli davranışları artırabiliceğinden dolayı, planlama gerektirebilen çeşitli konular ile ilgilendik. Bunların arasında, zamanlama kavramının yoğun orarak üzerinde duruyoruz. Karakter zamanı doğru bir şekilde kullanabilmek için pekiştirmeli Öğrenme tekniği ile nasıl ve ne zaman davranabiliceğini gösteren bazı politikalar öğrenir. Farklı yakalama stratejileri ile gerçekleştirilen bazı animasyon sonuçları sunarak yöntemimizin etkinliğini gösterilmektedir. Her bir simülasyon sırasında, toplar farklı uçuş süreleri ve yükseklik kouşullarını göz önünde bulundurmak için rasgele bir biçimde, ancak belirli limitler aralığında, fırlatıldı.

Anahtar sözcükler: Top Yakalama Simulasyonu, Fizik Tabanlı Karakter Animasyonu, Pekiştirmeli Öğrenme.

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor, Asst. Prof. Tolga Kurtuluş Çapın, for his guidance in supervision of this thesis. I would like to thank especially for his support and kindness during my M.S. study.

I would also like to thank to my jury members Prof. Dr. Uğur Güdükbay and Prof. Dr. Haşmet Gürcay for accepting to spend their valuable time for evaluating my thesis. I am thankful to their valuable comments and suggestions.

I am not sure there are enough words of thanks to express my gratitude to my family for their continuous love and endless support. They devoted everything they can to raise me in a wonderful environment. The achievements in my education would be a dream without them.

I just want to give big thanks to my friends Seher Acer, Elif Eser, Gizem Mısırlı and Bengü Kevinç for being there for me through some of the most difficult times. I want to thank them for being my second family. Your kind hugs, smiles, jokes, and memories will always remain in my memory.

I am grateful to my project friends Hacer lhan, Zümra Kavafoğlu and Ersan Kavafoğlu who share their time to motivate me during my research and thesis. I also would like to thank Zeynep Korkmaz, Gülden Olgun, Sinan Arıyrek, Can Telkenaroğlu and Seyhan Uçar for their helps and sincere supports.

Finally, I would like to the Scientific and Technical Research Council of Turkey (TUBITAK, project number 112E105) for the financial support.

Thanks a lot everyone.

to my family,

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For almost all daily activities, such as performing of standing, walking, holding objects, the human body both resists against forces and generates forces. In general, these forces can be categorized as internal forces and external forces. While external forces are resulted from the interaction of the human body with the environment, internal forces are produced by the musculoskeletal system. In a predictable way, the internal forces enable us to move body parts, but external forces are needed in order to move the center of mass of the human body. For example, the walking movement is performed using the internal forces in order to manipulate the external forces, such as, gravity, friction and contact forces. Therefore, the difficulty of generating physics-based character animations comes from the fact that we need to control properly the internal forces with the help of the external forces in order to interact with the virtual environment in a physically accurate way.

Even though the past few decades, promising results are achieved in physics-based character animation, including the simulation of most ordinary behaviors, such as standing, walking and running, physically simulating the highly dynamic motions, such as throwing, catching or striking are still one of the great challenges. The major reason is that these behaviors require predictions about the target, necessary velocity and time in order to respond to the impact correctly. The common features of these behaviors are that the movements of the character

should be coordinated accurately over time and handle safely the shock at the impact time to minimize the possible injury risk. These controls increase the believability and the quality of the performance.

1.1 Motivation

The key point that should be considered in such studies is to get the benefit of various principles based on physics and bio-mechanics while conserving the nature of human characteristics. Because of the intuitive non-determinism and complexity of the human motion, data-driven approaches are popular since they allow us to reach easily the low-level features of the motion. In that way, motion sequences are readily generated from pre-recorded motion databases. On the other hand, collecting these large databases that cover all variations of a human behavior is rather challenging and most of the time finding a specific motion that compensates the required constraints is very difficult. Supporting interactions with environment like reaching to a pre-defined target location in accurate time with end-effectors adds additional difficulties in the context of both data-driven approaches and physics-based approaches. These outcomes create a compelling reason for exploiting motor control strategies based on learning motor primitives for complex behaviors, such as, catching a flying ball which is the main component of this thesis.

For physically simulating the interaction with the environment of a character, motor control models can allow us to create realistic human animations. In dynamic environments where the character should adjust its movements to the changing environment, it is required to alter the motor control models to adapt itself according to the varying target within the required time and velocity. In sports science literature, behaviors such as striking, catching show similar phases [1]. This indicates that it is possible to define motor control models related to these phases and learn how to control them in such a way that it includes different interception points and arbitrary time and velocities. In this way, the character learns how to plan its behavior by knowing which motor control model should be

triggered or modified without changing the overall structure of the motion.

1.2 Approach Overview

It is almost impossible to develop a single model that controls the whole body for a specific task while creating robust interaction with the environment in physics-based animation. Therefore, one intuitive approach would be dividing one control strategy into partly task-oriented motor control models in such a way that combining these models, which are responsible for controlling rather simple tasks, can generate the required behavior. Standing, walking to a specific location, moving your hands to a predefined target are just a few examples for these control models. For any complex behaviors that require transporting your body while trying to reach to a target with your hand, such as ball catching, can be achieved by properly managing these simple control models.

With the light of these findings and additional inspirations from interdisciplinary areas, such as robotics and bio-mechanics, we present a new method that allow us to generate natural and physically plausible human catching motions in real time using physical simulation. The method does not require any motion capture data or pre-scripted motion sequence, and the approach can be used for other behaviors, which require interaction with environment where adaptation in timing and positioning needs to be planned.

We can summarize the three important steps for a successful catching process: (1) It requires developing a strategy that determines final pose in order to absorb safely the shock at impact while maintaining the balance of the character. (2) There is a need for a proper system for controlling the intermediate poses while trying to reach the desired pose targets for the character defined in the first step. This also ensures the character is driven to the pre-defined impact pose in accurate time. Lastly (3) The character should be able to learn using and managing the motor control skills, such as reaching, standing, stepping, in different circumstances and other specifications. The approach presented in this

thesis uses reinforcement learning in order to learn when and where to use motor control skills. During modeling these steps, our method is inspired from catching principles that are structured in recent biomechanics researches.

In this section, the overall system framework and the interaction between its components are presented. Figure 1.1 shows the general framework of the system in a brief diagram. There are three important components- high-level controllers, policy-based phase planning, final strategy planing.

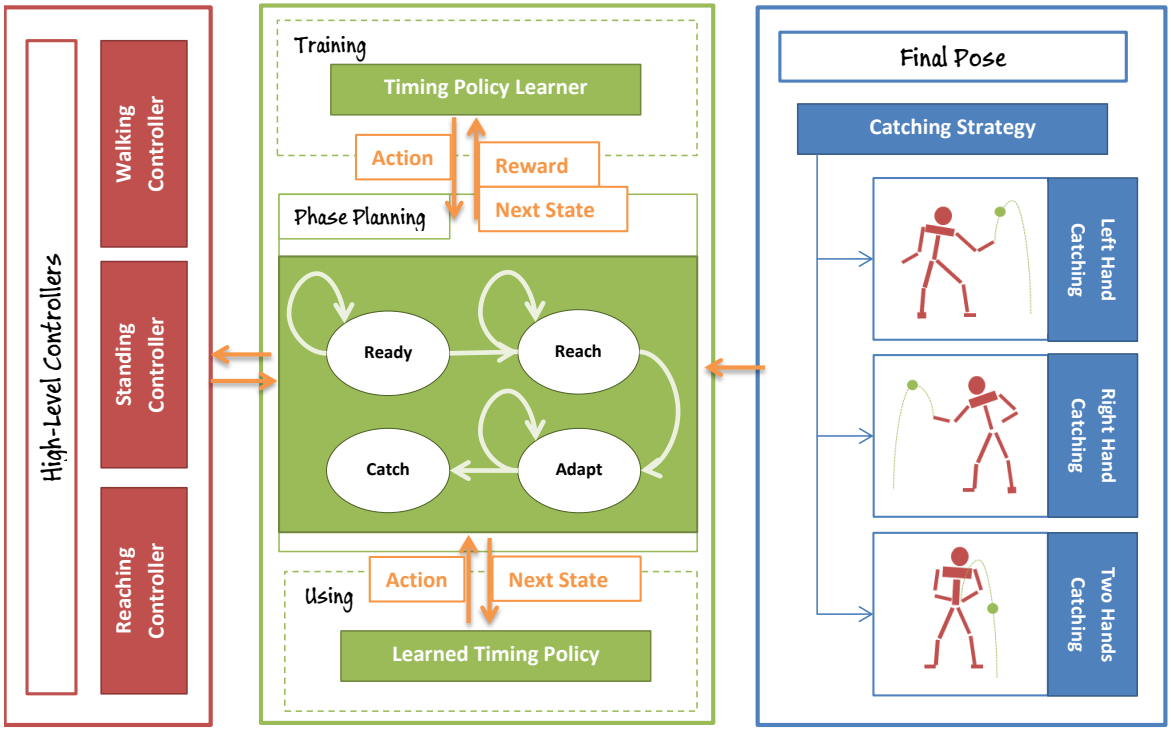

Figure 1.1: General system framework.

During a catching task, the first step of the system after the ball is thrown is to decide a catching strategy with an interception hand position and an interception com position according to the trajectory of the ball. Then, this final catching strategy including interception positions are send to the phase planner which is responsible to defining high-level inter poses of the character that drives it to this interception position with given strategy. Based on this, the phase planner

uses pre-trained policy in order to select the best action which decides which phase the character should be in according to the current state to be able to catch the ball right on time. The pre-trained policy is obtained after a number of ball catching sessions by trial and error. The movements of the character during the catching task are achieved with high-level controllers, such as walking and reaching. According to the current catching phase decided by phase planner, a combination of the high-level controllers are used to bring the character to the corresponding pose described in each catching phase.

During the process of learning the timing policy, balls were projected randomly, but within a interval of limits, in order to obtain different arrival flight time and height conditions.

1.3 Thesis Structure

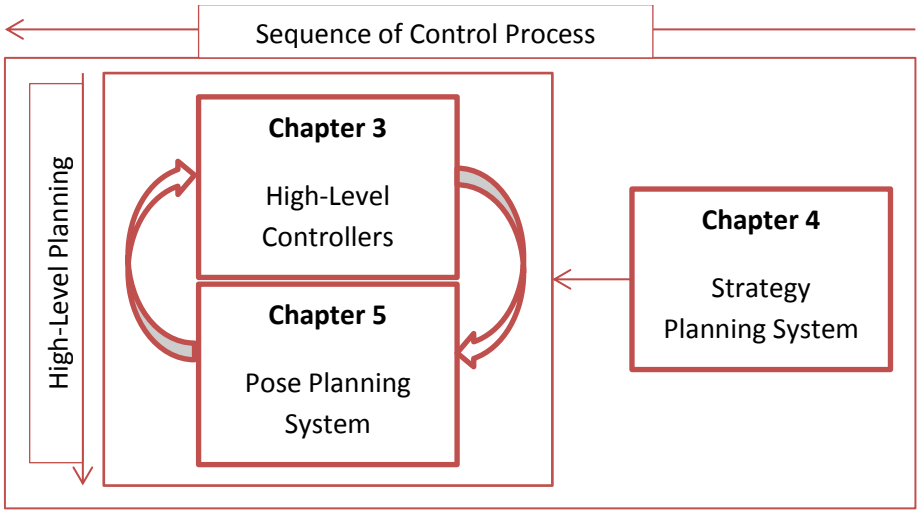

Figure 1.2: The overview of the thesis. The Strategy Planning Mechanism works almost independent from High-Level Controllers and the Pose Planning System. On the other hand, High-Level Controllers and the Pose Planning System are loosely related to each other such that there is a continuous interaction between them.

In this thesis, we introduce a real-time physics-based technique to simulate

strategic catching behavior, which combines three different mechanisms: (1) A human-like strategy planning system decides a target final pose, which defines the goal for the other controllers and planners to reach at the same time. (2) High-level motor controllers which are dedicated to simple behaviors, such as walking, standing, reaching, etc., defines the repertoire of motor skills for the character. (3) A policy-based pose planning system controls the inner poses that drive the character to the goal within the specified constraints by managing these controllers. The structure of the thesis examines these three distinct mechanisms in separate chapters. Figure 1.2 provides an overview of the structure of the thesis.

Rest of the thesis is organized as follows. In Chapter II, we begin with information about some related work in the fields of data-driven, physics-based and reinforcement learning based approaches. In Chapter III, the simple and robust High-Level Motor Controllers are presented in detail, as the starting point of the work. In Chapter IV, the process of defining specifications of final pose for ball-catching problem is presented. In Chapter V, the inner pose planning mechanism and its integration with the Reinforcement Learning (RL) is introduced. In Chapter VI, the simulation results of our approach are demonstrated and the findings based on our approach are examined. Finally, we the thesis is concluded in Chapter VI with discussion and suggestions for the future work.

Chapter 2

Related Work

This chapter presents an extensive summary of earlier related works that use control based character animation techniques, which is relevant to our work. There are several other methods in the field of control based character animation. We present the most common and relevant techniques- data-driven approaches, physics-based approaches and reinforcement learning-based approaches.

