Handling Motion Processing Constraints for Articulated Figure Animation

by

Yongping Luo

B.Sc. (Comp. Sci.) University of Science and Technology of China 1994

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
November 1997

0-612-24190-4

APPROVAL

Name:	Yongping Luo
Degree:	Master of Science
Title of Thesis:	Handling Motion Processing Constraints for Articulated Figure Animation

Examining Committee: Dr. James J. Weinkham
Chair

Dr. Thomas W. Calvert
Senior Supervisor

Dr. John C. Dill
Supervisor

Dr. Dave Fracchia
Examiner

Date Approved: 28 November 1997

Abstract

The animation of articulated models such as human figures remains a major challenge because of the many degrees of freedom involved and the complex structure of the models. A particular challenge is to find ways to generate new movement from existing movement sequences produced by animating or motion capture. Techniques from the image and signal processing domain can be applied to motion sequences. We have successfully applied motion multiresolution filtering, multitarget motion interpolation and waveshaping simultaneously to many or all degrees of freedom of an articulated figure. Thus we can edit, modify and blend existing motions in ways that would be difficult or impossible to accomplish otherwise.

These motion transforms result in interesting new movement sequences but may introduce constraint violations. The constraint violation problem is inherent to the motion transform techniques and cannot be avoided. In this thesis, we first show how to generally reduce the chance and degree of constraint violations by offsetting motion data and applying consistent dynamic timewarping before motion transforms. One way is provided to let the animator specify constraints, and the system checks the constraints and adjusts the constraint violations in a smooth manner before the new motion is presented to the animator. Most geometric constraint violations can be handled successfully at interactive speeds with our system. We also provide some tools for the animator to modify the new motion at a relatively low level, thus giving the animator more control over constraint violation adjustment; therefore these tools can be used to handle constraints which are hard to specify in the above system.

------ Dedicated to My Parents

Acknowledgements

I wish to thank all those who have helped me at various times during my research and writing of this thesis. I would like to convey my greatest applitude to my senior supervisor, Dr. Tom Calvert, for his guidance and support during my graduate study, and for his patient help during the writing of this thesis. Thank to Dr. John Dill and Dr. Dave Fracchia for their careful reading and detailed comments on my thesis. Thanks also to Armin Bruderlin for his kind help in the implementation of motion signal processing. Finally, my special thanks to my wife, Hui Zhang, for pushing me when I needed it most.

Contents

List of Figures

Chapter 1

Introduction

In recent years, three-dimensional computer animation has played an important role in areas such as advertising, entertainment, education, scientific visualization and simulation. Although computer animation raises technical challenges in several areas such as modeling and rendering, motion control remains a particularly difficult task, especially for the animation of complicated articulated structures such as the human figure.

Several facts make the animation of the human figure difficult and time-consuming. First, even a simple model of a human figure possesses many degrees of freedom and is capable of very complex movement. Secondly, since humans are very sensitive of each other's motion, erroneous movement can be detected easily, but the factor that causes the movement to look incorrect can be very hard to isolate. Thirdly, humans have personality and emotion that is hard to animate.

Much of the research in motion control for articulated bodies has been devoted to reducing the amount of motion specification; that is, to develop higher level controls to relieve the animator from the specificaton of tedious detail. The high level control approach suffers from lack of interactivity; i.e. it does not always produce the motion that the animator had in mind, and the predefined motion sequence may not be useful. An alternative method to generate movements of articulated figures is rotoscoping where the motion is captured from live subjects. However motion capture takes much time since a recorded movement that is not quite "right" requires the whole data capture process to be repeated.

Because of the complexity of articulated movements and the limitations of current motion control systems, it is desirable to develop tools to reuse and adapt existing motion data. For this purpose, techniques from the field of image and signal processing are adopted by A. Bruderlin[13] to provide simple and useful ways to edit, modify, blend and align motion signals of articulated figures. These techniques can also be used to produce specific motion from predefined motion data; for example, to produce motion with personality and emotion from a neutral motion. These techniques provide analytic solutions at interactive speeds and high level control over several or all degrees of freedom of an articulated figure.

There is a major problem with the use of motion signal processing: after applying these techniquess, physically impossible motion, or some motion which is not what the animator wanted, may be produced. To solve this problem, the system should be able to let the animator specify the constraints on the figure movement at a high level, and be able to detect all constraint violations and adjust the motion in a smooth way without changing the motion characteristics. In this work, we present a solution to this problem which provides the animator with a way of specifying constraints. By offsetting signal data and applying consistent dynamic time warping, some constraint violations can be prevented. Joint angle constraints can be detected and adjusted globally, non-linear constraints can be adjusted locally using inverse kinematics, and the motion can be smoothed around the violation frame by quaternion interpolation. Most geometric constraint violations resulting from the signal processing method can be prevented or adjusted successfully.

The following is the outline of the rest of this thesis:

In Chapter 2, animation of articulated figures such as the human figure is discussed to provide the background of this work, and the problem of constraint violation introduced by motion signal transformation is presented as the motivation of this work.

In Chapter 3, signal processing techniques are introduced, and some other techniques used in this work such as inverse kinematics and quaternion interpolation are discussed.

In Chapter 4, several signal processing techniques applied to motion parameters are described in detail; those techniques include motion multiresolution filtering, motion multitarget interpolation, and motion waveshaping.

In Chapter 5, we first describe two ways to prevent some constraint violations: offsetting motion data and synchronizing the motion signals to be blended. Then we give two methods, bound mapping and motion signal compression, to handle linear joint angle constraints such as joint angle limits. The major part of the chapter presents a system for specifying non-linear constraints, detecting and adjusting constraint violations, and smoothing the adjusted motion. The last part of this chapter describes signal editor and displacement mapping as complementary approaches for constraint handling.

In Chapter 6, experiments carried out on various motions are described, the results are presented, and the constraint hanling system is evualated.

In Chapter 7, the work is summarized and potential directions for future research are discussed.

Chapter 2

Background and Motivation

2.1 Hierarchical control of figure motion

Approaches to motion control in computer animation systems are moving from low-level to high-level techniques as computers become more powerful. High-level motion controls usually result in less manual work for animators; on the other hand, they also give animators less low-level control over the motion.

In traditional animation, the animator has full control over the motion by drawing the animated objects frame by frame. Computer animation tools relieve the animator of much of the tedious work by keyframing: the animator draws only the crucial frames, or key frames, and the computer animation tool builds up the in-betweens by spline interpolation or other techniques. Keyframe interpolation is well established [23, 27, 51, 53] and is used in all commercial animation systems. Here the animator loses some control over the interpolated frames, and this has led to intensive research on control of the interpolation [4, 5, 9, 26, 34, 36, 50, 57]. For movements involving a small number of degrees of freedom, this keyframing technique can produce believable motion, but when animating complex movements of complex objects (e.g. human movements), the work is still tedious since a large number of degrees of freedom, mostly joint angles, are involved and need to be specified. Even with keyframing techniques, thousands of values need to be specified to simulate a human walking with a simple human skeleton of about 20 body segments. For each frame, the animator may need to try several joint angle settings to get the body part to the desired place.

Inverse kinematics has been introduced to minimize part of the tedious work but also results in loss of some control for the animator[2, 17, 46, 58]. Animators now just give the desired place of the end effector, e.g. the finger or foot of human figure, and the rotations of the related joints will be calculated using inverse kinematics. LifeForms[31] is such an animation system for animating complicated human figure movements such as dances and sports. Inverse kinematics takes much manual work from animators, but animators still need to specify many parameters, and high skills of the animators still are in demand.

C. B. Phillips and N. I. Badler control bipedal articulated figures through kinematics constraints[45]. The system takes the kinematic constraints as input, and uses inverse kinematics to get the output joint angles; the animator controls the system by command, and is free from editing the key frames. While this approach is attractive, it is not so useful practically because general movements can not be generated since specific movements are programmed into the system. Moreover, since the control is at frame level, there is no obvious relationship between a particular high-level parameter such as step length and low-level parameters such as joint angles of a frame, and the system has no understanding of what motion is concerned. Therefore, to get the same kind of movements with different styles, (e.g. walking with different step length,) the animator has to work from the beginning, and the predefined motion cannot be fully used.

Dynamic simulation has been introduced to computer animation because it provides the animator with quite high-level control so that the animator just needs to specify a few high-level parameters. It is attractive also because the generated motion adheres to the laws of physics, and provides a level of realism that is extremely difficult to achieve with kinematic methods[3]. Forward dynamic animation can be used to simulate simple passive systems successfully[25], but it fails to simulate active and complex systems with "intention", such as human figures. Several reasons lead to this failure. First, the complexity of the human figure leads to a large system of equations, which takes considerable time to solve. The complexity also makes the force and torque propagation in the skeleton very complicated. Secondly, the movement of a human figure is activated by internal force and torque, but the internal working mechanism of biological control systems is not fully understood, and therefore hard to simulate. Thirdly, the equations of motion for articulated skeletons are inherently ill-conditioned, independent of their formulation[40, 58], and this leads to the numerical instability of the system.

Much work on procedural animation has attacked this problem. A. Witkin and M.Kass combined physical simulation and keyframing technique to produce some realistic animations[59]. The animator can use spacetime constraints to specify several key points for selected variables at specific times, and they are combined with Lagrangian equations to produce animation. C. Heer and B. Wyvill presented a dynamic system that allows easy user control through a simulation language and several high-level control primitives[29]. A. Bruderlin and T. Calvert used procedural techniques to animate human locomotion[10, 12]. This system uses three parameters: step length, step frequency, and velocity to specify a desired motion, and the motion is produced by a programmed control system. Furthermore, 15 locomotion attributes can be set to individualize the locomotion, and thus the animator can make the motion reflect the personal characteristics of the animated figure. H. Ko and N. I. Badler[33] presented an inverse dynamic method, in which strength data is defined to make the motion more natural.

Procedural animation systems relieve animators from tedious keyframe editing, and specification of motions at a high-level is very convenient. However, because the motion control is programmed into the systems, procedural animation systems cannot generate general motions, and because the control for different motion is very different, the control programs cannot be easily reused. The ideas of state machines are applied in several approaches to make the control programs reusable to some extent. J.K. Hodgins et al. use a state-machine-like technique to aid the design of control programs[30]. The system presented in [56] specifies a motion by defining a goal with a set of states of destination only, and therefore a state-space controller can be built to provide control torques that achieve desired goal from arbitrary initial states so that the controller can be concatenated easily to create complex sequence of motion. The system presented in [52] can select a controller automatically by monitoring world states. The animator can write an algorithm (an abstract control language) to control the object and run this algorithm on a physical simulator to produce the animation. This approach is very attractive in that it provides a general way to organize and reuse the controllers, but the animator has to learn to write the algorithms, which may be challenging to many animators.

Another effort for dynamic figure animation is in the context of optimization problem and optimal control theory[1, 42, 43, 55, 59, 56]. Those methods allow the specification of relatively high-level goals and constraints, and the forces and torques necessary for meeting the goals can then be calculated. The potential generality of these approaches makes them very promising, but the growth of the search space and poor convergence when applied to complex systems are still unsolved problems.

Even higher motion control level focuses on the interaction between figures, or between figures and the world[16, 35, 41]. At these levels, individual articulated motion is less important than issues such as path planning and task achievement. The behavior articulated figures is controlled by some rules in an intelligent way.

The reality today is that general high-level control is far from mature in theory and practice, and most commercially available animation systems still rely on low-level keyframing. Thus, an integrated multi-layered approach to motion control is desirable so that animators can specify and modify the animated movements at any level. A successful figure animation system is likely to incorporate all the techniques discussed above to some degree. A. Bruderlin and T. Calvert give an illustration hierarchy of the levels of human motion control[12]: in a joint, within a limb, between limbs, between figures, and within the world. In this thesis, we just focus on the lower levels– in a joint, within a limb and between limbs; no attention is paid to the between figures and within the world levels.

2.2 Why use motion signal transformations

The complexity of articulated movements and the limitations of current motion control systems lead to the need for tools that make reusing existing motion data possible and easy. One advantage of the use of computers in animation is that animation can be saved and reused on different occasions. A whole library of motion can be built and saved, and then new movements can be more readily created by modifying the existing ones. Most animation systems support motion modification at low-evels only, one painful degree of freedom a time.

Another reason for the need for efficient motion modification tools stems from the use of motion capture techniques. Since making a believable human movement at a low level demands lots of time for a highly skilled animator, and since efforts to develop high-level control tools are still less than completely successful, rotoscoping methods such as motion capture become an attractive choice for obtaining convincing life-like motion. Motion capture gets real-life data by attaching some sort of sensors to a performer's body so that position can be tracked by computer and stored for later playback. Although yielding fairly realistic human animation, several facts limit the use of motion capture. First, sophisticated software is required to reconstruct the original motion from sensor data. Secondly, the motion is represented by densely-spaced data, for which spline editing tools are not suitable, therefore a motion often needs to be recaptured even if a slight modification is desired. Thirdly, a figure animated in this way is limited to those movements which can be actually performed by a live subject, but in many cases such as special effects, a motion that cannot be performed by a live subject is needed. For example, motion capture techniques cannot record a jump longer than the subject can actually make. Finally, but most importantly, since the motion is captured on a finite number of subjects, without a generic process method, the animated motion will be limited to simple imitation. Due to these limitations, some kind of tools that can make it easy to reuse the precaptured motion are desired.

There has been some attempts to make use of existing motion data. Zeltzer [60, 61] presented a kinematic method for simulating human walking based on kinematic joint values obtained from a rotoscoped human walk. High-level walking instructions are decomposed into a set of motor control programs, which drive the motions of individual joints. There are several attempts to obtain generic properties of one particular motion, e.g. human walking. H. Ko and N. I. Badler proposed a generalization method to generate a step of an arbitrary subject with arbitrary step length based on the rotoscoped data with constraint satisfaction enforced within the generalization process[32]. The generalization is designed to preserve the original motion characteristics, anthropometry generalization is used to handle subjects with different "size", and step length generalization is used to change the steps to different step length for the same subject. While this type of method is good at preserving the characteristics of human walking, it is limited to one particular motion and particular parameter generalization; also in some cases, characteristics-preservation is not desired, since motions with different characteristics are needed.

A. Bruderlin and L. Williams presented a novel approach to modify or create motion using some techniques from the image and signal processing domain[13]. This approach provides analytic solutions at interactive speeds, and lends it to high-level control by acting on several or all degrees of freedom of an articulated figure at the same time. In this approach, a motion parameter like the degree of freedom of joint or the location of the whole body is taken as a sampled signal. A signal consists of the values of a motion parameter at each frame. The processed motion data could be derived from captured motion data, or come from the evaluation of a spline curve in a keyframing system, or even from motion data produced by procedural animation. This approach cannot only modify motion slightly and locally, but also change the motion dramatically and globally; it can not only generate similar motions with different style (e.g., excited and sleepy motion can be obtained from neutral motion), but also quite new motions (e.g., knocking on a horizontal table can be created from knocking on a vertical door, and walking-and-waving can be created from a walking and a waving). It is not guaranteed that the animator will get the motion he/she wanted at the first try, but since modifications are made at interactive speeds, it is affordable to try several times, and still saves much of animator's time. The image and signal processing techniques used in this work will be discussed in detail in Chapter 4.

2.3 Constraints in human figure animation

Constraints are everywhere in computer animation; there are constraints at all levels of the motion control and for animation techniques. For some approaches, constraints drive or guide the movement. Constraints can prevent the production of physically impossible motion, enable the production of comfortable-looking movement, or be used to specify the movement to be animated. The constraints may come from the understanding of the biological nature of body movement, the physical law of motion and the animator's requirements.

The constraints of articulated figure animation also fit into the movement hierarchy described in [12]. At the lowest level, it is the constraints of joint angles, usually the joint rotation limits or relations between joint angles due to the physical possibility or the animator's specification. Constraints of joint rotation limit are well handled in all animation systems since most animation system models use joint angle as the motion parameter. At highest level, constraints for avoiding collisions between figures or between figures and obstacles usually require that the articulated figure has the intelligence for its behavior control and path planning[16][41], and this kind of constraint will not be considered in this work.

Constraints can be imposed during inverse kinematics. Linear constraints like joint angle limits or other constraints may be considered. Inverse kinematics systems handle constraints either by maintaining constraints which are already satisfied, or trying to satisfying constraints after they are violated. The constraints are modeled either to reduce the number of coordinates needed to describe the system's state, or to introduce additional penalty forces into the system to maintain the constraints. The first approach, termed generalized coordinate approach, is used in S. Bawa's work [7], and the second approach, termed penalty force approach, is used in C.Welman's work[58].

Interpolation could produce intermediate values that violate the constraints although the key frames do not violate the constraints, or produce some unnatural movement since the physics is not considered during interpolation. For example, even if both feet are positioned correctly in a series of key frames, there is no guarantee that simply interpolating joint rotations will maintain the correct foot position in the interpolated frames, especially when the inbetween segments get long. This can be remedied by specifying additional key frames, carefully selecting interpolating spline, and providing interactive or parameterized control over interpolation. D. Kochanek proposed an interpolation technique based on a generalized form of piecewise cubic Hermite spline[34]. Three parameters - continuity, tension and bias - are provided to control the interpolation. S. N. Steletee and N. I. Badler[50] proposed a double-interpolant method which separates timing control from trajectory itself, G. Hanotaux and Bernard Peroche advocated interactive control of interpolations[26], and L. S. Brotman and A. N. Nelravali presented optimal control over motion interpolation to produce natural-looking motions in [9].

As we have mentioned before, sometimes constraints guide or drive the the production of movements, and this occurs often in dynamic animation. In [6], constraints are used not only to introduce forces and torques into a dynamic simulation of the movement, but also to model the articulated figure. D. R. Forsey and J. Welhelms accomplished position constraints by changing the mass of constrained body parts[20]. As are used in the constraints handling for inverse kinematics, the penalty-force approach and generalized coordinates approach are also used for constraints handling in dynamic simulation. Those techniques handle geometric constraints quite well.

Besides geometric constraints, non-geometric constraints such as minimum energy expenditure and minimum jerk of the end of limbs are used to improve the quality of the animated motion. Most approaches for handling non-geometric constraints in dynamic simulation are based on well-established techniques for optimizing functions subject to a set of constraints[9, 22]. The establishment of correspondences between optimization criteria and expressive qualities of movements remains an open area for research.

Optimization methods assume that the complete or partial motion paths for limbs are known in advance; this side-steps the fundamental problem of synthesizing the limb trajectories for coordinated movements, and leads to the intensive research on spacetime constraints[18, 39, 43, 59]. Spacetime constraints are first proposed in [59] to help define trajectories. The constraints are specified by the positions of body parts at given times, and termed spacetime constraints since they span both time and space domain. An objective function has to be provided too. Since the spacetime constraint formulation leads to a non-linear constrained variation problem, which has no closed form solution in general, M. F. Cohen developed an interactive spacetime control system using hybrid symbolic and numeric processing techniques, so that the user can interactively guide the optimization process to converge to an acceptable solution[18]. Z. Liu et al. represented the trajectories of the degrees of freedom in a wavelet basis to achieve hierarchical control[39].

