A Study on Physics-Based Real-Time Human Animation

Masaki Oshita

February 2003

A Doctoral Dissertation

Department of Intelligent Systems
Graduate School of Information Science and Electrical Engineering
Kyushu University

Abstract

This dissertation presents novel methods for genetating realistic human animation in real-time especially for interactive applications, such as computer games, virtual studio and virtual reality. In such applications, the motions of virtual characters and surrounding objects (hair, cloth, properties, and so on) should be generated in real-time and in response to the user's input and interaction between the characters and objects.

The primal approach of this work is to combine kinematics and dynamics. By combining the useful aspects of both currently used kinematic methods and physics-based dynamic simulation, we realize the dynamic motion control that has both physically-soundness and controllability in real-time applications. In this dissertation, we introduce the results that the approach was applied to the motion control of human-like articulated figures and the motion generation of non-rigid cloth objects. These results show the power and effectiveness of the approach.

This dissertation consists of five chapters. Chapter 1 introduced the background and necessarily of real-time animation and describes the approach of this work. Chapter 2 and 3 presents a dynamic motion control technique. In chapter 2, a motion control method of human-like articulated figure is described. This method controls the angular accelerations of an articulated figure's joints and generates dynamic reactions of human figures in response to physical interactions such as collisions, external and gravity forces. Chapter 3 extends the motion control method presented in the chapter 2 to realize continuous human motions in interactive applications and introduces two novel kinematic methods: motion transition and reaction generation. Chapter 4 presents a method for fast and plausible cloth animation by combining a particle-based dynamic simulation and geometrical surface control methods. Finally, chapter 5 concludes this work and shows the future direction of the research in real-time computer animation.

Acknowledgements

I am deeply grateful to my supervisor, Prof. Akifumi Makinouchi who always gave me thoughtful advices and led me to the right direction in six years since I have joined his laboratory.

I would like to thank to Prof. Susumu Kuroki, Prof. Kunihiko Kaneko, Prof. Hirofumi Amano, Prof. Norihiko Yoshida, and past and present members of Makinouchi laboratory for supporting my work and giving useful comments.

I also would like to thank to committee members of my dissertation, Prof. Akifumi Makinouchi, Prof. Tsutomu Hasegawa, and Prof. Rin-ichiro Taniguchi for their insightful comments on this dissertation.

Finally, I thank to my family and friends for their expectation and encouragement. Without them, I couldn't finish this dissertation.

Table of Contents

Chapter 1: Introduction

1.1. Background

Recently use of computer animation has been spreading in many areas. Formerly computer animation techniques were mainly used for making commercial films, such as motion pictures, TV program and promotion films. Therefore, animation techniques have been mostly developed to support to create the motion data of virtual characters and objects in an off-line process and in cooperate with skilled animators.

However, lately real-time animation of human figures has been required in many interactive applications, such as computer games, virtual studio, avatar control in virtual environment and virtual reality. In these applications, the motions of virtual characters and surrounding objects (hair, cloth, properties, and so on) should be generated in real-time and in response to the user's input and interaction between the characters and the objects in the virtual scene.

Currently, in such applications, real-time animation is realized by sequentially replaying exiting motion data that are created and stored in advance. Using edited motion capture data or keyframed motion sequences that are created by a skilled animator, we can generate realistic human motions in even interactive applications. However, in this framework, characters and other objects can do nothing but just repeating a set of fixed motions. We can slightly modify the prepared motion data on the fly using some kinematic techniques such as inverse kinematics and motion warping. However, because these techniques are kinematics and physical factors such as collision, contact and gravity forces are not considered, they cannot produce realistic motions that involve physical interaction between characters, objects, and the environments. In spite of that these kinds of interactions are very important in interactive applications, there is no effective method has been developed. In order to handle such interactions well, we need some dynamic control methods to generate realistic and changing motions on the fly while

taking into account the physics of the human figure and objects.

Researchers also have been trying to use physics-based dynamic simulation for physically-correct animation. By modeling physical properties of human figures and objects and developing dynamic controllers to determine joint torques on each frame to drive the figurers, the resulting motions are numerically computed using dynamic simulation. However, this approach has a lot of problem and it is difficult to be employed in practice. First, it lacks the controllability. Because the motions are computed from forces and joint torques, it is difficult to predict the results and control them as the programmers and users intend. Moreover physics-based dynamic simulation sometimes consumes too much computational time to be used in real-time. Because of these problems, physics-based approaches are not adopted in current interactive applications.

As explained so far, despite of that real-time human animations are used in many interactive applications, most of them just ignore the physics of human figures and objects now. As result, they cannot handle the physical interaction between characters and environments and this makes resulting animation look artificial and very unnatural. To solve this problem, some break through has been expected.

1.2. Methodology

The primal approach of this work is to combine kinematics and dynamics. By combining the useful aspects of both currently used kinematics-based methods and physics-based dynamic simulation, we achieve dynamic motion control techniques for real-time applications that have both physically-soundness and controllability.

To apply this methodology in practical, we had to solve two major problems:

- To merge kinematics-based method and dynamics-based method seamless. Fundamentally, the two methods control totally different aspects of motions. Kinematics-based method control the results directory, while dynamics-based method controls the sources of the motions. We should find an appropriate point where dynamics is combined into kinematics.
- To build a simple dynamics model on each subject. Existing physical models are normally very complex and require a lot of computational times. Our purpose is computer animation and we do not need so much accuracy. From this viewpoint, we should extract the essence of existing physical model that seems to be important for generating physically-soundness.

1.3. Developed Methods

In this dissertation, we introduce some results that the above approach was applied to the motion control of human-like articulated figures and the motion generation of non-rigid cloth objects. These results show the power and effectiveness of our approach.

Motion control techniques in human animation are categorized into primary and secondary motions. Primary motion means the motions of human figures. Human figure is usually modeled as an articulated figure, and their motions are controlled as time-varying rotations of their joints. Secondary motion means the motions of surrounding objects such as hairs, clothes, human skins, properties, and so on. The secondary motions are also very important to make the resulting animations look natural. Even if human motions were very life like, if its hairs and clothes were fixed as in most of current computer games, it gaves very unnatural impression to the users. Therefore, both the primary and secondary motions are important in human animation. From this viewpoint, we have tackled major subjects of both fields and tried to apply the above approach to the following subjects:

- Dynamic motion control of human figure in response to environmental physical interaction (as a primary motion).
- Real-time cloth simulation that has plausible appearance of clothes (as a secondary motion).

1.4. Contributions

The main contributions of this work in computer animation literature are

- We have proposed a methodology that combines kinematics and dynamics, and pointed two issues as in section 1.2.
- We have applied the methodology on two major subjects from primary and secondary motions, and developed effective methods to show the power of the methodology.

We believe that this methodology is very powerful and it also expected to be applied to other subjects of human animations such as human locomotion control, hair simulation, skin deformations and so on.

1.5. Organization

This dissertation consists of five chapters. Chapter 2 and 3 presents a dynamic motion control technique. In chapter 2, a motion control method of human-like articulated figure is described. This method controls the angular accelerations of an articulated figure's joints and generates dynamic reactions of human figures in response to physical interactions such as collisions, external and gravity forces. Chapter 3 extends the motion control method presented in the chapter 2 to realize continuous human motions in interactive applications and introduces two novel kinematic methods: motion transition and reaction generation. Chapter 4 presents a method for fast and plausible cloth animation by combining a particle-based dynamic simulation and geometrical surface control methods. Finally, chapter 5 concludes this work and shows the future direction of the research in real-time computer animation.

Chapter 2: Dynamic Motion Control of Human Figure

This chapter presents a dynamic motion control technique for human-like articulated figures. This method controls the joints of a human figure such that the figure tracks an input motion data specified by a user. When environmental physical input such as an external force or a collision impulse are applied to the figure, this method generates dynamically changing motion in response to the physical input. We have introduced comfort and balance control to compute the angular acceleration of the figure's joints. This algorithm controls the several parts of a human-like articulated figure separately through the minimum number of degrees-of-freedom. Using this approach, our algorithm simulates realistic human motions at efficient computational cost. Unlike existing dynamic simulation systems, our method assumes that input motion is already realistic, and is aimed at dynamically changing the input motion in real-time only when unexpected physical input is applied to the figure. As such, our method works efficiently in the framework that is used in current applications.

2.1. Introduction

Generating realistic character animation is a difficult challenge. Recently, many online applications such as computer games and virtual environments require the generation of realistic and continuous character animation in real-time. Currently, such animations are generated by dynamically composing motion sequences such as motion capture or keyframed motion data. These motion sequences need to be created in advance. Therefore, it is difficult to produce dynamically changing motion that respond to physical input from the environment, such as the gravitational force when carrying a heavy load, an external force, or a collision impulse from other objects. This kind of interaction between a character and the environment are frequent and important events in computer games. Nevertheless, very few methods have

been developed for dynamic motion control in such situations. This is one of the most important issues in real-time character animation.

In this chapter, we present a dynamic motion control technique for human-like articulated figures. This method controls a character based on input motion specified by a user, and environmental physical input in a physics based character animation system. In the system, the angular acceleration of character's joints are controlled so as to track the user-input motion. Dynamic simulation then generates the resulting animation. When environmental physical input is applied to the character, the dynamic motion control computes the angular joint accelerations in order to produce dynamically changing motion in response to the physical input. We introduce two kinds of dynamic control: comfort and balance control. Under comfort control, when a torque on a joint exceeds the available muscle strength of the joint, the angular joint accelerations are controlled so as to reduce the joint stress based on the moment of inertia. Under balance control, when the character is likely to lose balance, the angular joint accelerations are controlled so as to maintain balance. This approach produces human-like dynamic motion control, such as reducing the stress on the back by swing the arms and maintaining balance by moving the pelvis, when the character carries a heavy load or collides with other objects. This dynamic motion control method is specific to human-like articulated figures, controlling the arms, back and legs separately in order of importance. Each part is controlled through the minimum number of degrees-of-freedom (DOF). A number of minor factors are ignored in this method and the resulting motion is not perfectly physically correct. However, our method makes it possible to simulate realistic human motions at lower computational cost because the method does not include heuristics. The goal of our method was not to establish a stable control method but to produce realistic character reactions in response to physical interactions.

A number of techniques have been developed for generating character animation in real-time using dynamic simulation. However, most of these methods are aimed at generating physically correct motion from unnatural input motion such as specified keyframes and monotonous procedural motion. Because these methods cannot utilize existing realistic motion sequences such as motion capture data, they are not used in many applications. Our method assumes that input motion is already realistic, and is aimed at dynamically changing the input motion only when unexpected physical input is applied to the figure from the environment. As such, our method works efficiently in the framework of current computer games and other online applications.

The reminder of this chapter is organized as follows: Section 2.2 reviews related work and issues relevant to solve our problem. Section 2.3 describes the structure of proposed system and its components. Section 2.4 presents a simple tracking control algorithm to track an input motion directly. Based on the tracking controller, section 2.5 then introduces a dynamic control algorithm for comfort and balance control. In section 2.6, an experimental result is presented, and section 2.7 concludes this work and outlines future research.

2.2. Related Work

There are two main approaches for generating or editing physically correct motion based on dynamics: spacetime constraints and dynamic simulation. In addition, there are motion control techniques using dynamics for specific task.

2.2.1. Spacetime Constraints

The spacetime constraints approach is a popular technique to generate a motion trajectory while ensuring controllability and physically realism. Applying this technique to human animation, an input motion is optimized so that it minimizes an objective function. Researchers have proposed some objective functions such as minimizing joint torques [32], muscle forces and the reactive torque from the ground [21], balance error [37].

In the spacetime constraints approach, character motion is controlled in angular space. Therefore, it is difficult to realize motion that interacts dynamically with the environment. During static motion, joint stress and balance depend on primarily joint angles. However, during a dynamic motion in which the figure moves quickly, especially the effect of the moment of inertia and angular accelerations should also be considered. Motion optimization techniques are suitable for motion planning before the motion is executed, but are not suitable for dynamic control during dynamic motion.

Recently, Popović and Witkin [31] proposed a motion transformation technique based on a spacetime constraint approach and dynamics. This method extracts the essential physical characteristics from an original motion for a simplified model using spacetime constraints, modifies the extracted dynamics and then reconstructs the resulting motion for the original articulated figure. This technique has a similarity with our dynamic control method with both using a simplified human structure. However, the purpose of their method is to optimize the

spacetime constraints and not to realize human-like dynamic control, for which we use the simplified human structure. Moreover, they do not model the character's skeleton or strength.

2.2.2. Dynamic Simulation and Controllers

The combination of dynamic simulation and a controller is a popular technique for generating physically correct human motions. Researchers have developed dynamic controllers for specific character skeletons and for behavior such as for walking [43] and athletic movements [17]. These controllers consist of proportional-derivative (PD) servos and state machines. The state machine determines next desired state. The PD controller determines the output torque in proportion to the difference between the desired state $\theta_d, \dot{\theta}_d$ and the current state $\theta, \dot{\theta}$ (angles and angular velocities, respectively) for each joint according to

$$\tau = k_p\left(\theta_d - \theta\right) + k_v\left(\dot{\theta}_d - \dot{\theta}\right). \tag{2.1}$$

The controller does not take into account the dynamic characteristics of the system. Therefore, to produce motion that is both stable and natural, the state machine and the gain parameters k_p, k_v of all joints need to be tuned empirically for both the specific character and the specific kind of motion. Although parameter optimization [43] and transformation [14] techniques have been proposed, it is still difficult to construct a successfully working controller, or to adapt an existing controller to another character and another motion.

The controllers compute the output torque of each joint separately based on the angle and angular velocity of the joint. However, the output torque of one joint influences the angular acceleration of all joints. Therefore, to realize human-like active controls such as balance control or reducing stresses, multiple joints should be cooperatively controlled. The dynamic simulation and controller approach ensures physically correctness. However, to control characters in a framework, very sophisticated algorithms that could even control real robots are required. Although research in the robotics field has significantly progressed in recent years, it is still difficult to make robotic athletic movements in a human way. The dynamic simulation approach is suitable for computing passive motions based on physics, such as falling from a high place. However, it is difficult to simulate active motions in response to environmental physical interaction.

Recently, Faloutsos et al. [7] have proposed a framework for composing different controllers and determining the transition between the controllers. They implemented various everyday actions and dynamic reactions in their framework. Since their system also uses PD servos and

sate machines, controllers should be designed for each specific motion and the difficult to realize human-like active control.

Some researchers proposed more general controllers for tracking motion capture data. Zordan and Hodgins [49] used PD controllers with a parameter optimization technique for tracking human upper-body motion. Kokkevis et al. [20] introduced model reference adaptive control (MRAC) as a replacement for PD control. The aim of their work is to make use of existing motion capture data and to generate dynamic motions by considering the physics. Their motivation is similar to our work, but they still use simple control algorithms and the torque of each joint in the upper body is controlled separately. Therefore, human-like active control during tracking motions cannot be realized.

2.2.3. Dynamic Motion Control

A number of techniques have been developed for generating dynamically controlled motion based on dynamics for a particular kind of task.

Some researchers have introduced dynamics into an inverse kinematics method. The traditional inverse kinematics technique controls the joints of an articulated figure based on the trajectory of end-effectors (hands, foot, etc.). Lee et al. [22] introduced a muscle strength model into the inverse kinematics method, modifying the trajectory of an end-effecter and motion speed based on the muscle strength of the joints. Boulic et al. [1] developed the inverse kinetics method to control the trajectory of the center of mass of an articulated figure while controlling the trajectory of the end-effectors. These methods make it possible to create motion that includes comfort and balance control. However, they are unable to handle the change of velocity of an articulated figure due to a collision impulse, nor can they make use of existing motion data.

Ko and Badler [19] developed a real-time animation system that produces a human walking motion with balance and comfort control using inverse dynamics. The system moves the positions of the pelvis and torso during the procedurally generated walking motion, and controls walking speed in response to the joint torques. Their work has similarity with ours with it generates changing animation based on existing motion data and dynamics computed during the motion. However, in their work, the computation of displacement does not include dynamics, and remains dependent on empirically tuned parameters for a specific walking motion. Furthermore, since direct modification in angular apace does not ensure continuous motion, the method is unable to handle environmental interactions such as collisions.

2.3. System Structure

The structure of the animation system presented in this chapter is shown in Figure 2.1. The system consists of two main module: a controller and a simulator. At each simulation step, the controller computes the angular joint acceleration of the figure, based on the current state of the figure and an input motion that is given by a user. The simulator then updates the state of the figure through dynamic simulation. A human body model and external physical input are considered in both the controller and the simulator.

Unlike standard controllers [14][17][50][20] control joint torque, our controller controls angular joint acceleration directly. No forward dynamics are used in our system. Instead, inverse dynamics is used in the controller to take into account the torque required to realize a given angular acceleration.

The algorithm for the controller is presented in detail in section 2.4 and 2.5. The remainder of this section explains the other components in the system.

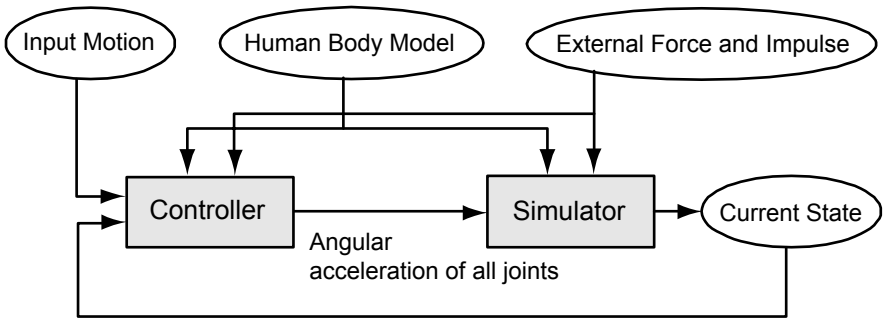

Figure 2.1 System structure.

2.3.1. Human Body Model

The human body model is considered as an articulated figure, which is a common representation in character animation. The articulated figure consists of segments and joints. Each rigid segment is connected by one, two, or three rotational joints. For example, the shoulder has three joints and the elbow has one. Based on this skeleton model, the configuration of a figure is represented by the set of angles of all joints and the position and orientation of the root segment. In this work, we use the pelvis segment as the root segment of a human figure. In addition, each segment has physical properties relevant to dynamic

simulation, such as mass and moment of inertia. These properties are calculated from the polygonal geometry of each segment using an integral calculation [26]. The polygonal geometries also are used for collision detection and for computing the contact surface between the segment and the ground. For our experiments, we use a skeleton model that has 18 segments and 39 joints (Figure 2.2).

The dynamic controller uses the available muscle strength of each joint as the criterion for comfort control. We adopt a simple muscle strength model [19][22] in which two muscle strength functions: the maximum and minimum available torque, are assigned to each joint.

$$\tau_{\max} = f_{\max}\left(\theta,\dot{\theta}\right),\ \tau_{\min} = f_{\min}\left(\theta,\dot{\theta}\right) \tag{2.2}$$

Pandya et al. [30] showed by collecting human strength data that these values can be approximated by functions of the joint angle and angular velocity. We assigned approximated strength functions to each joint, taking into account references including muscle strength data [30][22].

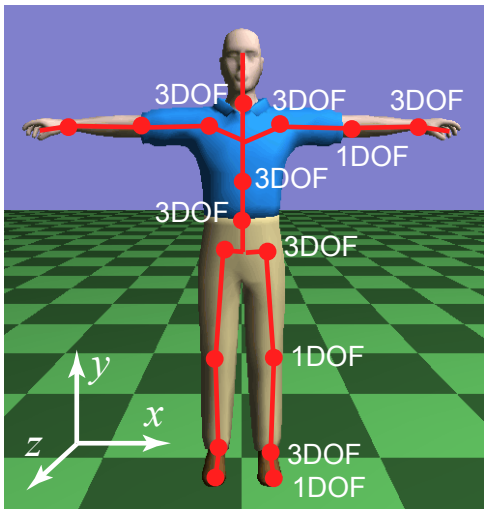

Figure 2.2 Human skeleton model.

2.3.2. Motion Data

Desired motion is specified in terms of the displacements of the configuration of a figure over time. Therefore, motion data are expressed as the angular trajectories of all joints and the spatial and orientational trajectory of the root segment. In addition, while a foot is in contact with the ground, the joints of the leg is controlled such that the foot is held in the same position (as explained in section 2.4.2). Therefore, the time when each foot lands on the ground and leaves again should also be indicated.

As input motion is represented kinematically, any form of motion capture data or keyframe motion sequence can be used as an input to our system.