We first begin with a review of recent data-driven approaches, which are based on using existing joint trajectories obtained from motion capture technique to generate character animation. We then focus on fundamentally different approaches in the literature, physics-based approaches, which create all motions as a result of a physics simulation process instead of directly manipulating the trajectories. Finally, we present the most relevant reinforcement learning-based approaches which are based on character controllers that use optimized control policies.

2.1 Data-Driven Approaches

As a result of the increasing popularity of the motion capture solutions, data-driven approaches have become more relevant in the recent years. The fundamental idea behind the data-driven based motion generation is that new motions can

be created by simply editing, concatenating and interpolating the pre-recorded clips of motion.

The most common method which is based on directly using motion clips is motion graphs (e.g. [2], [3], [4], [5], [6]). A motion graph structure is constructed of the nodes, the motion clips, and the edges, the transitions between the motion clips. Arbitrarily traversing the motion graphs gives a new continuous motion different from the original motions in the graph. Although motion graphs provide an efficient way of reusing motion clips, they have several limitations. The most important drawback of the motion graph technique is that the generation of a motion sequence that accomplishes a desired task using the available combination of motions may never be achieved. In addition, motion graphs do not allow significant diversity among the generated motions.

Notably, since the graph-based representation, such as motion graphs, cannot generalize motion data, many researchers started to explore the data-driven approaches which benefit statistical models The main reason is that statistical models are based on analysis of kinematic data, and synthesis of a new motion that is not in prerecorded motion database is not applicable. In general, these methods construct different low-dimensional spaces on the motion capture data and synthesize motions using optimization. (e.g. [7], [8], [9]). The most common methods used for dimension reduction are PCA [10] and GPLVM ([11], [12]).

Among the statistical data-driven approaches, several works investigated creating animations of character performing various tasks that include the interaction with objects using biomechanical rules. For example, the approach in [13] uses motor control mechanism in order to produce specific reaching task. In another work, [14] also exploit various biomechanical strategies in order to generate life-like reaching motion, and [15] presents a technique that relies on motion capture for data manipulation tasks, such as locomotion, reaching and grasping in real time.

Unfortunately, the approaches based on motion capture data have several important weaknesses. (1) The ability to produce good-quality motions strictly dependent on the size of the mocap data set. (2) The motions synthesized are

limited since collecting a database that comprises a full range of human motions, including the ones that require interaction with environment, is almost impossible. (3) In addition, It is well accepted that utilizing the knowledge of human motor control is very important for realistic motions, such as balance control.

2.2 Physics-Based Approaches

On the other side, physics-based approaches emerged as a promising strategy for creating more realistic motion as it occurs for humans in a real world since the motions are produced with intelligently managing the internal joint torques and forces. The main advantage of physics-based approaches is that they allow producing a wide range of diverse motions, including the interaction with objects by using no additional, or minimal amount of motion data.

The key challenge in physics-based character animation is to model motions and develop control solutions in order to make the character behave and respond realistically to the unforeseen circumstances in virtual environments. Despite this, it has been investigated across numerous tasks, such as balancing, standing, walking, etc., but no doubt the great amount of work has been released in walking and dynamic balance strategies (e.g. [16], [17], [18], [19]).

The important components used in this thesis share common features of the method used in SIMBICON [20] and the more recent work of Coros et al. [21]. SIMBICON is a robust and simple locomotion controller that employs feedback strategy for keeping balance under control while walking even in unforeseen disturbances. We employ a simplified inverted pendulum strategy similar to the strategy used in Coros et al. [21] to guarantee balancing by predicting foot placement in our walking controller. There is other components of our controllers in our study that is inspired from this work, such as canceling the gravity effect over body parts using Jacobian Transpose method.

Some researchers also investigated the simulation of reactive and responsive motions integrated with walking or standing behaviors with or without the help

of biomechanic strategies ([22], [23], [24], [25]). Furthermore, there other studies that control the flight-based rotational behaviors, such as falling and rolling ([26], [27]). Among them, our study is similar to the work of Ha et al. [26] in such a way that it divides the falling process into meaningful biomechanical phases and tries to control and plan the motions of the character to prepare it for the impact.

Since generating animation of physics-based characters is a complex process, it requires consideration of many different disciplines, such as biomechanics and optimization theory. Based on this, there are a number of physics-based studies supplemented with motion capture data in order to increase the stylistic characteristics of the motions. For example, trajectory tracking is one of the common techniques used in this context. To maintain balance while walking, some studies presented a strategy of driving upper body by tracking reference motion capture data while using some conventional balance control for lower-body (e.g. [28], [19]). Among them, the work in [29] uses a method that modifies continuously the reference motion capture data before tracking in order to maintain balance.

Even though the physics-based strategies mentioned here are successful for controlling the characters to react realistically. The purposeful characters that involve interactions with their environments to achieve a predefined task, however, may require more than react. They can benefit from a mechanism for anticipating and accordingly planning their movements. For example, the character should be able to learn under which circumstances transitioning to a different state in order to achieve the goal on time. Based on this, the work, presented in this thesis, attempt to integrate such a learning system in order to plan low-level control strategies using reinforcement learning.

2.3 Learning-based Approaches

In the previous section, it is mentioned that a number of control strategies that can perform various physically simulated motions, including walking, running, falling and standing is studied. Many of them leave the decision regarding when

the transition happens between different controllers to the user. Based on this, reinforcement learning is a promising approach that enables to formulate higher-level goals and develops control policies in order to plan behaviors that achieve the goals. There is no need to define manually the transitions for different conditions, instead, the character learns the transitions by itself.

There are a number of studies proposed to utilize reinforcement learning to learn policies for choosing best actions based on data-driven models ([30], [31], [32]) for interactive avatar control [33] to perform locomotion [34]. They are based on specifying best motion clips to continue from the current motion clip regarding to the task. Unfortunately, many of them are limited such that many of them do not allow physical interaction with the environment.

Reinforcement learning has also been studied in the field of robotics, mainly for controlling walking ([35], [36]). They are based on learning motor primitives for acquiring new behaviors for the robots by reinforcement learning. In robotics, there are also many works studied that learn motor primitives for more complex behaviors, such as playing table tennis ([37], [38], [39]).

Some approaches that have been proposed presents systems that control the switches between the physics-based controllers to carry out locomotion ([34] by optimizing control strategies ([40], [41]). Among them, the method in [42] presents a muscle-based control for locomotion of bipedal creatures. Coros et al. [43] developed a method that achieves robust gaits for quadrupeds. The approach presented in [44] creates a realistic swimming behavior for an articulated creature with a learning process through offline optimization for appropriate maneuvers. Recently, Tan et al. [45] presents an approach for controlling a character while riding a bicycle. It has two main components similar to our work- offline learning and online simulation. In the paper, the rider is controlled by an optimal policy which is learned through an offline learning process.

Many of these methods search for mapping between the state space and the action space by learning optimal control policy for the optimization through the low-level controllers using reinforcement learning. Our approach complements

especially to the works which investigate methods that capable of making transitions, since we focus on learning to plan between low-level controllers.

Chapter 3

High-Level Controllers

3.1 Introduction

Physics-based character animation offers movements that are physically accurate which obey the laws of the physics (angular momentum) and as a result appear realistic. Similar to the real world, the physics-based character is controlled via forces and torques. This creates a challenge in achieving simple human behaviors, such as balance, walking.

The fundamental components for the physics-based character animation are a physics simulator or physics engine, a physics-based character and a motion controller. A physics simulator predicts the kinematics of objects in the virtual environment and controls them through external forces and torques for every step of the simulation. Physical characters are most of the times simplified into low dimensional models which have small number of degrees-of-freedom (DOF) for performance issues and simplicity. A physics-based character are composed of a number of body segments which are usually modeled as primitive shapes, such as boxes or cylinders, and joints which connect two bodies. Figure 3.2(a) shows the simplified physics based character that used in our framework. It is constructed from 15 rigid bodies and 14 joints. Motion Controller forms the core of the concept of the physics-based control. It is responsible of generating the

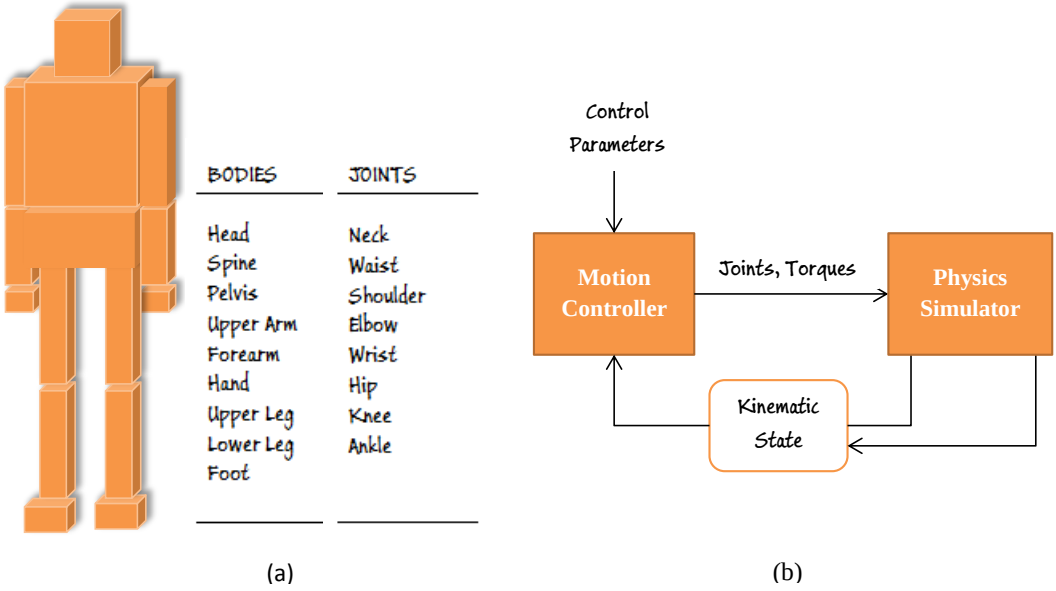

Figure 3.1: (a) Simplified physics based character. (b) Motion Controller and Physics Simulator.

necessary torques and forces to drive the character to perform predefined tasks, such as, walking, running, balancing etc. Figure 3.2(b) shows the flow between Motion controller and Physics simulator.

The Bullet Physics Engine is used as the physics simulator which is an open-source physical simulation library for collision detection and rigid-body dynamics. There are other popular other libraries, such as Open Dynamics Engine (ODE), PhysX, Newton and Havok. The reason of choosing Bullet Physics Engine is that it is very well supported by the community recently.

This chapter presents high-level controller responsible of controlling specific task, such as walking to target location in the environment. The high-level controllers developed for this work are Walking Controller, Standing Controller and Reaching Controller. Each of these controllers are composed of a number of low-level controllers. Therefore the chapter begins by explaining the fundamental control concepts behind these low-level controllers. Then it presents how these concepts are integrated to low-level and high-level controllers in detail.

3.1.1 Proportional Derivative (PD) Control

The proportional derivative controllers, also angular spring servos, are most widely used control loop feedback mechanism. They are basically used to generate joint torques. A PD controller computes the torque, τ, of a joint by calculating the error between the current state and desired state of the joint. The controller is responsible of minimizing this error.

$$\tau = k_p(\theta_d - \theta) + k_d(\dot{\theta}_d - \dot{\theta}) \tag{3.1}$$

In the equation, θ_d and $\dot{\theta}_d$ are the desired orientation and desired angular velocity. The control is also combined with a spring constant k_p and a damping k_d constant, also known as control gains. Tuning k_p and k_d into correct values is a crucial process since they effect how responsive the character is to the proportional and derivative error. That is, very high controller gains of a joint make the joint movement very stiff and rigid; very low controller gains make joint responses very weak to follow the desired orientation and the angular velocity. Besides, if the spring constant is kept very low while the damping constant is high, the joint movements become very slow and very stiff.

3.1.2 Jacobian Transpose Control

The Jacobian Transpose method is used to calculate the necessary internal joint torques in order to create the effect of virtual external force at a specific body segment. Directly applying an external force over a body segment can create unnatural and puppet-like motions, so that is something undesirable. Creating necessary force effect on a body segment by applying internal torques causes more natural and realistic animations.

Jacobian Transpose method defines the relation between the joint orientations and the position that the external force is applied, $J = \frac{dP}{d\theta}$. This relation can be extended to the relation between joint torques and external force applied.

$$\tau = J(p)^T F \tag{3.2}$$

In the equation, τ is a vector of torques applied to a chain of bodies when a virtual force F is applied at a point p. $J(p)$ represents the rate of change of a point p with the rotation α_i about DOF i.

$$J(p)^T = \begin{bmatrix} \frac{dp_x}{d\alpha_1} & \frac{dp_y}{d\alpha_1} & \frac{dp_z}{d\alpha_1} \\ \vdots & \vdots & \vdots \\ \frac{dp_x}{d\alpha_k} & \frac{dp_x}{d\alpha_k} & \frac{dp_x}{d\alpha_k} \end{bmatrix} \tag{3.3}$$

3.2 Low Level Controllers

Animating physics based characters as the product of internal joint torques offers the opportunity of creating human-like motions, but it requires low-level control which is very difficult. There are some control strategies that should be taken into consideration. (1) The characters should be able to maintain their balance while performing an action, otherwise they fall over. In the physical environment, the gravity creates an extra issue to be controlled. (2) They should be able to move to any specific location without losing their balance. (3) They should be able to use their end-effectors (i.e., feet and hands) in order to reach and manipulate objects in the environment. In this section, we introduce some low-level controllers which are the necessary components for the high-level controllers that are responsible to perform these strategies.