2.4 Constraints introduced by motion signal processing

Motion signal processing may introduce constraint violations into the resulting motion. Constraint violations may come from the amplitude adjustment of frequency bands, or from the poor timing between the motion signals to be blended. The constraint violation might be some physically impossible motion such as walking through the floor or walking above the floor, or motion simply failing to achieve the animator's goal, such as not putting the cup to the mouth for a drinking motion. Usually, constraints here are at a joint-angle level, within-limb level or between-limbs level defined in [12].

Since motion transformation does not use key-framing, inverse kinematics, dynamic or optimization approaches to produce new motion, none of the constraint handling methods discussed in the preceeding section can solve the problem here. Although some techniques could be used to prevent some constraint violations, in general, the constraint violations must be adjusted after the transformation. It is impossible to detect and adjust constraint violations or avoid constraint violations by modifying the transformation method during the transformation for several reasons. First, in most cases, more than one degree of freedom might be involved in a constraint, and the constraint is usually a function of the involved degrees of freedom, but since all degrees of freedom of the figure are processed individually and sequentially, it is impossible to evaluate those functions. Secondly, more than one frame might be involved in a constraint, which makes it impossible to evaluate the constraints at the current frame. Finally, the transformation is global in the sense that each frame of the resulting motion is related to all frames of the transformed motion or motions, and this makes the detection and adjustment of constraint violation even harder. Another reason for adjusting the motion after the transformation lies in the fact that many animators ignore the laws of physics, or even try to make some movements that violate physical laws to get some effects of traditional animation, in this case, decoupling motion transformation from constraints satisfaction is desirable.

While it is possible that non-geometric constraint violations might be introduced into movements produced by motion transformation, this kind of constraint violations can not be adjusted without great change of the characteristics of the motion. The work in this thesis will be devoted to geometric constraints.

In this thesis, linear constraints refer to constraints that are unconditional and involve only joint angles; they could be joint angle limits, or relations between joint angles. Non-linear constraints refer to all conditional constraints and constraints that involve data other than joint angles, such as position and orientation of body parts, distance between body parts. Unconditional constraints must be satisfied throughout the entire duration of the motion sequence, and conditional constraints must be satisfied at those frames where the condition is valid.

While it is relatively easy to handle linear constraints, non-linear constraint handling is non-trivial. It is even hard to find a way by which an animator can specify a constraint conveniently and the system can understand it and use it to detect and adjust the constraint violation easily. A constraint that could be stated simply in natural language might be quite hard to state in a formal way. Taking walking as an example, how can the animator specify the constraint "walking with feet on the floor"? Procedural animation does not have this problem since constraints are programmed into the system, and the animator does not need to specify them. Some animation techniques surveyed in this chapter are used for handling constraints in this work, although they are not used to produce new motions. Inverse kinematics and interpolation are used to adjust the constraint violations and get smooth motion, and constraint violations can be introduced by them. Therefore, constraint issues in inverse kinematics and interpolation must be considered, although they are not the research topic of this work.

An alternative way to adjust constraint violations is to give the animator full control: the animator can look at the result of the motion signal processing, find the constraint violations, and decide how to adjust the violations. To take this approach, the system must provide a way to let the animator modify the motion signal conveniently, not frame by frame; or edit several frames with constraint violations with a keyframe editing tool, and the system will be responsible for producing a smooth and violation-free motion. This approach gives animators more control, however, since animators are involved in the frame-level editing, which is what motion signal processing intends to avoid, this approach is provided as an complementary appoach for those constraints which are hard to be specified formally and adjusted in a desired way.

Before presenting the work of this thesis, other related ideas and techniques used in this work will be discussed in the next chapter.

Chapter 3

Related work

In Chapter 2, we have given the background and motivation of this thesis; in this chapter, related ideas and techniques will be discussed.

3.1 Skeleton modeling

Since articulated figures are the objects of animation in this work, the model of articulated figures needs to be derived first. Although the ideal computer-generated character would have muscle and tissue which deforms during movement, skin and clothing which wrinkles and stretches, hair that flows, and a face that has expression, modeling, animation, and rendering of these attributes are research topics in their own right, and much research is going on in each of those topics. Our attention is restricted to the skeletal structure of the articulated body in this thesis. Because muscle and tissue, skin and clothing, hair and face are not considered here, the animation of an articulated body is reduced to that of a skeleton.

A skeleton is represented by a collection of simple rigid objects connected by joints. A joint may allow rotation in 1, 2, or 3 orthogonal directions, and thus the degrees of freedom of a joint can be 1, 2, or 3. A detailed human body skeleton will have more than 200 degrees of freedom, although often fewer are enough to approximate the human body. By limiting the rotation angle in each degree of freedom of each joint, constraints on the allowable range of movement for a joint can be imposed.

The individual objects comprising the skeleton are all defined in their own local coordinate systems, and are assembled into a recognizable figure in a global world coordinate system by a nested series of rotation and translation transformations.

A human skeleton can be built up by arranging the segments in a tree-structured hierarchy. Each node in the tree maintains the current rotations of the corresponding joint in each degree of freedom; these rotations are with regard to the orientation of the parent segment in the tree, and the nested transformations in the skeleton tree ensure that segment will inherit the rotations applied to its ancestors in the tree. For example, a rotation applied at the shoulder joint will affect the entire arm, not just the upper arm segment. One joint is specified as the root of the tree, and transformations applied to the root will move the whole body in the world coordinate system. The transformation of a particular part in the skeleton can be computed by traversing the hierarchy from the root to that part and concatenating the local transformation at each joint visited by the traversal.

Most animation systems provide a way to build up a skeleton, and it is easy to define a grammar for specifying skeletons. No matter how a skeleton is created, it must specify the individual body segment lengths, the joint degrees of freedom and the overall hierarchy of the structure. In LifeForms[31], the skeleton is built up from a skeleton description file. After the skeleton is built up, it can be animated by varying the local rotations applied at each joint over time, and the global translation at the root joint as well.

3.2 Inverse kinematics

In this thesis, inverse kinematics techniques will be applied to pull a body part from a constraint-violating pose to a constraint-satisfied pose. The inverse kinematics problem has been studied extensively in the robotics literature, and in this section, we formally state the problem and review some common approaches to solving it, and one approach will be selected to meet our need.

In the inverse kinematics literature, a *manipulator* refers to a kinematic chain within a skeleton; one end of the manipulator, the *base*, is fixed and cannot move, and the *end-effector* is embedded in the coordinate frame of the most distal joint in the chain. The transformation M_i at a rotation joint i is a concatenation of a translation and a rotation, both are related to the coordinate frame of this joint's parent. That is:

$$\mathbf{M_i} = \mathbf{T}(x_i, y_i, z_i)\mathbf{R}(\theta_i) \tag{3.1}$$

where $\mathbf{T}(x_i, y_i, z_i)$ is the matrix that translates by the offset of joint i from its parent joint $i-1$, and $\mathbf{R}(\theta_i)$ is the matrix that rotates by θ_i about joint i's rotation axis.

The relation between any two coordinate systems i and j in the chain is found by concatenating the transformations at the joints encountered during a traversal from joint i to joint j:

$$\mathbf{M}_i^j = \mathbf{M}_i\mathbf{M}_{i+1}\cdots\mathbf{M}_{j-1}\mathbf{M}_j \tag{3.2}$$

So the position and orientation of the end-effector with respect to the base frame is just a matrix concatenation.

Given a vector $\mathbf{q}$ of known joint variables, the *forward kinematic* problem of computing the position and orientation is straight forward:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \tag{3.3}$$

If the goal is to place the end-effector at a specified position and orientation $\mathbf{x}$, determining the appropriate joint variable vector $\mathbf{q}$ to achieve the goal requires a solution to the inverse of Equation 3.3,

$$\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x}) \tag{3.4}$$

Solving this *inverse kinematic* problem is not simple. The function $\mathbf{f}$ is nonlinear, and in most cases, closed-form solutions cannot be found. A general analytic solution for arbitrary manipulators does not exist, and the problem must be solved with numerical methods for solving systems of non-linear equations.

3.2.1 Matrix inversion based methods

Since the non-linear nature of Equation 3.4 makes it difficult to solve, an alternative approach is to linearize the problem. The relation between joint velocities and the end-effector velocity is:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{3.5}$$

and the linear relation is given by

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \tag{3.6}$$

which maps changes in the joint variables $\mathbf{q}$ to changes in the end-effector position and orientation $\mathbf{x}$. Inverting the relationship of Equation 3.5 provides the basis for a matrix inversion based method:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{x}} \tag{3.7}$$

A simple iteration scheme for solving inverse kinematics problem can be based on Equation 3.7. At each iteration, a desired $\dot{\mathbf{a}}$ can be computed from the current and desired end-effector positions and orientations. The joint velocities $\dot{\mathbf{q}}$ can then be computed using the Jacobian inverse, and integrated once to find a new joint state vector $\mathbf{q}$, and this will be repeated until reach the desired goal. Note that the Jacobian $\mathbf{J}(\mathbf{q})$ and its inversion must be recomputed at each iteration.

Besides the high time-expenditure for matrix inversion, this approach suffers from *redundancy* and *singularity* problems. A manipulator is kinematically redundant if it possesses more degrees of freedom than are required to specify a goal for the end-effector. In general, any manipulator possessing more than six degrees of freedom is redundant for general 3-D positioning task, the Jacobian matrix is not invertable, and there is no unique set of joint values solving the inverse kinematic problem. A manipulator is in a singular configuration if its Jacobian matrix is singular; that is, two or more rows of the matrix are linear dependent. As a manipulator is in singular configuration, the Jacobian is invertible, and no combination of joint velocities will produce an end-effector velocity in the singular direction.

3.2.2 Optimization based methods

Another approach to the inverse kinematics problem is to cast the problem of Equation 3.4 as a minimization problem. For example, to position the end-effector $\mathbf{x}$ at a goal position $\mathbf{p}$, the distance from the current position $\mathbf{x}(\mathbf{q})$ to the goal position $\mathbf{p}$ serves as an error measurement:

$$\mathbf{E}(\mathbf{q}) = (\mathbf{p} - \mathbf{x}(\mathbf{q}))^2 \tag{3.8}$$

The problem now is to minimize $\mathbf{E}(\mathbf{q})$ subject to some constraints. A typical solver will converge from an initial state toward a solution state. At each step the state variables are perturbed slightly, and the objective function to evaluate its progress is reevaluated.

While conceptually simple, there are some practical difficulties in this approach. Constrained optimization of arbitrary non-linear functions is still an open research area, there is no guarantee that a solver will find a global minimum for a constrained optimization problem, and the convergence speed is another important issue. S. Bawa [7] gave an optimization based method for the inverse kinematics problem subject to non-linear constraints.

3.2.3 Jacobian transpose method

To avoid the expensive matrix inversion, C. Welman[58] proposed a Jacobian transpose method based on simplified dynamic model. In this approach, the error measurement:

$$\mathbf{e} = \mathbf{x_d} - \mathbf{x_c} \tag{3.9}$$

is taken as a force pulling the end-effector from current pose $\mathbf{x_c}$ toward desired pose $\mathbf{x_d}$, and the force can be propagated to internal joints by:

$$\tau = \mathbf{J^T F} \tag{3.10}$$

Then by simply using *f=mv* instead of *f=ma*, the velocities of the internal joints can be computed with:

$$\dot{\mathbf{q}} = \mathbf{J^T F} \tag{3.11}$$

Once $\dot{\mathbf{q}}$ is computed, a single iteration yields a new vector $\mathbf{q}$ which move the end-effector toward the goal, and this procedure repeats until the end-effector reaches the desired position. In this approach, constraints are handled with penalty force method.

This approach avoids matrix inversion, which makes the method good for interactive applications, but since this method is based on Jacobian, it still suffers from the singularity problem.

3.2.4 Heuristic method

To avoid the singularity problem, C. Welman[58] proposed a heuristic approach called *cyclic-coordinate descent* the (CCD) method. This method attempts to minimize position and orientation errors by varying one joint variable at a time, and each iteration involves a single traversal of the manipulator from the distal link toward the base. Each joint variable $\mathbf{q}_i$ is modified in turn to minimize an objective function. The minimization problem at each joint is simple enough to be solved analytically, and therefore the method can be performed quickly. This method does not suffer from the singularity problem.

3.3 Interpolation and quaternions

Interpolation is used to get the intermediate frames in keyframing animation systems. In this work, interpolation will be used to smooth the motion around the frames or segments with constraint violations. The interpolated data could be the location of the body or the rotation of the joints.

3.3.1 Interpolation with continuity, tension and bias

Interpolation with continuity, tension and bias was presented by Kochenek and Bartels in [34]. Kochenek interpolating splines give the user more control over the shape of the curve than cardinal splines. Three control parameters, *tension*, *continuity* and *bias* can be specified between -1 and +1. Continuity controls the continuity of the first derivative at input points. Tension controls how sharply the curve bends at an input point; a value of -1 produces more slack in the curve and a value of 1 tightens the curve. Bias controls the direction of the curve when it passes through an input point; a value of -1 undershoots the point while a value of 1 overshoots the point. These three parameters give the user broad control over the shape of the interpolating spline. Readers are referred to [34] for a detailed description.

In this thesis, interpolation with continuity, tension and bias will be used to produce shape function for motion waveshaping, to control quaternion interpolation for motion smoothing, to edit motion signals and to interpolate the displacement points to get a displacement mapping spline.

3.3.2 Quaternion and quaternion interpolation

Euler's angle coordinates specify orientation as a series of three independent rotations about pre-chosen axes; the three rotations must be used in exactly the given order since rotations do not commute, and the ordering of the axes is just a matter of convention. The geometry of orientation in Euler's angle coordinates is twisted, and varies with the choice of initial coordinate axes. There is no reasonable way to multiply or otherwise combine two rotations. Orientation and rotations can also be represented as a 3x3 matrix, but storing the matrix needs much more space and a conversion between angle coordinates and the matrix is required.

Quaternions were introduced into the computer animation field in 1985 by Shoemake[48] to represent rotations or the orientation of objects. A quaternion describes a rotation with four values in a natural way: three of them give the coordinates for the axis of the rotation, while the fourth is determined by the angle rotated through. Quaternions provide an easy way for specifying an arbitrary rotation about an arbitrary axis. To understand how quaternion works, readers are referred to [28], which gives a way to visualize quaternion rotations. Quaternions represent orientation as a single rotation, and rotations could be combined by quaternion multiplication. Quaternions preserve the interdependence between rotations naturally while Euler's angle coordinates ignore it. To represent a rotation or orientation, quaternions need less values than matrices, and therefore operations such as rotation multiplication will be more effecient.

The most attractive aspect of quaternions is that they are much better for interpolation than Euler angle coordinates or matrix. Angle interpolation treats the three angles of rotation at each key orientation as a three-dimensional vector whose components are interpolated from key to key independently, which ignores the dependence between rotation axes, and may lead to "gimbal lock": the loss of one degree of rotational freedom because two rotation axes collapse into one. Quaternion interpolation offers much smoother interpolation between orientations than three independent angle interpolations do.

Interpolating quaternions in a planar surface ignores some of the natural geometry of rotation space, and the animated rotation will speed up in the middle of the inbetweens[48]. For that reason, quaternions are usually interpolated on a spherical surface, and the problem is to construct smooth curves on spheres. Spherical Bézier curves are used by Shoemake[48, 49], and Spherical Biarcs are proposed by Wang and Joe[57]. In [4], spherical derivatives are used to impose angular velocity constraints on quaternion interpolation. The interpolation method used in this thesis is a combination of [49] and [34], more accurately, it is the algorithm of Shoemake[49] with bias control of Kochanek[34].

Conversion from rotation matrices to quaternions, and vice versa, are straightforward. Since most animators are familiar with Euler angle coordinates system, the user interface will still use a Euler angle system, and the conversion will be done before interpolation.

3.4 Signal processing techniques

In this section, we introduce some signal processing techniques which will be applied to the animation of articulated figures. Our interest lies in digital signal processing rather than continuous-time analog technology. A digital signal is defined by a sequence of numbers, and for our purpose, a signal contains the values of a motion parameter such as a joint angle for an articulated figure over time.

3.4.1 Fourier, multiresolution, and wavelet analysis

Several well-established techniques in image and signal processing such as Fourier analysis, multiresolution analysis and wavelet analysis are adapted to computer animation of articulated figures. Unuma et al. [54] applied Fourier transforms to joint angle data of human walking to extract a basic "walking" factor and a "qualitative" factor like "brisk" or "fast" from a walking sequence, and then those factors were used to generate new movements by interpolation and exterpolation in the frequency domain, such that a walk can be changed continuously from normal walk to brisk walk. Litwinowicz [38] used recursive filters to produce "lag, drag, and wiggle" effects to keyframed two-dimensional animated motion. While Fourier analysis could extract some characteristics from motion data, the power of frequency analysis is not fully exploited since the resolutions at each level are the same.

In multiresolution analysis, a signal is presented with different resolutions at each level, and the analysis can be carried out from coarse to fine. Generally, a signal could be stored as Gaussian (low-pass) or Laplacian (band-pass) pyramids of filter bands, where each level represents a different octave band of frequency, then some operations can be performed band by band before the signal is reconstructed by adding up the resulting bands. In this way, the fine features of a motion corresponding to the higher frequencies are treated separately from the course features corresponding to the lower frequencies.

Wavelet transforms have recently been proposed as a new multiresolution decomposition tool. The kernel of the wavelet transform is obtained by dilating and translating a prototype bandpass function. Wavelet transforms permit the decomposition of a signal into the sum of a lower resolution signal plus a detail signal, and each coarse approximation in turn can be decomposed further into yet a coarser signal and detail signal in finer resolution. Eventually, the signal can be represented by a low-pass at certain scale, plus a sum of detail signals at different resolutions. Liu et al. [39] used wavelet decomposition to speed up interpolation of physically-based keyframe animation.

In our system, Burt's multiresolution analysis[14, 15] is used. In this method, a

Gaussian pyramid is obtained by successively convolving the signal with a Gaussian filter kernel, and the signal is subsampled by a factor of 2 at each iteration until the signal become constant, or the "DC" value, and then the Laplacian pyramid is calculated by repeatedly subtracting 2 successive Gaussian signals in the Gaussian pyramid (the subtracted signal is expanded beforehand). Now operations can be performed on Laplacian bands and the DC value before they are added up to reconstruct the signal. Burt's multiresolution filtering is related to wavelet analysis: the Gaussian pyramid is the counterpart to a wavelet decomposition based on a cubic B-spline scaling function, while the Laplacian bands correspond to the high frequency wavelet component. We believe Burt's decomposition is more computationally efficient since convolution is applied only to obtain the low-pass Gaussian decomposition and the high frequency Laplacian are simple differences between successive Gaussian levels, and there is no need for two sets of filter coefficients.