2.3.3. Dynamic Simulation

Given the angular accelerations of all joints, the simulator updates the angles and angular velocities of all figures by Eular integration [1]. In addition to the angular acceleration of the joints, the rotational acceleration of the supporting segment of the figure (e.g. foot) is computed based on the angular joint acceleration, simulating falling motion. The segment upon which the moment of the center of mass of the figure is maximum is chosen as the supporting segment. To compute the rotational acceleration of a supporting segment, we use the zero moment point (ZMP) and minimum moment point (MMP). The details of the concept of the ZMP are explained in [37]. The ZMP is the point where the torque exerted by the figure on the ground is zero. When ZMP is within the support area (Figure 2.3 (a)), the figure is balanced and there is no rotational acceleration of supporting segment. Otherwise, rotational acceleration occurs around the MMP where the exerted torque is minimum. The MMP is the closest point from the ZMP within the support area (Figure 2.3 (b)). The support area is the convex hull of contact surfaces between the foot segments and the ground. The convex hull is computed from the vertices of contact faces in the foot segments [29]. The rotational acceleration of the supporting segment is computed from the torque exerted on the MMP and the moment of inertia of the whole body in that configuration.

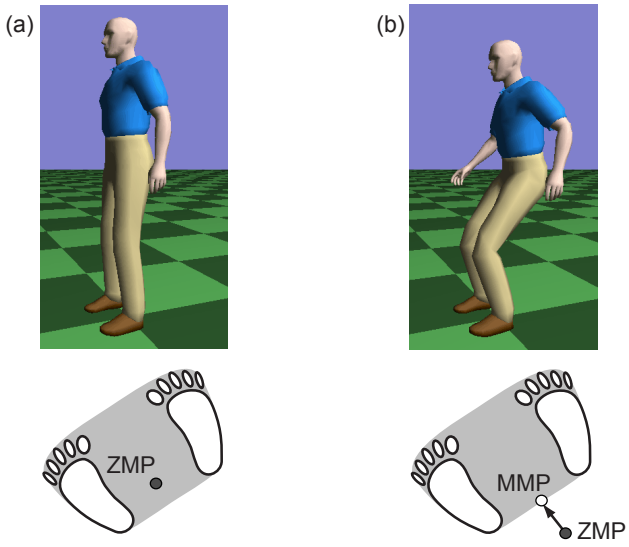

Figure 2.3 Zero-Moment Point in (a) balanced and (b) unbalanced state.

After the integration, collision detection and response are performed. When two figures collide, an impact force is imparted on each and their velocities change. The velocity changes are computed by solving the linear equation [27][20]. If the figures remain in contact, a reaction force acts between them. Reaction forces and other external forces are considered in the inverse dynamics component of the dynamic controller.

2.4. Tracking Control

This section presents the algorithm used to compute the angular acceleration of all joints so as to track a desired input motion, based on the current state of the figure and the desired motion. This algorithm controls joint angular acceleration directly rather than via joint torque, which is the case in standard dynamic simulation systems. As result, the desired motion is almost exactly tracked. However, unlike standard animation and game systems in which the joint angles of a figure are directly controlled according to a desired motion trajectory, our tracking controller produces continuous motion that approaches the desired motion even when the velocity of the figure is changed through a collision. In addition, when a figure loses its balance, a falling motion is generated as explained in section 3.3. The algorithm presented here is a simple and direct tracking control. A more advanced dynamic control for realizing human-like movements is presented in the next section as an extension of this tracking control system. This tracking control also can be used alone, if a user requires only continuous motion and lower computational cost. This tracking control scheme does not require dynamics computations or muscle strength model, making it easily to implement, with low computational cost.

2.4.1. Angular Acceleration of Each Joint

The angular acceleration for each joint is computed based on the figure's current state (joint angle and angular velocity), and the angular trajectory of the desired motion. As reviewed in section 2.2.2, PD control servos are widely used for this purpose in existing dynamic simulation systems [16][43]. Using a similar approach to PD controller, the output angular acceleration can be computed as follows:

$$\ddot{\theta} = k_p\left(\theta_d - \theta\right) + k_v\left(\dot{\theta}_d - \dot{\theta}\right), \tag{2.3}$$

where $\left(\theta, \dot{\theta}\right)$ is the current joint angle and angular velocity, $\left(\theta_d, \dot{\theta}_d\right)$ is the desired state obtained from a desired joint angular trajectory after Δt, and k_p, k_v are the gain parameters.

The parameters need to be tuned for each joint and each motion. Therefore, this makes it difficult to construct a general controller by this approach. In addition, to realize a stable control, a controller should take into account not only one state in the desired angular trajectory after Δt, but also the entire trajectory.

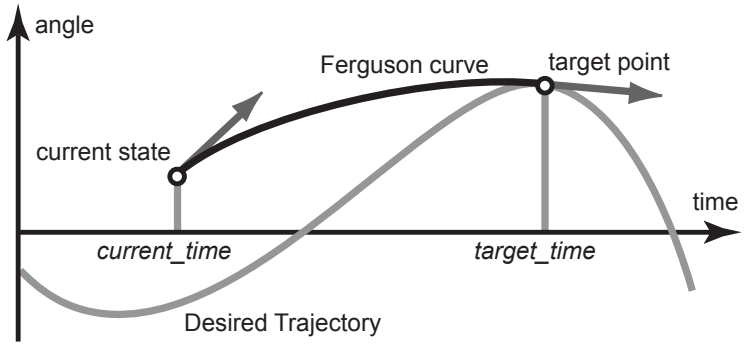

Figure 2.4 Tracking control using Ferguson curve.

Therefore, we have decided to use a Ferguson curve to compute output angular acceleration. A Ferguson curve is a kind of interpolation curve, such as a B-Spline or Bezier curve. However, while other spline curves are defined by a set of $(time, value)$ at knot points, a Ferguson curve is defined by $(time, value, derivative)$ at knot points. This feature makes a Ferguson curve suitable for use in our method because the current and desired state of a joint is defined by the joint angle and angular velocity. To compute the output angular acceleration, this method first determines the target point for which the motion is calculated to approach, taken as the closest extremity point to the desired angular trajectory. The projected trajectory, from the current state to the state approaching the desired position of the target point, is approximated by a Ferguson curve (Figure 2.4), as follows:

$$q(s) = \left(2s^3 - 3s^2 + 1\right)\theta + \left(-2s^3 + 3s^2\right)\theta_t + \left(s^3 - 2s^2 + s\right)\dot{\theta} + \left(s^3 - 2s^2 + s\right)\dot{\theta}_t, \tag{2.4}$$

$$T = target\_time - current\_time,\ s = \frac{t - current\_time}{T}, \tag{2.5}$$

where $\left(\theta_t, \dot{\theta}_t\right)$ is the desired state of the target point in the desired angular trajectory. By taking the second derivative of the trajectory and letting $s = 0$, the output angular acceleration can be determined, written as

$$\ddot{\theta} = \left(6\theta_t - 6\theta - 2\dot{\theta}_t - 4\dot{\theta}\right)/T^2. \tag{2.6}$$

While a target point is fixed, the output angular acceleration is continuous. At the target time,

the output angular acceleration becomes discontinuous, as the target point becomes the next extremity point in the desired trajectory. However, the effect on the output angular acceleration is minimal as long as the current state is close to the desired trajectory. In addition, an instant of discontinuity has little influence on the motion trajectory in angular space. Therefore, realistic continuous motion is always realized.

2.4.2. Angular Acceleration of Limb Joints

When several limbs (arms and legs) of the figure are constrained, all joints in the limbs should be controlled cooperatively. For example, during a double support phase, the joints in both legs should be controlled such that neither foot leaves the ground, and if a figure is holding a ladder with the right arm and right leg, all joints in both limbs should be controlled cooperatively.

We use a human body model in which each limb has 7 DOF (Figure 2.2). The angles of the 7 joints are determined from the position and orientation of the pelvis $(\mathbf{p},\mathbf{o})$ (6 DOF) and the swivel angle of the knee around the vector from the hip joint to the ankle joint s (1 DOF), by analytical inverse kinematics [39][25]. The tracking algorithm determines the spatial and rotational accelerations of the root segment $(\ddot{\mathbf{p}},\ddot{\mathbf{o}})$ and the swivel angular acceleration $\ddot{s}$ for constrained limbs using the tracking algorithm presented in section 2.4.1. The angular accelerations of the joints of the constrained limbs are computed using an analytical inverse kinematics method in the same way as inverse kinematics was used for joint angles.

The inverse kinematics algorithm for angular accelerations is easily derived from the inverse kinematics method for angles [39][25].

2.5. Dynamic Control

This section introduces a dynamic control method to compute an output angular acceleration in response to physical input from the environment. The angular acceleration computed by the tracking control algorithm in the previous section is used as the initial angular acceleration. The output angular acceleration is defined as the sum of the initial acceleration $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$ and the difference of the angular acceleration $\Delta\ddot{\boldsymbol{\theta}}_{\text{dynamic}}$ in dynamic motion control as follows:

$$\ddot{\boldsymbol{\theta}}_{\text{output}} = \ddot{\boldsymbol{\theta}}_{\text{tracking}} + \Delta\ddot{\boldsymbol{\theta}}_{\text{dynamic}}. \tag{2.7}$$

Here, $\ddot{\boldsymbol{\theta}}_{\text{output}}, \ddot{\boldsymbol{\theta}}_{\text{tracking}}, \Delta\ddot{\boldsymbol{\theta}}_{\text{dynamic}}$ are n-dimensional vectors, where n is the total number of joints. Each

row of the vectors corresponds to a single joint. Comfort and balance control are used in our method to realize dynamic motion control. Under comfort control, when a torque exerted on a joint exceeds the available muscle strength of the joint, joint angular accelerations are controlled so as to reduce the joint stress based on the moment of inertia. Under balance control, when a character is likely to lose balance, angular joint acceleration is controlled so as to maintain balance. When the joint torque is within the available muscle range for all joints and the body balance is maintained on $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$, no dynamic control processing is performed and the controller outputs the initial acceleration $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$ as the output acceleration. In this way, motion close to the desired motion trajectory is realized.

2.5.1. Control Foundation

The criteria for comfort and balance control are introduced here in terms of the dynamics of articulated figures.

The criteria and the use of comfort and balance control is not novel work. The novel part of our work is the dynamic control algorithm that controls angular joint acceleration. Here, the relationship between the criteria and the angular acceleration of a joint is derived for the dynamic control algorithm that is described in section 2.5.3.

Comfort Control

Joint torque that exceeds available muscle strength is considered as the criterion for comfort control. The joint torques $\boldsymbol{\tau}$ required to produce the joint acceleration $\ddot{\boldsymbol{\theta}}$ is computed by an inverse dynamics method, expressed as

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) + \mathbf{F}(\boldsymbol{\theta}), \tag{2.8}$$

where $\mathbf{H}(\boldsymbol{\theta})$ is the moment of inertia, and $\mathbf{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})$, $\mathbf{G}(\boldsymbol{\theta})$ and $\mathbf{F}(\boldsymbol{\theta})$ are the influences on torque due to coriolis and centrifugal forces, gravity, and external force, respectively. The dimension of all vectors is n, where n is the number of joints of the figure. For the inverse dynamics, we use the Newton-Eular method [13]. During a double support phase, an approximation [19][21] is used to determine the forces applied from the upper body to each leg. The required torque of all joints is computed in $O(n)$. The available torque of the i-th joint is given by equation (2.2).

The required change in joint torque to satisfy the muscle strength constraint is computed for each joint by the following equation:

$$\boldsymbol{\tau}_{\text{stress},i} = \begin{cases} \boldsymbol{\tau}_i - \boldsymbol{\tau}_{\max,i} & \text{if } \boldsymbol{\tau}_i > \boldsymbol{\tau}_{\max,i} \\ \boldsymbol{\tau}_i - \boldsymbol{\tau}_{\min,i} & \text{if } \boldsymbol{\tau}_i < \boldsymbol{\tau}_{\min,i} \\ 0 & \text{if } \boldsymbol{\tau}_{\min,i} < \boldsymbol{\tau}_i < \boldsymbol{\tau}_{\max,i} \end{cases} \tag{2.9}$$

Comfort control is performed so as to minimize $\boldsymbol{\tau}_{\text{stress},i}$ for all joints.

The relationship between the required change in joint torque $\Delta\boldsymbol{\tau}$ and the corresponding change in angular joint acceleration can be derived from equation (7). The relationship is dependent on the moment of inertia, as follows:

$$\Delta\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{\theta})\Delta\ddot{\boldsymbol{\theta}}. \tag{2.10}$$

Each column of the matrix $\mathbf{H}(\boldsymbol{\theta})$ is computed solely from the current angles in $O(n)$ [44]. Comfort control is performed based on the derivation of the joint torque in equation (2.10).

Balance Control

The zero moment point (ZMP) and minimum moment point (MMP), explained in section 3.3, are used as the criterion for balance control. The position of the ZMP is computed from the spatial accelerations of all segments on the assumption that the ground is defined as $ZMP_y = 0$ according to the following equations [37]:

$$ZMP_x = \frac{\sum m_i x_i (\ddot{y}_i - g) - \sum m_i y_i \ddot{x}_i}{\sum m_i (\ddot{y}_i - g)}, \tag{2.11}$$

$$ZMP_z = \frac{\sum m_i z_i (\ddot{y}_i - g) - \sum m_i y_i \ddot{z}_i}{\sum m_i (\ddot{y}_i - g)}, \tag{2.12}$$

where m_i is the mass of the i-th segment, (x_i, y_i, z_i) is the position of the i-th segment, and $(\ddot{x}_i, \ddot{y}_i, \ddot{z}_i)$ is the spatial acceleration of the i-th segment. As the spatial accelerations of segments are computed from the angular acceleration of joints, the position of the ZMP is represented as a function of angular joint acceleration. When ZMP is outside the support area, the MMP becomes the closest point from the ZMP within the support area. Balance control is performed to move the ZMP to the MMP (Figure Figure 2.3(b)).

The relationship between the position of the ZMP and the angular acceleration of a joint can be derived from equations (11) and (12) by considering the movements of an augmented body [4], defined as the imaginary rigid body supported by a single joint, consisting of all segments from the joint to the end-effecters. The relationship is shown in Figure 2.5, where M is the mass of the augmented body, l is the vector from the joint to the center of mass of the augmented body, r

is the rotational axis of the joint, and a is the spatial acceleration of the augmented body. Using these variables, the spatial derivation of the ZMP can be computed by

$$\frac{\delta ZMP}{\delta \ddot{\boldsymbol{\theta}}_i} = \frac{M}{\sum m_i (\ddot{y}_i - g)} \left\{ (p_x - ZMP_x) a_y - p_y a_x \right\}, \tag{2.13}$$

$$\frac{\delta ZMP}{\delta \ddot{\boldsymbol{\theta}}_i} = \frac{M}{\sum m_i (\ddot{y}_i - g)} \left\{ (p_x - ZMP_x) a_y - p_y a_x \right\}. \tag{2.14}$$

Balance control is performed based on the spatial derivation of the ZMP in equations (2.13) and (2.14).

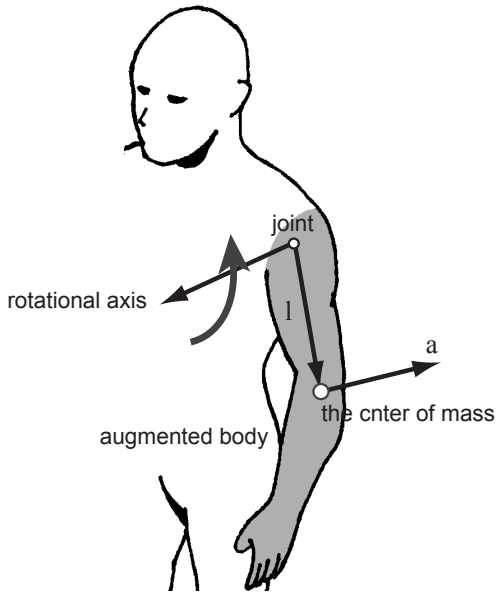

Figure 2.5 Velocity of ZMP from rotation of augmented body.

2.5.2. Control Strategy

One simple approach for computing $\Delta\ddot{\boldsymbol{\theta}}_{\text{dynamic}}$ is to solve an optimization problem so as to minimize an objective function such as

$$f\left(\Delta\ddot{\boldsymbol{\theta}}\right) = \left|\boldsymbol{\tau}_{\text{stress}}\right| + \left|ZMP - MMP\right| + \left|\Delta\ddot{\boldsymbol{\theta}}\right|. \tag{2.15}$$

However, solving the optimization problem requires significant computational time because this equation controls a large number of DOF. Although the optimization approach is a good strategy for generating stable human motion [21][32][37], it is difficult to handle dynamic reactions respond unexpected environmental input as sated in section 2.2.1. Moreover, this approach does not reflect motion control based on human experience.

Instead of that, we have developed a heuristic method to compute $\Delta\ddot{\boldsymbol{\theta}}_{\text{dynamic}}$ based on the observation of human movement. The method is specialized for human-like figures in a standing double-support phase.

The features of this algorithm is follows:

- Two kinds of control: active and passive control are employed.
- Each body part is controlled through a minimum number of DOF in active control.
- Each control step is performed based on a heuristic order.

In the reminder of this subsection, we explain the heuristic in detail.

Active and Passive Control

First, we categorize dynamic motion control into two types: active and passive control. Active control is aggressive movements in which some available joints are moved to reduce the stresses on other joints or to maintain its balance. In active control, we control a small number of selected primary joints in order to realize active control in the way that actual humans do in a lower computational cost. On the other hand, passive control is enforced movements by joint stresses. In passive control, joints under high stress are controlled so as to reduce their own stress.

For example, if a figure has a heavy load in the right hand, active control moves other parts to assist the motion of the right arm by reducing the stress on the right arm, while passive control moves the joints in the right arm based on joint stress.

Control of Each Body Part

In active control, the human figure is controlled through three parts: the arms, back and legs (Figure 2.6(a)). We choose primary DOF for the each part in order to control them efficiently.

The arms are controlled through the two angular joint accelerations for each shoulder joint $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ (4 DOF) (Figure 2.6(b)). The rotational acceleration of each shoulder around the x-axis (lateral axis) and z-axis (front and back axis) are controlled. The rotational acceleration around the y-axis is not used here because the influence of the motion component on other joints is smaller than that of the other axes in terms of dynamics. If the stress around the y-axis (vertical axis) exerted on a joint, the stress is reduced by swinging both arms around the x-axis in opposite direction. This means that both arms should be controlled cooperatively. The back is

controlled through the three angular accelerations of the back joint $\Delta\ddot{\boldsymbol{\theta}}_{\text{back}}$ (3 DOF) (Figure 2.6(c)). The legs are controlled through the spatial acceleration of the pelvis segment $\Delta\ddot{\boldsymbol{\theta}}_{\text{legs}}$ (3 DOF) (Figure 2.6(d)), because in this case the legs should be controlled cooperatively so as to satisfy the constraints of both feet, as explained in section 4.2.

For the lower body (legs), active and passive control is computed at the same time through $\Delta\ddot{\boldsymbol{\theta}}_{\text{legs}}$. For the upper body (arms, back), the angular acceleration $\Delta\ddot{\boldsymbol{\theta}}_{\text{stress}}$ (k DOF) is computed for the passive control of k joints with stress exceeding the available muscle strength. The number k depends on the initial torque that is required to generate the initial acceleration $\Delta\ddot{\boldsymbol{\theta}}_{\text{tracking}}$.

The difference of the angular acceleration $\Delta\ddot{\boldsymbol{\theta}}_{\text{dynamic}}$ (n DOF) in equation (2.7) is computed by

$$\Delta\ddot{\boldsymbol{\theta}} = \mathbf{S}_{\text{a}}\ddot{\boldsymbol{\theta}}_{\text{arms}} + \mathbf{S}_{\text{b}}\ddot{\boldsymbol{\theta}}_{\text{back}} + \mathbf{S}_{\text{s}}\ddot{\boldsymbol{\theta}}_{\text{stress}} + \mathbf{J}_{\text{p}}\ddot{\mathbf{p}}_{\text{pelvis}}, \tag{2.16}$$

where $\mathbf{S}_{\text{a}}$, $\mathbf{S}_{\text{b}}$ and $\mathbf{S}_{\text{s}}$ are the selection matrices that map each controlled joint to the corresponding joint in the entire body joints $\Delta\ddot{\boldsymbol{\theta}}$ (n DOF). For example, each element $\ddot{\boldsymbol{\theta}}_{\text{back}}$ are mapped to the joint in the back in $\Delta\ddot{\boldsymbol{\theta}}$. $\mathbf{J}_{\text{p}}$ is the Jacobian matrix ($n \times 3$) that maps the controlled special acceleration to the displacement of all joints in the lower body, computed by inverse kinematics. In equation (2.16), $\mathbf{S}_{\text{a}}$ and $\mathbf{S}_{\text{b}}$ are fixed, while $\mathbf{S}_{\text{s}}$ and $\mathbf{J}_{\text{p}}$ are dynamically computed on each frame.