3.2.1 Stance Swing State Controller

One walking loop can basically be divided into two states by taking into account the stance and swing legs. These states can be modeled as a Finite State Machine where state variables are $s \in \{left, right, double\}$ according to the current stance leg for a single step. Stance Swing Controller is dedicated to keep track of the

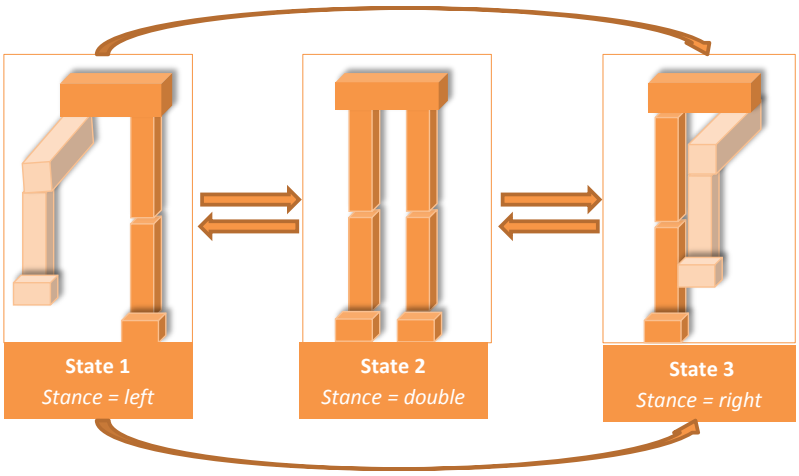

Figure 3.2: Stance Swing Controller.

transitions between these states. For walking motion, the transitions between states happens when the corresponding swing foot contacts the ground or the duration that is dedicated to that state is exceeded. The controller also holds information about how much of the current state has been completed with state phase parameter $\phi \in [1, 0)$ as t/T where t is the total time elapsed in the current state. The double stance state is different from left stance and right stance states such that the transition to other states from that state is not dependent to the step duration. It depends on an external command that comes from a different controller. The transition from any other states to the double stance state happens when the reaching controller sends the signal to Stance Swing State Controller to make the character stop moving and stand. For example, the reaching controller may send this signal when the center of mass (COM) of the character is close enough to the target location.

Stance Swing Controller is one of the common controller that is used by high-level controllers especially dominated with lower body movements, such as walking, standing. The usage of this controller also will be examined in detail in the other sections when explaining the high-level controllers.

3.2.2 Swing Foot Placement Controller

Carefully choosing step locations is one of the key tasks for the human walking behavior in order to restore balance. There is a direct relation between the foot placement and the balance restoration since we know that walking is a process of falling over and catching yourself just in time. During walking, we constantly try not to fall over by placing our swing foot down. If we lean forward with our upper bodies, we need to throw out a leg just in time to catch ourselves. Therefore, for this work, we use a simplified Inverted Pendulum Model in order to find the position that the swing foot should be placed for the next step.

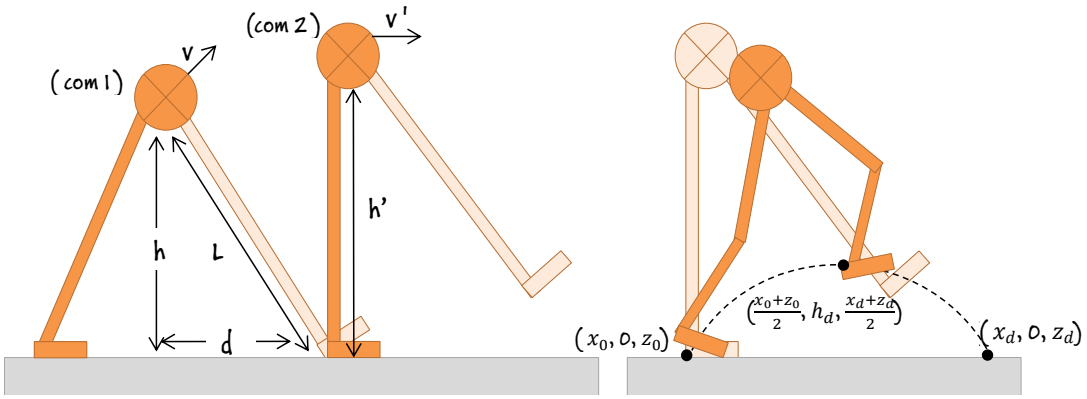

Figure 3.3: Swing Foot Placement Controller

Inverted Pendulum Model allow us to find the displacement $landing_d = (x_d, z_d)$ of the swing foot from the COM of the character (Figure 3.4). We can find $landing_d$ by equalizing the sum of potential and kinetic energy of the current COM with the sum of potential and kinetic energy of the the COM in balanced state where the character is upright position on its stance leg.

$$\frac{1}{2}mv^2 + mgh = \frac{1}{2}mv'^2 + mgh' \tag{3.4}$$

In the equation, $v' = 0$ and $h' = L = \sqrt{h^2 + x_d^2}$ since we want the velocity of the character's COM to be zero and we assume L is constant. If we modify the Eq. 3.4 according to this, we can solve it for d as given in Eq. 3.5.

$$x_d = v\sqrt{\frac{h}{g} + \frac{v^2}{4g^2}} \tag{3.5}$$

As a result, x_d gives the displacement of the foot in the sagittal plane to make the velocity of the character's COM zero. We can find the z_d which gives the displacement of the swing foot in the coronal plane in the same manner.

Defining the movement of the swing foot from the $leaving_d$ which is the position that it leaves the ground to the landing position $landing_d$ found by inverted pendulum model is also important for a natural walking behavior. If we define the ground leaving position of the swing foot $(x_0, 0, z_0)$ and the landing position $(x_d, 0, z_d)$, there is also a need for an extra third point in order to give an arc movement to the foot. We define this third point from the maximum swing foot height as $(\frac{x_0+x_d}{2}, h_d, \frac{z_0+z_d}{2})$. The step height h_d is one of the parameters that can be tuned for the Swing Foot Placement Controller. The trajectory of the swing foot is calculated with simple Catmull spline interpolation between these three points (Figure 3.4).

3.2.3 Virtual Anti-gravity Controller

Working in physical environment brings additional issues related to gravity. All body segments are affected from the gravity force proportional to their mass. Even though the torques can be calculated with PD controller in order to drive the character to the desired pose, the gravity force pulls down the body segments so that the desired pose can never be reached. This problem can be solved by giving high values to the control gains, k_p, k_d, in the PD control model, but it would create very stiff and rigid character movements. In addition, increasing control gains makes the character unstable to the contact forces.

While keeping the control gains, k_p, k_d, considerably optimum values, Virtual Anti-gravity Controller allows us to cancel the gravity effect on the body segments by creating negative virtual force that neutralizes the gravity force on them. If the gravity force on a body is mg, the virtual force that should be created on it should

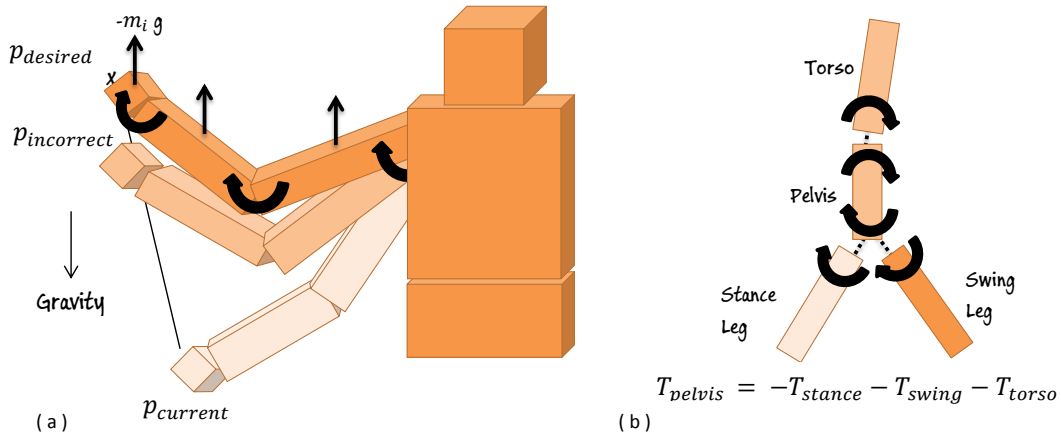

Figure 3.4: (a) Virtual Anti-gravity Controller. (b) Torque distribution Controller.

be $-mg$. In order to create this virtual force on the body, Jacobian Transpose Control can be used which calculates the necessary internal joint torques. To cancel the gravity effect on the whole body, this process is applied to all body segments separately. While applying Jacobian Transpose Control on a body, the chain of body segments from the corresponding segment to the root segment is used as the Jacobian Transpose Chain.

3.2.4 Virtual COM-Force Controller

A skillful simulated character should be able to move successfully to any desired position in the physical environment. Even if stepping toward a place is a very common action in daily life, it requires sophisticated control of body dynamics for physically simulated characters. In order to move the body a target, a force should be generated whose the direction is toward the target location. However, as mentioned in the previous sections, applying a direct external force will create unnatural motions such that the character is being pushed. Therefore, the Jacobian Transpose technique is used to create a virtual COM force which of the pivot point of the chain is the character's stance ankle.

$$F_{com} = k_{p_{com}}(P'_{com} - P_{com}) + k_{d_{com}}(V'_{com} - V_{com}) \tag{3.6}$$

In the equation, while P_{com}, V_{com} are the current position and the velocity of the center of mass of the character, P'_{com}, V'_{com} are the desired COM position and velocity respectively. The Virtual COM-Force Controller allows the character to move to any pre-defined location in the environment by calculating a virtual force (Eq. 3.6), and converting it into internal joint torques.

3.2.5 Virtual Hand-Force Controller

Another important skill that the character should have is that of being able to reach a specific point in the environment. This fundamental skill is rather crucial to allow the character to interact with the object in the environment. The similar approach is used with the one used in the COM-force controller, but the calculated virtual force here applied only end-effectors, hands.

$$F_{hand} = k_{p_{hand}}(P'_{hand} - P_{hand}) + k_{d_{hand}}(V'_{hand} - V_{hand}) \tag{3.7}$$

In this framework, once the necessary virtual force is computed, the desired joint torques for the shoulder, elbow and the hand are found by Jacobian Transpose Method if the character is within the reachable distance. Virtual Hand-Force Controller has two processors for two hands separately. Usage of the processors changes according to the strategy - left hand only, right only, or both hands.

3.2.6 Torque Distribution Controller

It has been studied that there is a relation between the torque that will be applied to the pelvis, and the stance hip and swing hip torques (e.g. [20], [21]). This simple relation is very crucial for the control of balance. In Torque Distribution Controller, Stance hip torque is calculated in a similar way as the one proposed in

SIMBICON. First of all, the torques of the swing hip and the torso are computed, then a virtual pelvis torque is calculated via a virtual PD controller. This torque is virtual because it is not applied directly, instead it is used to calculate the stance hip torque (Eq. 3.8).

$$\tau_{stance} = -\tau_{pelvis} - \tau_{swing} - \tau_{torso} \tag{3.8}$$

3.3 Walking Controller

The walking behavior is the core part of most of the motions, and it is most fundamental behavior for a skillful simulated character. Unfortunately, it is challenging to simulate a robust motion since physically simulated characters are mostly unstable and underactuated such that requires controlling high-dimensional actions.

In this section, a high-level walking controller is presented which allows the character to walk to a specific location. The idea behind the controller is to calculate the internal joint torques for the stance leg to drive the character to the target location while computing the internal torques of the swing leg that enable the character to maintain balance by using inverted pendulum model (Figure 3.5).

The starting point of the controller is a simple finite state machine, Swing-Stance State Controller. It defines the current swing foot and the current stance as well as the duration of one step cycle. Knowing which one is the swing foot and which one is the stance foot is crucial for a walking controller because each foot has different tasks during the walking behavior. While swing foot involves the processes that are responsible of keeping the character balance, the stance foot is the pivot point of the chain that drives the character to any specific location. Therefore, Swing-Stance State Controller is a global controller that can be reached by all other controllers.