3.4.2 Displacement mapping

Displacement mapping can be used to change the shape of a signal locally through a displacement map while maintaining continuity and preserving the global shape of the signal[13]. To alter a signal, only a few samples need to be modified, the changes made are called displacements. A spline curve is then fitted through these displacements, and then the spline is added to original signal to obtain a new smoothly modified signal. Figure 3.1 illustrated the procedure. Note that the first and last data point are always taken as displacement points. Displacement mapping will be used to handle constraints in this work, which will be presented in Section 5.4.2, and the idea of displacement mapping will also used for linear constraint handling, which will be presented in Section 5.2.2.

The techniques presented in this chapter will be used to handle constraints for motion signal processing. In next chapter, we will discuss motion signal processing techniques in detail.

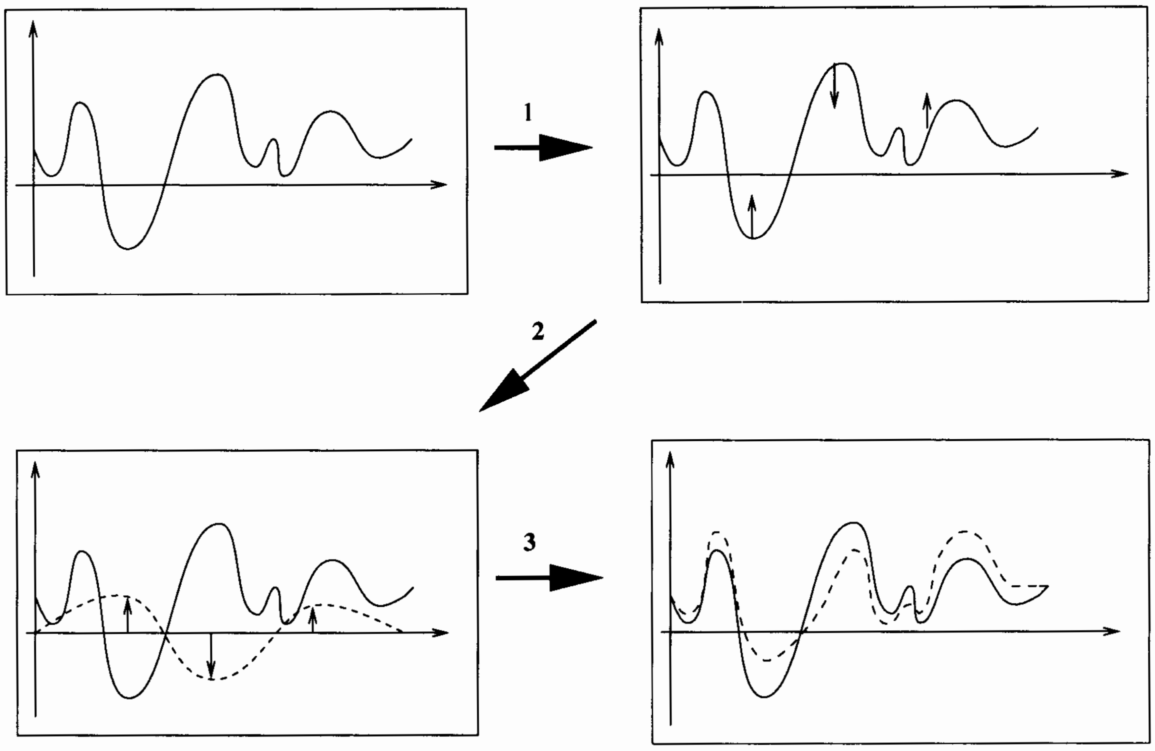


Figure 3.1: The procedure of displacement mapping.(1): define displacements; (2): generate a spline through the displacements; (3): add the interpolate spine to original signal

Chapter 4

Motion Signal Processing

As analyzed in Section 2.2, because of the complexity of articulated figure movements, the availability of motion libraries and the increasing use of motion-capture technology, it is desirable to provide tools that make it easy to adapt existing motion data for reuse. Techniques from the image and signal processing domain are well suited for this purpose. With signal processing techniques, existing motions can be modified and combined in such a way and at such a level that would be hard or impossible to accomplish otherwise. This is because the motion signal processing acts above the "in a joint" level by affecting several or all degrees of freedom of an articulated figure at the same time and over the entire duration of the motion.

For the discussion below, we treat a series of values for a motion parameter such as a degree of freedom of a joint as a sampled signal, and the signal contains the values for a particular degree of freedom at each frame instead of just at key frames. The values for a parameter must be sampled at the same fixed interval during the whole motion. These values could come from evaluating a spline curve in a keyframing system, or from captured motion. In this chapter, we will explain how several signal processing techniques can be adapted to produce new motion sequences for computer animation. These techniques have been implemented in a system where the user can interactively set certain parameters and then view their effects on the motion sequence. Original work on the techniques presented in this chapter has been done by Armin Bruderlin[13][11].

4.1 Motion Multiresolution Filtering

The multiresolution filtering technique used by Burt et al. [14][15] for certain kinds of operations on images can be modified to a one-dimensional version to be used for motion signal processing. Intuitively, low frequencies of the motion signal contain general and gross motion patterns, whereas high frequencies contain details, subtleties, and noise.

4.1.1 Algorithm

The length m of each signal determines how many frequency bands fb can be computed:

$$\text{let} \quad 2^n \leq m < 2^{n+1}, \qquad \text{then} \quad fb = n \tag{4.1}$$

In our implementation, we set an upper limit for the number of frequency bands.

The following is the motion multiresolution filtering algorithm in detail, as described by Armin Bruderlin [13].

1. Calculate Gaussian sequences G_k for all fb low pass signals by successively convolving the signal with the expanded kernel w, where G_0 is the original motion signal and G_{fb} is the zero frequency (DC) value ($0 \leq k < fb$):

   $$G_{k+1} = w_{k+1} \times G_k; \tag{4.2}$$

   This can be calculated efficiently by keeping the kernel constant and skipping signal data points:

   $$G_{k+1}(i) = \sum_{m=-2}^{2} w_1(m) G_k(i + 2^k m) \tag{4.3}$$

   where i ranges over all data points of a signal. Several strategies are implemented to handle the edge of the signal, i.e., when $i + 2^k m$ lies outside the range of a signal. The two most promising approaches have proven to be reflecting the signal and keeping the signal values constant (i.e., equal to the first/last data point) outside its boundaries.

2. Obtain the Laplacian filter bands $L_k (0 \leq k < fb)$:

$$L_k = G_k - G_{k+1} \tag{4.4}$$

3. Let the user adjust gains for each Laplacian band and multiply L_k's by their current gain values.

4. If the filtering is for multitarget interpolation (see section 4.2), blend bands of different motion signals at the each level.

5. Reconstruct the motion signal:

$$G_0 = G_{fb} + \sum_{k=0}^{fb-1} L_k \tag{4.5}$$

4.1.2 Joint angle vs. joint position

In articulated figure animation, either joint angles or joint positions can be used to define the motion of a figure. While the motion signal processing techniques we implemented can process both joint angle data and joint position data, the use of joint angle data is advised for the most desirable results. Several facts make joint angle data a better choice than joint position data, although motion-captured data are usually expressed as positions.

First, processing joint position data leads to a scaling effect on the animated figures. For example, if the low frequency bands of the motion signal are emphesized, the figure will get much larger than the original one. Although this scaling effect is desirable on certain occasion (e.g., in traditional animation), it should be avoided in most cases. Figure 4.1 shows the scaling effect; the original figure is shown beside the resulting figure for comparison.

Secondly, in most models for articulated figures, the segments are rigid and their lengths are fixed. Even if scaling is permitted, the ratio of length between any two segments should be fixed, but after joint position data is processed, the resulting position of each joint might not be achievable at the same time if the segments in the skeleton must be scaled at the same ratio. For the links shown in Figure 4.2, suppose

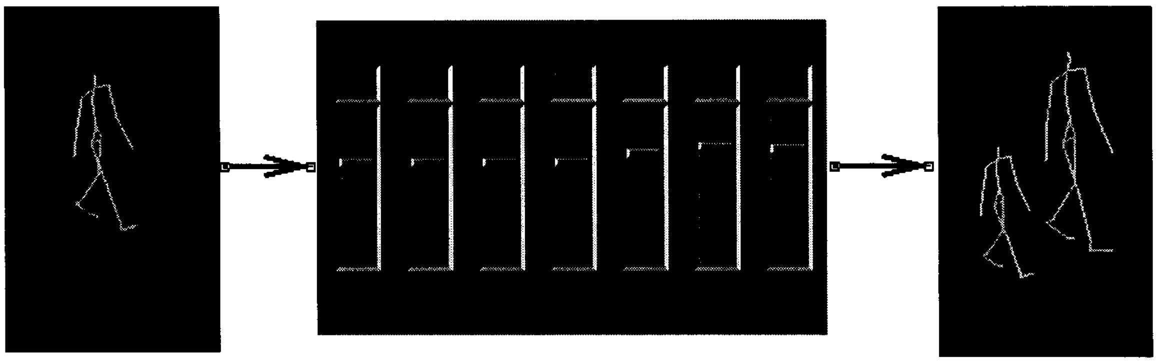


Figure 4.1: Scaling effect of processing position data. When low frequency bands are emphasized, the human figure is enlarged.

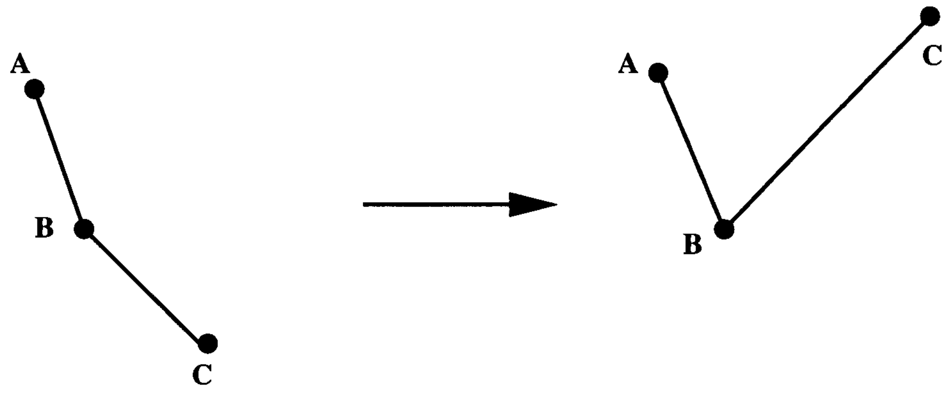


Figure 4.2: Change of skeleton configuration when processing position data

most of the movement of joint **C** (in position) falls in the low frequency bands and most of the movement of joint **B** (in position) falls in the high frequency bands. If we emphasize the low frequency bands of the movement, joint **C** extands too far from joint **B**. To allow joint **B** and joint **C** to be in the resulting positions, segment **BC** must be lengthened with a higher scaling factor than segment **AB**, thus the ratio between the lengths of segment **AB** and **BC** is changed (again this might be desirable in some occasion in traditional animation). Applying motion signal processing techniques on some parts of the figure also results in such problems. Processing joint angle data will not have these effects since the model itself guarantees that the articulated skeleton is maintained.

The third reason for working with joint angle data is that the motion produced by processing joint position data does not reflect modifications of the gains for the frequency bands. If we set the gains of all frequency bands to 2.0, the motion produced is exactly the same as the original one except that there is a scaling effect by factor 2.0.

4.1.3 Body part mask and band mask

Since there usually are many degrees of freedom (DOF) in articulated figures, it will likely be too much work for the animator to control the motion transformation if the filtering control is applied separately to each DOF. In this thesis, the same filter is applied to all DOFs, and the animator controls the figure motion at a level above joint angles.

While this high level control saves work for animators, it also limits the flexibility available to the animators. Animators may wish to use different filters on specific body parts or joint angles on some occasions. The reasons for this include the following. First, each DOF has different frequency property – for some DOFs, the major movement pattern is in low frequency bands, while for some other DOFs, it is in middle frequency bands, and an animator might want to process those DOFs separately. Secondly, animators might just want to transform the motion of selected parts of the figure, as you will see in an example in Section 4.2, where we just need upper body movement of one figure, and lower body movement of another figure. Finally, to achieve a specific movement of the figure, some constraints will be violated for certain parts of the figure, and the constraint violations can be avoided by using different filters on those parts of the figure.

To let the animator control the movement at a high level and still have some flexibility, body part masking and band masking functionality should be provided. Motion transformation can then be applied to all the DOFs of the articulated figure except the masked body parts or joints, and thus different filters can be applied to different body parts or joints.

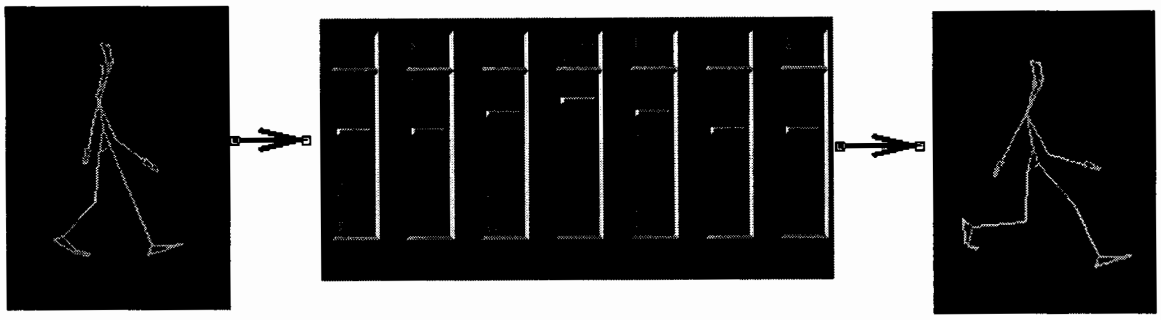
4.1.4 Examples

Here we give some examples to illustrate the effects and applications of multiresolution filtering, and more examples will be presented in Chapter 6 to explore how to make full use of this technique. With a display like an equalizer in an audio amplifier, Figure 4.3 shows a kind of graphic equalizer for motion, where the amplitude (also called *gain* in this thesis) of each frequency band can be individually adjusted via a slider before summing all the bands together again to obtain the new motion. In these examples, the multiresolution technique is applied to a skeleton with more than 70 DOFs built with the skeleton modeling method presented in Chapter 3. The same frequency band gains are used for all DOFs. In Figure 4.3 (a), increasing the gains of the middle frequencies of a walk sequence resulted in a smoothed but exaggerated walk, and in Figure 4.3 (b), increasing the gains of the low frequencies generated an attenuated, constrained walk at a high speed. Note that the gains do not have to be positive, and this is shown in Figure 4.3 (c), where the gain for one band is negative for a motion captured sequence of a figure knocking at the door, resulting in a motion like knocking at a table.

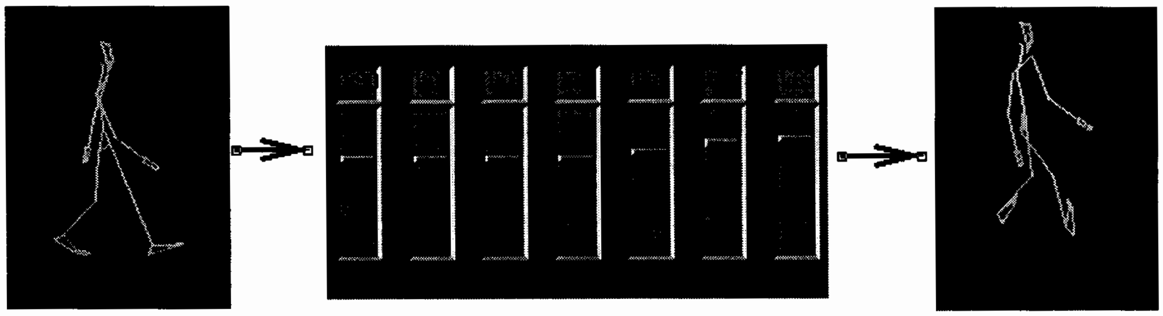
From the examples, you can see that some constraints such as joint limits or walking on the floor can be violated in the resulting movements. As has been presented in Chapter 2, our motion-editing philosophy is to employ constraints after the general character of the motion has been defined. Techniques for preventing and adjusting constraint violations are presented in Chapter 5.

4.2 Multitarget Interpolation

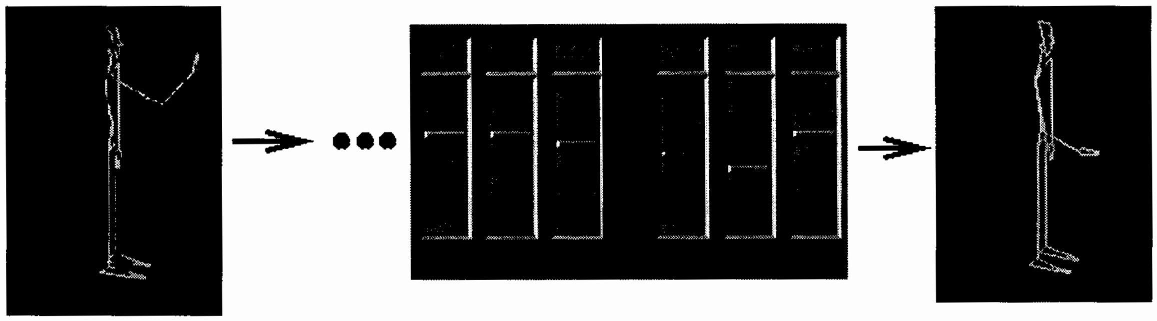
Multitarget interpolation is a process widely used in computer animation to blend between different models. The technique was originally applied in facial animation [8, 44]. We might have detailed models of happy faces, sad faces, quizical faces, angry face, etc., and those models are similar but with their own particular control parameters. The control parameters can be at a high level (like "raise left eyebrow by 0.7"), or a very high level (like "be happy"). The expression of the face can be



(a)



(b)



(c)

Figure 4.3: Examples of motion signal filtering. (a) middle frequency of walking emphasized; (b) low frequency of walking emphasized, (c) negative gains used

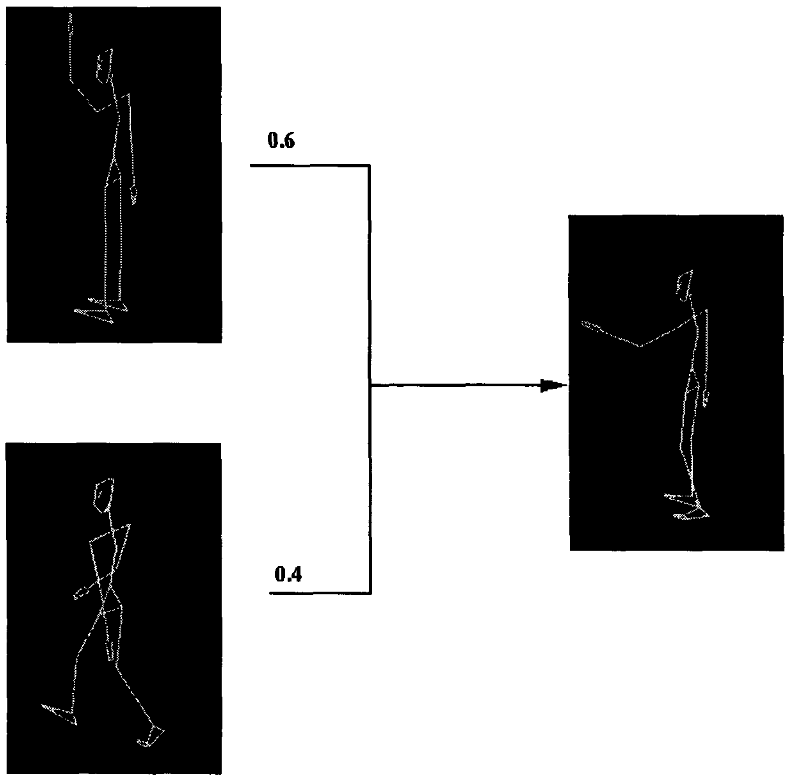


Figure 4.4: Blending waving and walking without filtering

controlled by blending the corresponding parameters of different models to varying degrees.