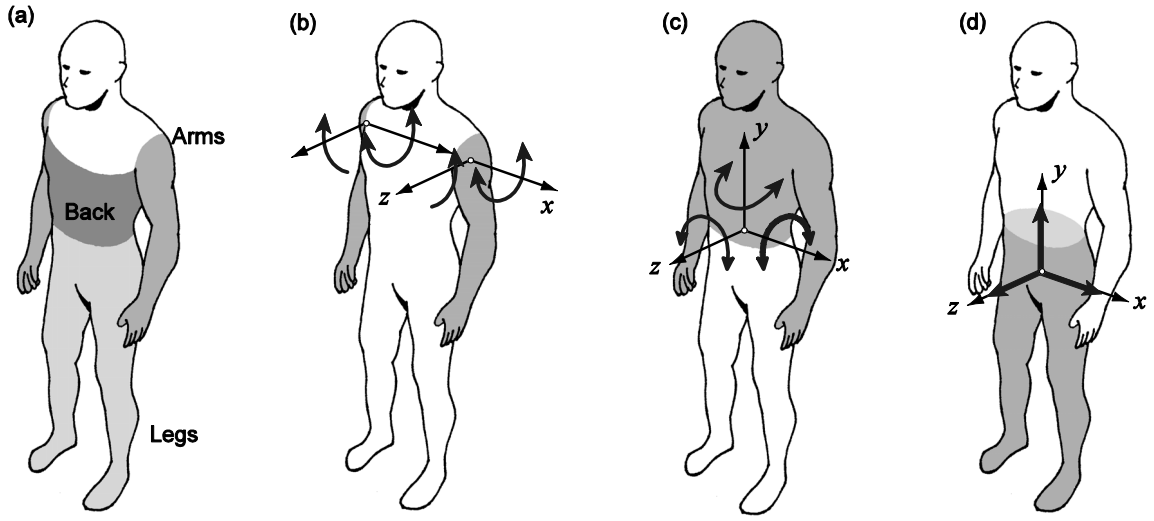

Figure 2.6 Control of body parts: (a) all parts, (b) arms, (c) back, and (d) legs.

Order of Control Steps

In the control algorithm, $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ (4 DOF), $\Delta\ddot{\boldsymbol{\theta}}_{\text{back}}$ (3 DOF), $\Delta\ddot{\mathbf{p}}_{\text{pelvis}}$ (3 DOF) and $\Delta\ddot{\boldsymbol{\theta}}_{\text{stress}}$ (k DOF) are controlled, each having an effect all the others. This interaction makes it difficult to control all these targets at the same time. Therefore, the algorithm computes each term in order, based on the order of importance.

Active control is applied to the arms, back and legs, in that order. The control of the upper body is more applicable than the control of the lower body in human motion control. The control of the lower body has a significant influence on body balance and the stability of motion, and hence it is desirable to minimize control of the lower body for a stability reasons. Therefore, comfort and balance control, using $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ (4 DOF) and $\Delta\ddot{\boldsymbol{\theta}}_{\text{back}}$ (3 DOF) are performed first. If the joint stress cannot be reduced or balance cannot be maintained, the lower body is then controlled using $\Delta\ddot{\mathbf{p}}_{\text{pelvis}}$ (4 DOF).

In control upper body, passive control is applied before active control. When environmental input is large and the current state is significantly different from the desired motion, the initial acceleration necessarily becomes large. As a result, because the joint stress becomes large and the figure is likely to lose balance, control based on the conditions causes unstable motion. To avoid this, the initial acceleration is first reduced through passive control. Active control is then applied based on the reduced acceleration in order to realize output acceleration close to the initial acceleration.

2.5.3. Control Algorithm

Based on the strategies described in the previous section, the overall algorithm of dynamic motion control is applied in following steps:

1. The initial acceleration is computed. $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$ is computed using the tracking algorithm.
2. The joint stresses and balance error are computed based on $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$. If there is no stressed joint and its balance is maintained, $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$ is used as $\ddot{\boldsymbol{\theta}}_{\text{output}}$ and following steps are not executed.
3. Passive control of upper body. $\Delta\ddot{\boldsymbol{\theta}}_{\text{stress}}$ is computed for all stressed joints.
4. Active control of upper body. $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ then $\Delta\ddot{\boldsymbol{\theta}}_{\text{back}}$ are controlled so as to reduce joint stress, and $\Delta\ddot{\boldsymbol{\theta}}_{\text{stress}}$ is recomputed based on $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$, $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$, $\Delta\ddot{\boldsymbol{\theta}}_{\text{stress}}$, and $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$.

5. Passive and active control of lower body. $\Delta\ddot{\mathbf{p}}_{\text{pelvis}}$ is controlled so as to reduce the stress on joints in the lower body and maintain body balance.

6. Output acceleration $\ddot{\boldsymbol{\theta}}_{\text{output}}$ is computed from $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$, $\Delta\ddot{\boldsymbol{\theta}}_{\text{back}}$, $\Delta\ddot{\boldsymbol{\theta}}_{\text{stress}}$, $\Delta\ddot{\mathbf{p}}_{\text{pelvis}}$, and $\ddot{\boldsymbol{\theta}}_{\text{tracking}}$.

In the reminder of this section, control algorithm in above steps is described.

Passive Control for Upper Body

Passive control of the upper body involves controlling the change of the angular acceleration of k joints where k is the number of joints that its torques exceed the available range of the joints that is determined by the muscle strength model. If the initial angular acceleration of one joint of the k joints is small, then the influence of that joint on other joints is also small. In passive control, the angular acceleration of each joint of the k joints is controlled separately considering only the moment of inertia $\mathbf{H}_{ij}$ which represents the relationship between the angular acceleration and torque of the individual joints. However, when the current state of a joint differs significantly from the desired motion, the initial angular acceleration of the joint is large and control becomes unstable because large angular accelerations cause large stress and balance error.

Therefore, we compute the change of the angular acceleration of each joint $\Delta\ddot{\boldsymbol{\theta}}_{\text{stress},i}$ in two phases. First, $\Delta\ddot{\boldsymbol{\theta}}'_{\text{stress},i}$ is computed such that $\ddot{\boldsymbol{\theta}}_{\text{tracking},i} + \Delta\ddot{\boldsymbol{\theta}}'_{\text{stress},i}$ is realizable within the available torque range of the i-th joint when the moment of inertia from other joints is ignored, given by

$$\boldsymbol{\tau}_{\min,i} < \mathbf{H}_{ii}\left(\ddot{\boldsymbol{\theta}}_{\text{tracking},i} + \Delta\ddot{\boldsymbol{\theta}}'_{\text{stress},i}\right) + \mathbf{C}_i + \mathbf{G}_i + \mathbf{F}_i < \boldsymbol{\tau}_{\max,i}. \tag{2.17}$$

Second, $\Delta\ddot{\boldsymbol{\theta}}'_{\text{stress},i}$ is computed such that $\ddot{\boldsymbol{\theta}}_{\text{tracking},i} + \Delta\ddot{\boldsymbol{\theta}}'_{\text{stress},i}$ is realizable when the moment of inertia from the angular acceleration of other joints $\ddot{\boldsymbol{\theta}}_{\text{tracking}} + \Delta\ddot{\boldsymbol{\theta}}'_{\text{stress}}$ is considered, given by

$$\boldsymbol{\tau}_{\min,i} < \mathbf{H}_{i}\left(\ddot{\boldsymbol{\theta}}_{\text{tracking}} + \Delta\ddot{\boldsymbol{\theta}}'_{\text{stress}}\right) + \mathbf{H}_{ii}\left(\ddot{\boldsymbol{\theta}}_{\text{tracking},i} + \Delta\ddot{\boldsymbol{\theta}}'_{\text{stress},i}\right) + \mathbf{C}_i + \mathbf{G}_i + \mathbf{F}_i < \boldsymbol{\tau}_{\max,i}. \tag{2.18}$$

Active Control for Upper Body

Active control of the upper body involves calculating $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ and $\Delta\ddot{\boldsymbol{\theta}}_{\text{back}}$. We compute $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ first, then $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$. The rotational acceleration of each part is computed for the comfort control of the j-th stressed joint ($\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},cj}$ or $\Delta\ddot{\boldsymbol{\theta}}_{\text{back},cj}$) and for balance control ($\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},bj}$ or $\Delta\ddot{\boldsymbol{\theta}}_{\text{back},bj}$). The largest acceleration is then used to control the part. When an environment input is applied to figure, the stress of joints and the positional error of the ZMP often occur in the same direction. In that case, the rotational acceleration for reducing the largest stress or for maintaining

balance can be expected to help the other stresses and imbalance. If unresolved stresses and imbalance remain, the next part is controlled so as to solve them.

The rotational accelerations $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},cj}$ and $\Delta\ddot{\boldsymbol{\theta}}_{\text{back},cj}$ for reducing joint stress are computed for the j-th composite joint consisting of rotational joints. For example, if the wrist consists of three rotational joints, as in our model, the rotational acceleration required to reduce the stress of the three joints in the wrist $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms,wrist},j}$ is computed for each joint j simultaneously. As mentioned in section 5.1.1, the relationship between the displacements of the i-th composite joint and the rotational acceleration of the arms or back is expressed using a submatrix of the moment of inertia matrix $\mathbf{H}(\boldsymbol{\theta})$, given by

$$\Delta\boldsymbol{\tau}_j = \mathbf{H}'\ddot{\boldsymbol{\theta}}_{\text{arms},ci}. \tag{2.19}$$

The required change of torque $\Delta\boldsymbol{\tau}_j$ is computed from $\boldsymbol{\tau}_{\text{stress}}$. The dimension of $\Delta\boldsymbol{\tau}_j$ is always equal to or less than $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},cj}$. Thus, $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},cj}$ is redundant. The solution so as to minimize $\left|\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},cj}\right|$ can be computed using the pseudo inverse matrix $\mathbf{H}'^{+}$ of $\mathbf{H}'$, given by

$$\ddot{\boldsymbol{\theta}}_{\text{arms},ci} = \mathbf{H}'^{+}\Delta\boldsymbol{\tau}_j. \tag{2.20}$$

$$\mathbf{H}'^{+} = \mathbf{H}'\left(\mathbf{H}'^{t}\mathbf{H}'\right)^{-1}. \tag{2.21}$$

The rotational acceleration of the arms for balancing $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},b}$ is computed in the same way such that the ZMP is moved to the MMP. As described in section 5.1.2, $\delta ZMP/\delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ is computed using equations (2.13) and (2.14).

Within the rotational accelerations, $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},cj}$ is computed for all stressed composite joints and $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms},b}$ is computed for the position of the ZMP, the largest of which is used to control the arms or back. When $\Delta\ddot{\boldsymbol{\theta}}_{\text{arms}}$ or $\Delta\ddot{\boldsymbol{\theta}}_{\text{back}}$ is too large, the stress on joints in the shoulders or back exceed the available torque. In this case, the rotational acceleration of each joint is reduced later by passive control using equations (2.17) and (2.18).

Active and Passive Control for Lower Body

Control of the lower body is achieved by controlling the change of the spatial acceleration of the pelvis in the same way as active control is applied for the upper body. The change of angular acceleration for all joints in the lower body is controlled indirectly through control of the spatial acceleration of the pelvis. For comfort control, the spatial acceleration of the pelvis is computed for all composite joints in the lower body. The relationship between the joint torques in a

composite joint and the spatial acceleration of the pelvis can be derived from equations (16) and (10), written as

$$\Delta\boldsymbol{\tau}_j = \mathbf{H}'\mathbf{J}_p\ddot{\mathbf{p}}_{\text{legs}} . \tag{2.22}$$

For balance control, the relationship between ΔZMP and $\ddot{\mathbf{p}}_{\text{legs}}$ is derived from equations (2.13) and (2.14) using the weight and center of mass of the upper body.

Passive control for the lower body is included in this control algorithm. The pelvis is controlled in the same way to active control for the upper body. The spatial acceleration of the pelvis is computed for the stressed composite joints of the legs and the ZMP. Within the spatial accelerations, the largest acceleration is used to control the lower body. As a result, the joint torque for the output acceleration may exceed the available torque range in this algorithm.

2.5.4. Limitations

On human movements, when some large stresses work on joints in the lower body or it likely to lose balance, the foot leaves from the ground. During a single phase, by swinging the moving leg or moving the foot to a stable position, more efficient and flexible control is achievable. However, to realize this kind of control, the motion needs to be controlled not only in angular acceleration space but also in angular space. This is beyond the scope of this method. Therefore, the algorithm is unable to control a figure successfully, when excessive forces or impulses are applied to the figure and the leg must be moved for stabilization. In such case, joint stresses on some joints are ignored and unnatural results could be generated. To overcome this limitation, we have to introduce some control in angular space. We are going to introduce an extension for this in next chapter.

2.6. Results

In this section, we present an experimental result. We created animations based on a keyframe motion sequence and environmental physical input. We used a squatting motion as input. The trajectories of the input motion were represented by a B-Spline. The interval between each frame of dynamic simulation was 1/30 second.

Figure 2.7, 2.8 and 2.9 show the images of the generated animation under four conditions. The upper images of each animation are normally rendered figures. In addition, in lower images,

both the input and generated motions are rendered as stick figure, and the control information is visualized using arrows. In those images, the orange figures represent the input motion, and the white figures represent the generated motion. Red arrows at joints indicate the stress on the joint. Blue, green and yellow arrows at the joints indicate comfort, balance and passive control, respectively.

When no physical input is applied (a), only tracking control was applied and the input motion was almost directly tracked. With an 8 kg weight (b), the arms were slightly controlled for balance and to reduce the stress on the back when raising the back. With a 15 kg weight (c), because active control of the arms could not sufficiently reduce the stress on the back, the back was forced to bend. Subsequently, the figure recovered to the input motion by swinging the arms. In the last animation (d), an impulse is applied to the figure from the front at the first frame. The read corn that appears in frame no. 3 shows the position and direction of the impulse. After the impact, the figure attempted to track the input motion while maintaining balance. These results show that our method produces dynamically changing motion based on input motion and environmental physical input.

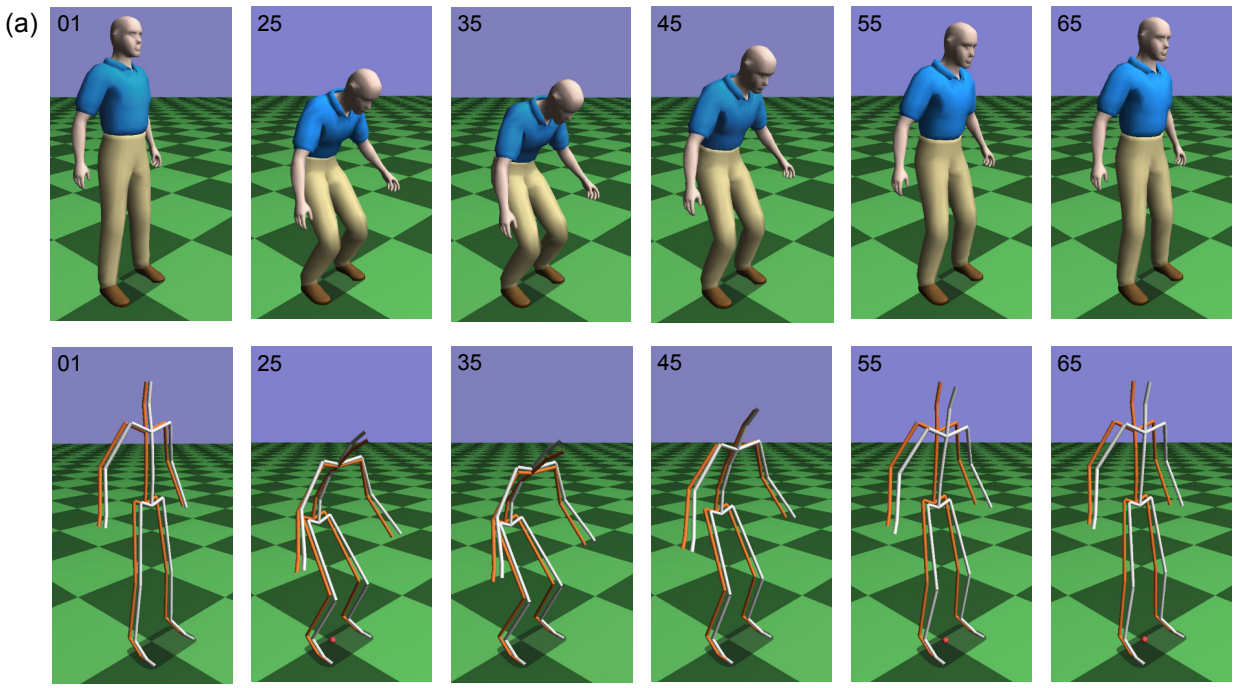

Figure 2.7 Images from resulting animations of a squatting motion with various environmental input. (a) no environmental physical input. The numbers on the corner of images shows its frame number.

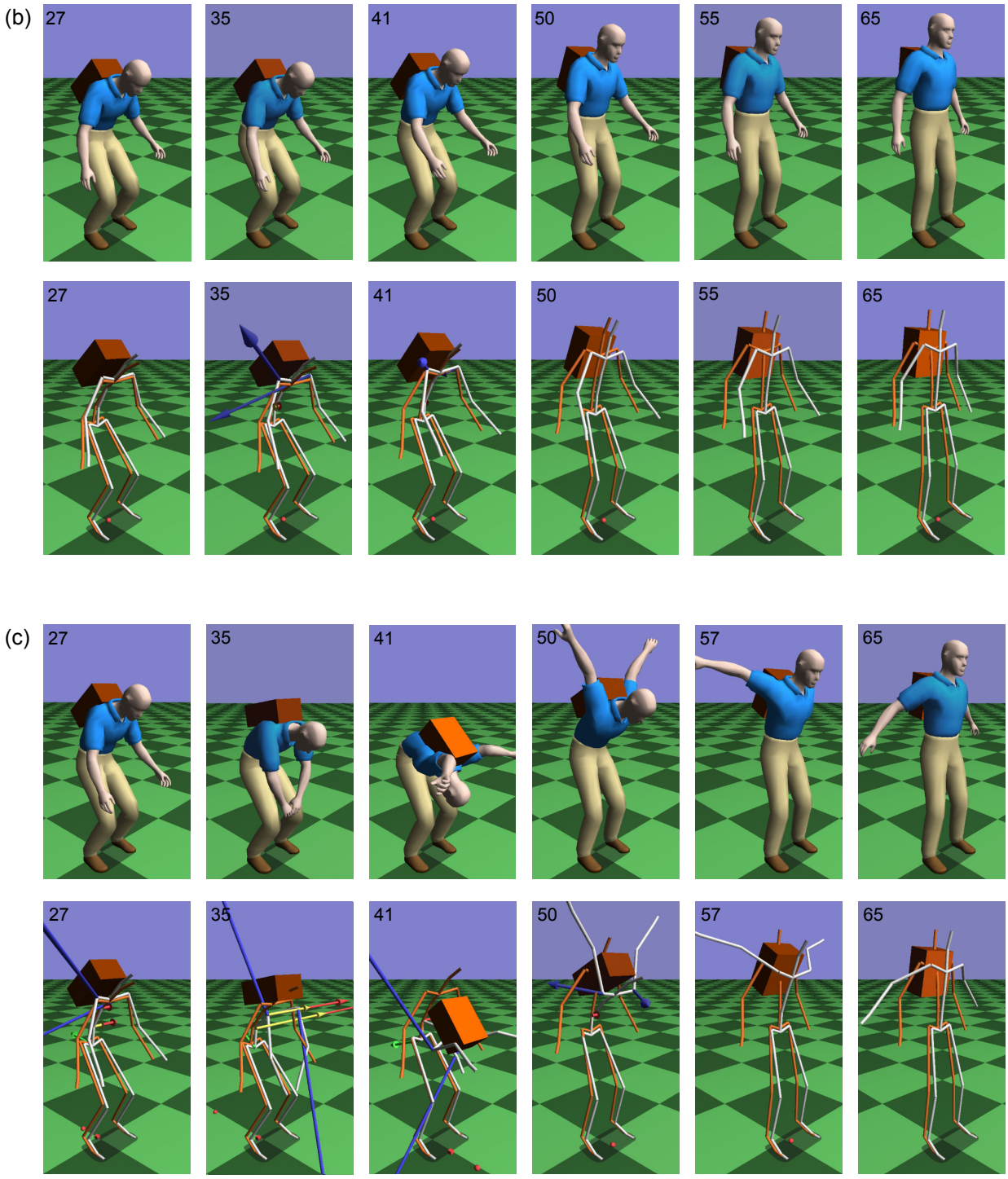

Figure 2.8 Images from resulting animations of a squatting motion with various environmental input. (b) with 8 kg weight. And (c) with 15 kg weight.