The walking controller introduced here concerns only about the lower body

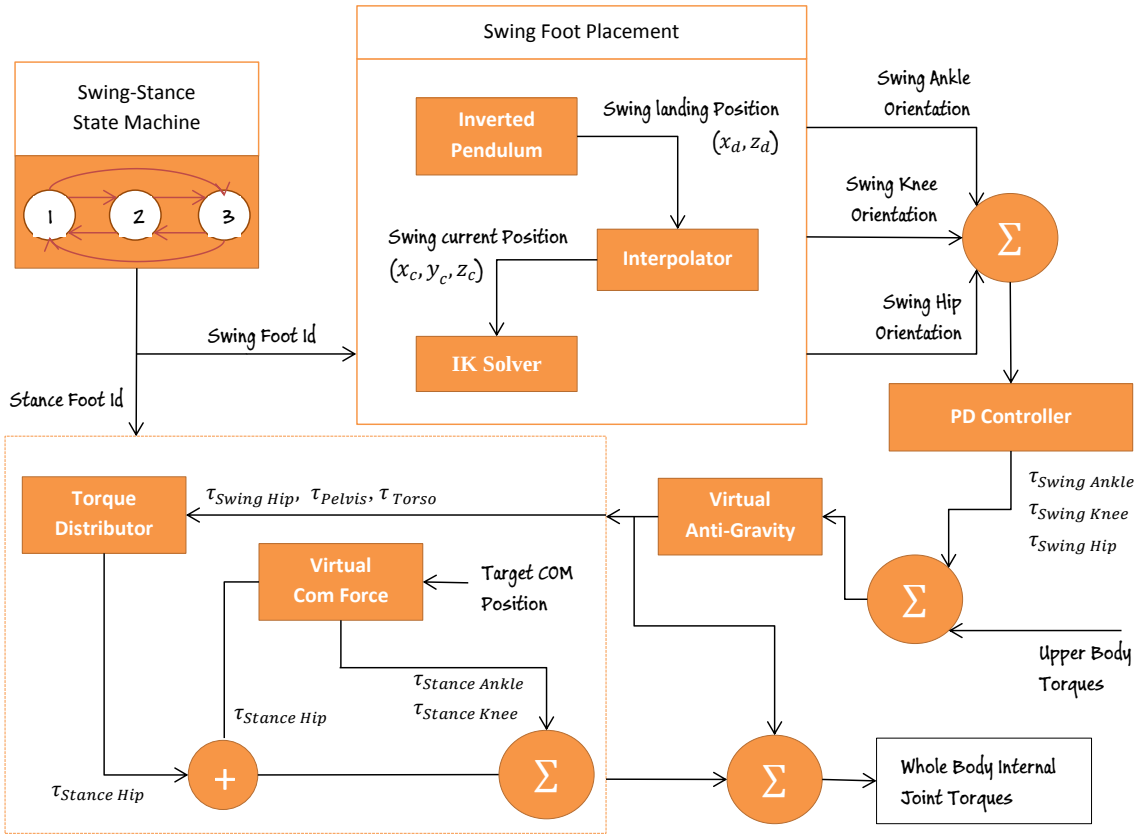

Figure 3.5: General structure of walking controller

joint torques. That is, the torques applied to the upper body joints do not affect the controller and balance of the character. The second process in the controller is the Swing foot placement. First, the landing position (x_d, z_d) of the swing foot which is taken from the Swing-Stance State Controller is calculated using inverted pendulum model. Then, the desired swing foot position (x_d, y_d, z_d) is found using an interpolator. At the end of the Swing Foot Placement process, the desired joint orientation of the swing ankle, swing knee and swing hip are found with the help of a simple Inverse Kinematic (IK) solver from the desired swing foot position. Next, these desired orientations are converted to desired torques for swing ankle, knee and hip using Proportional Derivative Controller. All torques including upper body torques are strengthened with the anti-gravity torques that are calculated in the Virtual Anti-Gravity Controller.

The torques of stance ankle, knee and hip are calculated in the Virtual COM-Force Controller given the target center of position. At the same time torque distributor computes the stance hip torque given the swing hip, pelvis and torso torques. Finally, the stance hip torque from COM-Force Controller and the stance hip torque from Torque distributor are summed up. At the end, the internal joint torques are found that allow the character walk to any specific location.

3.4 Standing Controller

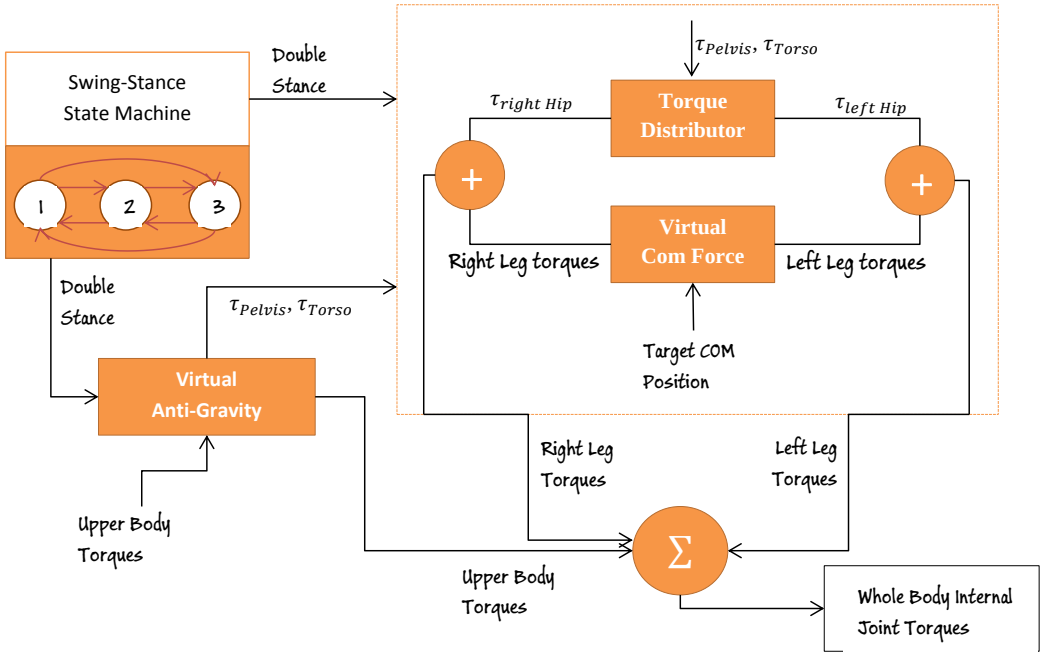

Figure 3.6: General structure of standing controller

Maintaining a natural standing posture is one of the simplest actions for the humans. In biomechanics, the standing control phenomenon depends on some factors, such as the position of the center of gravity over the base of support. Basically the strategy the humans employ is to instinctively move the rest of your body to adjust the position of your center of gravity. The center of gravity (CG) is the average of the character's weight distribution. Since the gravity is

constant, the center of gravity is the center of mass at the same time. While standing, the base of support of a human is the area under his/her feet. Based on this, if the center of mass of the character is inside of the character's base of support, then the character maintains its balance. The closer the character's COM is to the center of the base of support, the more balanced the character becomes.

In this section, a standing controller which is responsible for maintaining a standing pose is presented (Figure 3.6). It is based on calculating the internal joint torques that keep the center of mass of the character closer to the center of the support base of the character. In a similar way as in the walking controller, the global Swing-Stance Machine is the starting point of the controller. When the state machine shows the double stance state, the virtual COM-force controller calculates the internal leg joint torques that create the virtual force that keep the character's center of mass closer to the base of support. The internal joint torques for the left leg, $\tau_{leftAnkle}, \tau_{leftKnee}, \tau_{leftHip}$, and for the right leg, $\tau_{rightAnkle}, \tau_{rightKnee}, \tau_{rightHip}$, are calculated separately by using Jacobian transpose. At the same time, the torque distributor distributes the torques coming from τ_{Pelvis} and τ_{Torso} to the right and left hip equally such that $\tau_{leftHip} = \frac{\tau_{Perlvis}+\tau_{Torso}}{2}$ and $\tau_{rightHip} = \frac{\tau_{Perlvis}+\tau_{Torso}}{2}$. The torques calculated by the virtual COM force controller and the torque distributor are summed up together that form the lower body torques. As same as the walking controller, the standing controller is not affected from the upper body torques.

3.5 Reaching Controller

In contrast to lower limbs, the upper extremity (UE) tasks, such as reaching, require executing fine movements due to its extensive functionality. Therefore, an ordinary task in daily life of a human, reaching, is still a challenge to synthesize with inverse kinematic based models or data-driven methods. The main reason is that there are possibly many poses. Sometimes, it may require only moving your arms to a target position while standing, but there may be cases that requires

complex whole body motions, such as taking a step to reach while keeping balance.

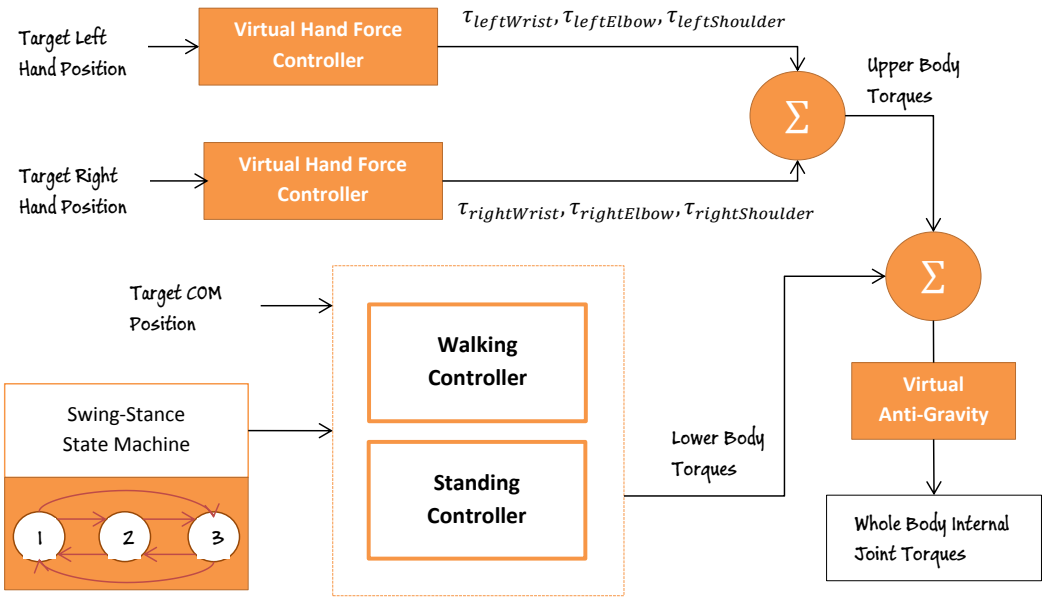

Figure 3.7: General structure of reaching controller. It dependent on the standing and walking controller.

In this section, a high-level controller for generating reaching motions is explained that conforms the flexibility of the human reaching behavior. The main component of the reaching controller is the virtual hand force controller. It calculates the necessary arm torques $\tau_{shoulder}, \tau_{elbow}, \tau_{wrist}$ by computing a virtual force according to the desired hand positions. There are three important inputs for these controller- target left hand position, target right hand position and target COM position as shown in Figure 3.7. While target hand positions contribute the computation of the upper body torques, target COM position contributes the lower body internal joint torques. The lower body torques are either coming from the standing controller or walking controller as explained in the previous sections. After combining the upper body torques with lower body torques, all final body torques are obtained by canceling gravity effect over them by using Virtual anti-gravity controller.

Chapter 4

Planning of a Ball Catching

4.1 Introduction

After developing computational models of motor control as examined in the previous sections, getting the initial conditions correct in kinematics and kinetics terms is the key to be able to use these models in many sport activities. This requires a research that is collaborated with the literature which describes best biomechanical strategies. In Interception tasks, such as kicking, throwing, catching, the learned strategies increase the consciousness in motor control which allows the performer to achieve the goal successfully. At the same time, they prepare the body to adjust the time accurately which enables the performer to prevent possible injuries.

In this work, we focus on modeling a highly complex task that requires the combination of important motor skills - ball catching. We study the movements during ball catching motions and present a system that imitates the human movements while performing a ball catching behavior.

The reason behind choosing ball catching behavior is that it requires accurate planning of the usage of several elementary motor primitives, time optimization.

We model the human movements in a ball catching behavior using discrete movement phases in the light of Biomechanics studies. For such a complex behavior, the human central system has a little time to plan and react. Hence, we propose a system such that the character anticipates the interception point and the amount of time needed to reach the desired hitting position, and adjusts its movements accordingly.

The remainder of this chapter is organized as follows. For the following section, we introduce the ball catching problem, which is the central topics of this chapter. Then the phase analysis of a ball catching task and its details are presented which involve bio-mechanically breaking down the movement into distinct phases. Then, the approach used for predicting the target interception positions based on the naturalness and efficiency concerns are explained. In the last section of this chapter, the control parameter assigned for each phase are presented.

4.2 Ball Catching Behavior

In real world situations, ball catching is a challenging task which requires spatio-temporal control. It requires locating the ball and determining the proper interception place as well as timing by identifying the speed of the ball. Humans are able to successfully predict the ball trajectory in order to move hands to the right place at the right time after some training. This behavior can be adapted to the physically simulated characters such that they can learn to move to a target position at the right time by anticipating the interception point from ball trajectory. On the other hand, there are two important problems that should be addressed at this point: (1) infinitely many different interception points can be found along the trajectory of the ball, (2) infinitely many different body motions can be found to reach this target position in a specific flight time of the ball. The solution to these problems is to define tasks to reduce the redundancy in spatial position and timing, such as, energy minimization, trajectory smoothness, time accuracy, etc.