4.2.1 Multitarget motion interpolation

The same technique can be applied to the motion of articulated figures. If we have a happy walk, a sad walk, an angry walk, etc., we can blend those walks freely to provide a new result. Figure 4.4 shows an example of blending two different motions of a human figure: a waving sequence and a walking sequence. In this case, the blend is linear: add 0.6 of the waving and 0.4 of the walking. Guo et al. [24] gave a good discussion of this approach which they termed parametric frame space interpolation. In our approach, the motion parameters to be blended, such as joint angles, are completely decoupled from one another, and have no implicit range limits, which may lead to constraint violations in the blended motion.

As indicated in step (4) of the multiresolution algorithm in 4.1.1, we can mix multitarget interpolation and multiresolution filtering to separately blend the frequency bands of two or more motions. Figure 4.5 illustrates this technique for two walking sequences: regular walking and tired walking. By treating different frequency components in a different way, this technique can produce more meaningful new movements. For example, after exploring the effect of each frequency band on the movement, the frequency bands contributing to the tiredness of a tired walking can be extracted, and the animator can obtain new walking sequences with different degree of tiredness by adjusting the gains of those bands.

4.2.2 Normalization

There is a normalization issue in multitarget motion interpolation. When the animator sets the gains of certain frequency bands of the tired walk to 1, and sets those of a regular walk to 1 also, what the animator wants might be "not-so-tired" walking. If there is no normalization, the resulting walk will not be what the animator expected; in fact, the resulting walk violates many constraints.

There are also some reasons for not using normalization. Look back at the example of blending a waving and a walking movement; with normalization, the waving movement in the blended motion can never be as big as the original waving, and the walking movement in the blended motion can never be as big as the original walking, because:

$$x < \frac{ax+by}{a+b} < y \qquad \text{with} \quad x < y \quad \text{and} \quad a,b > 0; \tag{4.6}$$

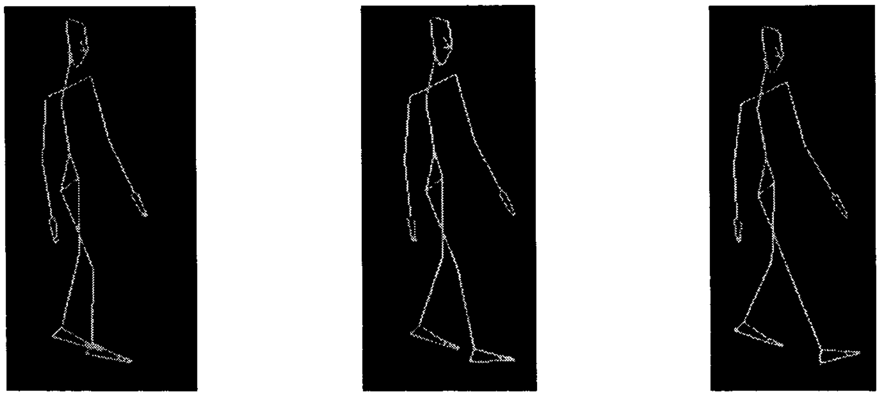
Another reason for no normalization is that the gains might be negative numbers, which makes the normalization nonsense ($a + b$ might be 0 !).

In our system, it is the animator who decides whether normalization is necessary. For example, in Figure 4.7, by setting the gains for waving to 0 as applied to the lower body, and setting the gains for walking to 0 as applied to upper body, we can get a walking and waving motion with the waving the same as in the original waving and the walking the same as in the original walking. You may be confused by the settings of the gains at this point: since there is no movement in the lower body for the original waving sequence, why bother setting the gains to 0 and applying them only to the lower body? We will explain this at the beginning of Chapter 5.

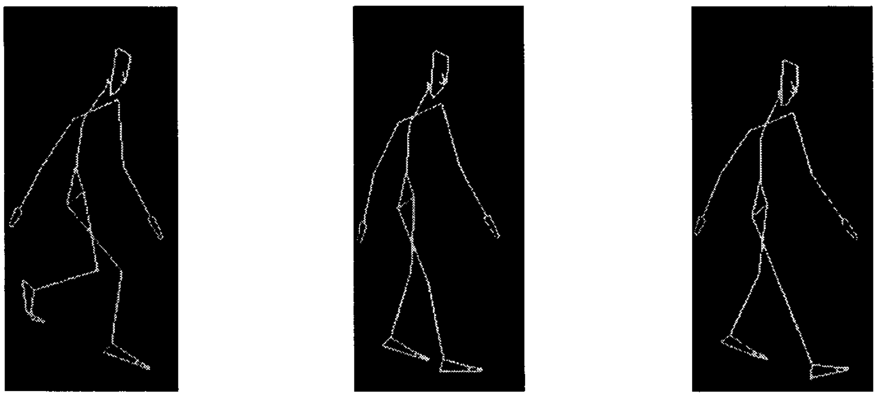
4.3 Motion Waveshaping

LeBrun [37] introduced waveshaping for the generation of musical sound by synthesizing steady-state or time varying harmonic sound spectra. Waveshaping changes the amplitude of a signal in a non-uniform manner. The basic idea involves a non-linear shaping function $\mathbf{f}$ of an arbitrary shape which operates as a special filter on a signal $\mathbf{x}$. The shaping function $\mathbf{f}$ is normalized, i.e., it accepts and outputs values in the range [-1, +1]. For example, if we define $\mathbf{f}(\mathbf{x}) = \mathbf{x}$, the signal will pass through unchanged; if $\mathbf{f}$ is slightly changed, for example, to have a small bump near 0, signal $\mathbf{x}$ will be changed in that it will have positive values around where it was 0 before; if $\mathbf{f}$ is defined as a cosine function in the [-1, +1] range, the values of $\mathbf{x}$ will be exaggerated in the middle and attenuated at the extremes.

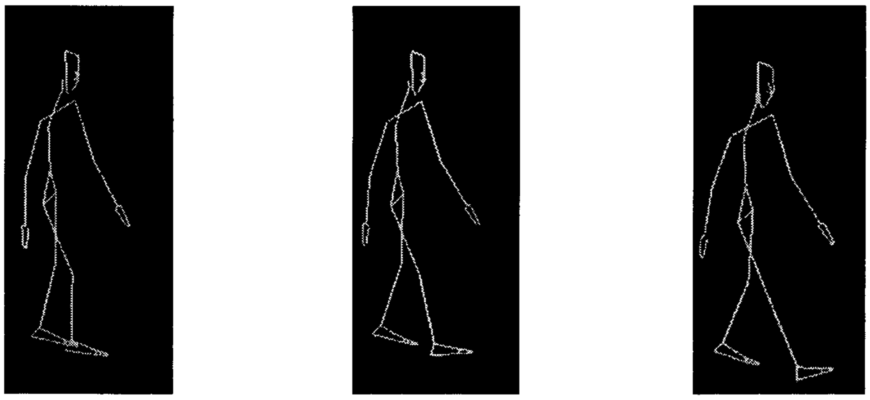
The waveshaping technique can be adapted to generate new animations[13], and it is also useful to efficiently edit densely-spaced motion signals such as live motion capture data where each frame is a keyframe. One application is to map the shape of the input motion to a "characteristic" function. In this way, we can build up a library of shaping functions which will permit rapid experimentation with different styles of movement. Another application of waveshaping is to limit the joint angles to a specific range - this will be presented in detail in Chapter 5.



Snapshots of a normal walk



Snapshots of a tired walk



Snapshots of the blending result

Figure 4.5: Blending two walks with filtering

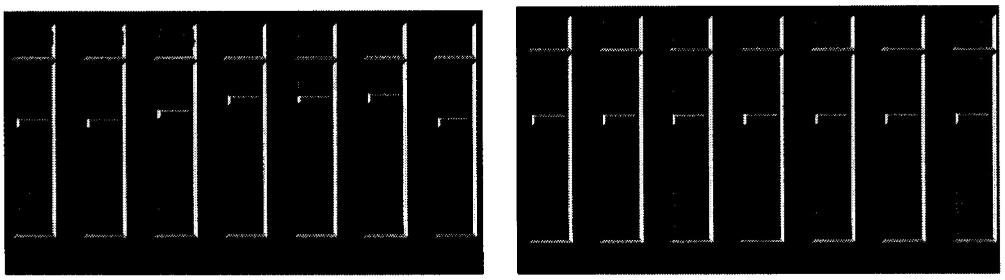


Figure 4.6: Filters used for blending in Figure 4.5

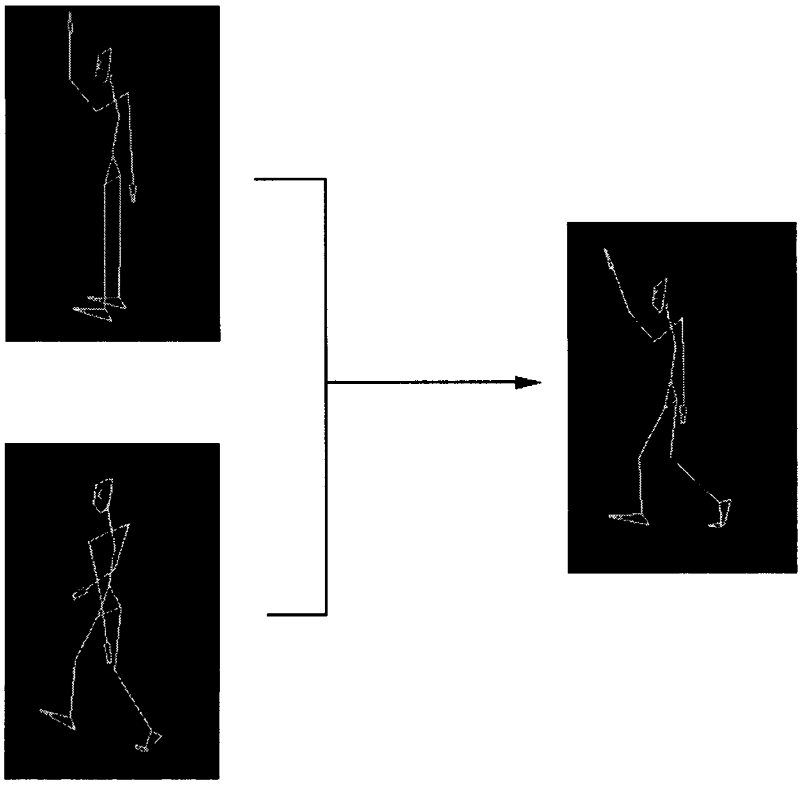


Figure 4.7: Blending waving and walking without normalization

Chapter 5

Handling Motion Processing Constraints

As we have seen in Chapter 4, all of these motion signal processing techniques we have used may produce some motions which violate constraints. The constraints may involve the definition of the skeleton, such as joint angle limits. Those constraints may define the behavior of the animated body; for example, the constraints imposed on the skeleton of a human being make the skeleton behave in a way natural to human beings. Violation of those constraints leads to motions physically impossible for the animated skeleton. Other constraints are not imposed by the skeleton, but are defined by animators to achieve particular goals, and the violation of those constraints results in motions that fail to meet those goals.

Constraint violations resulting from motion transformations may come from the adjustment of the amplitude of frequency bands, or from the poor time synchronization between signals to be blended if multitarget interpolation is used. While we can apply time warping techniques to prevent the constraint violations due to poor timing, it is impossible to prevent constraint violations due to amplitude adjustment. As analyzed in Section 2.4, it is impossible to detect constraint violations during the transformation, therefore, although we can apply some techniques to generally reduce constraint violations, we do not make an effort to prevent all constraint violations for a motion transformation. Instead, we try to adjust motions with constraint violations after the motion transformation and before providing the resulting motion to the animator.

In the first section of this Chapter, processing offset data and applying consistent dynamic time warping are presented as two general methods to avoid some constraint violations. In the second section, bound mapping and motion signal compression are used to handle linear constraints. Non-linear constraint handling is discussed in Section 5.3, and some tools for editing motion signals at a low level are provided as alternative approaches for constraint violation adjustment.

5.1 Prevention of constraint violation

In general, constraint violations cannot be prevented during the motion signal transformation and there is no filter that can avoid constraint violations. However, it is possible to reduce the chance of violations or the degree of violations by processing appropriate data and by pre-processing the signals to be processed. As presented in Chapter 4, we can reduce the chance of constraint violations by processing joint angle data instead of joint position data, and by applying motion signal transforms selectively to different parts of the body. In this section, we will give methods to prevent some of the constraint violations by pre-processing the signal data.

5.1.1 Processing offset data

The first technique used to prevent constraint violations is to process offset data. We do not treat the joint angle values of a parameter sampled over a period of time as a signal; instead, we treat the differences between the initial status and each frame of the motion as the signal to be transformed. For example, if the values of a parameter are: 50.5, 52.5, 55, 60, 64.5, $\cdots$, the signal to be transformed will be: 0, 2.0, 4.5, 9.5, 14, $\cdots$. In the following discussion, we will call the latter offset data, and the former absolute data.

The first reason for processing offset data lies in the fact that it is the changes of

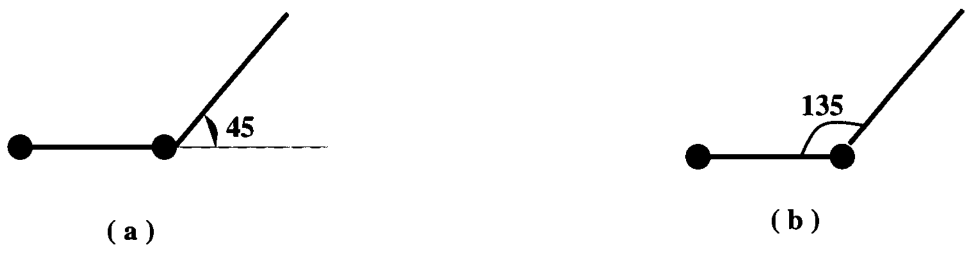


Figure 5.1: different definition of joint angles

the joint angles that represent the motion, not the values of the joint angles. When the animator says "increase the low frequency band by 50 percent", she/he means to emphasize the low frequency of the movement, not including that of the initial status. Recall that in Chapter 4 we had a question about the blending of a waving motion and a walking motion. Although there is no movement in the lower body of the waving figure, the initial angle value for the ankle is about 90 degrees. If there is no normalization and if we do not process offset data, the angle value for ankles will be about 180 degrees after blending even if the gain for each frequency band is not changed, and the feet will be upward instead of forward. This will not happen if offset data are processed.

Processing offset data also makes the motion transformations independent of the definition of the joint angles. For example, for a link in Figure 5.1, the joint angle might be defined as 45 degree in one model, and 135 degree in another model. If the angle is changed due to motion transformation, those two models will get two totally different movements. There is no such problem if offset data are processed.

Transforming absolute data may lead to constraint violations, as explained above. Thus some constraint violations can be prevented by processing offset data instead. We have seen the example of walking and waving; Figure 5.2 shows two other examples. We emphasize the low frequency bands of a human walking and a human drinking; (a) and (c) are the results of processing absolute data. You can see that there are constraint violations in ankles in walking (a), and the hand goes beyond the mouth in drinking (c). Figure 5.2 (b) and (d) are the results of processing offset data with the same filters, and there is no constraint violation, or the violation is reduced to an acceptable limit.

5.1.2 Consistent dynamic timewarping

When multitarget interpolation is applied, constraint violations may result from poor timing between the signals to be blended. For example, if we want to blend two walk cycles, the steps of the walks must coincide so that the feet strike the ground at the same time. If one walk is at a slower pace than the other, and we simply blend them without establishing a correspondence between the steps beforehand, the resulting motion may be an uncoordinated movement, and the feet may no longer strike the ground at regular intervals, or not strike the ground at all. This problem can be solved by timewarping, a technique involving remapping a signal in time to make it coincide with a template signal. A dynamic timewarping technique is used in this work.

Dynamic timewarping is a nonlinear signal matching procedure which was first used in the field of speech recognition to compare templates with input utterances [19]. Since the speeches can be made at variable speeds, each acoustic input signal also shows variations in speed with respect to the template signal, and the timewarping procedure identifies a combination of expansion and compression which can best warp the input signal to match the template most closely. In our case, timewarping is applied in a discrete time domain to make one motion coincide with another. The problem can be decomposed and solved in two steps: find the optimal sample correspondences between the two signals first, and then apply the warp.