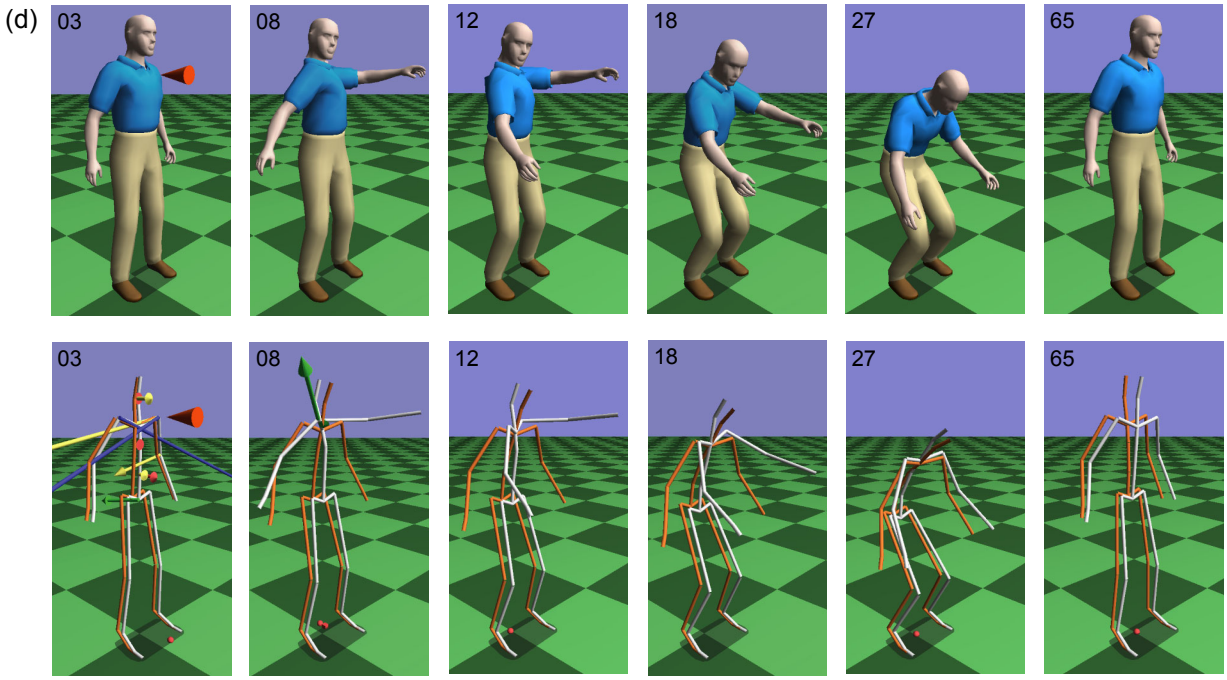

Figure 2.9 Images from resulting animations of a squatting motion with various environmental input. (d) impulse applied at the first frame.

The computational time for dynamic motion control on the generated animation is shown in Table 2.1. The computational time become large when the joint torque of many joints in the initial angular acceleration exceeded the available range because comfort control is computed for each stressed joint. The computational time required for one step of dynamic motion control was 3 milliseconds in the worst case (c). This system generated the animations in real-time.

interaction	total	average	Max
(a) no interaction	70.6	0.78	1.0
(b) with 8 kg weight	78.8	0.89	2.4
(c) with 15 kg weight	102.9	1.43	3.1
(d) impulse applied	72.9	0.81	2.5

Table 2.1 Computational time (milliseconds) for dynamic motion control on PC (Pentium III, 800 MHz), the total time for 90 frames (3 seconds), average frame time, and maximum frame time.

2.7. Summary and Remaining Problem

This chapter described a dynamic motion control technique for human-like articulated figures. The primary difference between our method and former methods is to control the angular accelerations of a human figure's joints instead of the angles and torques. This approach ensures continuous and realistically changing motion. The algorithm controls each part of the figure through a minimum number of DOF, and computes the output angular acceleration in carefully designed steps. This approach has made it possible to generate dynamically changing motion in real-time. In experiments, our system generated changing motions in response to the weight of a load and an external impulse.

Physics based approaches are yet to be widely adopted in computer games. However, such applications require dynamically and realistically changing motion, otherwise are limited to replaying motion sequences created in advance. We believe that the proposed technique will break the limitations of physics based approaches.

As discussed in section 2.5.4, a limitation of this method is that it is difficult to control resulting motions in angular space. Therefore, it cannot generate appropriate reaction when a large influence is applied. To overcome this problem, the dynamic control algorithm is integrated into a new framework in the next chapter.

Chapter 3: Extended Motion Control Framework

3.1. Introduction

This chapter describes a framework that is extended from the motion control method that is presented in chapter 2. A dynamic motion control scheme for generating dynamic reactions based on a given motion data was presented in the last chapter. However, in practice, it alone cannot generate continuous motions in an interactive application because of two reasons.

First, the algorithm lacks the ability of control in angular space. The dynamic control algorithm generates direct reactions when unexpected physical interaction is applied to the figure. It works well when the physical influences from the interaction are small enough. However, when large physical influences are applied to the figure, a more careful and planned reaction is expected. For example, if too large stresses are applied to the pelvis or under body, the control algorithm may generate a very large arm swing. In such case, it is expected to stop following the input motion data and plan a new motion data such as moving a foot to maintain its balance or stopping motion tracking and returning to a stable posture. It is difficult to realize such control in a control scheme that controls figure's joints in angular acceleration space. In order to handle such reactions, we need a motion planning scheme what works in angular space.

Second, it is difficult to provide a continuous input motion to the dynamic motion controller. The control algorithm generates resulting motions based on the input motion sequence. As introduced in section 2.1, current applications mostly generate continuous human motions by sequentially synthesizing shot motion clips that are prepared in advance. However, those applications do not suppose that the synthesized motion data is changed on the fly. The prepared motion clips are carefully designed so that one motion can be continued from another motion by making the differences the terminal posture of a previous motion and the initial posture of a next motion small. If the terminal posture of resulting motion is changed by

dynamic control method, they cannot continue the next motion because such situations are not supposed. In order to provide continuous target motions to dynamic controller, more sophisticated motion synthesis motion is required.

To solve these problems, we have developed two new kinematics based techniques: motion transition and reaction generation and combined them with the dynamic motion control algorithm. The motion transition method generates continuous target motions by synthesizing stored motion clips considering the constraints of the end-effectors. The dynamic reaction methods generate an appropriate motion data of dynamic reactions such as protective steps for balancing and recovery motions when large physical interactions is detected. The dynamic control method that is presented in Chapter 2 the last chapter is used to control angular accelerations of the character to track a given target motion data considering the physics of the character, such as joint stress and balance. The input motions passed to dynamic control algorithm are usually executed as given, and only when an unexpected physical interaction happens are dynamically changing motions generated. By combining the three techniques, our framework produces dynamic character motions on the fly while making use of existing motion collections.

The remainder of this chapter is organized as follows. Section 3.2 explains the structure of the developed framework. Section 3.3 describes action data representation. Sections 3.4, and 3.5 present dynamic motion transition and generation of dynamic reactions, respectively. In section 3.6, an experimental result is presented and discussed. Section 3.8 summarizes this chapter.

3.2. System Structure

The real-time animation system that is described in this chapter is designed to produce dynamic motions in cooperate with an interactive application such as computer games. The system makes use of the existing motion collections and has the ability to generate dynamically changing motions in response to physical interaction between the character and the environment. The structure of the proposed system is shown in Figure 3.1. The controller of the proposed system consists of three components: motion synthesizer, dynamic controller, and reaction generator.

The application gives short motion sequences that it wants the character to execute in context one after another. The motion synthesizer generates continuous target motion by synthesizing shot motion data. The smooth transitions from one action to next action are realized

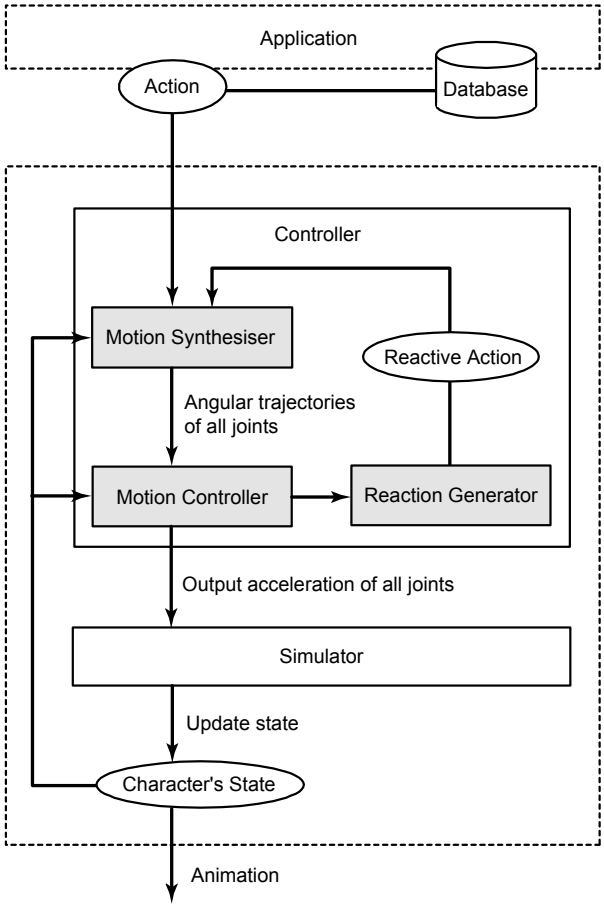

Figure 3.1 System structure.

considering the constraints of the end-effectors, even if the postures of the current motion and the following motion are slightly different.

The dynamic controller computes the angular accelerations of the character's joints on each frame to track a target motion that is generated by the motion synthesizer. The target motion is usually executed as it was given, and only when unexpected physical interactions was happen, dynamically changing motions are generated considering the balance and joint stresses of the character. This dynamic control method is specific to human-like articulated figures. It controls each body part through the minimum number of degrees-of-freedom (DOF).

In addition, when it is difficult to realize the synthesized motion because of an unexpected physical interaction, dynamic reactions such as protective steps for balancing and recovery motion are generated and executed. The control in an angular acceleration space used in dynamic controller is suitable for generating reactive motions in response to physical interactions while completing the given motion. However, when it is difficult to execute the

given motion, and the character tries to recover to a stable posture, the reactive motion in angular space must be planned. In our framework, the reaction generator realizes these reactions.

3.3. Action Data

This section explains data representation for short motion clips. In this chapter, we have call short motion clips "action" data for differentiating them from synthesized continuous motions that are delivered to the dynamic controller. Both action data and synthesized motions have the same representation internally.

Action data are supposed to be prepared so that each action represents a unit motion such as a kick, punch, a walking step, etc. For action data, any kinds of existing motion data such as motion captured or keyframed motion clips are used. In addition, dynamically generated motions using inverse kinematics or a step generator for locomotion can also be used as action data. Our system allows action data to vary itself on the fly.

Action data consists of

- Time-varying angles for all joints: $\theta_i(t)$.
- Time-varying position and orientation of the root segment: $\mathbf{p}_{\text{root}}(t), \mathbf{q}_{\text{root}}(t)$.

This is a common way to represent motion data of an articulated figure.

In addition, action data have additional information for motion transition and dynamic control in our system as below:

- Total, initial, and terminal duration: $T_{\text{total}}, T_{\text{init}}, T_{\text{term}}$, respectively
- Time-varying constraints on the root segment and each limb: $C_l(t)$ where l = {root, right_foot, left_foot, right_hand, left_hand}.
- Time-varying position and orientation of each end-effectors: $\mathbf{p}_l(t), \mathbf{q}_l(t)$.

The initial and terminal durations are the lengths of the initial and terminal phases that are used for motion transitions.

The time-varying constraints have one condition on each frame, chosen from the choices below, for the root segment and end-effectors of each limb (arms and legs):

- Angular constraints
- Spatial constraints
- Supporting constraints

Angular constraints preserve the joint angles of the limb linkage during motion transition and dynamic control. Spatial constraints preserve the position and orientation of end-effector (foot or hand), and the joint angles in the limb linkage are computed using an inverse kinematics. Supporting constraints is a special case of spatial constraints, and keeps the end-effector motionless on the ground as a supporting limb.

The trajectories of the end-effectors of each limb $\mathbf{p}_l(t), \mathbf{q}_l(t)$ are automatically computed from the action data using forward kinematics. To realize human-like movements, our system handles supporting limbs with meticulous care.

3.3.1. Inverse Kinematics for Human Limbs

We use an analytical inverse kinematics [25][35] for computing the joint angles in a limb linage. Using the inverse kinematics technique, the angles of seven joints in the limb $(\theta_0 ... \theta_6) \in \mathbf{R}^7$ are uniquely determined from the position $\mathbf{p}_{ee}$ and orientation $\mathbf{o}_{ee}$ of the end-effecter and the swivel angle w_{mid} of the middle joint of the limb around the axis from the root joints to the end-effector.

$$\mathbf{p}_{ee} \in \mathbf{R}^3, \mathbf{o}_{ee} \in \mathbf{R}^3, w_{mid} \in \mathbf{R}^3 \Leftrightarrow \theta_0 ... \theta_6 \in \mathbf{R}^7 . \quad (3.1)$$

This computation of analytical inverse kinematics is faster than numerical inverse kinematics methods designed for general body structures, and does not include redundancy.

This analytical inverse kinematics technique makes it possible to edit a motion on the fly based on the constraints of the positions of the end-effectors. Because of these benefits, this method is used widely for motion retargeting and editing [25][35]. However, the details of these editing techniques are out of the scope of this paper as here we focus on novel methods that use the inverse kinematics for motion transition and dynamic control.

3.4. Dynamic Motion Transition

This section describes a motion transition method for generating continuous and long motions by sequentially synthesizing short action data. However, it is difficult to realize the transition

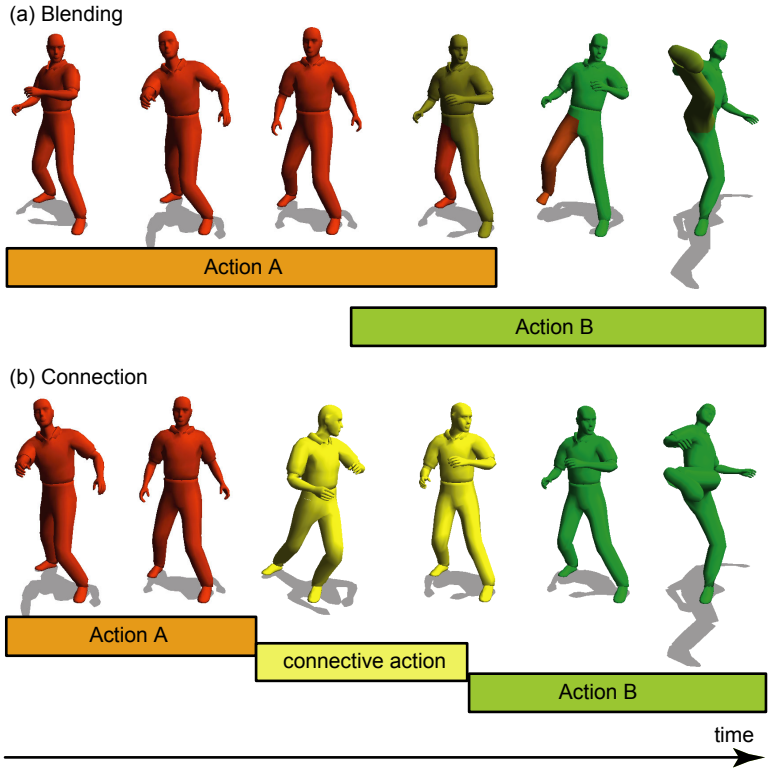

Figure 3.2 Two transition types: (a) blending and (b) connection.

between any kinds of actions. As explained in Section 3.2, action data are edited by the animators so that the last of an action is continued with the beginning of another motion. The purpose of our transition method is to make as much use as possible of motion data in our framework and to realize a smooth transition in case when character motions are slightly changed through environmental interactions. To do so, we provide the users clear rules for the judgments of transition and a sophisticated transition method that works in real-time.

3.4.1. Transition Interface

Our system provides two kinds of transition type: "blending" and "connection" (Figure 3.2). The application is allowed to choose the appropriate transition type based on its context. "Blending" overlaps the last part of the current motion and the first part of the next motion. "Connection" inserts a short action for connecting two actions. "Blending" provides a smoother transition while "connection" executes actions more precisely.

When an action is given by the application to the motion synthesizer, it is first determined if a transition to the given action is possible based on the condition of the character at the point the given motion is activated. If the transition is not possible, the input action is rejected and the

application is able to entry another action. If the transition is applicable, a smooth transition is realized while preventing unnatural results such as a gap of motion or a sliding of the foot. To realize such judgments and transitions, additional information about action data such as constraints on end-effectors are considered.

The applications can request to initiate a new action after the current action is completed, or to terminate the current action and to execute the new action immediately. Judgment whether the transition is possible or not depends on the condition when the new motion is to be executed.

3.4.2. Motion Blending

Motion blending realizes a transition from an action to another action by blending the initial phase of the following action with the terminal phase of the current action. Action data are given the duration of their initial and terminal phases by the animators who edit the actions. Specifying these durations in this blending method, a very common approach used in many commercial animation tools, is not a difficult task for animators.

We allow motion blending only when two actions satisfy the following conditions:

- The constraints of the root segment and all limbs of the two actions are the same: $C_{/1}\left(T_{\text{total1}}\right) = C_{/2}\left(T_{\text{initial2}}\right),\ \forall l \in \{\text{root}, \text{right\_foot}, \text{left\_foot}, \text{right\_hand}, \text{left\_hand}\}$.

These conditions are required for the following motion blending method to be executed.

Motion blending for non-supporting limbs

The blending method that is used for each limb depends on the constraints on the limb. When angular constraints are specified, the angles of joints are simply blended using the weight function for each action. This is common way to blend two motion data.

$$\theta(t) = \left(\theta_1(t_1)\cdot w_1(t_1) + \theta_2(t_2)\cdot w_2(t_2)\right) \Big/ \left(w_1(t_1) + w_2(t_2)\right), \tag{3.2}$$

where t_1 and t_2 are the local time on each action which is computed from t, and the start time of each action $t_{\text{init1}}, t_{\text{init2}}$ respectively. In addition, As a weight function, the linear function is used as follows:

$$t_1 = t - t_{\text{init1}}, w(t_i) = \left(t_i - T_{\text{total1}} + T_{\text{term1}}\right) / T_{\text{term1}}, \tag{3.3}$$

$$t_2 = t - t_{init2}, w_2(t_2) = (t_2) / T_{init2}. \tag{3.4}$$

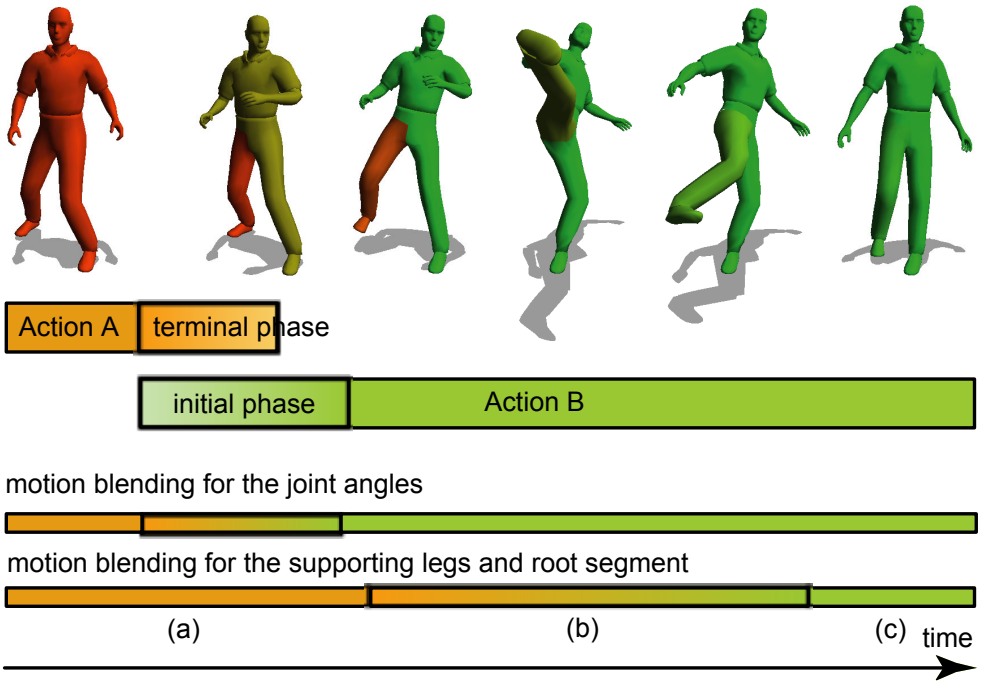

Figure 3.3 Motion blending with supporting constraints.