Based on these findings, in this work, we exploit some specific cost functions to achieve the redundancy in spatial interception position. Besides, the timing of interception is controlled by scaling temporal window using a control strategy. This control strategy is based on dividing up the movement into relevant phases and controlling them.

4.2.1 Phases of Ball Catching

In phase analysis, ballistic movements, such as throwing, kicking, hitting, can be divided biomechanically into three phases: preparation, action and recovery. Each phase is defined different biomechanical features and boundaries. In preparation phase, the performer gets ready for the performance. The action phase is the end of the preparation phase and is the part where the main effort of the performer takes place. After action phase, the recovery phase takes under control the deceleration of the movement.

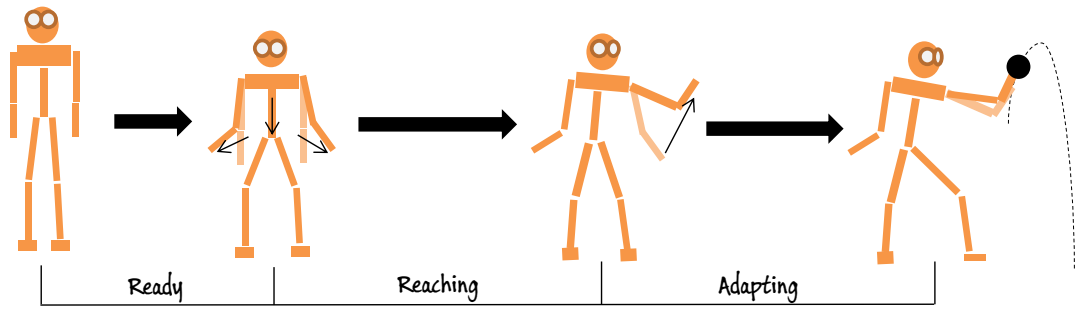

Figure 4.1: The three phases of catching motion.

According to phase analysis, as other many ballistic movements, the catching movement exhibits similar phases independent of the location and the velocity of the ball. Inspired by this, we divides a regular catching movement into following four phases which are labeled related to their functionality (Figure 4.1).

Ready Phase The ready phase starts when the ball is thrown, and the interception hand position and center of mass position of the character is determined. The character slowly starts to step to the target COM position while hands that are responsible of catching the ball begin to move to a position that is above the waist, slightly in front of the chest. Even though the interception hand position is decided before the ready phase, the hands move independent from that position. The main goal of the ready phase is to prepare the character to reaching movement by slightly accelerating the body and bringing the hands to a location that can optimally reach any position in the reachable arm space.

Reaching Phase The reaching phase comes after the ready phase. In reaching phase, the character starts to extend its arms towards virtual interception hand positions that are at the same distance from the COM of the character in the desired final pose at the interception moment. In this phase, the character increases its COM speed to reach the desired interception position for the center of mass of the character. For a successful and realistic catching behavior, the reaching phase should start to execute when the character is optimally close to the interception position, or ball in order to give an impression that the character can track the ball in a more healthy way when the vision system is more comfortable to see the ball.

Adapting Phase After the reaching phase, the adapting phase is important for the character to accurately reach to the interception hand positions. In this phase, while the COM velocity of the character starts to decelerate, the hands starts to move directly to the interception hand position that is found on the ball trajectory. As comparison to the reaching phase, the character should be more close to the interception COM position and the interception hand position such that there is no need to step toward the interception point anymore. If the character executes the adapting phase when it is not close enough, the movement of the character seems as the hands drives the character to the interception point.

Catching Phase The catching phase comprises the movements where the character meets the ball and the recovery from the impact. The reason for

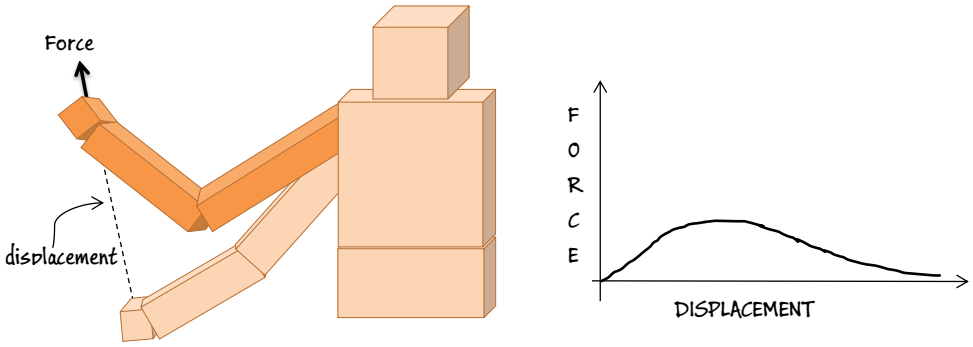

Figure 4.2: The force and displacement relation of catching. The longer displacement reduce the effect of high force.

naming this phase as 'catching phase' comes from that it controls the reduction of the speed of the ball.

In a typical catching behavior, humans tend to extend the time to make the ball stop in order to reduce the risk of hurting themselves. This is because the hands are responsible of applying negative force to decrease the momentum of the ball to zero in order to bring the ball to stands still. Choosing smaller force to apply over a longer time makes the catching motion more comfortable because it decreases the pressure on the hands and the stress on the joints.

Since increasing the time gives the same effect as increasing the displacement after the ball hits the hand, the work-energy principle is used to calculate the necessary displacement to reduce the ball's kinetic energy to zero while reducing the potential damageable stress on the joints. The difference between the final kinetic energy and the initial kinetic energy of the ball gives the work as the multiplication of a constant force and the displacement.

$$\frac{1}{2}M_b V_{bfinal} - \frac{1}{2}M_b V_{binitial} = F_{max} d_s \tag{4.1}$$

In the equation, F_{max} represents the the maximum force that can be applied without any damage. Since the final kinetic energy of the ball is required

to be zero, the work is equal to the initial kinetic energy of the ball in magnitude. Therefore, the necessary displacement for the hand with the ball can be calculated for a natural and realistic catching movement.

$$d_s = -\frac{1}{2} M_b V_{binitial} \frac{1}{F_{max}} \tag{4.2}$$

Increasing F_{max} decreases the displacement and increases stress of the arm joints, while decreasing the value of F_{max} increases the displacement, but reduces the stress on the joints (Figure 4.2).

The phases, ready, reaching, adapting, catching are consecutive. Among these phases, the only phase that we cannot control the transition to is the catching phase because starting to execute catching phase is strictly dependent on the condition that the ball is touching, or almost touching, the hand. The remaining phases, ready, reaching, adapting are very important to achieve desired final catching position. The condition for a successful catching relies on the control of the transitions between these phases. The process that the character learns how to control the transitions for a better timing and successful catching is explained in the chapter 4.

4.2.2 Catching Strategy

At the beginning of a catching process, just after the ball is thrown, the character first decides an appropriate catching strategy. Generally, the strategies in catching can be divided into three categories: left-handed, right-handed and two-handed catching. Deciding the catching strategy is the first step because it describes the body parts that will be used during the execution of the catching phases. Figure 4.3 shows the process of deciding catching strategy according to the trajectory of the ball.

In general, one-handed catching strategies are chosen in order to save time while reaching to the interception position. Whether the left-handed or the right-handed strategy is chosen depends on the direction of the ball. The reason for this

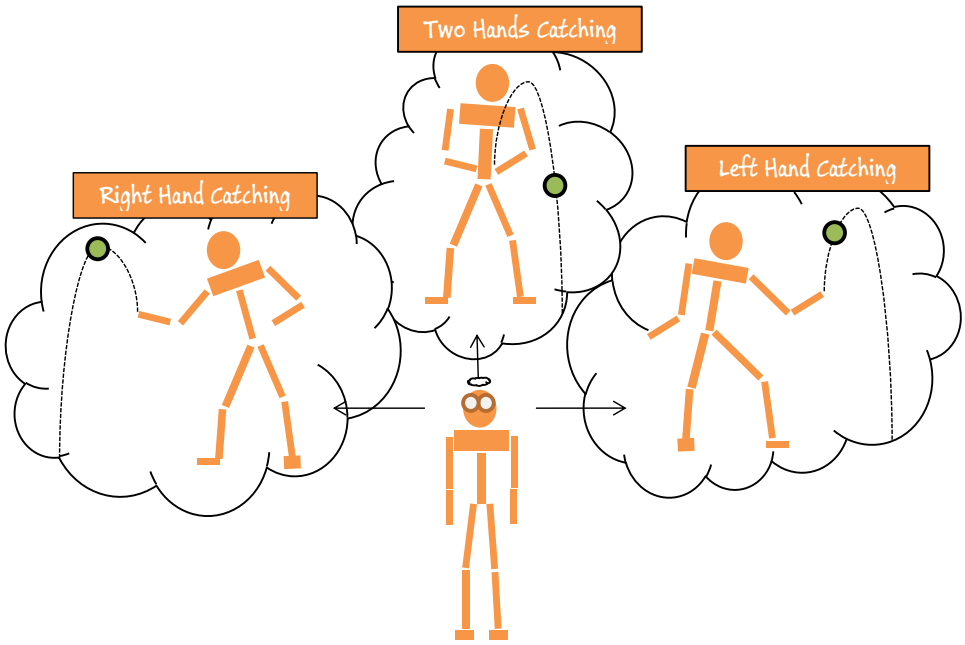

Figure 4.3: The process of the character's decision making for choosing an appropriate catching strategy.

catching a ball using left-handed strategy while the ball is coming towards the right side of the character would be unrealistic and challenging. For aesthetic and realistic purposes, the two-handed catching strategy is chosen in some occasions, especially when the ball flies directly over the character.

4.3 Interception Point Prediction

Parallel to the process of choosing an appropriate catching strategy, the character determines an ideal hitting point on the ball trajectory, called Interception Hand Position, and a standing position for the center of mass while the hands are extended to the interception hand position, called interception COM position. Determining an optimal interception hand position and interception COM position is substantial because they are two important ingredients for the reaching controller (Figure 4.4).

Considering the purposes of naturalness and efficiency, we make the following assumptions while selecting optimal interception hand position by taking into account a typical human ball catching behavior.

Reachability During a catching behavior, people make judgements on reachability of the ball, and give up making an effort if they think they could probably miss the ball.

Maximum time People tend to reach a position over the ball path where the flight time of the ball is possibly maximum. This instinct comes from the fact that how much the flight time is, is there enough time to reach the interception point in time.

Minimum displacement For energy minimization, people are also tend to select positions that require minimum displacement for the body, and as a result minimum effort.

Based on these assumptions, we set up a simple quadratic optimization for finding an optimal interception hand position q^*.

$$\begin{aligned} q^* = \arg\min_{q \in T} \quad & \underbrace{\alpha E_d}_{\text{displacement}} + \underbrace{\beta E_t}_{\text{time}} \\ \text{subject to} \quad & q_y \leq P_{maxy} \\ & q_y \geq P_{miny} \end{aligned} \tag{4.3}$$

In the energy function, the minimum displacement criterion is considered with E_d by calculating the quadratic distance of the character to the point.

$$E_d = ||P_c - q||^2$$

where P_c is the center of mass position of the character. The maximum time criterion is taken into account with E_t which calculates the time required for ball to reach to the point from the current position and the velocity of the ball.

$$E_t = -(P_b - q)/V_b$$

where P_b, V_b are the position and velocity of the ball, respectively. Each of the position vectors in the overall energy function is projected on the $x-z$ plane, and for the convenience, the velocities are horizontal. The constants, α and β, are used to control the weights of two criteria on the process of selecting interception hand position.

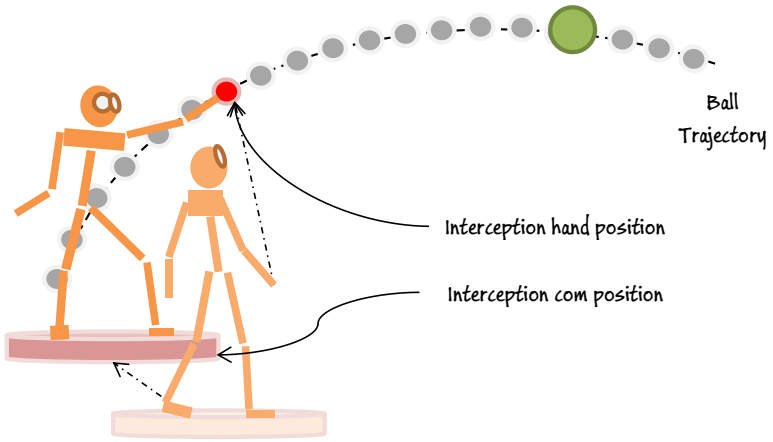

Figure 4.4: The target interception points. The red point in the ball trajectory shows the interception hand position. The claret red color area under the character shows interception com position.

The reachability factor is integrated into the optimization process by defining constants that limit the minimum and maximum reachable height for the interception point search.