The sample correspondence problem is defined as finding the globally optimal correspondence between the samples of the two signals [11]: to each sample of one signal, assign at least a vertex in the other signal such that a global cost function measuring the "difference" of the two signals is minimized. This problem is related to contour triangulation [21] and shape blending [47], which can be solved by dynamic programming optimization techniques. The solution space can be represented as a two-dimensional grid, where each node corresponds to one possible vertex assignment. Figure 5.3 is an example for two signals with length 10, and the optimal vertex correspondence solution is illustrated in the grid by a path from (0, 0) to (9, 9). In

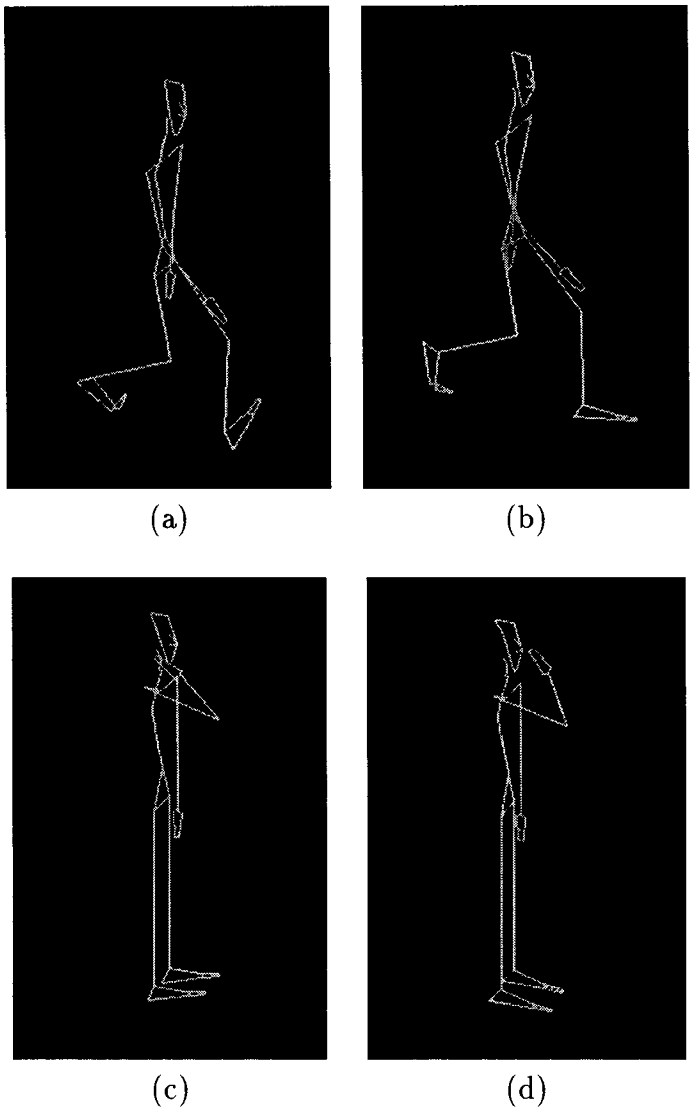
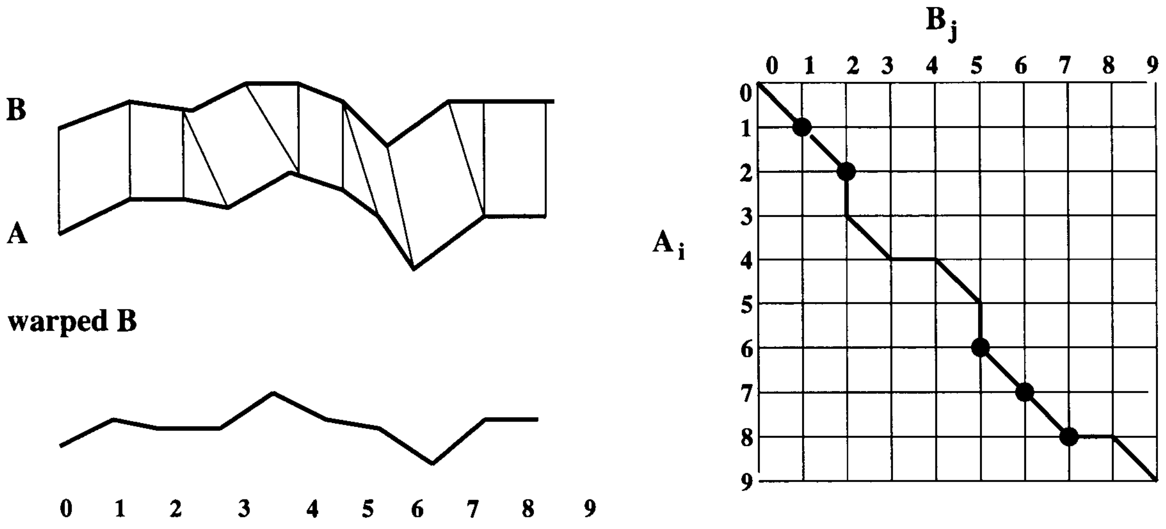


Figure 5.2: The effect of processing offset data; (a) and (c) absolute data processed; (b) and (d) offset data processed.



substitution: **1:1 correspondence of successive samples.**
deletion: **multiple samples of B map to a sample of A.**
insertion: **a sample of B maps to multiple samples of A.**

Figure 5.3: Dynamic time warping algorith

general there are $O(n^n/n!)$ possible paths. Sederberg presents a method which gives a globally optimal in $O(n^2)$ time[47]. The length of the two signals can be different, in general. For more detail of the algorithm, readers are referred to [47] and [13].

The second part of the problem is to actually apply the warp given the optimal sample correspondences. As in speech recognition [19], three cases are distinguished: substitution, deletion and insertion, which are indicated in the optimal path by a diagonal, horizontal and vertical line, respectively, between two nodes. Suppose we want to warp signal **B** to **A** as show in Figure 5.3, and the warped signal is denoted by B_w. If B_j and A_i are related by a substitution, it follows that $B_{w_i} = B_j$. In the case of a deletion, where multiple samples of B, $(B_j, B_{j+1}, \cdots, B_{j+k})$ correspond to one A_i, we let B_{w_i} to be the mean of B, $(B_j, B_{j+1}, \cdots, B_{j+k})$. Finally, an insertion implies that one sample of B, B_j, maps to multiple samples of A, $(A_i, A_{i+1}, \cdots, A_{i+k})$. In this case, the values for $B_{w_i}, B_{w_{i+1}}, \cdots, B_{w_{i+k}}$ are determined by calculating a Gaussian distribution around the original value B_j.

The complexity of an articulated skeleton may cause some problems here. First, the dynamic timewarping algorithm is $O(n^2)$ in time complexity, where n is the number of frames of the warped signal. If dynamic time warping is applied to each DOF of the skeleton (remember even a simple skeleton model for a human figure has around 80 DOFs), the animator must wait for quite a long time, which is not acceptable in an interactive work environment. Furthermore, the optimal sample correspondences will be different for each DOF. For example, in the signal for left knee, the ith frame is mapped to the j_1th frame of the template, but in signal for left ankle, the ith frame might be mapped to the j_2th frame of the template. This inconsistent timing between DOFs may lead to some strange movements, and some constraints may be violated.

To solve these problems, we use consistent dynamic timewarping. We just need to calculate the optimal sample correspondences once on a key DOF which can best represent the timing of the motion. This DOF is selected by the animator according to the property of the concerned motion; for example, one of the knees can be selected as such a DOF for a walk, and the elbow can be selected for a drink. The optimal sample correspondences for this DOF are then used to warp the signals for all DOFs. Since the time complexity for finding the optimal correspondences is $O(n^2)$, and the actual warping is linear, therefore the time is now not a problem due to the number of DOF, and since we warp all the DOFs with the same correspondences, the skeleton has consistent timing for all DOFs. The result of this technique may depend on which DOF is selected by the animator as the key DOF. Sometimes it is hard to find a DOF that can serve as such a key DOF. In those case, different body part may need to be processed separately, and then a key DOF is selected for each part of the body.

The $O(n^2)$ complexity of the algorithm makes the method non-interactive for very long motion sequences, but since the motion transformation system we built is intended to let animators interactively edit motion sequences, the sequences would not be too long (usually no more than 3,000 frames).

Figure 5.4 shows the improvement of the result by using consistent dynamic timewarping. Two walk sequences with different timing and length are blended to make a new walk. The results with and without time warping are both shown in Figure 5.4 for comparison. In each snapshot, the left human figure shows the result of the blending

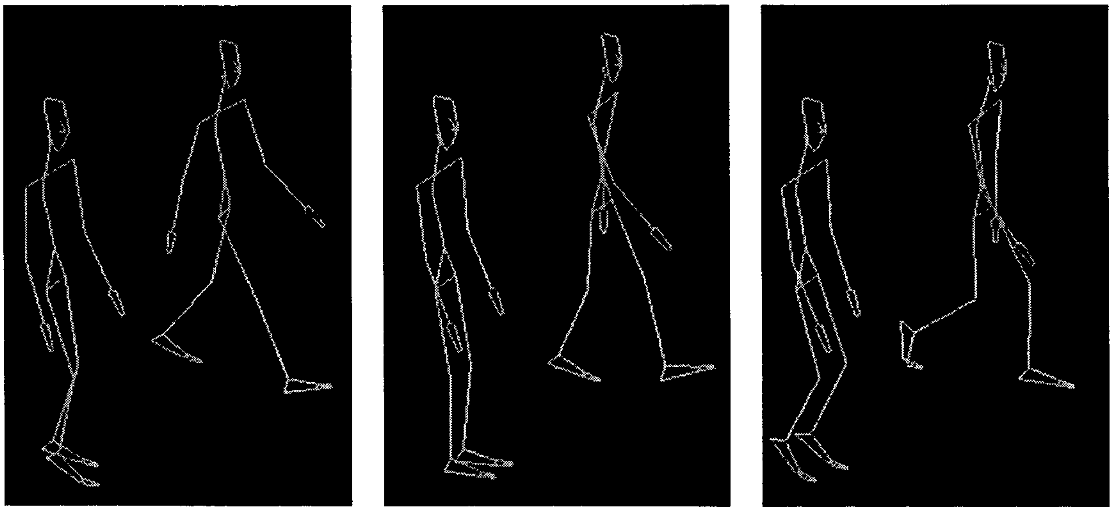


Figure 5.4: Example of consistent dynamic time warping algorithm

without time warping, and the right human figure shows the result of blending with consistent time warping. The movement of the left figure is not a walk any more.

We can avoid some and only some of the constraint violations introduced by motion signal processing by offsetting the signal to be processed and synchronizing the signals to be blended; in the following sections we will present some techniques to adjust those constraint violations which cannot be avoided.

5.2 Handling linear constraints

As we have stated in Chapter 2, we will focus only on geometrical constraints in this thesis. We will handle linear constraints in this section and non-linear constraints in next section. If there are linear constraints and non-linear constraints at the same time, linear constraints will be checked and adjusted first, then the non-linear constraint handler will work on the adjusted data, and the linear constraint will still be considered in the adjustment of non-linear constraint violations.

When we say a constraint is linear or non-linear, what we mean is whether the relation between joint angles is linear or non-linear. Formally, a linear constraint is:

$$\theta_1 \quad R \quad k\theta_2 + \alpha \tag{5.1}$$

where θ_1, and θ_2 are joint angles of the skeleton, k and α are given constants, and the comparison relation operator R could be: $=, >, <, \geq$, and $\leq$. (Since float values are concerned, we do not need to distinguish between '$>$' and '$\geq$', '$<$' and '$\leq$'). A nonlinear constraint is:

$$\mathbf{f_1}(\theta) \quad R \quad k\mathbf{f_2}(\theta) + \alpha \tag{5.2}$$

where θ is the vector of the joint angles, $\mathbf{f_1}$ and $\mathbf{f_2}$ are non-linear functions such as the position or orientation of certain point of the body, or distance between two points. As in linear constraints, the comparison relation operator R could be: $=, >, <$. Another type of nonlinear constraint is a conditional constraint:

$$\textbf{Condition} \longrightarrow \textbf{Constraint} \tag{5.3}$$

where **Constraint** can be a linear or non-linear constraint as defined before; the **Constraint** part must be satisfied when the **Condition** is valid.

For example, "the angle of knees must be greater than 0 degree and less than 180 degree" is a linear constraint; "the hand can not go beyond the mouth during drinking" is a non-linear constraint. The statement "during the drink, when the distance between the hand and the mouth reaches a minimum value, this value must be zero" is a conditional nonlinear constraint.

5.2.1 Linear constraint specification

In Equation 5.1, k could be any value, but it is usually 0 or 1. If $k = 0$, the constraint becomes a joint angle limit, and this is commonest linear constraint; if $k = 1$, the constraint becomes the comparison between two joint angles. If k is not 0 or 1, the user must be careful about the definition of the joint angle, as analysed in Section 5.1.1.

The default joint limits can be specified in the skeleton description file. The user may want to ignore all of the default joint limits, change the default limit setup for some joint angles, or set up a relation between two joint angles. We should provide a convenient way for users to describe their requirements easily, and for the constraint handler to understand the requirements. It is quite easy to achieve this for linear constraints compared to nonlinear constraints. The only problem is how to let the user specify a joint angle. The following is the simple grammar for linear constraints:

```
LINEAR-CONSTR::  JOINT-ANGLE COMPARE [COEF'*']JOINT-ANGLE|
                 JOINT-ANGLE COMPARE [[COEF'*']JOINT-ANGLE '+'] CONSTANT;
JOINT-ANGLE ::  '('JOINT, AXIS')';
JOINT ::  JOINT-NAME|JOINT-INDEX;
COMPARE ::  '>'|'<'|'=';
COEF ::  float;
CONSTANT ::  float;
JOINT-NAME ::  string;
JOINT-INDEX ::  integer;
AXIS ::  'x'|'X'|'y'|'Y'|'z'|'Z';
```

For example, $(``leftknee", x) > -2.0$, $(30, x) > -2.0$, $(30, x) < (31, y) + 20.0$, $(30, x) < (2 * (31, y)$ are all linear constraints with correct grammar. JOINT-NAME must be one of the joint names given in the skeleton description file; JOINT-INDEX is indexed from 0 to the total number of joints in the skeleton according to the order in which they appear in the skeleton description file; if the index is equal to the number of joints in the skeleton, it refers to the location of the whole body, which is taken as a phantom joint. Since a joint may have more than one degree of freedom, therefore, AXIS is used to specify which degree of freedom is concerned; the AXIS must be a valid degree of freedom for the joint.

It is easy to understand these constraints. Only one point needs to be noted: if there are two joint angles concerned in the constraint and the constraint is violated, we will adjust the joint angle on the left side. For example, if constraint $(30, x) > (31, x)$ is violated, the value of joint angle $(30, x)$ will be modified.

When the comparison operator is '=', the constraint is a "joint angle lock" which keeps the joint angle unchanged. This might be a very important constraint in other contexts of computer animation, but it is not useful in our context. If the joint angle is unchanged in the original motion, it would not change in the transformed motion if offset data is processed; if the joint angle changed in the original motion, it would change in the transformed motion unless the gains are set to zero. Therefore, we will not consider "joint angle locks" in the following discussion. If the user does want to impose a joint angle lock, we will simply set the value of the locked joint angle to the lock value at each frame of the whole signal.

We have built a very simple interface to let the user specify linear constraints using the grammar given above. The list of joint names and their index can be accessed by a button click so that the user does not need to remember the index of each joint or the spelling of its name. The user can also make the default joint limits valid or invalid, and select one of the two constraint violation adjustment methods provided: bound mapping or motion signal compression.

In the following discussion, we will convert a linear constraint to the following form if k is not 0:

$$(\theta_1 - k * \theta_2) \quad R \quad \alpha \tag{5.4}$$

and then we will take the left part of Eaqution 5.4 as a signal. Before applying bound mapping or motion signal compression, the value of $\theta_1 - k * \theta_2$ is calculated at each frame, and it is treated the same as a joint angle signal. α will be taken as the bound. After the constraint is checked, if there are constraint violations, θ_1 will be modified according to the modification made to $\theta_1 - k * \theta_2$.

5.2.2 Bound mapping

The idea of bound mapping stems from displacement mapping. If the signal for a joint angle goes beyond its range, the overshoots and undershoots from the range are taken as displacements, and a spline through the displacement points is calculated and added to the signal under consideration. The algorithm is as follows:

1. Take the first and last point as displacement points; if their values are greater than the upper bound or less than the lower bound, set their displacement value to be the difference between the value and the bound; otherwise, set the displacement value to be 0;

2. Check the signal at each frame, if the frame has a local maximum or minimum value, and this value is greater than upper bound or less than lower bound, the frame is taken as a displacement point, and the difference between the value and the bound is the displacement value.

3. If there are no displacement points other than the first and last frame, there are no violations, return; otherwise, calculate a cubic interpolation spline through the displacement points. A cubic spline instead of a Kochanek spline is used here because the displacement points are all maximum or minimum points, therefore at those points, the slope must be 0;

4. Add the spline to the signal.

Figure 5.5 illustrates the first two steps of the algorithm, and the rest of the algorithm is similar to displacement mapping presented in Section 3.4.2. Looking at the point A and point B, we find a problem. The violations at point A and point B cause the whole segment between A and B to be "pulled" down by almost the same value, and this is not desired. What is even worse is that, this may lead to new constraint violations: suppose point C is close enough to the lower bound, it may be less than the lower bound after the "pull". To avoid this problem, if there are

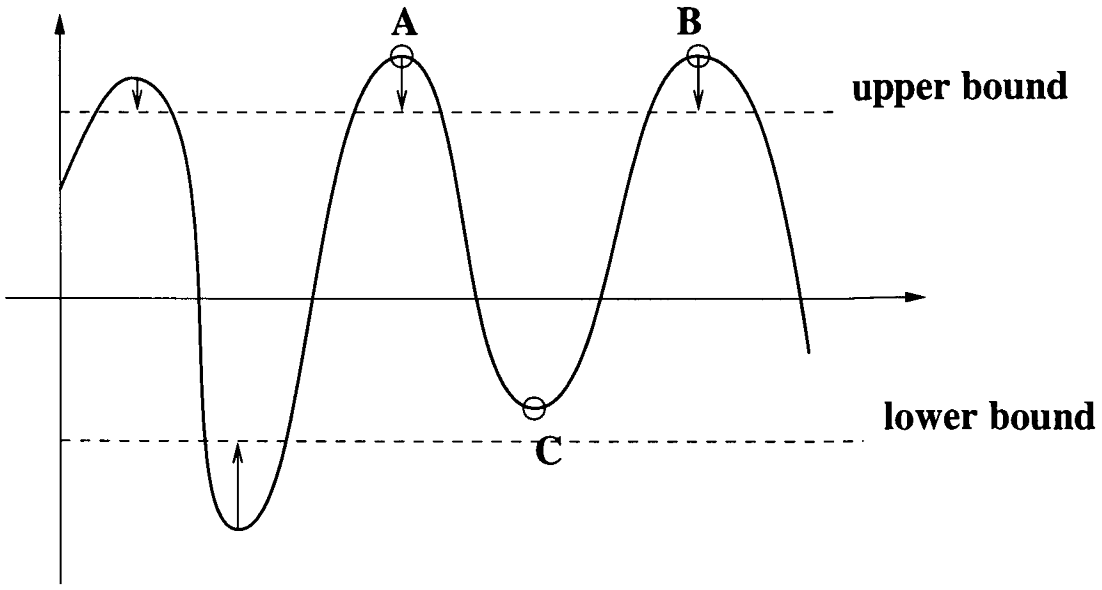


Figure 5.5: Bound Mapping

two adjacent displacement points both have positive values (or negative) values, we will find the maximum (or minimum) point between them, and this point will be an extra displacement point with the displacement value of 0. This extra displacement can always be found because there is a maximum (or minimum) point between two minimum (or maximum) points. In Figure 5.5, point C will be selected as the extra displacement point.