The angles of the joints that are not included in any limbs such as, back, neck and a finger are also blended in this way.

When spatial constraints are specified, the position and orientation of the end-effector is blended using a similar blending function. For blending positions, the x, y and z coordinates are blended separately using the equation (3.2). For blending orientations, quaternion and great circle interpolation is used. The joint angles in the limb are then computed from the trajectory of the end-effectors using inverse kinematics as explained in Section 3.3.1.

Motion blending for supporting limbs

When supporting constraints are specified, the blending method becomes more complicated. If the position of a foot that is expected to support the body is blended in the same way with spatial constraints, it causes an unnatural motion such as foot sliding. Therefore, under supporting constraints, the configuration of the end-effectors at the end of the previous action is basically maintained during the next action (Figure 3.3 (a)). When the foot is moved in the next action, the position and orientation of the foot are smoothly blended with the original trajectories (Figure 3.3 (b)).

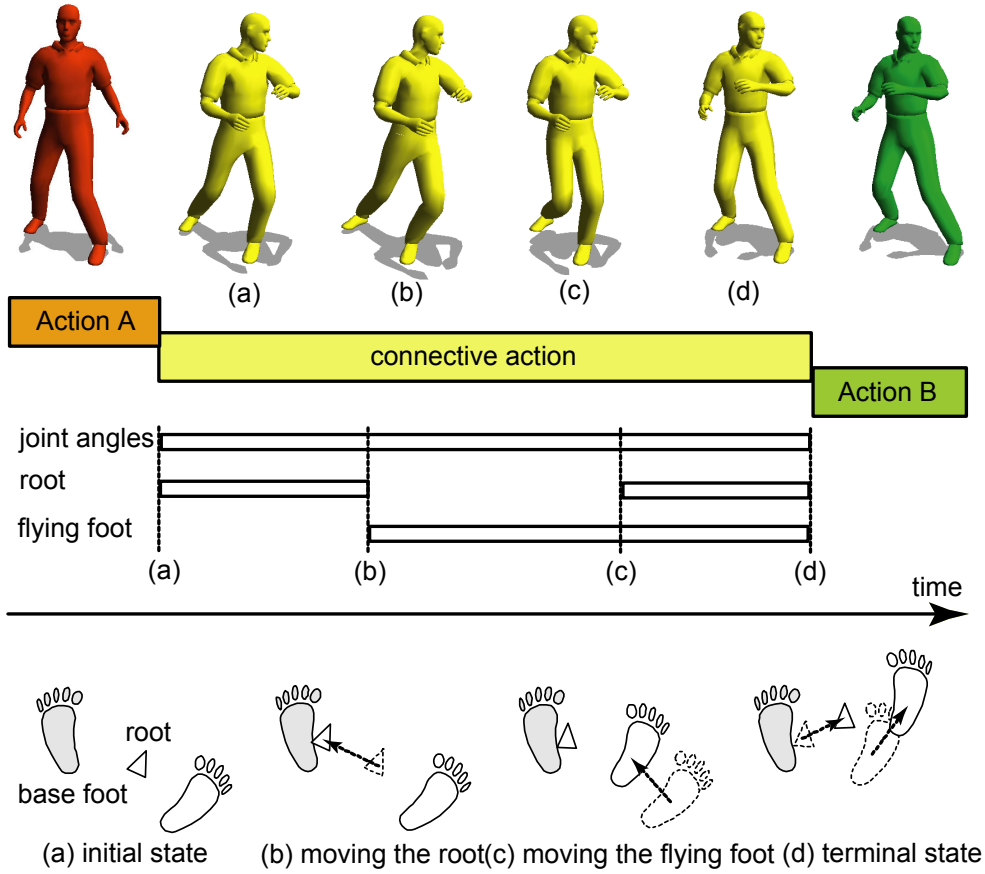

Figure 3.4 Motion connection with supporting constraints.

3.4.3. Motion Connection

Motion connection inserts a procedurally generated connective action between two actions to realize the transition. The generated actions do not necessarily ensure that the transition is as realistic as that in motion capture data or motions edited by a skilled animator. However, as explained here, it is difficult to prepare all possible motions for connecting any kind of action. Our method realizes a smooth transition by at least preventing unnatural results.

We allow motion connection when the following conditions are satisfied on the termination of current action and the beginning frame of the following action.

- The conditions of the root are the same: $C_{\text{root1}}(T_{\text{total1}}) = C_{\text{root2}}(0) \neq \text{SUPPORT}$.
- The same limbs have supporting constraints:
  $C_{\text{root1}}(T_{\text{total1}}) = C_{\text{root2}}(0) = \text{SUPPORT}, \exists l \in \{\text{root}, \text{right\_foot}, \text{left\_foot}, \text{right\_hand}, \text{left\_hand}\}$

The second condition means that at least one limb must have supporting constraints during the transition. The supporting limb is used as the base foot during the connective action. The conditions for motion connection are looser than those for motion blending. Therefore, when the application tries to insert an action with a "blending" mode and it is rejected by the motion synthesizer, the application can try again with an "insert" mode.

Generation of connective action

A connective action is dynamically generated as a keyframe motion trajectory based on the terminal posture of the current action and the initial posture of the following action by considering the constraints of the limbs. The angles of all joints in non-constrained limbs are simply interpolated. An interpolation function is computed for each joint from the two keyframes.

During the connective action, the base foot is fixed, and the root segment and the flying foot which is the other end-effector are moved cooperatively (Figure 3.4). To move the flying foot, the center of mass should first be moved above the base foot to reduce the weight applied to the flying foot. The trajectories of the root and flying foot are computed from four keyframes. The postures in the intermediate frames are computed as follows. The position of the root (Figure 3.4 (b)) is chosen as the point closest to the flying foot in the initial posture inside the contact polygon of the base foot. The position of the flying foot (Figure 3.4 (c)) is computed so that the distance between the root and the flying foot is same as the distance at the terminal posture. For the heights of the root segment and the flying foot at the midpoint, constant values are used.

Duration of connective action

The duration of the connective action is automatically determined based on the differences between the two actions. If the initial and terminal postures of a connective action are similar, the connective action does not need much duration. The larger the difference between the two postures, then a larger duration is required. In addition, the speed of the two actions needs to be considered. If a connective action connects quick actions, a quick connective action is expected.

The total duration of a connective action is computed so that the velocities of all joints, the root, and the end-effectors during the connective action do not exceed the maximum velocity of the following velocities:

- The terminal velocity of the current action $\left|\dot{\theta}_{i1}(T_{\text{term1}})\right|, \left|\dot{\mathbf{p}}_{l1}(T_{\text{term1}})\right|$
- The maximum velocity in the following action $\left|\dot{\theta}_{i2}(t)\right|, \left|\dot{\mathbf{p}}_{l2}(t)\right|$
- A constant value (to avoid too small velocities) $\dot{C}_{\text{angular}}, \dot{C}_{\text{spatial}}$

For example, a duration of the connective action is computed based the velocity of the i-th joint $\dot{\theta}_{i,\text{connective}}(t)$ so that the following equation is satisfied:

$$\left|\dot{\theta}_{i,\text{connective}}(\forall t)\right| <= \max\left\{\left|\dot{\theta}_{i1}(T_{\text{term1}})\right|, \left|\dot{\theta}_{i2}(\forall t)\right|, \dot{C}\right\}. \tag{3.5}$$

From the durations that are computed for all joints $\dot{\theta}_{i,\text{connective}}(t)$, and root and end-effectors $\left|\dot{\mathbf{p}}_{l,\text{connective}}(t)\right|$, the maximum duration is used as the duration of the connective action.

The trajectories of connective actions are computed from the keyframes as interpolation functions. Therefore, by taking the derivatives of the interpolation functions, velocities are calculated. The duration of each phase of the connective action is determined based on the distance of the root segment and the flying foot between the keyframes.

3.5. Dynamic Reactions

This section describes a method to generate dynamic reactions when it is impossible to execute a target motion by dynamic controller because of external interaction such as large impulse or external force. The dynamic control algorithm explained in Chapter 2 generates dynamic motions such as moving the pelvis and swinging the arms while still executing the target motion. For this kind of dynamic control, the control in angular acceleration space is appropriate as discussed in Section 2.2. However, when it is difficult to continue to execute the input motion, the target motion should be changed in the angular space rather than by dynamic control in the angular acceleration space.

In our system, when dynamic reaction is initiated based on some conditions in the reaction generated, a keyframe action is dynamically generated and given to the motion synthesizer with the terminating of the current action. On each time step, after the dynamic control is performed, the conditions are checked to initiate dynamic reactions. Currently three kinds of reactions are supported in our system: a protective step for balancing, falling down, and recovery to a stable posture.

3.5.1. Protective Step for Balancing

In dynamic control, the balance of the character is maintained by moving body parts such as pelvis or arms. However, when it is difficult to maintain balance by such controls, a protective step by moving a foot in the direction in which the figure is losing its balance is executed in order to keep the ZMP inside the support polygon. Moreover, when the character is losing balance more quickly and it does not have enough time to move the foot, the character must fall down. The generation of a falling down reaction is discussed in the next subsection.

The condition to initiate a protective step is determined using the spatial velocity $\dot{p}$ and acceleration $\ddot{p}$ of the root.

$$0.5 < \dot{p} < 1.0 \text{ m/sec} \quad \text{and} \quad \ddot{p} > 10.0 \text{ m/sec}^2 . \tag{3.6}$$

The protective step is initiated only when both feet of the character are on the ground. It is an approach where it is possible to use a learning algorithm to determine the conditions for falling down. Faloutsos [12][7] developed a learning method with and applied to dynamic controllers in a two-dimensional space. However, as mentioned [12], this is difficult because the dimension of the parameter space become large in three-dimensional space. Therefore, experimentally determined parameters are currently used in our system.

Step actions should be generated dynamically, because they depend on posture and velocity of the character when they are initiated. Although some methods for generating the step motions of human walking using motion capture data have been proposed [36], it is difficult to apply them in this case because position and velocity of the root segment should be considered, and

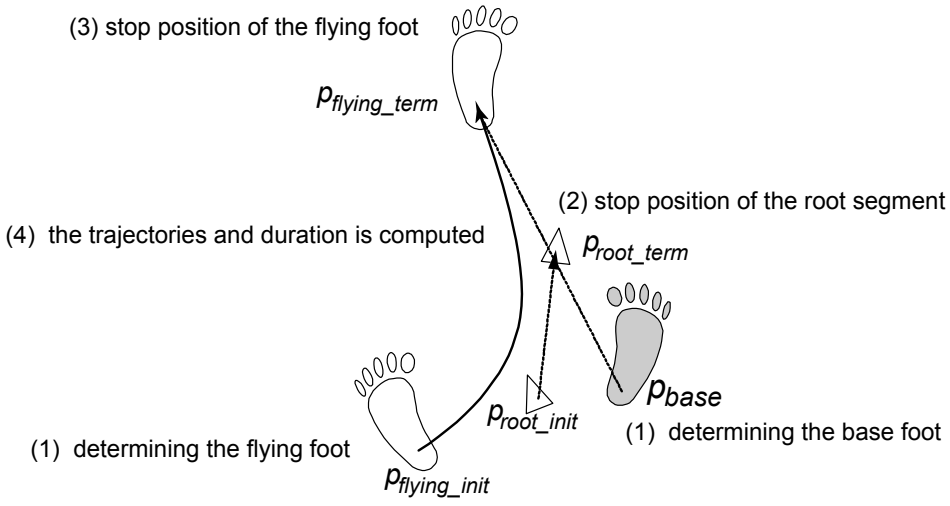

Figure 3.5: Motion planning for protective step.

the parameter spaces become large. Therefore, we decided to use a procedural method for generating step actions.

The step action is generated by using the algorithm for convective action explained in Section 3.4.3 based on the current and terminal posture. The terminal posture of the lower body in the step action is computed as shown in Figure 3.5. (1) First, the base and flying foot are determined. The one that is close to the ZMP is moved as the flying foot. (2) Second, the terminal position of the root segment is determined based on the velocity of the root segment using an empirically determined parameter k_{root}.

$$p_{\text{root\_term}} = k_{\text{root}} \cdot \dot{p} + p_{\text{root\_init}} . \tag{3.7}$$

The terminal position of the flying foot is determined so that the root segment is placed middle between the base and the flying foot using a parameter k_{foot}.

$$p_{\text{flying\_term}} = k_{\text{foot}} \left(p_{\text{root\_term}} - p_{\text{base}} \right) + p_{\text{root\_term}} . \tag{3.8}$$

In addition, the orientation of the root segment and the flying foot in the terminal position is determined from the direction to the initial position of the root segment. The terminal posture of the upper body of the original action is used as the terminal posture of the step action.

The generated step action is given to the motion synthesizer in "blending" mode by setting the initial duration of the section as being from the beginning to the point when the flying foot touches the ground.

3.5.2. Falling Down

The condition for falling down is defined in the same way as the condition for protective steps.

$$\dot{p} > 1.0 \text{ m/sec} \quad \text{and} \quad \ddot{p} > 10.0 \text{ m/sec}^2 . \tag{3.9}$$

For a falling action, a fixed action is used because the distance of the step in a falling action is basically considered to be fixed. Currently in our system, two actions are prepared for falling actions .

- Falling back and touching the hip and hands on the ground
- Falling to the front and touching the knees and hands on the ground

To continue to animate the character after a falling action, the application has to prepare some actions for a standing up action from the terminal postures of the falling actions.

3.5.3. Recovery to a Stable Posture

Target motions are sometimes not executable in the dynamic controller when large stresses are applied to the joints. For example, if a character is holding a significantly heavy load compared to the muscle strength of the character, it may be not able to complete a task. In such a case, a recovery action is generated to put the character back to a stable posture.

Recovery action is initiated when the character cannot generate enough angular acceleration to track the target motion in a joint within a certain time T_{recovery} which is specified by the user. The recovery action from the current posture to a stable posture is also generated using the algorithm for convective action, the same as for the protective step action. The initial posture of the current action is used as the stable posture.

3.6. Results and Discussion

This section shows some animations of two human figures that are dynamically generated in response to the interaction between the figures. The human model and keyframe motion data are created using commercial animation tools. In this experiment, we prepared and used still, step forward, punch, and kick actions in addition to falling down action that is explained in Section 3.5.2. Our prototype program generated 30 Hz animations of two figures in real-time on a Pentium4 1.2GHz PC.

Figure 3.6 and Figure 3.7 shows images from animations of dynamic reactions respond collision impulse that were generated in our experiment. In these animations, the figure standing in right side executed a punch action and hit the figure in left side. In this experiment, we used the same punch action for all animations and changed the initial positions of the two figures. Because the collision point and velocity varied based on their positions, various results are generated.

In Figure 3.6, because the influence from the collision impulse was small enough, only dynamic control algorithm was used and no reaction generation method was applied. After two figures collided each other, the left figure swung the arms and moved the pelvis to keep its balance, and returned to the target motion.

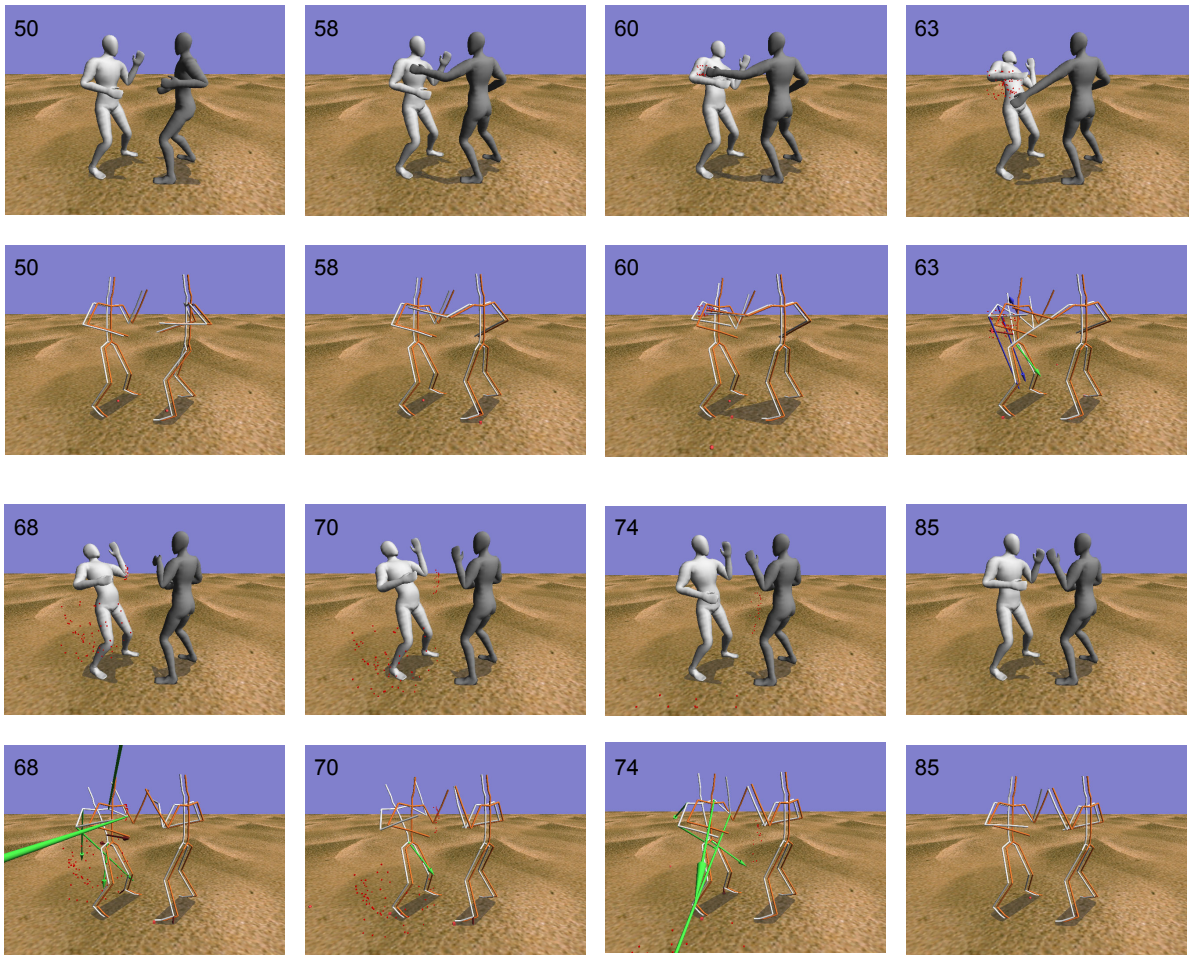

Figure 3.6 Dynamic reactions that were generated in response to collision impulse between two figures. The numbers on the corner of images shows its frame number. At frame 60, right figure hits the left figure and the collision impulse changes the velocity of their motions. In this case, only dynamic motion control algorithm was used to produce the reaction responds the collision impulse, because the impulse was small enough. Lower images depict visualization of the dynamic control. Orange figures are target motion and white figures are generated motion. Blue and green arrows from joints shows active controls for reducing stress and balancing control respectively.