The interception COM position is estimated after the interception hand position is found in a way that puts the body into an advantageous position for the catching phase and increases the deceleration path of the ball to be catched. The key point of determining the interception COM position is that it should provide an area which allows the hand to move freely while dissipating the energy of the ball.

4.3.1 Catching Phase Parameters

As described in the previous sections, we employ a catching strategy that breaks down the movement into phases.

Phases	COM_{K_p}	COM_{K_d}	$Hand_{K_p}$	$Hand_{K_d}$
Ready	340	40	200	30
Reaching	200	20	120	12
Adapting	100	10	400	40

Table 4.1: Control parameters of catching phases.

Each phase has control parameters, including K_p, K_d for center of mass of the character and K_p, K_d for hands according to the catching strategy. While control parameters for the COM, COM_{K_p}, COM_{K_d}, describe how fast the character can reach to the interception position, the control parameters for hands, $Hand_{K_p}, Hand_{K_d}$, indirectly describe the duration of the phases to reach to the target posture specifically defined for each phase. Table 4.1 shows the control parameters the catching phases, ready phase, reaching phase and adapting phase.

The high-level reaching controller takes different hand positions and COM speed according to the current catching phase. Therefore, controlling the flow of these phases changes the overall timing and structure of a catching task since each phase has a specific timing related to these control parameters. Correctly planning when the character should be in which phase is also very crucial to catch the ball just on time.

Chapter 5

Learning Timing in Motor Skills

5.1 Introduction

A key challenge of interception based tasks, as catching, is the process of planning how to choose an optimal sequence of sub-actions in order to generate natural-looking motions. In such a dynamic motion, learning action planning using a trial-and-error approach is a solution to this problem. Here, as the trial-and-error approach, the reinforcement learning is used. Reinforcement learning is a promising framework for controlling difficult behaviors that require planning for a distant goal by enabling us to formulate higher-level targets and generate control policies in order to achieve them.

The timing is very crucial in interception based motions because using time accurately enables more control over the impact force. Therefore, in this part, the character learns policies to know how and when to react by using reinforcement learning. The policy plans the phases of the ball catching behavior, which is explained in the previous chapter, to provide correct timing during the activity. The correct timing not only helps the character reach to the right place at the right time, but also makes movements of the character more reliable.

The Reinforcement Learning Toolbox. After a comprehensive search

for an available and open-source reinforcement learning toolkit, the reinforcement toolbox (RL Toolbox) is chosen since its repertoire includes many standard reinforcement learning algorithms and it is easy to extend for new additional algorithms. The RL Toolbox is a c++ based framework that presents most common RL algorithms and user-friendly interface for implementing new algorithms. It provides a rich variety of learning algorithms, such as Q-learning, TD learning, Actor critic learning, etc. The tool also employs a logging and error recognition system. On the other hand, it is limited to the choices of reinforcement learning functionality that can be used. Some specific features are selected because that the toolbox supports, even if it is not the best solution.

This chapter starts with introducing the key elements of the reinforcement learning framework in a broad sense. Then, the following section focuses on the process of adapting the ball catching behavior to an action planning problem whose goal is to optimize the timing using reinforcement learning.

5.2 Reinforcement Learning

Reinforcement Learning is one of the Machine learning approaches that use a process of learning from interactions with environment to achieve a goal. The learner, called agent, interacts with its environment and observes the results in a similar way of humans or animals.

The trial-and-error based interactions take place such that the agent senses the environment, and then uses this sensory input, also called state, in order to choose an action to perform. After performing an action in a state, the agent receives some reward in a form of scalar value. In this way, the system learns a mapping from state/action pairs by trying to maximize the long-term reward from some initial state to a final state. This learned mapping from states to actions is called policy. That is, the policy tells which action should be performed in a particular state. Based on these findings, there are three basic requirements of a reinforcement learning system designer based on the goal: defining reward

function, states and devising learning algorithm to estimate value functions.

value Function gives the value of the states, or the expected return value for an action performed in that state. This defines how good a particular action is in a given state. There are two common notations for value functions - $V^{\pi}(s)$ which gives the value of a state under policy π, $Q^{\pi}(s, a)$ which gives the value of the performed action a in a state s under policy π.

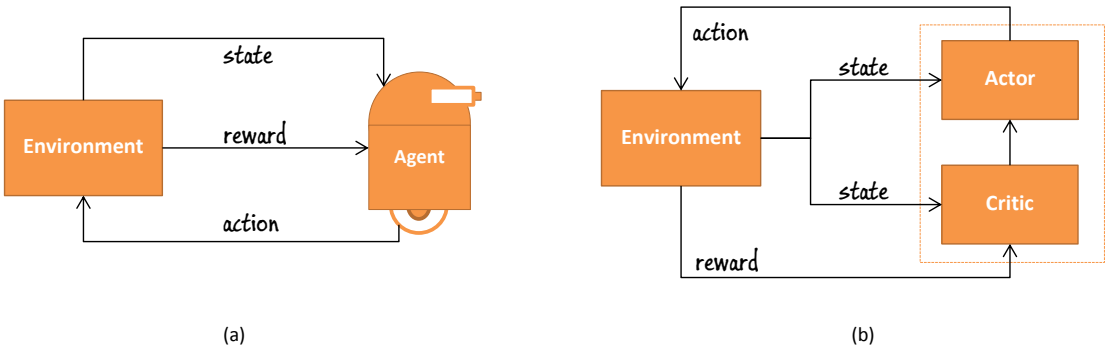

Figure 5.1: (a) Simplified physics based character. (b) Motion Controller and Physics Simulator.

5.2.1 Reinforcement Learning Algorithms

The Reinforcement learning algorithms that have been introduced in the literature can be categorized into three groups: actor-only, critic-only and actor-critic methods.

Critic-only methods are based on deriving optimal policy from the optimal value function that is found earlier. There are a number of different critic-only reinforcement learning algorithms, but Q-learning, SARSA are well known among them. These learning algorithms derives an optimal policy from estimated value function based on observation of the transitions from one state to another state when taking an action and the returning reward.

The algorithm of Q-learning method which is one of the famous learning

technique in this category. In the algorithm, there are two important parameters should be carefully set. α is the learning rate such that giving a high value leads to learning quickly. γ is the discount factor which is used to adjust how the future rewards will worth less than current rewards.

Algorithm 1 Q-Learning

```
1: procedure Q-Learning
2:     Initialize Q(s, a) arbitrarily
3:     for all episode do
4:         Initialize s
5:         for all steps in episode do
6:             Choose a from s using policy derived from Q
7:             Take action a, observe reward r, and next state s'
8:             Q(s, a) = Q(s, a) + α(r + γmaxQ(s', a') − Q(s, a))
9:             s ← s'
10:        end for
11:    end for
12: end procedure
```

The main disadvantage of the critic-only methods is that they usually need to discretize the continuous action space because the optimization process to find the action is computationally expensive. Therefore, this discretization causes problems for finding the true optimal policy.

Actor-only methods, in contrast to Critic-only methods, directly use optimization procedures to search for an optimal policy without trying to find an optimal value function beforehand. Policy gradient methods are an example for actor-only methods. The biggest advantage of actor-only methods over the critic-only methods is that in actor-critic only methods, the policy can create actions in continuous action space.

Actor-Critic methods combine the advantages of the actor-only and critic-only methods together where the actor and critic are represented separately. Figure 5.1b shows the schematic structure of an actor-critic architecture and the interaction of the actor and the critic. In the system, the agent is divided into two separate parts. The actor, also known as policy, is responsible of selecting actions. The critic, also known as value function, monitors and criticizes the these actions

by processing the rewards. The critic is also responsible of evaluating the policy and determining when the policy should change. That is, the critic evaluates the new state after an action selection in the context of Temporal Difference (TD) error by using current value function. This TD error is estimated as,

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{5.1}$$

In the equation, V is the value function and s_{t+1} is the new state. In this way, the critic can evaluate the action a_t taken in state s_t. The positive result of the equation suggests that this action should be selected more next time, while negative result suggests that the selection of this action should be avoided more in the future.

The main advantage of Q-learning over actor-critic learning is that it is not required to follow a current policy. However, Q-learning can only be considered when the states and the actions are both discrete. On the other hand, many problems in real world, the states and the actions are both continuous variables. Based on this, the actor-critic approach has numerous advantages over actor-only and critic-only methods. The major advantage is that they can learn with continuous state variables and continuous valued actions and require minimum computation while selecting actions since the policy can be explicitly stored.

5.3 Planning with Reinforcement Learning

The ball catching, as an interception task, requires a correct timing for reaching to the right place at the right time. Even though the goal, in this case catching the ball, and the phases that are required for a successful catching are well known, the character should learn how to achieve the goal by planning these phases. In this case, the reinforcement learning is a useful framework for learning with a critic similar to a human learning system.

For such a complex task like catching a ball, it would be tedious to produce

a model for the character that performs in every possible situations. Therefore, reinforcement learning makes it easy by evaluating the results of many random actions, and learning the best actions from any state at the end. In this section, we introduce the reinforcement learning system used in our work, and we explain the process of designing the components of the reinforcement learning for our problem.

The system consists of two distinct process - training and using, as shown in Figure 5.2. The training process corresponds to the part where the learning happens. In this part, the character learns a policy that defines which actions are best by interacting with the environment. As mentioned in the previous section, there are different algorithms to learn policy. In our system, we preferred to use Actor-critic method. In general architecture for actor critic learning, there are two learners. A value function learner, Critic, which critiques the actor's actions, and an Actor which learns the policy.

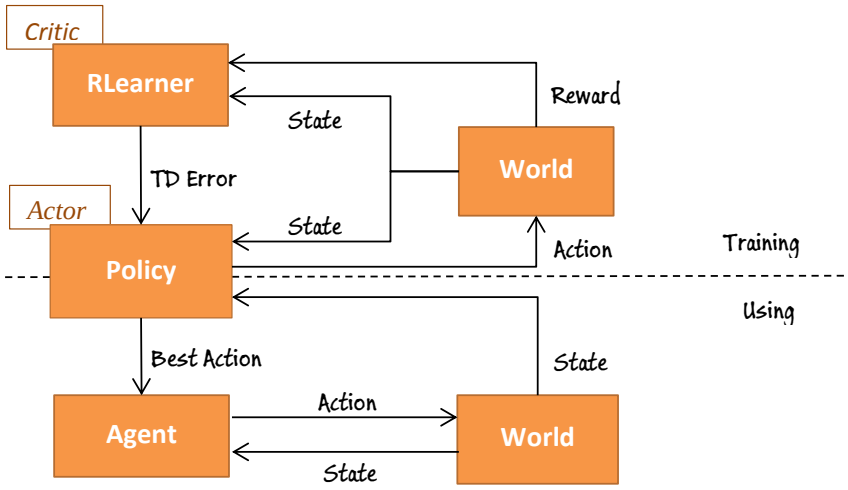

Figure 5.2: Reinforcement Learning system overview. It has two distinct parts: offline training process and online simulation process.

5.3.1 State Variables

The 'State' in a reinforcement learning system is a parameter or a set of parameters that are used to describe the specific situation the agent is in. For example, the coordinates of a ball can be used as a state variable in our case. It is very important to determine state variables tightly associated with the goal of the learning system. Otherwise, the unrelated state variables would only increase the dimensionality of the system. The size of the state space, and hence the complexity of the problem, grows exponentially with the number of state variables. Therefore, the optimal number of state variables are determined for the ball catching problem as follows:

Distance of ball d_{ball} - Since ball catching behavior is directly related to the current state of the ball, one of the state variables are considered as the distance of the current position of the ball to the calculated interception position. This state corresponds to the behavior of a person tracking the ball position while catching.

Time of ball t_{ball} - One another state variable related to the ball is the remaining time of the ball to the interception position. This state variable gives the information about not only the remaining time for the character but also the dynamics of the ball (i.e., velocity) together with the state variable d_{ball}.

Distance to interception hand position d_{hand} - One of the state variables associated with the character's current posture is the distance between the current hand position of the character and the interception hand position according to the catching strategy.

Distance to interception COM position d_{com} - One another state variable related to the character's current posture is the distance of the character's current center of mass position to the interception COM position.

There is no doubt that there possibly exist different state variables that can give better results, but for our problem, these state variables are found acceptable.

Even though the number of state variables are kept to be small, all of the state variables, in our case, are continuous data which increase the complexity of the model. Discretization is one of the applicable approaches when the state space is large due to the continuous state variables in order to reduce the size of the state space. Discretization is the process of dividing the state space into many small chunks. On the other hand, a discrete state representation can be problematic. If the state discretization is too coarse, the agent cannot distinguish between the states. In order to compensate for this problem, a function approximator for the value function is used which combines dividing the state space and saving the generalization between states.

5.3.2 Action Variables

The action defines what the agent can do in the current state. For our case, the motion of the character can be controlled by defining actions. After learning best actions from each state, the character should be able to decide the sequence of actions that make it reach to the target state from any arbitrary state.