Bound mapping guarantees that the signal will be set to the bound values at the maximum or minimum violation points; theoretically, this does not guarantee that the interpolated spline will "pull" the whole violation segment into the range between the lower bound and upper bound, but practically, this rarely happens, and if it does happen, motion signal compression which will be discussed in next section can be used.

Figure 5.6 illustrates the effect of bound mapping. The back figure shows the original walk, the middle figure shows the result after transformation without constraint checking, and the front figure is the walk after bound mapping. You can see that the joint angle of the knee in the middle figure is obviously out of the limit range and it is adjusted in the front figure.

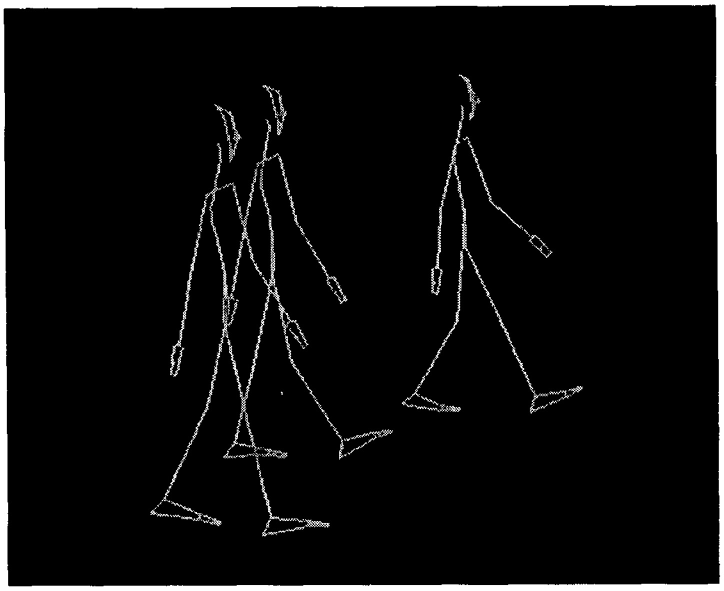


Figure 5.6: The result of bound mapping; the middle figure shows the result without constraint check, the front figure shows the result with bound mapping

5.2.3 Motion signal compression

To get a smooth motion, bound mapping needs to modify almost every frame of the motion, no matter what the value of the signal is at that frame. Motion compression is intended to modify the frame with violations and those frames with a signal value near the bounds. Motion signal compression is actually a type of waveshaping, where the shape function is a capping function. The idea is to "squash" the signal if it goes out of the range of the joint angle limits. The algorithm is very simple:

1. Check whether there are any violations of bound limit, if there are not, return;
2. Normalize the signal under consideration to range [-1, 1], and get the normalized value of the bounds;
3. Calculate the capping function;
4. Apply the capping function to the normalized signal;
5. Denormalize the capped signal.

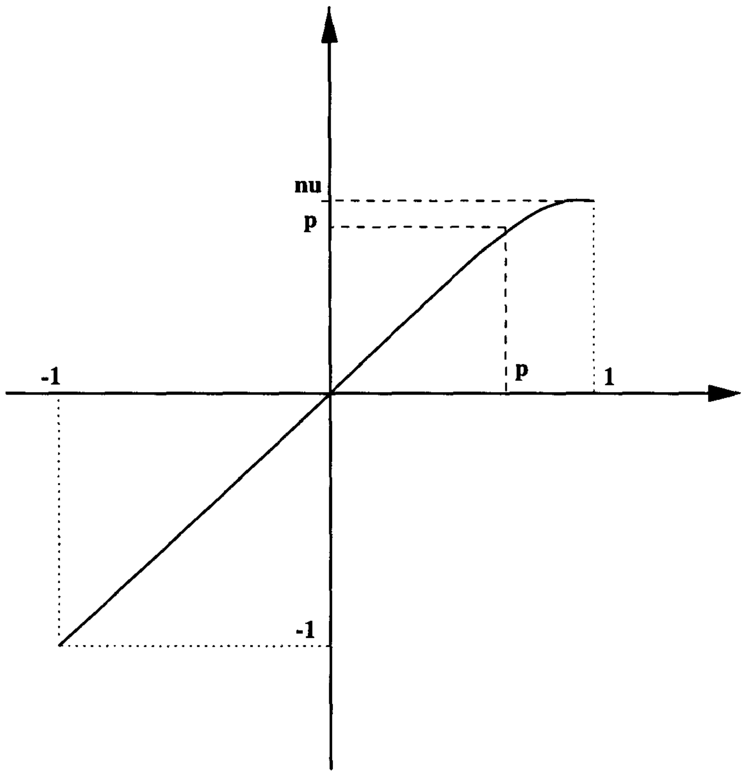


Figure 5.7: Capping function

Some explanations are needed on how to calculate the capping function. The capping function can simply map all the values greater than upper bound to the upper bound, and those less than lower bound to the lower bound, but this may lead to pauses for certain part of the skeleton during the motion. In our implementation, as the values exceed the bounds, they are mapped to values that gradually approach the bounds. As show in Figure 5.7, our capping function is a cubic interpolation spline between (p, p) and $(1, nu)$, where nu is the normalized upper-bound. The slope is 1 at (p, p), and 0 at $(1, nu)$. The figure shows only the capping of the upper bound; a similar capping function can be calculated for lower bound, and the upper bound capping and lower bound capping can be imposed at the same time.

Motion signal compression is local in the value dimension of the joint angle under consideration because only the values near or beyond the bounds are modified; but it is not local in the time dimension. First, the capping function is not calculated

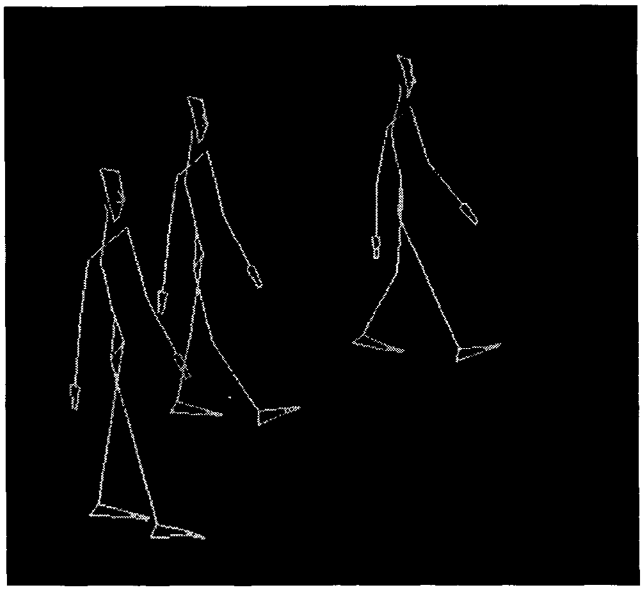


Figure 5.8: The result of motion signal squashing; the middle figure shows the result without constraint check, the front figure shows the result with motion signal squashing

at each violation segment, only the global maximum and minimum value contribute to the calculation of the capping function and the capping function is then applied to the whole signal; second, from Figure 5.7, we know p must be less than nu, those frames whose normalized value fit in range (p, nu) will be modified, even if they are not near to any violation segments.

Figure 5.8 illustrates the effect of signal compression. The back figure shows the original walk, the middle figure shows the result after tranform without constraint checking, and the front figure is the walk after bound mapping. You can see the joint angle of the knee in the middle figure is obviously out of the limit range and it is adjusted in the front figure. We cannot see significant differences between the result of bound mapping and that of signal compression in this example.

5.3 Handling non-linear constraints

While it is easy to specify and check linear constraints, it is very hard to handle non-linear constraints. Because the motivation of motion signal processing is to provide animators with high level control and release them from tedious editing work at the frame level, the ideal constraint handler for motion signal processing should also provide animators with high level control over the detection and adjustment of constraint violations. In this section, we will present a constraint handling method in which the animator just specifies constraints, and the system will detect and adjust constraint violations without any more instructions from the animator. The constraint handler has the following four steps:

1. Let the animator specify constraints;
2. Check the constraints on motion data produced by motion transformation, and get a list of constraint violations;
3. Adjust the constraint violations using inverse kinematics, and get a violation-free skeleton pose at each adjusted frame;
4. Smooth the motion locally by interpolating quaternions around the adjusted frames;

We choose to smooth the motion locally to preserve the characteristics of the checked motion as much as possible. In our system, interpolation for smooth motions is applied on joint angle data in the form of quaternions after the desired pose at the adjusted frame is calculated using inverse kinematics. An alternative approach might be to first interpolate the checked data (position, orientation or distance), and then applying inverse kinematics on each interpolated frame. We do not use this approach because inverse kinematics must be applied many times. Moreover, because two goal positions with a small difference may result in two poses with quite big difference after applying inverse kinematics, the motion adjusted by this approach may not be smooth even if the ckecked data have been smoothed before applying inverse kinematics.

Although there are all kinds of constraint violations resulting from motion transformations, most of the violations are geometric, i.e., related to joint angles, positions, orientations or distances. In general, violations of non-geometric constraints such as the balance of the body occur only when the gains of frequency bands are adjusted dramatically, providing the original motion does not violate the constraints. When this does occur, it is usually impossible to adjust these constraint violations without great change to the characteristics of the motion. Therefore, instead of trying to adjust these violations, we would like to take them as a failure of the motion transformation. Because motion transformation is at interactive speeds, it is acceptable to try several times before getting a satisfactory result. In this thesis, we will focus on geometric constraints.

5.3.1 Non-linear constraint specification

Dynamic simulation and procedural animation concentrate on particular motions of particular skeletons, and constraints can be built into the system; therefore, there are no constraint specification problems. Constraint specification in our context is not as easy as it appears to be; in fact, it is the most challenging part in the constraint handler. The difficulty of specifying constraints results from the generality of our motion transformation system. The motion transformation system we have built is intended to process any motion of any articulated skeleton as long as the skeleton description file is given. This makes it very difficult to specify the objective part of constraints at a high level; for example, you cannot use only the "right hand" to specify a human body point accurately. The diversity of motion makes it impossible to specify a constraint with high-level language such as "the feet strike the floor during a walk" because the system could not understand it. The diversity of skeletons and motions leads to the diversity of constraints, and it is impossible to build a constraint base containing all kinds of motions for all kinds of skeletons. It is very hard to find a way for animators to specify constraints conveniently and for the system to understand the specification easily.

We notice that geometric constraints usually involve some restrictions imposed on certain geometric elements, or some relations maintained between geometric elements. For example, "feet must not intersect with the floor" is a restriction on the position of feet, and "the hand cannot go beyond the mouth" for a drink movement is a relation between the position of the hand and that of the mouth. More complicated constraints might be such restrictions and relations that must be satisfied only when a specific condition is valid. One example is "the feet must touch the floor on each strike during walking".

Based on the observation above, we designed a way that animators can conveniently specify constraints on any skeleton and the system can easily understand the specified constraints. Animators can specify most of the geometric constraints they want to impose on a skeleton during motion transformation, although we cannot guarantee that all geometric constraints can be specified and checked this way.

We have formally defined non-linear constraints in Equation 5.2 and Equation 5.3. In this section we will discuss non-linear constraint definition in detail and give some restrictions on the definition. In the following discussion, we call Equation 5.2 a constraint unit, and Equation 5.3 a non-linear constraint. A constraint unit itself can be used alone as a non-linear constraint without condition.

- In the definition of non-linear constraints in Equation 5.2, we term $f_1(\theta)$ *goal data*, $f_2(\theta)$ *reference data*, and α *given value.* $f_1(\theta)$ is termed goal data because $f_1(\theta)$ will be modified if the constraint is violated, while $f_2(\theta)$ will remain unchanged.

- Although there is no big difference in the implementation, k in Eqution 5.2 is not really useful in practice, therefore, we will consider only the cases of $k = 1$ and $k = 0$. For different comparison operators and different values of k, we have the following *unit types*:

  - ABOVE GIVEN : $k = 0, ' >'$, the goal data is greater than given value;
  - BELOW GIVEN : $k = 0, ' <'$, the goal data is less than given value;
  - EQUAL GIVEN : $k = 0, ' ='$, the goal data is equal to given value;

  - ABOVE ANOTHER : $k = 1, ' >'$,
    the goal data is greater than the sum of reference data and given value;
  - BELOW ANOTHER : $k = 1, ' <'$,
    the goal data is less than the sum of reference data and given value;
  - EQUAL ANOTHER : $k = 1, ' ='$,
    the goal data is equal to the sum of reference data and given value;

- In Equation 5.2, f_1 and f_2 are functions of the joint angle vector θ. In this thesis, we support four types of functions: joint angle, position, orientation and distance. In practice, f_1 and f_2 are different but with the same type, therefore, the *concerned data type* of a constraint unit can be:

  - JOINT ANGLE : a joint angle of the skeleton;
  - POSITION : X, Y, or Z coordinate of a point;
  - ORIENTATION : rotation around X, Y, or Z axis of a segment in the world coordinate system.
  - DISTANCE : distance between two points;

  Constraint units with JOINT ANGLE data type are recommended for conditional constraints only; otherwise, it becomes a linear constraint.

- Constraint conditions can also be represented with constraint units because a condition is also usually a restriction on certain data or a relation between two data. Another kind of condition frequently used is the maximum or minimum frames; i.e., the condition is valid only at the frames where the concerned data reaches a maximum or minimum value. We introduce four more unit types which can be used only in the condition of constraints:

  - GLOBAL MAXIMUM : valid at the frame with global maximum value of concerned data;
  - GLOBAL MINIMUM : valid at the frame with global minimum value of concerned data;

  - LOCAL MAXIMUM : valid at the frames with local maximum value of concerned data;
  - LOCAL MINIMUM : valid at the frames with local minimum value of concerned data;

  A constraint condition is a list of constraint units linked with logical 'AND'. We do not use the logical 'OR' because its effect can be achieved by multiple constraints.

- Skeleton modeling presented in Section 3.1 is used in our system. Joint angle data is specified by a joint and a valid rotation axis; orientation data is specified by a joint and an axis of the world coordinates. A point is specified by a joint and an offset to the joint, and the point does not have to be in the figure body; if a point is specified without a joint, the offset is taken as the coordinates of the point in the world coordinate system. Position data is specified by a point and an axis, and distance data are specified by two points. To specify data in constraints, animators have to be familiar with the coordinate system of the skeleton, which is determined by the skeleton description file.

- The animator can let the constraint handler check a constraint only on a part of the motion by specifying the beginning and end frame. Usually the animator has a rough idea of which segment of the motion sequence may violate the constraint. Checking only the necessary part of the motion can save much time. Furthermore, it may turn a local maximum or minimum condition to a global maximum or minimum condition, which is much easier to check.

Based on the definition of non-linear constraint given above, we have the grammar for nonlinear constraint specification as follows:

```
constraint ::  condition-->body
body ::  unit
condition ::  unit_list | nil
unit_list ::  unit and unit_list
```

```
unit ::  '('unit_type, data_type, data1 [, data2] [, data3] [, data4],
          ignored_error [, given_value][, noise_width]')'
unit_type ::  'ABOVE_GIVEN' | 'BELOW_GIVEN' | 'EQUAL_GIVEN'
              |'ABOVE_ANOTHER' | 'BELOW_ANOTHER' | 'EQUAL_ANOTHER'
              |'GLOBAL_MIN' | 'GLOBAL_MAX' | 'LOCAL_MIN' | 'LOCAL_MAX'
data_type ::  'CMP_ANGLE' | 'CMP_POS' | 'CMP_DISTANCE' | 'CMP_ORNT'
data ::  '('joint[, offset][,axis]')'
joint ::  string | integer
offset ::  '('float, float, float')'
axis ::  'x' | 'y' | 'z' | 'X' | 'Y' | 'Z'
ignored_error::  float
given_value::  float
noise_width::  integer
nil ::  'NULL' | 'null'
and ::  'and' | 'AND'
```

ignored_error is the greatest error that can be ignored for comparison when the constraint is checked. The following are the rules for the usage of the optional fields in the constraint specification grammar:

- *data1* and *data2* represent the distance between those two points, and so do *data3* and *data4*, therefore, *data2* and *data4* are used only in the constraint unit with distance data;
- *data3* and *data4* are related to reference data, therefore, they are used only in *ABOVE_ANOTHER*, *BELOW_ANOTHER* and *EQUAL_ANOTHER* constraint units;
- *offset* cannot be used in constraint units with joint angle data or orientation data;
- *axis* cannot be used in constraint units with distance data;