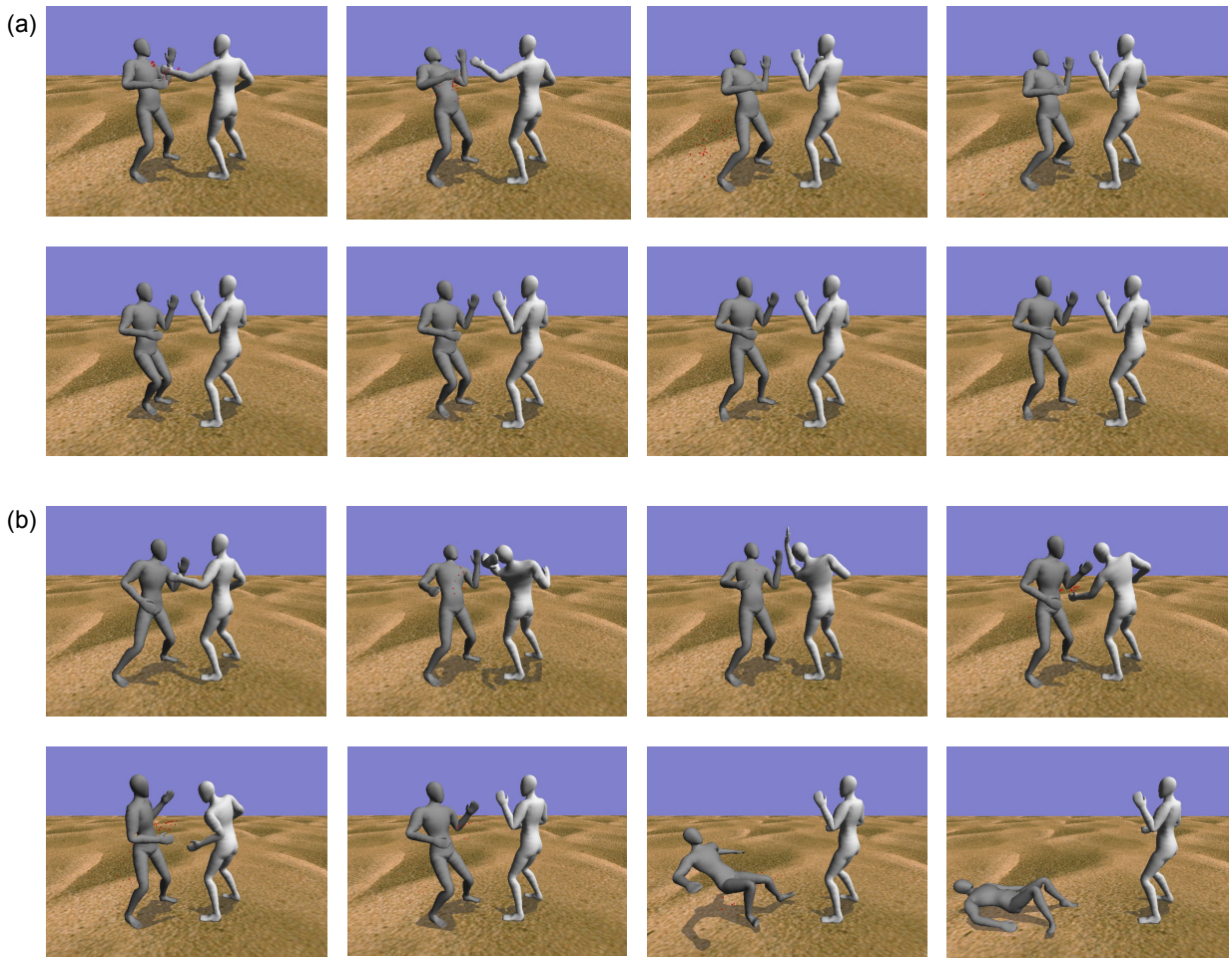

Figure 3.7 Dynamic reactions that were generated in response to collision impulse between two figures. At the first frame of each example, right figure hits the left figure and the collision impulse changes the velocity of their motions. In these cases, dynamic reactions are generated and executed: (a) Protective step for balancing. (b) Falling back.

In Figure 3.7 (a) and (b), some dynamic reactions were generated and executed. Figure 3.7 (a) shows a result of protective step for balancing. After the collision in the first image of Figure 3.7 (a), the velocity of the left figure exceeds the threshold, and a protective step was generated. The following images show the backward step for balancing. Finally, Figure 3.7 (b) is falling back motion. The figure fell down after the large impulse.

Figure 3.8 and Figure 3.9 show an example of motion transition continued after step back reaction. Figure 3.8 (a) shows three original action sequences. As shown here, still, step forward, and kick actions are created so that all the initial and terminal postures are same in order to perform smooth transition between each other. However, when the terminal posture of the previous motion was changed by some physical interaction and dynamic reaction (Figure 3.8 (b)), we need a motion transition method in order to continue another action. In Figure 3.8 (c), motion blending was used to make a transition. The terminal position of the left foot of the step action was kept during still action after the step action. Then, when the left foot is moved in the next kick action, the trajectory of the left leg was smoothly blended. On the other hand, Figure 3.8 (d) is the resulting animation when motion connection technique was used in the same situation. The connective action was inserted between the still action and kick action in order to move the left foot from the terminal position of the still action to the initial posture of the kick action. In Figure 3.8 (c) and (d), both motion blending and transition worked well.

In this experiment, we have generated various reactions using a same set of actions. However, we found that the resulting animations change based on even a small initial condition and it is difficult to control the results of animation. If some application requires controlling the results of motions, some kind of method to control the parameters such as the magnitude and direction of impact when a figure hits another should be developed. We also found that the dynamic control algorithm is sensitive to physical interaction. Because the angular accelerations are determined based on an instant condition, a large acceleration and exaggerated reactions sometimes occur as result. To handle more smooth reactions, the physical influences in wider duration should be considered. In addition, because the style of generated reactions is currently fixed, it cannot express the characteristics of individual characters. To realize more realistic motions, some methods that analyze motion capture data of actual person and utilize them for generating personalized reactions are anticipated.

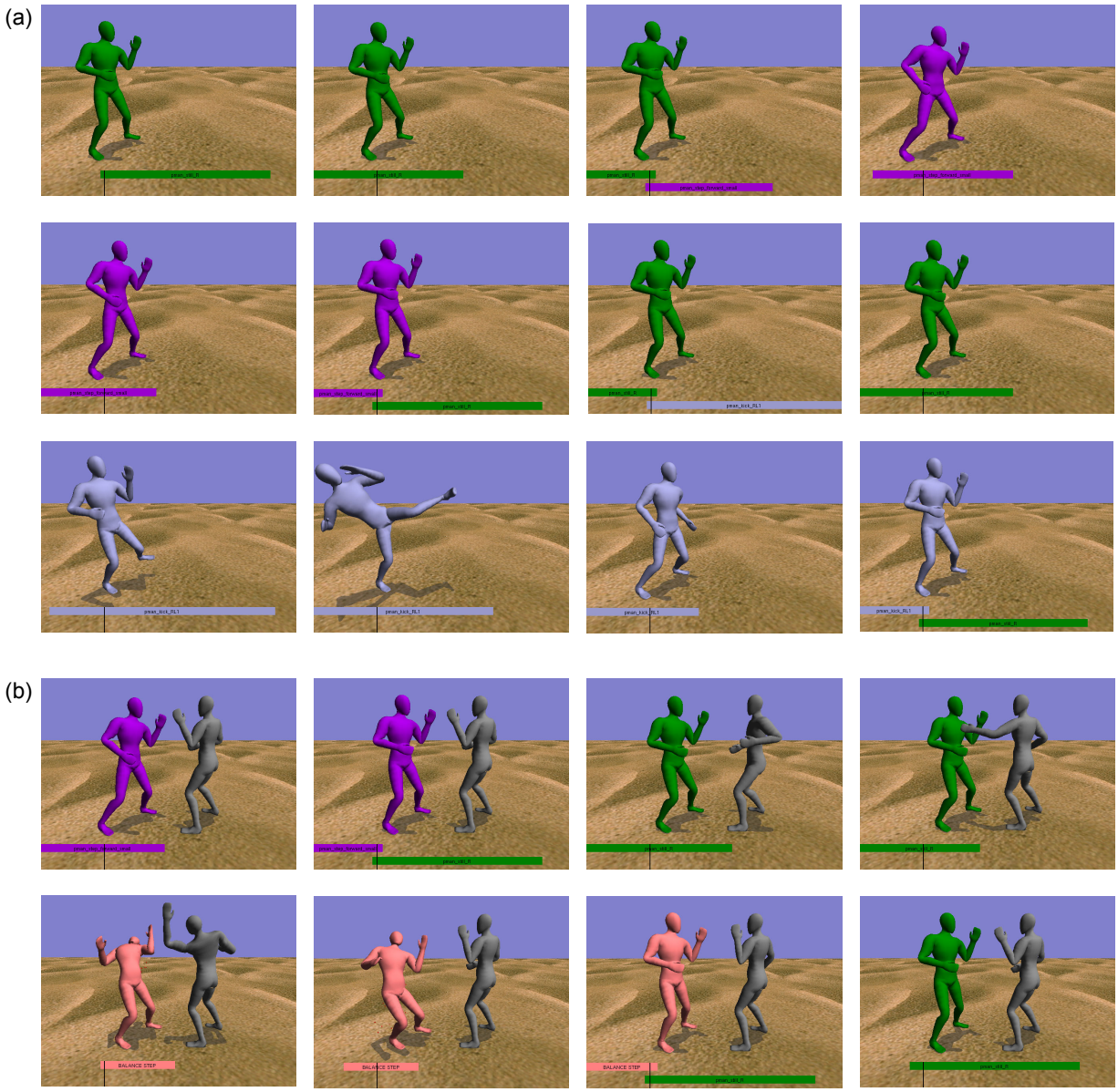

Figure 3.8 Example of motion transition. (a) original action sequences: still (green figure, first to third image), step forward (purple figure, fourth to sixth image), still again (seventh to eighth image), and kick (light blue figure, ninth to twelfth image). Each action has the same posture at the initial and terminal frames so that a motion transition from one to any other action works well. (b) An example in which the transition will not work well, because a step back reaction is executed by the collision impulse (fourth image) and it changed the terminal posture (the positions of the left foot are different between in the last image of (b) and in the first image of (a)).

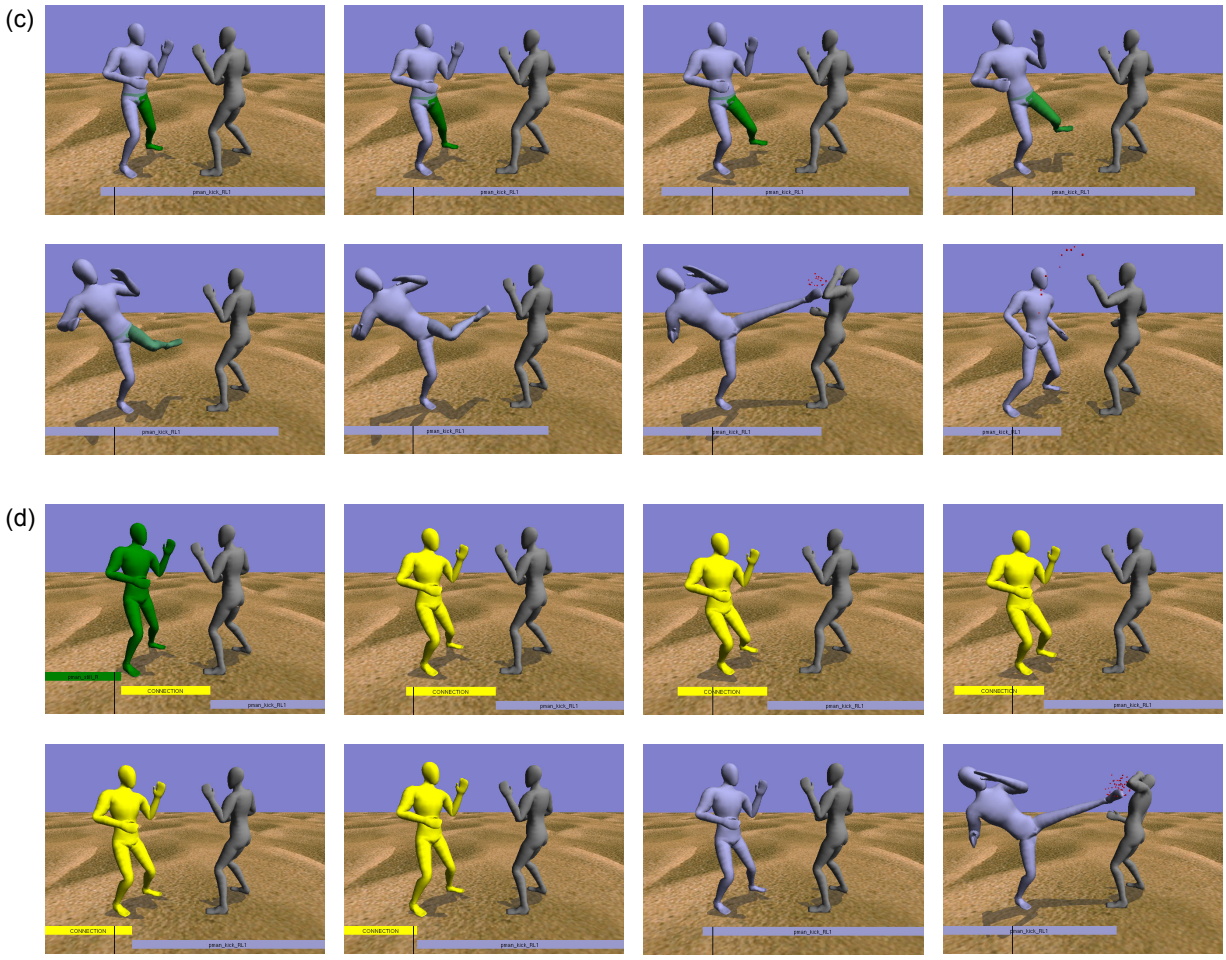

Figure 3.9 Example of motion transition. Two kinds of transition methods are applied after Figure 3.8 (b) in order to realize a smooth transition. (c) motion blending in which the motion of left foot is smoothly blended to the kick action. (d) motion connection in which an connective action is executed to move the position of the left foot.

3.7. Summary

We have described an approach that generates continuous dynamic character animation in response to physical interactions in virtual environments. A novelty of the system presented in this chapter is to combine dynamic motion synthesis in angular space with dynamic motion control of angular acceleration. This system solves the problems of the standard dynamic simulation approach in which the motion control in torque space makes it difficult to track a target motion correctly or to control the character's multiple joints cooperatively. Our approach makes use of the framework of current computer games and existing motion data.

Presently in real-time computer animations, the motions of the characters are very limited and monotonous. Moreover, unnatural results are sometimes generated when the characters interact with each other. We have presented a solution for this problem, which has become a very important issue in the generation of dynamic character motions. In the future, fluent action scenes that currently are seen only in action movies will be presented in real-time in computer games and other interactive applications. We think that our method will become the first step of the next generation of the interactive human animation.

Chapter 4: Fast and Plausible Cloth Simulation

In this chapter, a novel technique for real-time cloth simulation is presented. The method combines dynamic simulation and geometric techniques. Only a small number of particles (a few hundred at maximum) are controlled using dynamic simulation to simulate global cloth behaviors such as waving and bending. The cloth surface is then smoothed based on the elastic forces applied to each particle and the distance between each pair of adjacent particles. Using this geometric smoothing, local cloth behaviors such as twists and wrinkles are efficiently simulated. The proposed method is very simple, and is easy to implement and integrate with existing particle-based systems. We also describe a particle-based simulation system for efficient simulation with sparse particles. The proposed method has animated a skirt with rich details in real-time.

4.1. Introduction

Physics based cloth simulation is an important current topic in computer animation. At present the most efficient and commonly used approach is particle-based cloth simulation. In particle systems, clothes are modeled as mass-distributed particles that are grouped into a fixed topology. In addition, forces that work on the particles, such as the fabric, gravity, and air forces are also modeled. Then, by solving a differential equation based on Newton's law, particle motions are computed. Currently this method is mainly used in off-line processes, such as for making motion pictures and real-time simulation is still difficult. One major problem is the large number of particles required for accurate simulation. Typically, from a few hundred to a hundred thousand particles are required to simulate a garment. Such a large number of particles needs a lot of computational time per step of the dynamic simulation and requires small time steps to avoid numerical instability. Although many techniques have been developed

for fast and robust simulation, dynamic simulation requires too much computational time to animate clothes in real-time.

In this chapter, a novel technique for real-time cloth simulation is presented. The method combines dynamic simulation and geometric techniques. Only a small number of particles (a few hundred at maximum) are controlled using dynamic simulation to simulate global cloth behaviors such as waving and bending. The cloth surface is then smoothed based on the elastic forces applied to each particle and the distance between each pair of adjacent particles. Using this geometric smoothing, local cloth behaviors such as twists and wrinkles are simulated in a lower computational cost (Figure 4.1).

The proposed method assumes that the motion of a cloth is computed using a coarse mesh. Therefore, the application of the proposed method to waving cloths, such as skirts or jackets, is appropriate because tight garments, such as sleeves or tight pants, need a fine mesh for dynamic simulation. In this paper, animation of a skirt is presented as an example. The proposed approach that has potential uses in computer games, real-time animation, virtual environments, and virtual fashion shows.

The reminder of this chapter is organized as follows: Section 4.2 reviews related work. Section 4.3 describes a geometrical smoothing method for generating a fine cloth mesh from a small number of particles. In Section 4.4, a particle-based dynamic simulation system is described. An experimental result is presented and discussed in Section 4.5. Finally, Section 4.6 concludes this chapter and states the future work.

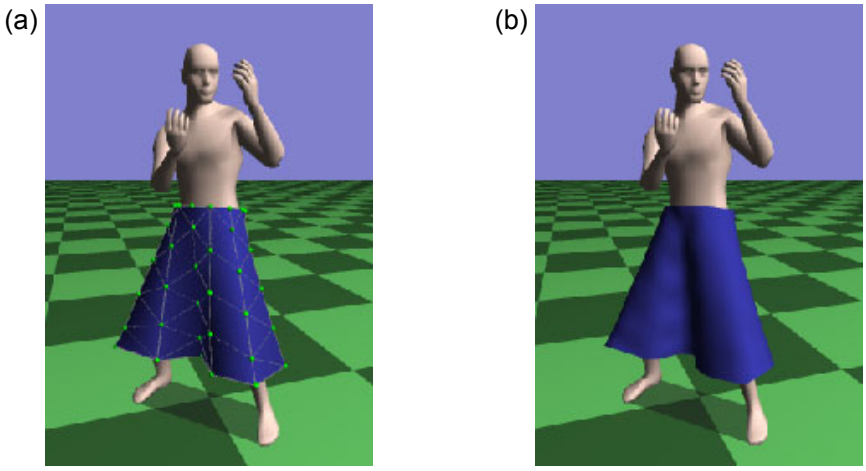

Figure 4.1 An example of the proposed approach. (a) A small number of particles (66 particles in this case) are controlled using dynamic simulation. (b) The cloth surface is then smoothed using geometrical techniques (1,944 faces were rendered).

4.2. Related Work

Many researchers have developed numerical models to simulate fabric objects [40][41][6][5][45]. One of the important recent advances is implicit integration method by Baraff [2] for solving the differential equation of particle systems. The implicit integration ensures stability with large time steps compared to explicit integration and this makes it possible to efficiently compute cloth simulation.

There is a tradeoff between the complexity of a cloth model and computational speed. The larger number of particles are used to model a cloth, the more computational time is required. Moreover, we need small time steps in numerical integration to avoid numerical instability when we use a large number of particles. Although implicit integration [2][48] and more simplified techniques of the implicit integration [8][18] largely improve the computational speed, it is still difficult to simulate clothes that have enough complexity to capture the details of clothes in real-time.

Generating a smooth surface from a coarse model that has lower complexity can be one solution for this problem. Smoothing of surfaces is a common technique in computer graphics. Many methods have been developed for modeling and compressing meshes, such as subdivision faces, parametric surfaces, and so on. These methods can generate a smoothed cloth surfaces from a coarse mesh [18][7]. However, just smoothing is not enough for generating a realistic cloth surface. Some kinds of geometric method specifically for cloth surfaces are required.

Some researchers have developed heuristic models for computing a static cloth shape [28]. These methods combine dynamics and geometric techniques. Taillefer [39] proposed a computational model for modeling the folds of a hanging cloth. Dhande et al. [9] modeled a wept surface. Kunii et al. [22] developed a formulation and deformation method for wrinkles. However, the deformation of wrinkles does not consider physical interactions with the underlying body. In these hybrid methods, an appropriate geometric cloth model is used to compute a specific and static cloth shape. Therefore, it is difficult to integrate them with an existing dynamic simulation system and animate them interacting with the changing environment. In this paper, we present a geometrical technique that is generically designed and easily integrated with standard particle-based cloth simulation systems.

Recently, Volino et al. [47] proposed a method for generating wrinkles using height maps. The method was expanded by Hadap et al. [14]. Using good looking wrinkle patterns provided by the designer as height maps, varying wrinkles are produced by modulating the height maps

based on the compression of the cloth face similar to our method and the multi-layering of the height maps. Although this method generates realistic wrinkles, the pattern is basically designed for a specific posture. Therefore, it may produce a unnatural result under unexpected conditions, such as nonstandard posture of the underlying body, a strong wind, other external forces, constraints and so on. The method proposed in this paper produces a curved cloth surface based on the condition, is generically designed, and does not need extra data such as wrinkle patterns.

4.3. Geometric Smoothing for Cloth Surfaces

In this section, a geometric smoothing method is described. The smoothing method generates a fine cloth mesh based on a coarse cloth mesh that consists of a small number (typically a few hundred) of particles that are grouped into triangular faces. In smoothing a cloth surface, this method simulates the cloth's local behaviors within each original triangular area rather than just smoothing angular faces.