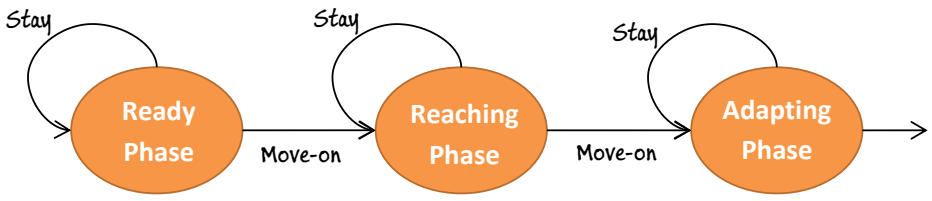

Figure 5.3: Actions and possible results of them in a FSM structure.

There are two high-level actions for our system that are described as (see Figure 5.3):

(1) Stay This action make the character stay in the current catching phase.

(2) Move-On This action is chosen if the character is expected to be move on another catching state.

These two simple actions enable us to plan the motions of the character by controlling transitions between the catching phases. By determining which state the character should be in at right time, the character can manage to control the timing that is making it to reach to target interception position on time.

5.3.3 Reward

In Reinforcement learning system, the goal of the agent is formalized in terms of the rewards. It describes the way of saying to the agent what is wanted from it to achieve. Designing the rewards correctly is very important due to the fact that agent should learn to reach to the goal by trying to maximize the total amount of reward it receives.

In our case, we designed the rewards by taking into account several considerations. There are two types of reasons for the termination of current catching trial during the learning process. The character takes a reward based on the type of termination as a simple number, $R_t \in \mathbb{R}$.

$$R_t = \begin{cases} -20000 & \text{if } result \equiv case1 \\ 1000 - 200{d_{com}}^2 - 800{d_{ball}}^2 & \text{if } result \equiv case2 \end{cases}$$

Case1 *Failed* - The first case happens when the ball passed through the interception point before the hands of the character reach to the interception hand position. In this case the character considered to be failed and get a negative reward.

Case2 *Succeeded* - This case refers to the situation where the character reaches the interception position with hand before the ball. In this case, the character is granted with a positive reward, but not completely. Arriving to the interception point with the hands when the ball is very far away from it is an unwanted situation. It creates an unnatural look because the human vision system cannot foresee the position of the impact accurately when the

ball is so distant. In addition, even though the hands reach to the interception hand position accurately, if the center of mass position is still not close enough to the interception COM position, this creates an unrealistic motion, and gives an impression that the body is under control of the hands. Therefore, we added negative reward related to the distance of the ball the COM to the positive reward.

Chapter 6

Results

In order to evaluate the generality of our approach, we simulates a number of catching motions with various different initial conditions and catching strategies (left-handed catching, rigth-handed catching, two-handed catching). In this section, we describe the results of our system.

Figure 6.1 summarizes the choices of catching phases for one catching trial of every 20 iterations during learning process.

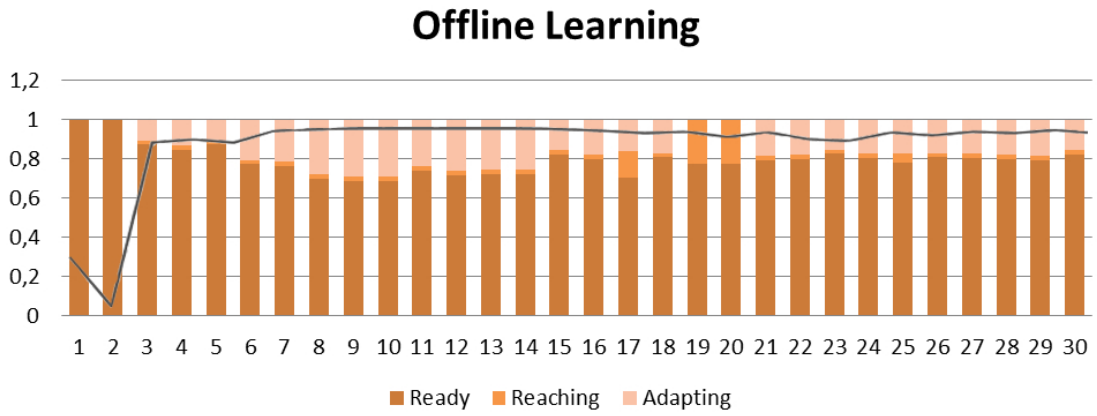

Figure 6.1: Distribution of catching phases during learning period.

The learning time changes according to the number of iterations and the conditions of trials. We perform 100 steps per iteration and 600 iterations for

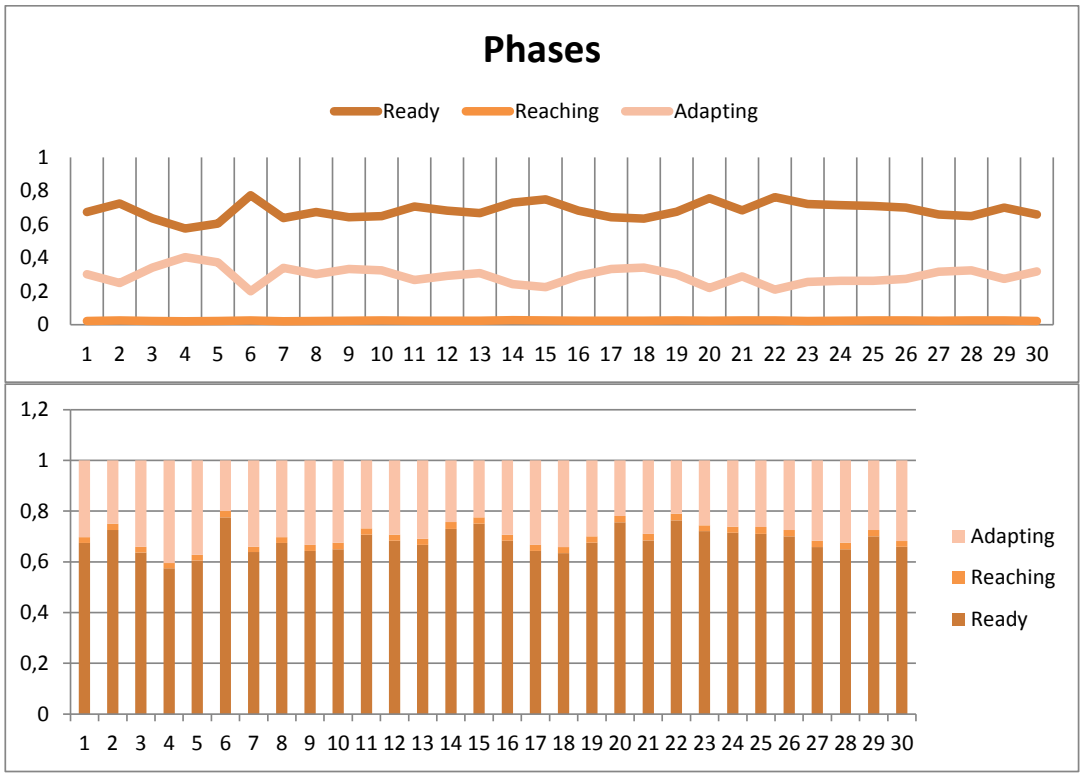

Figure 6.2: Distribution of catching phases for several simulation results.

offline learning process. All simulation results were produced on a computer with a 2.00GHz CPU and 6GB of memory.

Figure 6.2 shows the usage of the catching phases of the trained policy over 30 catching simulation samples, after learning sessions is completed. Compared to data in the offline learning diagram, this diagram demonstrates a more regular behavior. We observe that for the catching trials that require more body translations, the ready phase is chosen more, while the adapting phase is preferred more when the less body translations are necessary and when there is less time.

Some of the catching animation results executed in different catching strategies are demonstrated in Figure 6.3, 6.4 and 6.5. We use Bullet Physics Engine to simulate the character. The time step is 0.1 milliseconds.

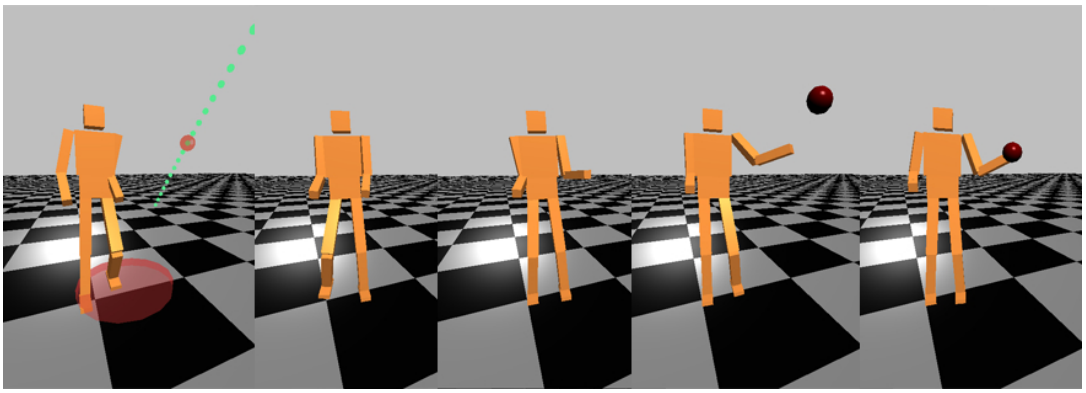

Figure 6.3: Simulation result of left-handed catching.

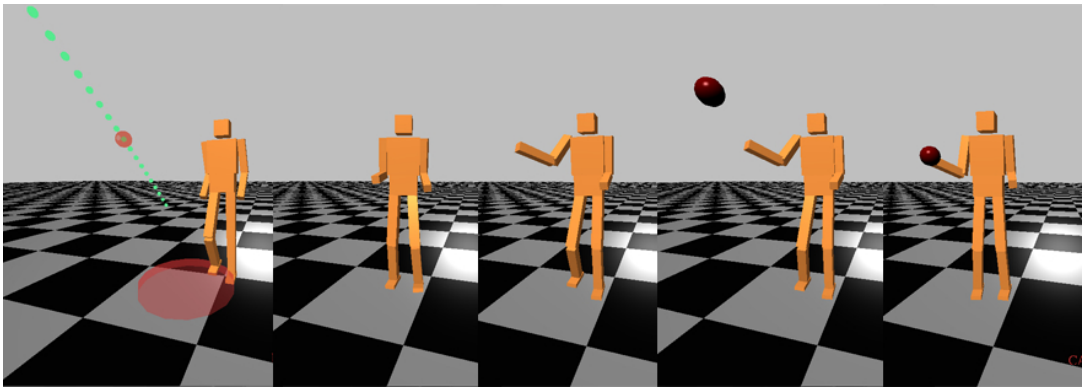

Figure 6.4: Simulation result of right-handed catching.

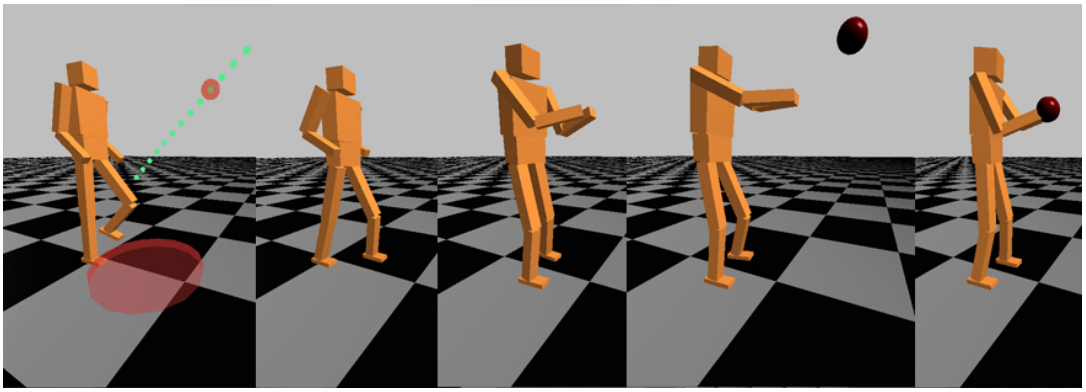

Figure 6.5: Simulation result of two-handed catching.

In the first sequence of actions, the character performs a left-handed catching. Figure shows the frames of the animation. In the first frame, the process of determining the interception hand position and the interception com position are demonstrated. The green dotted line shows the trajectory of the ball. The red point in the trajectory is the visual representation of the optimal interception hand position. The center of the red cylindrical region represents the interception COM position that the character should step into.

In all example animation results, each frame represents an illustration from ready phase, reaching phase, adapting phase and catching phase in the last frame, respectively. After the step of planning catching strategy, the character starts stepping into the interception COM position during the ready, reaching and adapting phase. If the character reaches to the interception COM position, it stops walking and switches to the standing behavior in the catching phase.