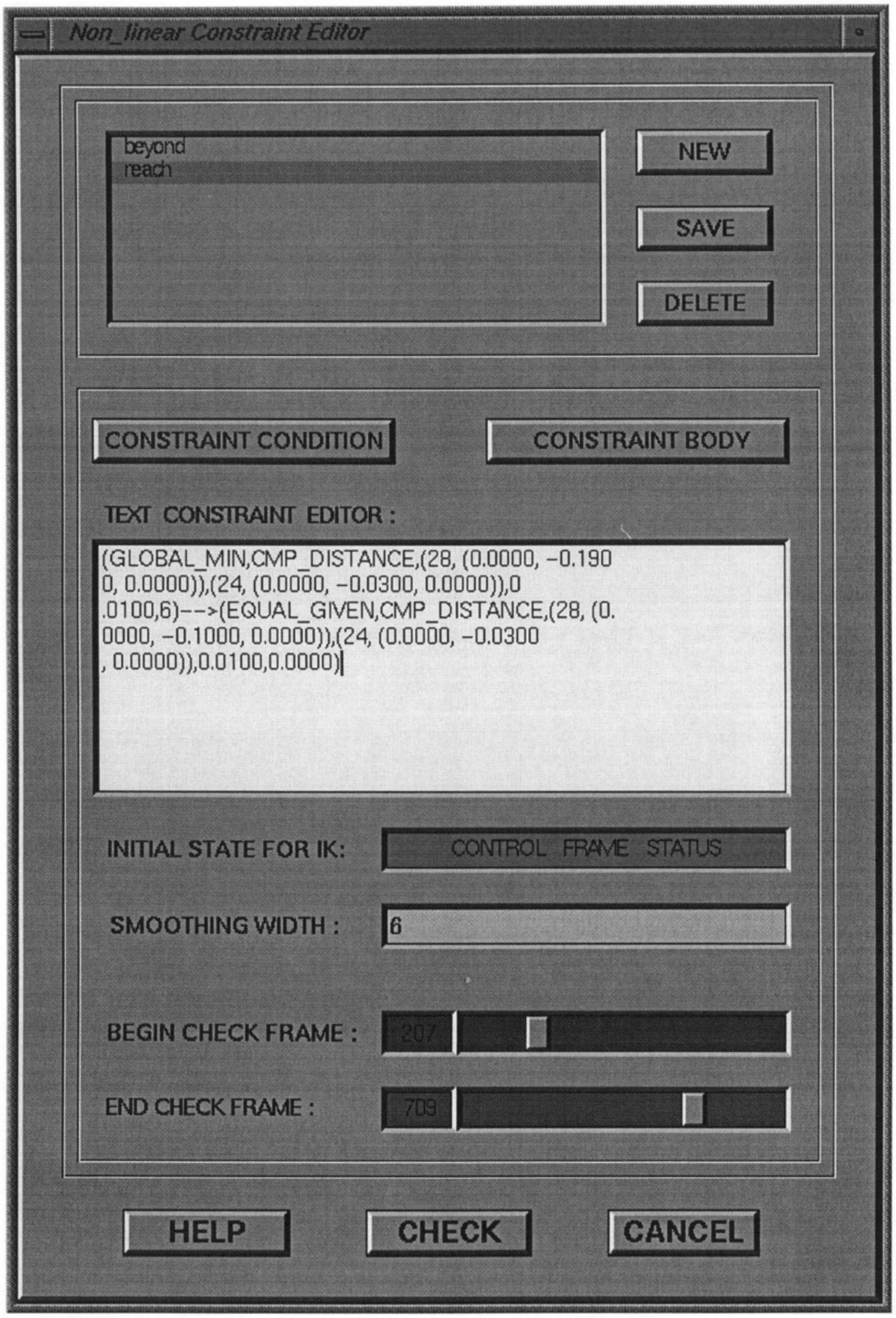
- *given_value* cannot be used in *LOCAL_MIN*, *LOCAL_MAX*, *GLOBAL_MIN* or *GLOBAL_MAX* constraint units;
- *noise_width* is used only in *LOCAL_MIN* or *LOCAL_MAX* constraint units, and will be explained later;
- *joint* can be the name or index of a joint; -1 (or 'non') means no joint is selected, and the system will use the offset as the point, and this can not be used in *data1*; '*phantom*' refers to the location of the skeleton;

For example, if we want to specify the constraint: "put cup to the mouth but do not go beyond the mouth" for a drink movement, we can decompose it into two constraints for specification: (1). the position of the cup in the X direction is greater than that of the mouth; and (2). at the frame where the distance between the cup and the mouth gets a global minimum value, the distance must be 0. The specification for these two constraints is:

```
null-->(ABOVE_ANOTHER, CMP_POS, (28, (0, -0.19, 0), x),
(''cervical_1'', (0, -0.03, -0.1), x), 0.01), 0)
```
and
```
(GLOBAL_MIN, CMP_DISTANCE, (''right wrist'', (0, -0.19, 0)),
(''cervical_1'', (0,-0.03, -0.1)), 0.01, 6)
-->(EQUAL_GIVEN, CMP_DISTANCE, (''right wrist'', (0, -0.19, 0)),
(''cervical_1'', (0,-0.03, -0.1)), 0.01, 0)
```

The point for the cup is specified by an offset to right wrist joint, and the mouth is specified by an offset to cervical segment #1. As we have said, the animator has to be familiar with the skeleton.

Figure 5.9 is the interface through which animators can specify constraints by entering the constraints in the constraint editor field using the grammar and the rules for the usage of optional fields in the grammar, or by filling in the form for constraint units. A constraint browser is provided for editing multiple constraints. The form for editing the constraint condition and the constraint body is available



```
(GLOBAL_MIN,CMP_DISTANCE,(28, (0.0000, -0.190
0, 0.0000)),(24, (0.0000, -0.0300, 0.0000)),0
.0100,6)-->(EQUAL_GIVEN,CMP_DISTANCE,(28, (0.
0000, -0.1000, 0.0000)),(24, (0.0000, -0.0300
, 0.0000)),0.0100,0.0000)
```

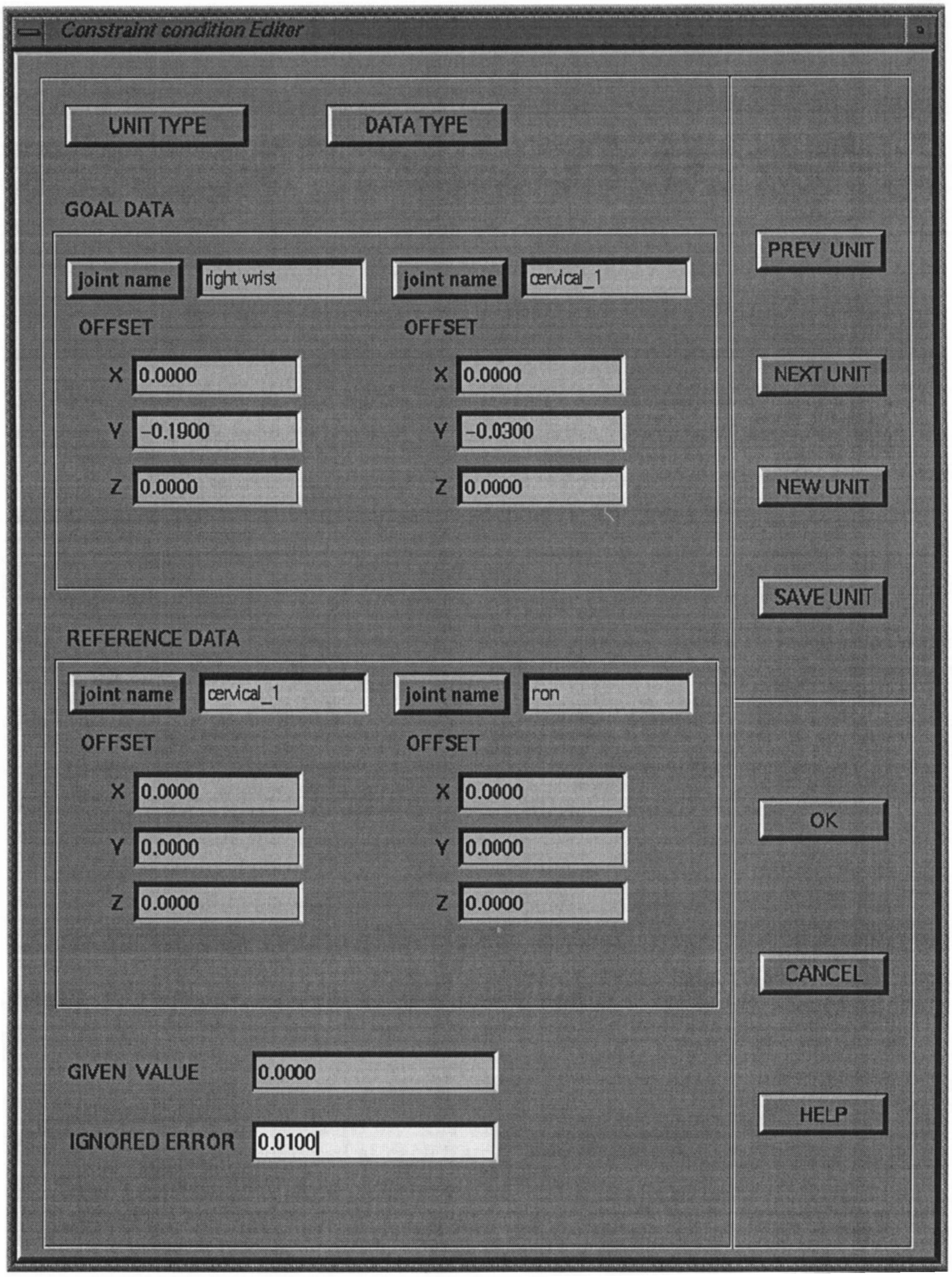
INITIAL STATE FOR IK: CONTROL FRAME STATUS

SMOOTHING WIDTH : 6

BEGIN CHECK FRAME : 207

END CHECK FRAME : 709

Figure 5.9: Interface for constraint specification



GOAL DATA

joint name: right wrist
OFFSET
X 0.0000
Y -0.1900
Z 0.0000

joint name: cervical_1
OFFSET
X 0.0000
Y -0.0300
Z 0.0000

REFERENCE DATA

joint name: cervical_1
OFFSET
X 0.0000
Y 0.0000
Z 0.0000

joint name: ron
OFFSET
X 0.0000
Y 0.0000
Z 0.0000

GIVEN VALUE 0.0000

IGNORED ERROR 0.0100

Figure 5.10: Form for constraint unit specification

to the animator by clicking on the corresponding buttons. Figure 5.10 shows the form for editing a constraint unit (the field for setting *noise_width* is not shown in the figure). A constraint or constraint unit can be loaded for modification at any time. The contents in the constraint editing field and in the forms are consistent, i.e. editing constraints in one way will be reflected in the other, and the animator can choose whatever s/he prefers and switch between them. It may seem that the form for specifying a constraint unit is very crowded and many fields need to be filled in. However, because many of those fields will be deactivated for each particular data type and unit type according to the rules for the usage of the optional fields in constraint specifications, and some of the fields have default value which are seldom modified, not too many fields need to be filled by the animator. The name and index of all the joints in the concerned skeleton are available to the animator at any point so that the animator does not need to memorize the spelling or the index of each joint.

If a constraint is specified by entering it in the editing field, the text typed in will be parsed, and the grammar and rules for the usage of the options in the specification will be checked by the parser; if a constraint is specified by filling in the forms, there will be no errors in the specification since the grammar and the rules are built into the forms.

5.3.2 Constraint violation detection

The second step for non-linear constraint handling is to detect violations of the constraints. First, the relevant data for each constraint unit are collected, then the condition of the constraints is checked, finally, the constraint body is checked at the frames where the condition is valid, and information about the violations is recorded for later adjustment. If a constraint has no condition, the constraint is checked at every frame.

A constraint unit may need both goal data and reference data, or only goal data. The relevant data are collected at each frame. No calculation is needed for joint angle data; the positions of points need to be calculated for position data and distance data, and the orientation of points need to be calculated for orientation data. The time required for calculating the required data at all frames is not a problem if the skeleton is not too complicated. For example, for the human figure skeleton used in our experiments, time is not a problem (no more time than the motion transformation itself).

Since the constraint units in a constraint condition are linked with logical 'AND' only, it is very easy to check the condition: for each condition unit, only the frames that have "passed" the check of all previous condition units need to be checked. The only challenge lies in how to find the real local minimum and maximum frames of relevant data. Figure 5.11 illustrates the problem: most likely, point A and B are not the local maximum points the animator wanted. Point A might just be noise, and one possible way to get rid of it is to smooth the concerned data first, but a simple smoothing algorithm may shift the real minimum or maximum point slightly. We take a simpler approach: intuitively a noise perturbation is very narrow, and motion transformation techniques are applied to densely-sampled data, therefore, if the width of a local minimum or maximum point (defined as the w shown in figure 5.11) is less than a threshold specified by the animator, we can consider it as noise instead of a minimum or maximum frame. To handle the circumstances of point B, we first get the list of all local minimum and maximum points, calculate all the slopes between two adjacent points, and get the average of the absolute value of the slopes, say *slope_average*. For each local minimum and maximum point, if the slopes on both sides are less than p**slope_average*, where p is a threshold value, then the point will not be taken as a local minimum or maximum point.

Although in most cases we can get only the local minimum and maximum frames the animator really wants if the threshold values are carefully selected, we have not essentially solved the problem; animators are advised to check constraints on a segment of the movements and thus turn a local minimum or maximum condition to a global minimum or maximum condition whenever it is possible.

After the condition is checked, the constraint body is checked at the frames where the condition is satisfied. If there is no constraint condition, or the condition is valid at all the frames that the constraint body is violated, then, as shown in Figure 5.12, the violation segment (the segment between *violation-begin* frame and *violation-end*

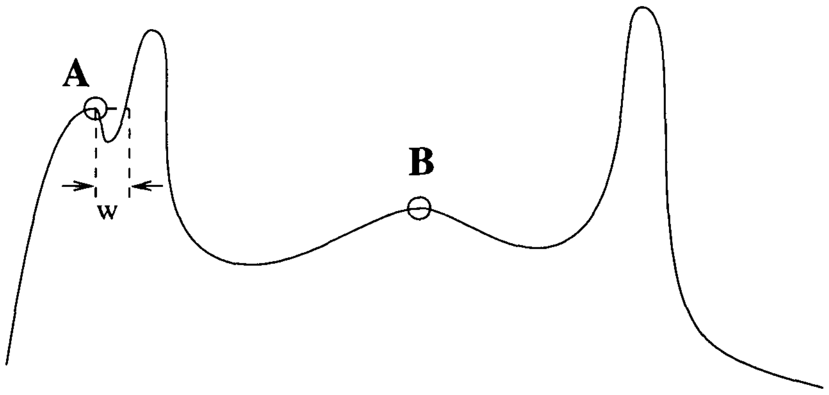


Figure 5.11: Problem of local minimum and maximum

frame) will be the segment where the constraint body is violated, and the frame with greatest violation will be the frame for later adjustment (called *adjustment frame* in the following discussion). Note that the curve in the figure represents the difference between goal data to reference data if there are reference data in the constraint unit, otherwise, it is the goal data. If the constraint has a condition, some of the frames where the constraint body is violated may not satisfy the condition, and therefore those frames does not violate the constraint, as shown in Figure 5.13. When only a part of the motion sequence is specified to be checked, this situation may also occur. In these cases, the constraint is satisfied does not mean the constraint body is satisfied. The most common special case of Figure 5.13 (b) is when the *violation-begin* and *violation-end* frame are the same, as shown in Figure 5.13 (c). This special case is common because whenever there is a maximum or minimum condition. After all the constraints are checked, all the violations are listed in the order of their adjustment frame.

5.3.3 Constraint violation adjustment with inverse kinematics

After the violations are detected, inverse kinematics is used to adjust the violations at their adjustment frame if the relevant data is not a joint angle. Inverse kinematics is applied only to the adjustment frame in the case of Figure 5.12, and used to the

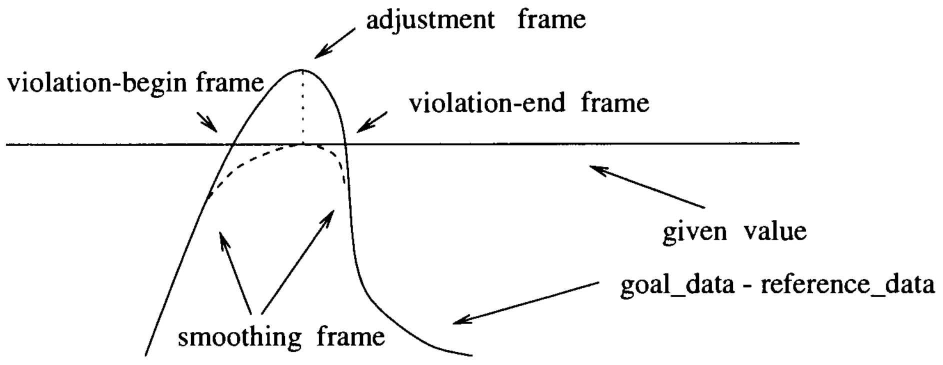


Figure 5.12: Detecting, adjusting and smoothing violations of uncoditional constraint

adjustment frame, *violation-begin* frame and *violation-end* frame in the cases of Figure 5.13.

If more than one violation needs to be adjusted at the same frame, multiple goals are set up for the inverse kinematics; if the goals are in conflict and the constraints cannot be satisfied at the same time, we assume that the animator made a mistake in constraint specification. For example, when two points in the same link of the skeleton need to be set to their desired positions, it is possible those two goals can not be reached at the same time. Therefore, animators should avoid specifying two potentially conflictive goals in the same link.

As presented in Section 3.2, there are many approaches to the inverse kinematics problem, and three implementations are available: the optimization based method implemented by S. Bawa [7], the Jacobian transpose method and the CCD method implemented by C. Welman [58]. In our problem, the manipulator is often in a singular configuration, for example, for a walking movement, a possible constraint violation is the intersection of a foot into the floor, and the foot should be adjusted to be on the floor. In most cases, the leg is fully stretched at the frame to be adjusted, and this is a singular configuration. Also our system is intended to work at interactive speeds, therefore, in this work, the CCD method is chosen to pull the end-effector from a constraint violation position to its desired position because the Jacobian transpose method suffers from singularity problems and the optimization based method is too slow to be used in an interactive environment. The CCD method implemented by C.