The proposed smoothing method consists of two stages: surface generation and deformation. Surface deformation involves particle normal control and edge length control. For surface generation, an existing method is used. Therefore, the original part of the smoothing method is the surface deformation method. In each simulation step, a cloth surface is rendered by the following steps:

1. The positions of all particles of the coarse cloth mesh are computed using dynamic simulation.
2. Surface generation. A cloth surface is generated by smoothing the triangular faces of the coarse cloth mesh.
3. Particle normal control. The generated surface is deformed so that its shape reflects the elastic forces applied to the particles.
4. Edge length control. The generated surface is deformed so that it keeps its original dimension by maintaining the length of each edge.
5. Triangular tessellation. The cloth surface is rendered as a number of small triangular faces.

4.3.1. Smoothing Triangular Faces

In the proposed method, PN triangles [44] are used to smooth triangular faces in a coarse cloth mesh. PN triangles substitute a three-sided cubic Bézier patch for each triangular face

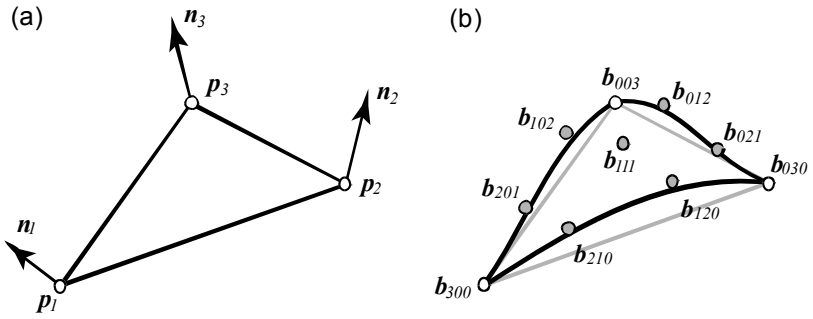

Figure 4.2 PN triangles replace a triangular face by a three-sided cubic Bezier patch computed from the three positions and normals. The cubic Bezier patch is represented by coefficients or control points (B_{ijk}).

facilitating control of the smoothing mesh using the normal of the vertices (Figure 4.2). In addition, PN triangles ensure that the generated surface matches the original mesh on each vertex. These features are suitable for the control of the shape of clothes. Using PN triangles, the ten control points for representing a three-side cubic Bézier patch are computed from three positions, and from the normals of the vertices of the triangular face as illustrated by Figure 4.2.

Many smoothing techniques have been proposed for modeling and compressing geometrical meshes. However, most smoothing techniques such as subdivision surfaces (e.g. Catmull-Clark) and parametric surfaces (e.g. NURBS) generate a smoothed mesh from the topology of a coarse mesh and the positions of the vertices without considering their normals. Moreover, the generated surfaces are so smooth that they are away from the original vertices. These features make it difficult to control the shape of the produced surface. This is why we choose the PN triangles.

DeRose et al. [7] applied Catmull-Clark subdivision surfaces to cloth simulation. However, the subdivision surfaces were used for modeling clothes and the fine cloth meshes are produced for dynamic simulation. They discussed the problem regarding computing the continuous fabric parameters in a subdivision surface. Biermann et al. [3] proposed a subdivision surface technique with normal control. Although the shape of the generated surface can be controlled by the normal of its vertices, the generated surface is still too smooth and difficult to control. Volino et al. proposed SPHERIGON [46] that generates curved polygons based on the position and normal of the vertices of an original flat polygon in a similar way to PN triangles. SPHERIGON provides us the same facilities as do PN triangles that control the shape of a curved surface with their normal and ensure a curved face match on vertex positions. Therefore,

SPHERIGON can also be used with the proposed surface deformation method. We have chosen PN triangles for first implementation since it just seems to be faster than SPHERIGON which involves a normalization process in computing a point on a curved polygon, and hardware accelerations for PN triangles [44] are expected. A more explicit comparison of the two methods in both visual results and computational time is a future topic.

4.3.2. Particle normal control

In particle-based systems, particles usually have no orientation. Only the position and velocity of particles are used in dynamic simulation. When the cloth faces are rendered using Gouraud shading, the particle normal is simply computed by taking an average of the normals of the adjacent faces.

Szeliski and Tonnesen [37] introduced oriented particles for a particle-based surface modeling system. They used oriented particles to keep the shape of surfaces in dynamic simulation. However, this approach makes the dynamic simulation more complex, and it is difficult to integrate with existing particle-based simulation systems. In our method, particles have no explicit orientation in dynamic simulation, and the particle normal is computed in the surface deformation step.

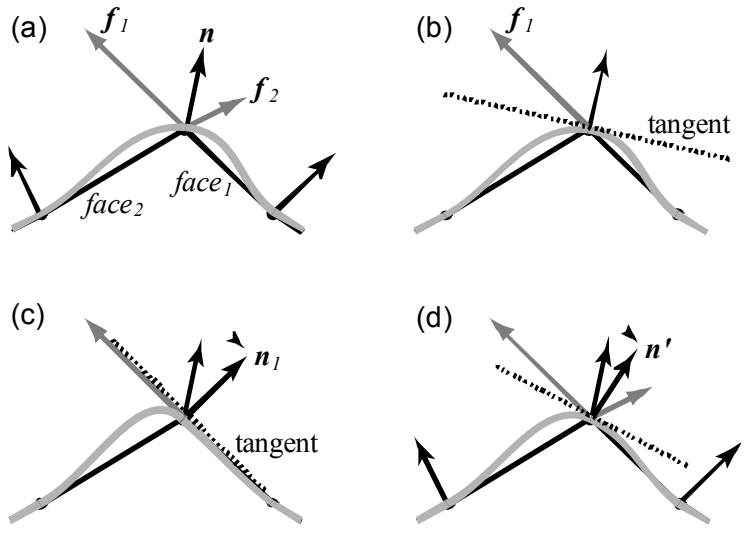

Figure 4.3 Normal control in section view. (a) The curved surface is computed based on the original normal using PN triangles. The elastic forces f_1 and f_2 are applied to p from $face_1$ and $face_2$ respectively. (b) The energy on p from f_1 is assumed to be proportional to inner product between f_1 and n. (c) f_1 makes the tangent proportional to f_1, and n proportional n_1 which is the normal of f_1. (d) The normal n' is computed by taking a weighted average.

The proposed method computes particle normal so that the curved patch reflects the in-plane forces that are applied to each particle. Many force models such as elastic, bending, shearing, and twisting forces have been proposed [28]. Of these, elastic forces are the strongest and are responsible for the in-plane deformation of the clothes. Therefore, only elastic forces are considered in our implementation. Elastic forces are usually defined as a spring model in which a spring force works along each edge as follows.

$$f_{ij} = k_{elastic} \frac{\left|\mathbf{p}_i - \mathbf{p}_j\right|}{L_{ij}} \left(\left|\mathbf{p}_i - \mathbf{p}_j\right| - L_{ij} \right), \tag{4.1}$$

where i, j is a pair of adjacent particles, $\mathbf{p}_i$ and $\mathbf{p}_j$ are the current positions of the two particles, L_{ij} Lij is the distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ in their rest positions, and $k_{elastic}$ is the spring coefficient. The elastic force is computed for each edge and it works to make the distance between the two side particles close to the original distance.

Normal control is based on the principle of energy minimization, as illustrated in Figure 4.3. When the position of a particle is fixed, the normal on the particle is computed so as to minimize the energy on the particle (Figure 4.3 (a)). When a particle receives an elastic force from an adjacent face, it can be assumed that the energy on the particle is proportional to the inner product between the elastic force and the normal vector on the particle (Figure 4.3 (b)).

$$E_i^{\,j} = \mathbf{n_i} \cdot \mathbf{f_i^j}. \tag{4.2}$$

The more parallel the normal vector is to the elastic force, the more the energy on the particle from the elastic force. Consequently, an elastic force on a particle works to make the tangent on the particle parallel to the elastic force (Figure 4.3 (c)). Because a particle usually receives some elastic forces from all adjacent faces, to minimize the energy on the particle, the particle normal is computed by taking a weighted average of the normals of all adjacent faces with the magnitude of the elastic forces applied from each face (Figure 4.3 (d)) as follows.

$$\mathbf{N}_i = \sum_{j=1}^{k} \mathbf{n}_j \frac{\left|\mathbf{f}_i^{\,j}\right|}{\sum_{j=1}^{k} \left|\mathbf{f}_i^{\,j}\right|}, \ \mathbf{n}_i = \frac{\mathbf{N}_i}{\left|\mathbf{N}_i\right|}, \tag{4.3}$$

where k is the number of adjacent faces, $\mathbf{n}_j$ is the normal on the jth face, $\mathbf{f}_i^{\,j}$ is the elastic force that applied to the ith particle from the jth adjacent face, and $\mathbf{f}_i^{\,j}$ is computed by taking the average of the two elastic force on the two edges on the jth face.

$$\mathbf{f}_i^{\,j} = \mathbf{f}_{ia} + \mathbf{f}_{ib}, \tag{4.4}$$

where *i*, *a*, *b* are three particles in the *j*th triangle, and $\mathbf{f}_{ia}$ and $\mathbf{f}_{ib}$ are computed by equation .

4.3.3. Edge Length Control

In particle-based dynamic simulation, the length of an edge on a cloth mesh sometimes varies from its original length in a resting state because of the movements of particles and the constraints on the particles. When using a coarse cloth mesh, the variation of each edge length has a large influence on the entire cloth dimension because the initial edge length is longer than when using a fine cloth mesh. Therefore, each edge length is controlled to keep its original edge length.

When the length of a cubic Bézier curve is shorter than its original length (Figure 5(b)), the shape of the curve is deformed to make it longer while keeping the positions of the two terminal particles. To control a curve length, we first divide the curved patch into four curved patches (Figure 4.4). This divides each curve into two curves. The position of the midpoint $\mathbf{p}$ on the curve is then moved in the normal direction $\mathbf{n}$. The position $\mathbf{p}$ and the normal $\mathbf{n}$ on the midpoint is obtained from the original curved patch (Figure 4.5 (c)).

$$\mathbf{p}' = \mathbf{p} + h\mathbf{n} \quad (h > 0). \tag{4.5}$$

In equation (4.5), h is computed so that the curve length is close to the original edge length. The divided four patches are also computed from the three corner particles and the three midpoints by again using PN triangles (Figure 4.5 (d)).

Unfortunately, the length of a cubic Bézier curve cannot be computed in a closed form. Therefore, the length should be computed numerically. By subdividing a Bézier curve into a number of segments and summing the length of all segments, we can approximate the curve

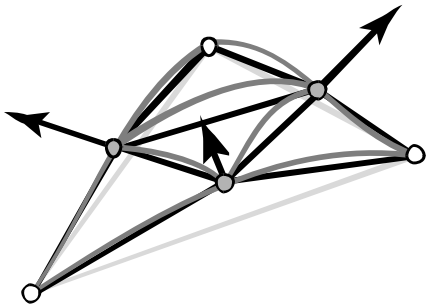

Figure 4.4 A cubic Bézier patch (Figure 2(b)) divided into 4 cubic Bézier patches. The 3 midpoints are obtained from the original PN triangle.

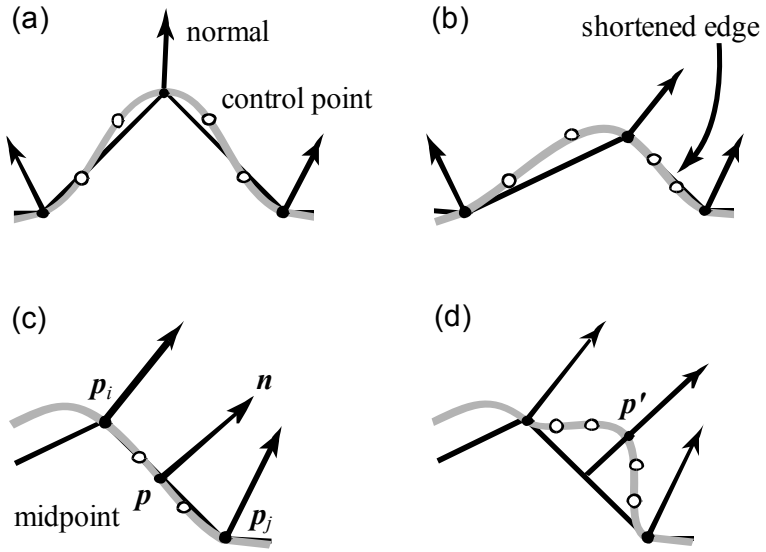

Figure 4.5 Edge length control in section view. (a) Original edges in the at rest length. (b) When an edge is shortened, (c) the midpoint of the shorter curve is controlled (d) so that the curve length is close to the original edge length.

length. Similarly, the edge length is controlled numerically and incrementally. Using a recursive bisection method [32], h in equation (4.5) is computed. We found the following equation to be a good approximation of the control distance as an initial guess.

$$h_0 = k\left(L_{ij} - \left|\mathbf{p}_i - \mathbf{p}_j\right|\right), \tag{4.6}$$

where L_{ij} is the initial edge length and k is an empirically determined parameter.

However, when a curve length is longer than its original edge length, it is difficult to make it shorter by using this approach. Therefore, in our system, instead of curve deformation in geometric smoothing, spatial constraints are applied in a dynamic simulation to avoid such elongated edges.

4.4. Dynamic Simulation with Sparse Particles

The smoothing method presented in the previous section is very simple, and is easy to integrate with an existing particle-based system because only particle positions are required for generating a smoothed surface. Many methods for particle-based simulation have been developed [28][49], but developing a successful particle-based system is still difficult. In developing a particle-based system, it is necessary to choose an appropriate method from many

alternative methods for the purpose, trading-off between accuracy, speed and stability [2]. This section describes the chosen methods in our system and how they were modified to work efficiently with a small number of particles and the proposed smoothing method.

4.4.1. Numerical Integral method

Numerical integration is the essence of particle-based simulation and is the most important factor in the design of a particle-based simulation system. Using numerical integration, a partial differential equation is solved to compute the time-varying state of the particles.

There are two major methods for numerical integration: explicit and implicit. An explicit method was chosen for our implementation. Recently, implicit methods are widely used in many systems [2][48], because they ensure stability which is the most important issue in interactive systems, and allow large steps instead of losing some accuracy. However, they have difficulties handling non-continuous constraints in dynamic simulation. On the other hand, explicit methods are a classic but accurate integral method. Each step of an explicit method is faster than an implicit method. Moreover, they are easily integrated with non-continuous constraints. However, a larger number of particles is used, so an explicit method requires smaller time steps. Therefore, as Baraff described [2], when many particles are used, explicit methods are slower than implicit methods and they become unstable.

However, when, only a few hundred particles are used, as here, an explicit method works faster than an implicit method. In addition, an explicit method is easily integrated with spatial constraints on penetrating particles and elongated edges that are introduced in our dynamic simulation. Therefore, an explicit numerical integral method, technically 4th-order Runge-Kutta, is currently used in our system. Further discussions about numerical integration are to be found in the literature [32][1].

4.4.2. Cloth Model

For modeling a cloth surface, triangular faces are used in our proposed method to apply the proposed smoothing method as explained in section 4.3. Elastic and bending force are introduced as internal cloth forces. The elastic force is described in section 4.3.2. The bending force is computed for each edge, and prevents bending between the two adjacent faces. In addition to the fabric forces, additional forces such as gravity, air velocity and damping forces are also introduced. In some particle-based systems, square faces are used for representing a

cloth instead of triangular faces. Here, diagonal forces such as shearing and twisting forces are used to prevent skews [28]. However, these are not introduced in our system because elastic and bending forces in triangular faces are considered to have similar affects on cloth surfaces.

4.4.3. Elastic Forces and Constraints

Clothes strongly resist stretching but allow bending, sharing or twisting. To realize such stiffness, the spring coefficient $k_{elastic}$ in equation (4.1) should be of a large value. Otherwise, the cloth changes length and acts like a rubber or spring. Such a large coefficient value causes instability in numerical integration and requires very small time steps in dynamic simulation making the simulation slower. However, our system allows a comparatively small value of $k_{elastic}$ because the edge length is maintained at its original length and large elastic forces are not required. Therefore, our system allows large time steps in dynamic simulation making the simulation faster.

In our system, two different methods are applied for handling shortened and elongated edges. As explained in section 4.3.3, when an edge length is shortened, the edge curve is deformed to keep the original edge length in geometric smoothing. However, this method cannot handle elongated edges. Therefore, when an edge length is elongated, spatial constraints are applied on the elongated edges in dynamic simulation in a way similar to that proposed by Provot [33]. If a curve length is longer than its original edge length, the positions of the two end-points of the elongated edge are directly altered to close the distance between the two particles.

4.4.4. Collision Detection and Constraints

Collision detection and response are also important issues in particle-based systems. To prevent a cloth penetrating the underling body contacts between the cloth and the body object should be detected and resolved. In our system, a simple state correction approach is used as explicit methods work well with this approach, while implicit methods require some extra techniques [2][48] to maintain stability.

If a fine cloth mesh with a large number of particles is used, the collisions between the cloth and any solid object can be handled by keeping the particles out of the solid object [2][10]. However, when using a coarse cloth mesh, as in our case, a cloth face may intersect with a solid object although the corner particles of the cloth face are kept away from the solid object. Therefore, in our system, we need to handle both vertex penetration and face intersection. Even though this

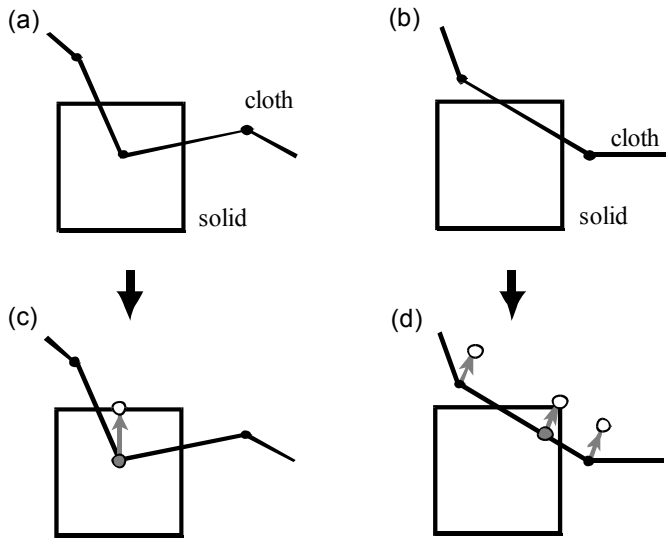

Figure 4.6 Collision detection and penetration avoidance. (a) Cloth vertex and solid penetration. (b) Cloth face and solid intersection. (c) Cloth particle or (d) cloth face position is corrected and constrained.

needs extra computational time, using a coarse mesh makes collision detection and the related response faster.

For collision detection, a proximity query package (PQP) [23] a collision detection library is used in our system. PQP reports an intersection between a cloth face and a solid face. When an intersection is reported, an initial determination is made whether a vertex of the cloth face penetrates into the solid (Figure 4.6 (a)) or the cloth face intersects with the solid but its vertices do not penetrate (Figure 4.6 (b). When a cloth particle penetrates inside the solid, the position of the particle is corrected to the nearest point on the solid face (Figure 4.6 (c)). When a cloth face intersects the object, the cloth face including its vertices is moved to the farthest point on the solid face (Figure 4.6 (d)). When a particle contacts a solid face, the acceleration of the particle is directly constrained in dynamic simulation so that the relative velocity between the particle and the contacting face maintains the normal direction of the contacting face.

Collision detection and penetration avoidance is performed based on the original flat triangles. Therefore, the curved faces may penetrate into a solid object or the cloth itself. In terms of cloth-solid collisions, it is not a large matter because any generated surface basically matches the original mesh. To handle cloth-cloth collisions, collision detection between curved faces is required. Although a penalty-force approach could be a solution, this would involve large computational costs, and so a cloth-cloth collision detection method is yet to be introduced in our system.