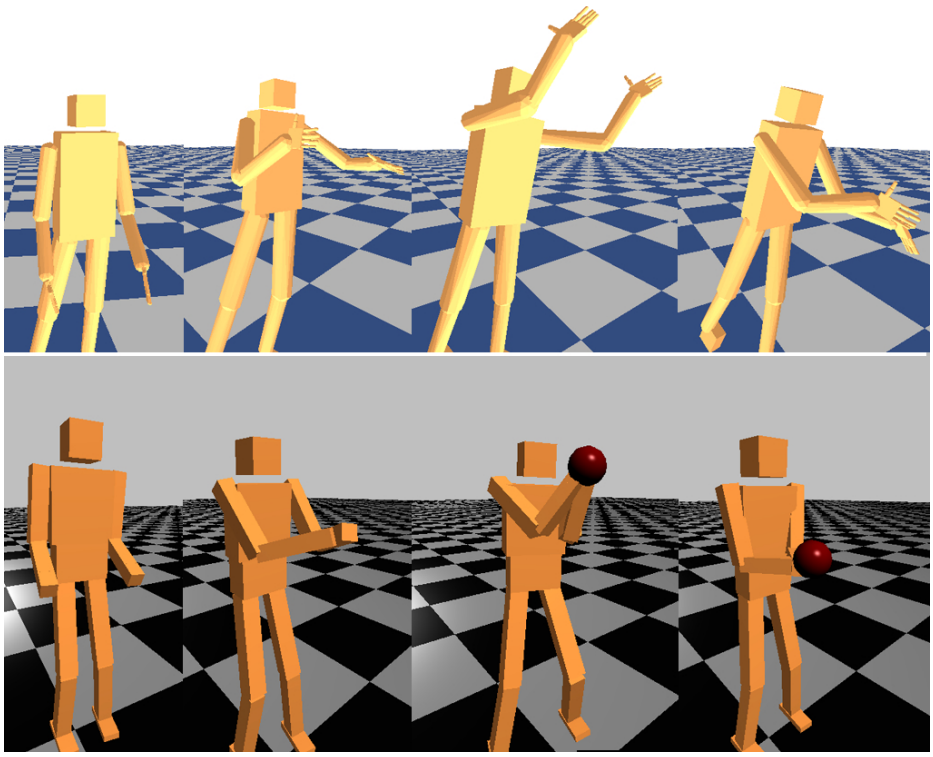

Figure 6.6: Our simulation results compared with a reference motion capture data.

We also compares our simulated motion with a reference motion capture data frame-by-frame. The catching simulation is based on the same catching strategy and the initial conditions as far as our guess from the data. Even though the two motions are not identical, results show that our simulation is qualitatively similar to the reference motion capture data (Figure 6.6).

Chapter 7

Conclusion

7.1 Conclusions

Due to the unstable nature of the physical characters, creating appropriate interactions with the environment, such as ball catching, is a challenging problem. In this thesis, we show that such a complex task can be achieved by dividing it into smaller phases and simpler task-oriented motor control models. We presented three different controllers including standing and walking controllers for stepping to a specific location and standing there, and reaching controller for reaching a specific target location with hands.

We show that properly planning the phases of the task and managing these high-level controllers according to the phases creates natural-looking and accurate ball catching animations. The usage of reinforcement learning for planning the transitions between the phases increases the accuracy in timing in catching while properly designing the control parameters for each specific catching phase increases the realistic looking in animations because these control parameters directly affect the inputs of the high-level controllers.

Simulation results indicate that the character can perform reasonable accurate and realistic catching of the randomly projected balls with different catching

strategies. The character appropriately controls the speed (timing) and plans the phases in order to reach on time to catch the ball accurately even if each thrown ball has different arrival flight time and height.

7.2 Discussion

The same approach used in this thesis can be applicable for physically simulated characters to perform different variety of high-level skills and tasks, such as throwing, striking, etc., where the skill requires controlling timing. This chapter outlines the important points of the thesis and suggests possible improvements.

Brief introduction in Chapter I, some important related works are mentioned in the fields of data-driven approaches, physics-based methods and reinforcement learning adapted works in Chapter II. In Chapter III, some high-level controllers with their own low-level modules for different purposes including standing, walking and reaching are presented, as the starting point of the thesis. In Chapter IV, ball catching problem is explained in detail and different catching phases (ready reaching, adapting and catching phase) are introduced developed from a phase analysis. The Chapter V presents the approach planning the catching phases using reinforcement learning in order to achieve proper timing in catching. In addition, some catching animation results with different catching strategies are demonstrated.

The proposed method has its own strengths and weaknesses. Some limitations of the system are given as follows:

(1) The reaching controller proposed here does not consider other human reaching strategies such as jumping for higher ball trajectories, bending for lower ball trajectories and tiptoeing, which requires raising over the tips of the feet. These different catching strategies can be achieved defining additional controllers.

(2) The phases defined for ball catching are task-specific so that different tasks,

such as striking, may require different phases and different control parameters. This is one of the biggest weakness of the approach.

(3) The control parameters specifically defined for each catching phase should be better optimized. The learning algorithm is so sensitive to the different control parameters that if they are not properly tuned, it affects the performance of the learning algorithm and the results of the training process.

7.3 Future Works

The work presented in this thesis provides different directions for future studies and it can be extended for different specific applications. We presented an approach that try to emulate the learning process in some sense that humans go through as they learn skills in a similar process.

As an extension, more complex policies can be learned for learning low-level control parameters in the high-level controllers by considering the strategies the humans use in their day-to-day activities and other biomechanical observations. To extend that even further, data from motion capture information can be integrated during the process of the learning process in order to improve the creation of stylistic human-like motions. In addition, there is a wide body of work studying the integration of biomechanical models and kinesiology investigating human reaching strategies. Finally, one future direction to pursue would be more exploiting these models for our control methods.

Bibliography

[1] R. Bartlett, *Introduction to Sports Biomechanics: Analysing Human Movement Patterns*. Routledge, 2007.

[2] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Trans. Graph.*, vol. 21, pp. 473–482, July 2002.

[3] A. Safonova and J. K. Hodgins, "Construction and optimal search of interpolated motion graphs," *ACM Trans. Graph.*, vol. 26, July 2007.

[4] R. Heck and M. Gleicher, "Parametric motion graphs," in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, (New York, NY, USA), pp. 129–136, ACM, 2007.

[5] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin, "Motion-motif graphs," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pp. 117–126, Eurographics Association, 2008.

[6] Y. Lee, S. J. Lee, and Z. Popović, "Compact character controllers," in *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, (New York, NY, USA), pp. 169:1–169:8, ACM, 2009.

[7] J. Chai and J. K. Hodgins, "Constraint-based motion optimization using a statistical dynamic model," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

[8] X. Wei, J. Min, and J. Chai, "Physically valid statistical models for human motion generation," *ACM Trans. Graph.*, vol. 30, pp. 19:1–19:10, May 2011.

[9] J. Min and J. Chai, "Motion graphs++: A compact generative model for semantic motion analysis and synthesis," *ACM Trans. Graph.*, vol. 31, pp. 153:1–153:12, Nov. 2012.

[10] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM Trans. Graph.*, vol. 23, pp. 514–521, Aug. 2004.

[11] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," *ACM Trans. Graph.*, vol. 23, pp. 522–531, Aug. 2004.

[12] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun, "Continuous character control with low-dimensional embeddings," *ACM Trans. Graph.*, vol. 31, pp. 28:1–28:10, July 2012.

[13] W. Huang, M. Kapadia, and D. Terzopoulos, "Full-body hybrid motor control for reaching," in *Proceedings of the Third International Conference on Motion in Games*, MIG'10, (Berlin, Heidelberg), pp. 36–47, Springer-Verlag, 2010.

[14] P. Lv, M. Zhang, M. Xu, H. Li, P. Zhu, and Z. Pan, "Biomechanics-based reaching optimization," *The Visual Computer*, vol. 27, no. 6-8, pp. 613–621, 2011.

[15] A. W. Feng, Y. Xu, and A. Shapiro, "An example-based motion synthesis technique for locomotion and object manipulation," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, (New York, NY, USA), pp. 95–102, ACM, 2012.

[16] S. Coros, P. Beaudoin, K. K. Yin, and M. van de Pann, "Synthesis of constrained walking skills," *ACM Trans. Graph.*, vol. 27, pp. 113:1–113:9, Dec. 2008.

[17] M. da Silva, Y. Abe, and J. Popović, "Interactive simulation of stylized human locomotion," *ACM Trans. Graph.*, vol. 27, pp. 82:1–82:10, Aug. 2008.

[18] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers," *ACM Trans. Graph.*, vol. 28, pp. 168:1–168:8, Dec. 2009.

[19] Y.-Y. Tsai, W.-C. Lin, K. B. Cheng, J. Lee, and T.-Y. Lee, "Real-time physics-based 3d biped character animation using an inverted pendulum model," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, pp. 325–337, Mar. 2010.

[20] K. Yin, K. Loken, and M. van de Panne, "Simbicon: Simple biped locomotion control," *ACM Trans. Graph.*, vol. 26, July 2007.

[21] S. Coros, P. Beaudoin, and M. van de Panne, "Generalized biped walking control," *ACM Trans. Graph.*, vol. 29, pp. 130:1–130:9, July 2010.

[22] T. Komura, H. Leung, and J. Kuffner, "Animating reactive motions for biped locomotion," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '04, (New York, NY, USA), pp. 32–40, ACM, 2004.

[23] T. Shiratori, B. Coley, R. Cham, and J. K. Hodgins, "Simulating balance recovery responses to trips based on biomechanical principles," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, (New York, NY, USA), pp. 37–46, ACM, 2009.

[24] B. Kenwright, "Generating responsive life-like biped characters," in *Proceedings of the The Third Workshop on Procedural Content Generation in Games*, PCG'12, (New York, NY, USA), pp. 1:1–1:8, ACM, 2012.

[25] N. H. Nguyen, R. Arista, C. K. Liu, and V. Zordan, "Adaptive dynamics with hybrid response," in *SIGGRAPH Asia 2012 Technical Briefs*, SA '12, (New York, NY, USA), pp. 5:1–5:4, ACM, 2012.

[26] S. Ha, Y. Ye, and C. K. Liu, "Falling and landing motion control for character animation," *ACM Trans. Graph.*, vol. 31, pp. 155:1–155:9, Nov. 2012.

[27] D. F. Brown, A. Macchietto, K. Yin, and V. Zordan, "Control of rotational dynamics for ground behaviors," in *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, (New York, NY, USA), pp. 55–61, ACM, 2013.

[28] V. B. Zordan and J. K. Hodgins, "Motion capture-driven simulations that hit and react," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, (New York, NY, USA), pp. 89–96, ACM, 2002.

[29] Y. Lee, S. Kim, and J. Lee, "Data-driven biped control," *ACM Trans. Graph.*, vol. 29, pp. 129:1–129:8, July 2010.

[30] L. Ikemoto, O. Arikan, and D. Forsyth, "Learning to move autonomously in a hostile world," in *ACM SIGGRAPH 2005 Sketches*, (New York, NY, USA), ACM, 2005.

[31] A. Treuille, Y. Lee, and Z. Popović, "Near-optimal character animation with continuous control," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

[32] W.-Y. Lo and M. Zwicker, "Real-time planning for parameterized human motion," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, (Aire-la-Ville, Switzerland, Switzerland), pp. 29–38, Eurographics Association, 2008.

[33] J. Lee and K. H. Lee, "Precomputing avatar behavior from human motion data," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, (Aire-la-Ville, Switzerland, Switzerland), pp. 79–87, Eurographics Association, 2004.

[34] S. Coros, P. Beaudoin, and M. van de Panne, "Robust task-based control policies for physics-based characters," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 170:1–170:9, 2009.

[35] R. Tedrake, T. Zhang, and H. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2849–2854 vol.3, Sept 2004.

[36] J. Morimoto, J. Nakanishi, G. Endo, G. Cheng, and P. Map, "Poincare-map-based reinforcement learning for biped walking," in *Proc. IEEE Int. Conf. Robotics and Automation, ICRA05*, pp. 2381–2386, 2005.

[37] Z. Wang, C. H. Lampert, K. Mlling, B. Schlkopf, and J. Peters, "Learning anticipation policies for robot table tennis.," in *IROS*, pp. 332–337, IEEE, 2011.

[38] K. Mlling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[39] K. Muelling, A. Boularias, B. Mohler, B. Schlkopf, and J. Peters, "Learning strategies in table tennis using inverse reinforcement learning," *Biological Cybernetics*, pp. 1–17, 2014.

[40] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Optimizing walking controllers for uncertain inputs and environments," *ACM Trans. Graph.*, vol. 29, pp. 73:1–73:8, July 2010.

[41] J. M. Wang, S. R. Hamner, S. L. Delp, and V. Koltun, "Optimizing locomotion controllers using biologically-based actuators and objectives.," *ACM Trans. Graph.*, vol. 31, no. 4, p. 25, 2012.

[42] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Trans. Graph.*, vol. 32, pp. 206:1–206:11, Nov. 2013.

[43] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne, "Locomotion skills for simulated quadrupeds," *ACM Trans. Graph.*, vol. 30, pp. 59:1–59:12, July 2011.

[44] J. Tan, Y. Gu, G. Turk, and C. K. Liu, "Articulated swimming creatures," *ACM Trans. Graph.*, vol. 30, pp. 58:1–58:12, July 2011.

[45] J. Tan, Y. Gu, G. Turk, and C. K. Liu, "Learning bicycle stunts," *ACM Trans. Graph.*, 2014.