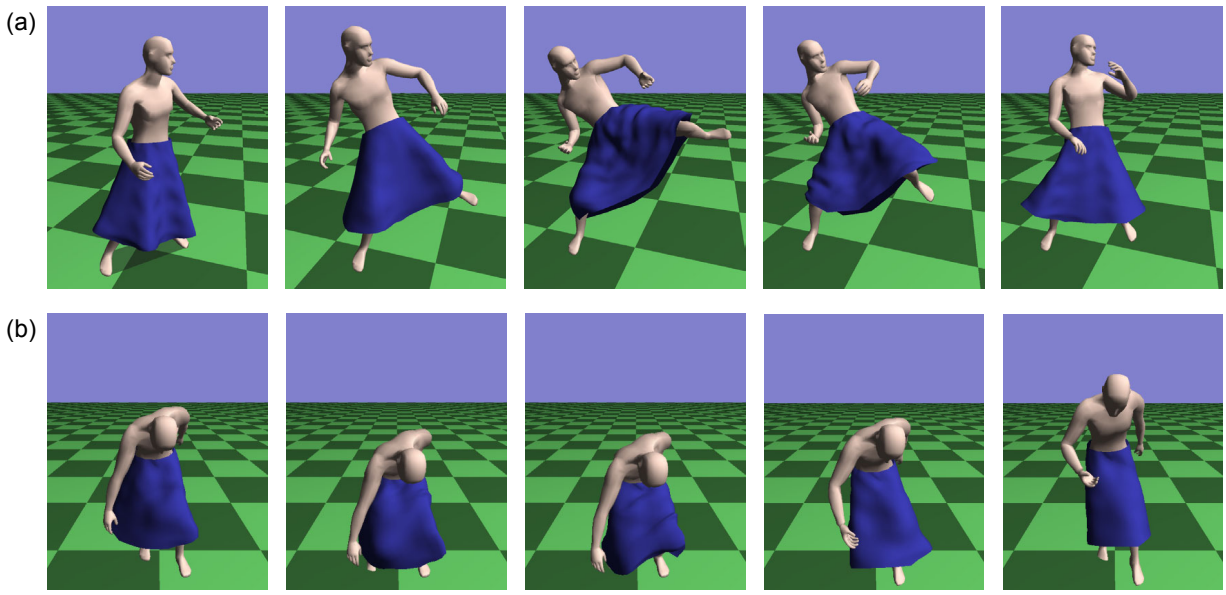

Figure 4.7 Images from animation of a skirt with (a) kick motion and (b) picking up motion.

4.5. Results and Discussion

In this section an experimental result generated using the proposed method is presented and discussed. For the first test, we chose a skirt. Figure 4.7 shows the images from generated animation of a skirt. The original skirt mesh used for the dynamic simulation consists of 66 particles grouped into 108 triangles. For the smoothed surface, each triangular face is rendered as 4 cubic Bézier patches using 18 small flat triangles. Consequently, 1,944 faces are tessellated using PN triangles and rendered. The skirt model is generated procedurally and the figure animated in the motion-captured sequence of kicking.

Efficiency of the proposed method. Figure 4.8 shows smoothing stages in a frame of the animation. Figure 4.8 (b) is the smoothed surface using only PN triangles and Figure 4.8 (c) is the smoothed surface using our proposed method. Figure 4.8 (d) shows the particle normal and edge length control used in Figure 4.8 (c). Sticks in yellow and orange from the particles show the original and modified normals, respectively, and purple sticks show the edge length control. Although it is difficult to judge the visual results, it is found that the smoothed surface using our proposed method has rich details such as folds and twists around the left thigh in Figure 4.8 (c), leading we believe to improved results. It is also found that the particle normal and edge length control are applied mainly on the left leg and on other shortened parts. When a cloth is

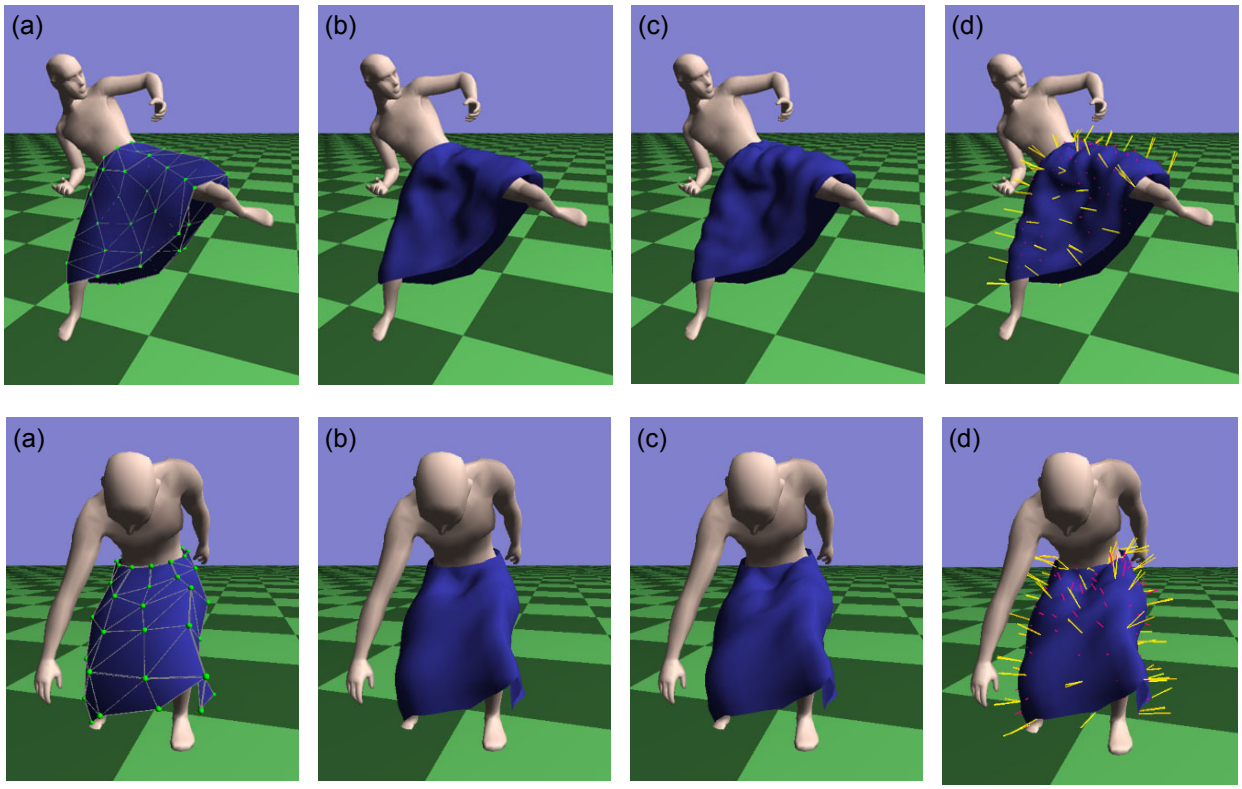

Figure 4.8 Smoothing surface in a frame of the animation. (a) Original mesh. (b) Smoothed surface using only PN triangles. (c) Smoothed surface using our proposed method. (d) Visualization of the surface deformation.

relaxed and hanging naturally, the difference between the generated surface using only PN triangles and that using our proposed method is small. Our proposed method works more effectively when a part of the cloth is exposed to external constraints or is shortened.

Computational time and its optimizations. In the experiments, the duration between each simulation step was 1/30 seconds fixed. Each frame was divided into three steps in an explicit Eular integration. As a result, a 30 Hz animation was generated in real-time on a PC (PentiumIII 800MHz CPU + GeForce2MX video card). The computational time is shown in Table 1. For the proposed smoothing method, 2.4 extra milli-seconds were used compared to when using only PN triangles. This means that, by allowing a few extra mill-seconds per a hundred faces in addition to that allowed for the PN triangles, more realistically deformed cloth surfaces are achieved. Although the triangular tessellation is currently computed in the CPU and the tessellated triangles are stored in memory, hardware supports for PN triangles [44] could accelerate the smoothing method. Since the program used in our experiments has not been optimized yet, the computational time in Table 4.1 could become faster after optimization

for both numerical computation and rendering.

Used method	Computational time (ms / frame)				Speed (fps)
	Integration	Collision	Smoothing	Total	
Proposed method	3.5	14.7	6.4	23.7	29.8
PN triangles	3.6	14.6	4.0	21.9	32.1

Table 4.1 Maximum computational time per frame in the animation of skirt when our proposed method or only PN triangles were used. The computational time for cloth simulation (numerical integration, collision handling, and smoothing surface) was about 60% of the entire computational time.

4.6. Summary and Future Work

The smoothing method presented in this paper is very simple and efficiently adds rich details on a coarse cloth mesh. However, the generated surfaces are slightly limited. The edge length control adds at most one fold on the center of a shortened edge and the direction of the fold is determined from just the normal of the two end points. To generate more realistic surfaces, the number, position, and direction of the folds should be controlled. This we think might be possible by taking into account bending forces and compression speed. For more subtle and more wrinkles, wrinkle patterns provided by a designer [47] or pre-computed using dynamic simulation [15], could be appropriate and could be integrated with our proposed method. However, such wrinkle patterns require extra modeling processing or pre-computation time, and cannot produce appropriate wrinkles under unexpected conditions. The goal of our research has been to provide a geometric method that is easily usable and generates cloth surfaces under any conditions using heuristic methods that capture the characteristics of clothes rather than using static and expensive wrinkle patterns.

We intend to apply our proposed method to garments such as jackets, shirts and pants in which the interaction with the underlying body is more complex and critical. To simulate a difficult interaction, such as the figure putting on a garment or picking up a cloth, while keeping a fast and robust simulation, an adaptive technique that inserts and removes appropriate particles into the cloth mesh would be required.

Currently only coarse cloth meshes achieve real-time performance. Many on-line applications such as computer games require real-time cloth simulation at a small computational cost. We believe that our proposed methods that add detail on a physically simulated coarse cloth mesh are an effective and reasonable approach.

Chapter 5: Conclusion

5.1. Summary

In this work, two methods are developed for real-time physics-based human animation

- Dynamic motion control of human figure in response to environmental physical interaction.
- Real-time cloth simulation that has plausible appearance of clothes.

The methodology of this work is to combine kinematics and dynamics to ensure both physically-soundness and controllability.

In order to apply this methodology on above subjects, two major issues had to be solved.

- To merge kinematics-based method and dynamics-based method seamless by finding appropriate parameters to control both aspects.
- To build a simple dynamics model by extracting an essence from an existing dynamics model that is complex and requires much computation times.

For dynamic motion control of human figure, we have solved them by adopting a motion control in angular acceleration space and by developing a control model that is specified to human-like articulated figure.

For fast and plausible cloth simulation, we have solved them by employing two levels of cloth models and by modifying the existing particle-based simulation method so that it works well with dynamic simulation of small numbers of particles and geometric surface generation method.

We believe that a novel methodology has been established by this work.

5.2. Future Work

We have developed novel techniques for real-time animation of human motion control and cloth simulation. We believe that they broke through a major unsolved issue on each field. However, many problems remain to be solved. For dynamic motion control, more stabilized control algorithm and some technique for precious control of the results of dynamic control would be required. For cloth simulation, more accurate geometric method is needed to handle contact between cloth surface and the underlying human body and to generate more rich detailed cloth surface.

Although we have built a physics based model and made some resulting animation, there is no guarantee that the model is correct and the animation is realistic. There is no way established of judging the reality of computer generated animation and this is a major problem in the field. The study of human behavior and recognition model can be a promising approach. We are going to research human motion data that are collected by using motion capture devices in order to build more sophisticated motion control scheme and human behaviors.

The ultimate goal of this work is to combine all techniques that are developed based on the methodology and to achieve a system for interactive animation of virtual humans that is as realistic as off-line animations in motion pictures that are created by skilled animators and their large efforts.

Bibliography

[1] David Baraff and Andrew Witkin , "Physically Based Modeling: Principles and Practice", SIGGRAPH '97 Course Notes,1997.

[2] David Baraff, and Andrew Witkin, "Large Steps in Cloth Simulation", Proc. of SIGGRAPH '98, pp. 43-54, 1998.

[3] Henning Biermann, Adi Levin, and Denis Zorin, “Piecewise Smooth Subdivision Surfaces with Normal Control”, Proc. of SIGGRAPH 2000, pp. 113-119, 2000.

[4] Ronan Boulic, Ramon Mas-Sanso, and Daniel Thalmann, “Complex Character Positioning Based on a Compatible Flow Model of Multiple Supports”, IEEE Transactions on Visualization and Computer Graphics, Vol. 3, No. 3, pp. 245-261, July-September 1997.

[5] David E. Breen, Donald H. House, and Michael J. Wozny, "Predicting the Drape of Woven Cloth Using Interacting Particles", SIGGRAPH '94 Proceedings, pp. 365-372, 1994.

[6] Michel Carignan, Ying Yang, Nadia Magnenat-Thalmann, and Daniel Thalmann, "Dressing Animated Synthetic Actors with Complex Deformable Clothes", Computer Graphics (SIGGRAPH '92 Conference), Vol. 26, No. 2, pp. 99-104, July 1992.

[7] Tony DeRose, Michael Kass, and Tien Truong, “Subdivision Surfaces in Character Animation", Proc. of SIGGRAPH '98, pp. 85-94, 1998.

[8] Mathieu Desbrun, Peter Schroder, and Alan Barr. "Interactive Animation of Structured Deformable Objects", Graphics Interface '99, pp. 1-8, 1999.

[9] S. G. Dhande et al., “Geometric Modeling of Draped Fabric Surface”, Proc. of IFIP International Conference on Computer Graphics, pp. 349-356, 1993.

[10] Bernhard Eberhardt, Andreas Weber, and Wolfgang Strasser, “A Fast, Flexible, Particle-System Model for Cloth Draping”, IEEE Computer Graphics and Applications, vol. 16, no. 5, pp.52-59, September 1996.

[11] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos, “Composable Controllers for Physics-Based Character Animation”, SIGGRAPH 2001, pp. 251-260, 2001.

[12] Petros Faloutsos, “Composable Controller for Physics-Based Character Animation”, PhD thesis, University of Toronto, DCS, Toronto, Canada, 2001.

[13] Roy Featherstone, Robot Dynamics Algorithms, Kluwer, 1987.

[14] Sunil Hadap, Endre Bangerte, Pascal Volino, and Natia Magnenat-Thalmann, “Animating Wrinkles on Clothes”, Proc. of IEEE Visualization ’99, pp. 175-182, 1999.

[15] Daniel L. Herman, “Using Precomputed Cloth Simulations for Interactive Applications”, SIGGRAPH 2001 Sketches and Applications, p. 273, 2001.

[16] Jessica K. Hodgins, and Nancy S. Pollard, “Adapting Simulated Behaviors for New Characters”, SIGGRAPH '97 Proceedings, pp 153-162, 1997.

[17] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien, “Animating Human Athletes”, SIGGRAPH '95 Proceedings, pp. 71--78, 1995.

[18] Young-Min Kang, Jeong-Hyeon Choi, Hwan-Gue Cho, and Do-Hoon Lee, "An Efficient Animation of Wrinkled Cloth with Approximate Implicit Integration", The Visual Computer, Vol. 17, No. 3, pp. 147-157, 2001.

[19] Hyeongseok Ko, and Norman I. Badler, “Animating Human Locomotion with Inverse Dynamics”, IEEE Computer Graphics and Applications, Vol. 16, No. 2, pp. 50-59, 1996.

[20] Evangelos Kokkevis, Dimitris Metaxas, and Norman I. Badler, “User-Controlled Physics-Based Animation for Articulated Figures”, Proceedings of Computer Animation '96, 1996.

[21] Taku Komura, Yoshihisa Shinagawa, and Tosiyasu L. Kunii. “Creating and retargeting motion by the musculoskeletal human body model”, The Visual Computer, Vol. 16, pp. 254-270, 2000.

[22] T. L. Kunii, and H. Gotoda, “Modeling and Animation of Garment Wrinkle Formation Processes”, Proc. of Computer Animation ’90, pp. 131-147, 1990.

[23] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha, "Fast Proximity Queries with Swept Sphere Volumes", Technical report TR99-018, Department of Computer Science, University of N. Carolina, Chapel Hill, 1999.

[24] Philip Lee, Susanna Wei, Jianmin Zhao, and Norman I. Badler, “Strength Guided Motion”, Computer Graphics (SIGGRAPH '90 Proceedings), Vol. 24, No. 3, pp. 253-262, 1990.

[25] Jehee Lee, and Sung Youg Shin, “A Hierarchical Approach to Interactive Motion Editing for Human-like Figures”, SIGGRAPH '99 Proceedings, pp. 39-48, 1999.

[26] Sheue-ling Lien and James T. Kajiya, "A Symbolic Method for Calculationg the Integral Properties of Arbitrary Nonconvex Polyhedra", IEEE Computer Graphics & Applications, pp.35-41, October 1984.

[27] Matthew Moore, and James Wilhelms, "Collision Detection and Response for Computer Animation", Computer Graphics (SIGGRAPH '88 Proceedings), Vol. 22, No. 3, pp. 289-298, 1988.

[28] Hing N. Ng and Richard L. Grimsdale, “Computer Graphics Techniques for Modeling Cloth”, IEEE Computer Graphics and Applications, Vol. 16, No. 5, pp.28-41, September 1996.

[29] Joseph O'Rourke, "Computational Geometry in C (Second Edition)", Cambridge University Press, 1998.

[30] Abhilash K. Pandya, James C. Maida, Ann M. Aldridge, Scott M. Hasson, and Barbara J. Woodford, "The Validation of a Human Force Model To Predict Dynamic Forces Resulting From Multi-Joint Motions", Technical Report 3206, NASA, Houston, Texas, 1992.

[31] Zoran Popovic, and Andrew Witkin, “Physically Based Motion Transformation”, SIGGRAPH '99 Proceedings, pp 11-20, 1999.

[32] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, "Numerical Recipes", Cambridge University Press, 1992.

[33] Xavier Provot, “Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior”, In Proc. Graphics Interface '95, pp. 147-154, 1995.

[34] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. “Efficient Generation of Motion Transitions using Spacetime Constraints”, SIGGRAPH '95 Proceedings, pp. 147-154, 1995.

[35] Hyun Joon Shin, Jehee Lee, Michael Gleicher, and Sung Young Shin, “Computer Puppetry: An Importance-Based Approach”, ACM Transactions on Graphics, Vol. 20, No 2, pp.67-94, 2001.

[36] Harold Sun, and Dimitris Metaxas, “Automating gait generation”, SIGGRAPH 2001, pages 271-269, 2001. (NEW)

[37] Richard Szeliski, and David Tonnesen, “Surface Modeling with Oriented Particles”, Computer Graphics (Proc. of SIGGRAPH '92), Vol. 26, No. 2, pp. 185-194, 1992.

[38] Seyoon Tak, Oh-young Song, and Hyeong-Seok Ko. “Motion Balance Filtering”, Computer Graphics Forum, Vol. 19, No. 3, pp. 435-446, 2000.

[39] F. Taillefer, “Mixed Modeling”, Proc. Compugraphics, pp. 467-478, 1991.

[40] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer, "Elastically Deformable Models", Computer Graphics (SIGGRAPH '87 Conference), Vol. 21, No. 4, pp. 205-214, 1987.

[41] Demetri Terzopoulos, and Kurt Fleischer, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture", Computer Graphics (SIGGRAPH '88 Conference), Vol. 22, No. 4, pp. 269-278, 1988.

[42] Deepak Tolani, Ambarish Goswami, and Norman I. Badler, “Real-time inverse kinematics techniques for anthropomorphic limbs”, Graphical Models, Vol. 62, pp. 353-388, 2000.

[43] Michiel van de Panne, “Parameterized Gait Synthesis”, IEEE Computer Graphics and Applications, Vol. 16, No. 2, pp. 40-49, March 1996.

[44] Alex Vlachos, Jorg Peters, Chas Boyd, and Jason L. Mitchell, "Curved PN Triangles", Proc. of the 2001 ACM Symposium on Interactive 3D Graphics, pp. 159-166, 2001.

[45] Pascal Volino, Martin Courchesne, and Nadia Magnenat-Thalmann, "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects", SIGGRAPH '95 Proceedings, pp. 137-144, 1995.

[46] Pascal Volino, and Nadia Magnenat-Thalmann, “The SPHERIGON: A Simple Polygon Patch for Smoothing Quickly your Polygonal Meshes”, Proc. of Computer Animation ’98, pp. 72-78, 1998.

[47] Pascal Volino, and Nadia Magnenat-Thalmann, "Fast Geometrical Wrinkles on Animated Surfaces", Proc. of the 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media ’99 (WSCG ‘99), 1999.

[48] Pascal Volino, and Nadia Magnenat-Thalmann, “Implementing Fast Cloth Simulation with Collision Response”, Proc. of Computer Graphics International 2000, 2000.

[49] Pascal Volino, and Nadia Magnenat-Thalmann, Virtual Clothing - Theory and Practice, Springer-Verlag, 2000.

[50] M. W. Walker, and D. E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", Journal of Dynamic Systems, Measurement, and Control, Vol. 104, pp. 205-211, September 1982.

[51] Victor B. Zordan, and Jessica K. Hodgins, “Tracking and Modifying Upper-body Human Motion Data with Dynamic Simulation”, Computer Animation and Simulation '99 (Proceedings of Eurographics Workshop on Animation and Simulation '99), 1999.