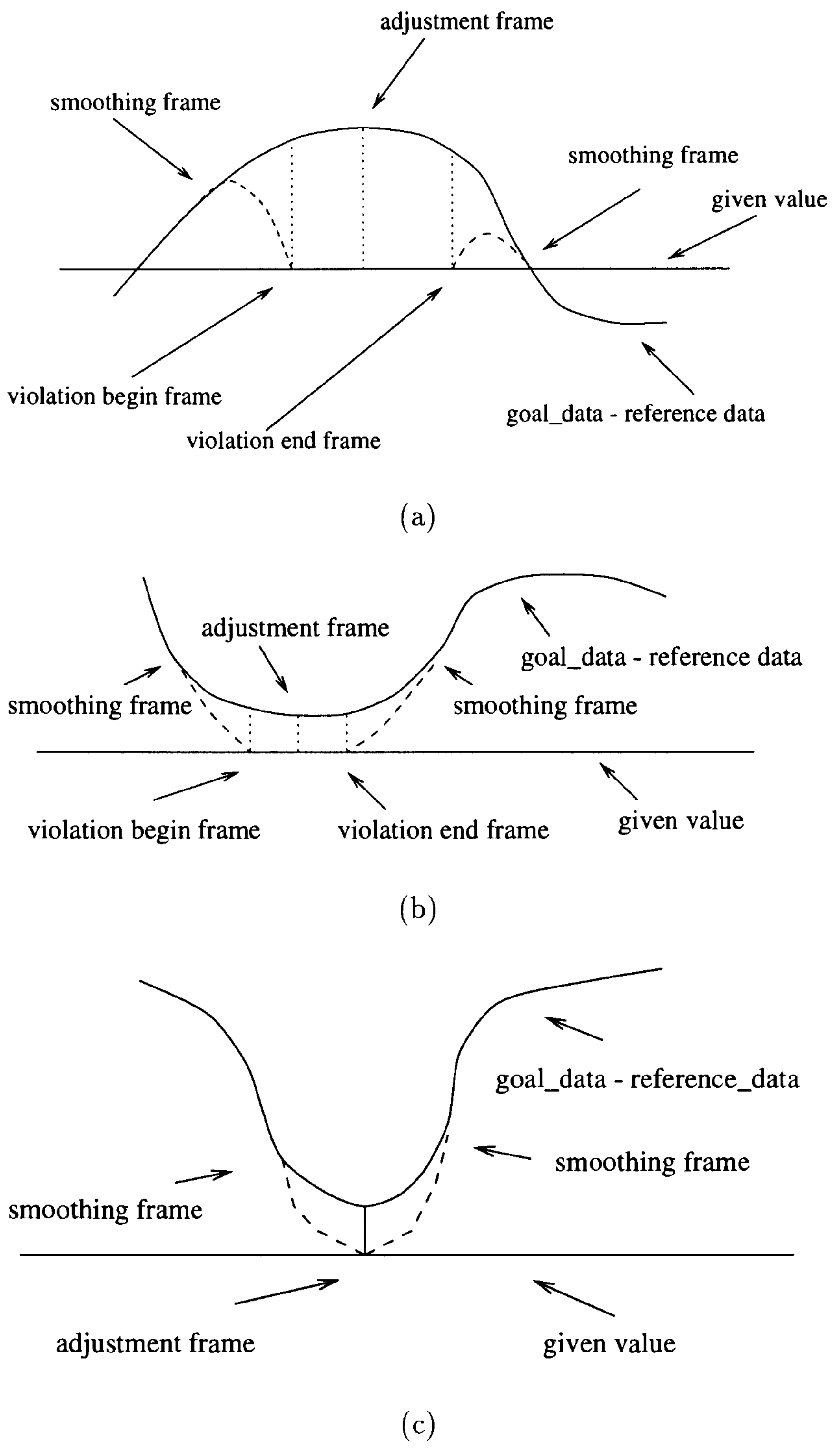


Figure 5.13: Detecting, adjusting and smoothing violations of conditional constraint

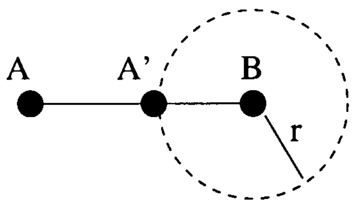


Figure 5.14: Goal for distance data

Welman can handle linear geometric constraints during the inverse kinematics procedure, and this enables us to impose the linear constraints on the skeleton, and thus consider linear constraints during the adjustment of non-linear constraint violations.

The goal for the inverse kinematics problem can be to achieve a position, an orientation, or a weighted combination of both of them. As shown in Figure 5.12 and 5.13, if reference data is available, the difference between goal data and reference data is checked against the given value; if there are violations which need to be adjusted, goal data will be adjusted, i.e., inverse kinematics will be applied only to the goal point (*data1* in the constraint specification). We need more explanation on the goal setup for inverse kinematics when distance data is concerned. In this case, the goal data is a distance between two points, say AB, and the position of A (*data1* in the constraint specification) will be modified to make the distance between A and B equal to r, while B (*data2* in the constraint specification) will be fixed. As shown in Figure 5.14, A can be "pulled" to any point on the circle centered at B with radius r to make $|AB| = r$. We choose A' to make the change minimal.

Two choices for the initial state for the inverse kinematics problem are provided to animators: the current state of the adjustment frame, or the state of one of the smoothing frames. The initial location is always the location of the adjustment frame. Generally, the former choice makes the resulting posture of the skeleton closer to the original posture at the adjusted frame and thus makes the change to the original motion smaller; the latter choice makes the resulting posture closer to that of the smoothing frame and thus makes the motion smoother. In most cases, there is no obvious difference.

To make the change to the original motion as small as possible, if the point to be adjusted is specified by joint J, we choose the nearest ancestor of J with multiple children to be the anchor for the inverse kinematics. Only the state of joints between J and the anchor will be changed during the inverse kinematics procedure. We have assumed that the reference point is fixed during the inverse kinematics procedure, therefore, one restriction of the constraint body is that the reference point should not be in the same link with the goal point and below the goal point's anchor.

If the goal is outside the space that can be reached by the point to be adjusted, for example, the foot can not reach the floor even if the leg is fully stretched, the location of the skeleton has to be changed, and this change will also be smoothed locally.

5.3.4 Motion smoothing with quaternion interpolation

After the violations are adjusted at adjustment frames, we need to providing local smoothing around the violations. Only the joint angles that have been modified for adjustment are interpolated. If the relevant data are joint angles, interpolation is applied directly to those frames between the adjustment frame and smoothing frames, as shown in Figure 5.12, or between the smoothing frame and the *violation-begin* or *violation-end* frame in Figure 5.13; otherwise, the dashed curve in Figure 5.12 and 5.13 is not interpolated directly from the relevant data; instead, the joint angles are interpolated in the form of quaternions.

In the cases of Figure 5.13 (a) (b), we have to get the pose of the skeleton at *violation-begin* frame and *violation-end* frame, then interpolate between the those five key frames: the smoothing frames, the beginning and end frames, and the adjustment frame. We cannot just interpolate between the adjustment frame and the smoothing frames as in the case of Figure 5.12, because that interpolation would not adjust the entire violation segment to valid state, as shown in Figure 5.15.

The *smoothing width* is the number of frames between the smoothing frame and the *violation-begin* or *violation-end* frame, and it is specified by the animator. When a smoothing frame lies out of the range of the motion signal being checked because the adjustment frame is near to the beginning or end of the signal, the motion signal is extended to that range with the value of the *violation-begin* frame or *violation-end*

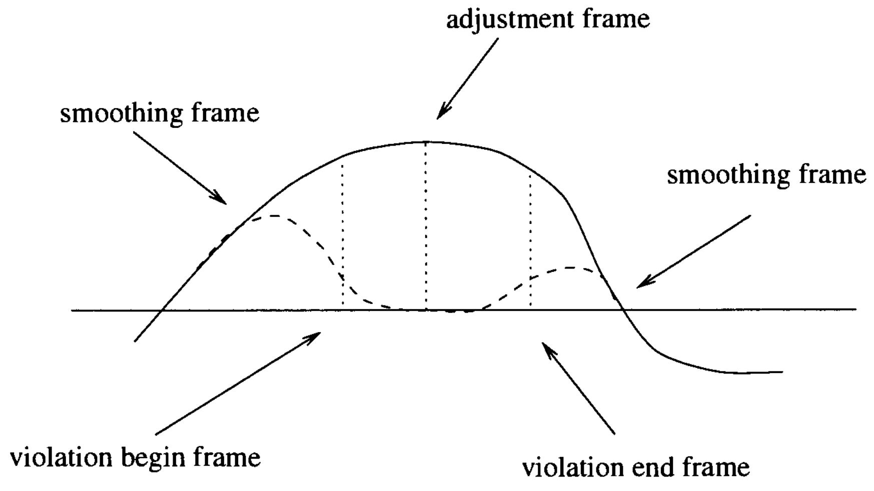


Figure 5.15: More than two interpolation segments needed

frame.

As discussed in Section 3.3.2, quaternion interpolation offers much smoother interpolation than with three independent angles and it is often used in keyframing animation. In this work, joint angles at each joint are converted into quaternions and after the quaternions are interpolated, they are converted back to joint angles.

We used the quaternion interpolation algorithm implemented in LifeForms[31], which was intended to interpolate quaternions between key frames. It is an implementation of Shoemake's algorithm [49], but modified to add the automatic bias control discussed by Kochanek [34]. This algorithm can handle an irregular keyframing space and the "undershoot" and "overshoot" problems, which is very important in our situation. The number of frames between the control points may be different; because the adjustment frame is the minimum or maximum point in Figure 5.12 and Figure 5.13 (c), it is very important to ensure that the interpolated curve does not have any minimum or maximum points between the adjusted frame and the smoothing frames, i.e. no undershoot or overshoot problem.

Constraint body with *EQUAL* unit types are usually used along with minimum or maximum condition. This is very useful when the animator wants to impose a constraint like "reach some place". Although "lock" constraint is not useful for linear constraint as we have discussed before, it is a useful type of nonlinear constraint.

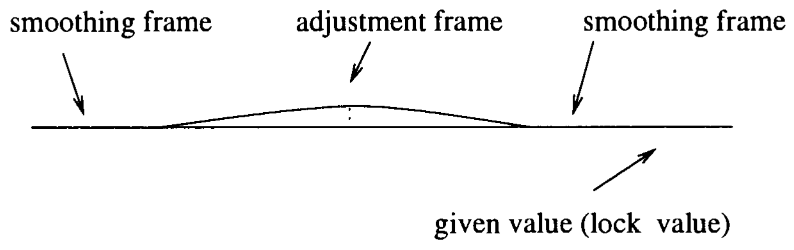
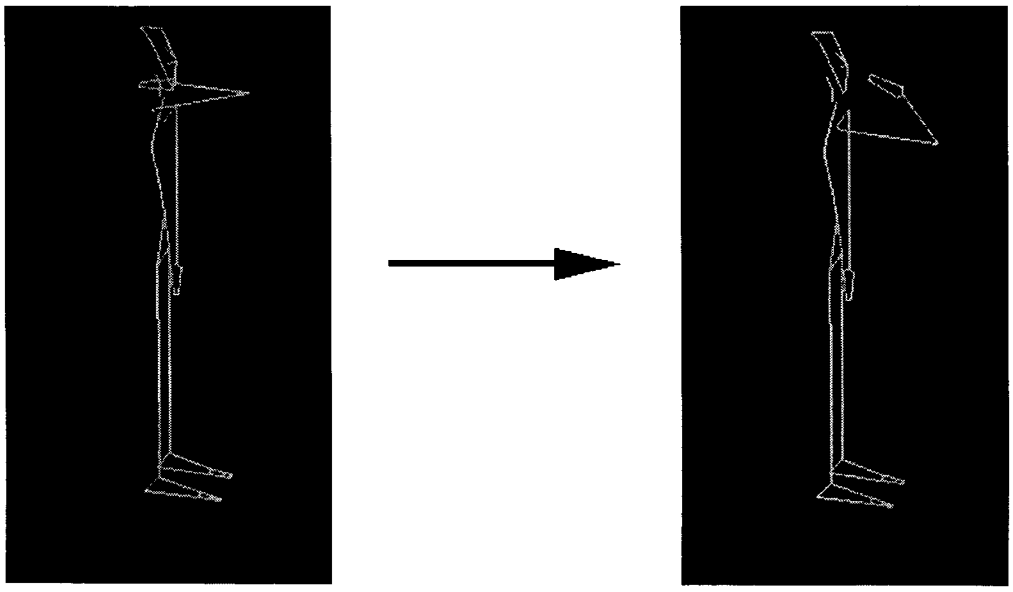


Figure 5.16: impose lock constraint

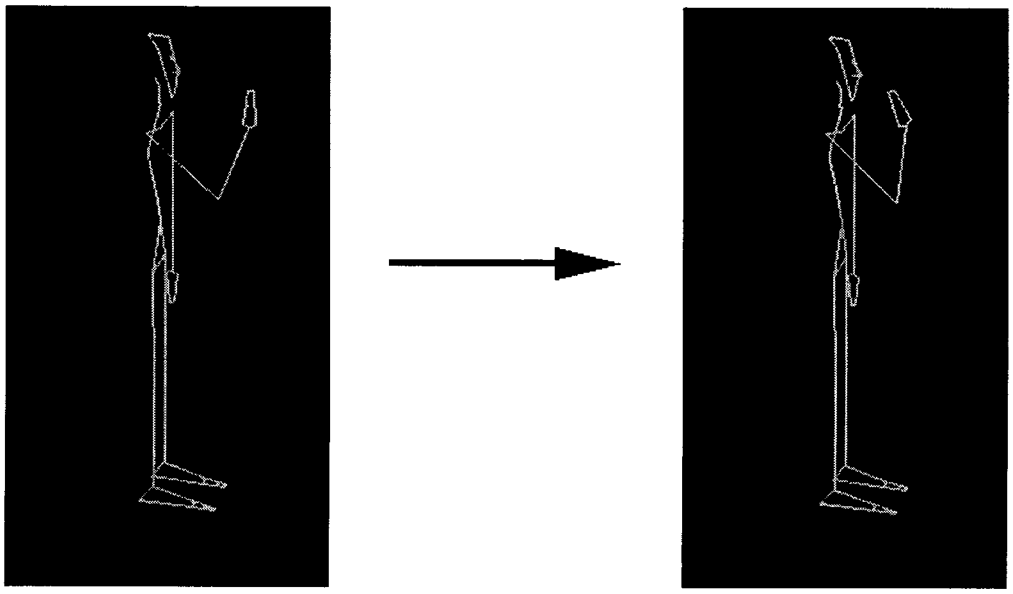
We do not separately define "lock" constraint as a constraint type, but it can be imposed using *EQUAL* constraint type. One problem is obvious if we look at Figure 5.12: after the adjustment, the relevant data is locked to the lock value only at the adjustment frame. This is because the smoothing frames are not in the lock state, therefore the frames in the interpolated segment are not in the lock state. Imposing nonlinear lock constraint in this way can get desirable result only when the frames on both sides of the violation segment are in the lock state, as shown in the Figure 5.16. The violation segment cannot be too long, otherwise there may be an undershoot or overshoot problem. If it is not the case in Figure 5.16 and the animator has to impose a lock constraint, the corresponding body part should be masked during the motion transformation.

5.3.5 Examples

As an example, Figure 5.17 shows the result of handling the non-linear constraints specified for a drink in section 5.3.1. In Figure 5.17 (a), the hand goes beyond the mouth due to the emphasis on low frequency bands of a captured normal drink movement. In (b), the cup does not reach the mouth because the gains of low frequency bands are set to low values; the picture is the snapshot when the hand comes nearest to the mouth. After the adjustment, the cup reaches the mouth exactly in both (a) and (b), and the characteristics are preserved in the modified movement. The cup is not displayed in the figure, therefore there is a distance between the hand and the mouth after the adjustment.



(a)



(b)

Figure 5.17: Result of non-linear constraint handling

5.3.6 Discussion

In the system which has been developed, the animator can specify a constraint by filling in forms or by entering the constraints following a simple grammar. Text-typing takes less steps, but all the elements of a constraint must be in a certain format and order, and might only be prefered by animators who are very familiar with the constraint grammar. Form-filling takes more steps, but the elements of a constraint do not need to be filled in with a particular order, and the system will ensure the form is filled correctly; furthermore, many of the fields in the forms are provided with default values which are seldom changed, therefore, new users may prefer form-filling.

Multiple constraints can be specified at the same time, and then checked separately, and the violations will be modified in the order of their adjustment frames without knowing which constraint a particular adjustment results from. In this way, all constraints are checked before any modification is made, and the modification of one constraint has no influence on the checking of another constraint. If animators want to check the constraints on the result of handling previous constraints, they can specify , check, and adjust one single constraint each time, and the process is repeated once for each constraint.

Obviously, after constraint checking, the interpolated segment of the motion does not preserve the characteristics of the original motion, and it is impossible to adjust the constraint violations without losing some characteristics of the motion. Because we only use the information of the frame with greatest violation for adjustment, all other information contained in the violation segment is lost. A possible solution to this problem is to record all minimum and maximum frames in a violation segment, set their goals for adjustment correspondingly, and those minimum and maximum frames can be used as key frames for interpolation. The challenge is how to set up their goals. We assume that constraint violation segments are short, and the interpolated segments of motion do not affect the overall characteristics of the whole motion.

5.4 Constraint handling with more control from animators

In the constraint handling methods presented in Section 5.2 and 5.3, constraints are handled totally by the system after they are specified by the animator. This makes it possible for the animator to control the constraint handling at a high level. We believe that most, but not all geometric constraints can be handled this way by our system. Furthermore, some animators want to play a bigger part in the procedure of constraint handling for full control over the animated motion; therefore, we provide some tools through which animators can control the constraint handling by manipulating relevant data in an interactive way or by editing the posture of the skeleton at a few key frames. These tools are provided as complementary approaches to those we have discussed in Section 5.2 and 5.3. The tools presented in this section can also be used for more general purpose, e.g., editing movements.

5.4.1 Signal curve editor

In the previous discussion, we assumed that the animator had a rough idea about the nature of the motion under consideration, and that the constraint specification is based on this rough information. In most cases, this rough information is enough, but sometimes more accurate information may be desirable. To provide more accurate and detailed information about a motion, we provide the specific information the animator requires at all frames with a curve, which we call a *signal curve* , as shown in Figure 5.18. We can provide information about joint angles, positions, orientations, and distances. Animators can specify the information they want in the same way that they specify the relevant data in non-linear constraints.

In Section 4.3, we edited a wave function, then applied it to a motion signal to get a new motion; here we can edit the motion signal directly, therefore, aside from providing accurate information which animators require, the signal curve also enables animators to detect and adjust constraint violations at a frame level, and if it is the information about joint angles, animators can also control the smoothing of the

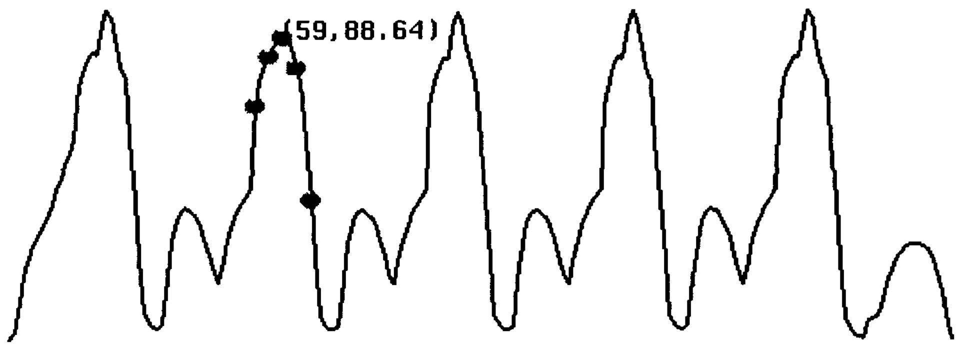


Figure 5.18: Signal editor

motion. For example, with the displayed signal curve, it is not difficult for animators to determine at which frame a local minimum or maximum frame occurs. Animator can also select the control frames on the displayed signal for constraint adjustment and motion smoothing as shown in Figure 5.18, where the dark point are the adjustment frame, and the grey points are constrol points for smoothing. Animators can edit the curve by pulling control points, and in this way they can decide how to adjust the constraint violations.

After the animator has detected the violations, decided how to adjust them, and selected the control frames for smoothing of the adjustment, the violations will be adjusted in a smooth way. If the signal curve is for joint angle data, the animator can control the interpolation curve between the control points by setting the control parameters of interpolation, thus having control over the smoothing of the adjustment. If the signal curve is for position, orientation, or distance, the procedure is the same as that for handling non-linear constraints. First, get the posture of the skeleton at the frame which is to be adjusted using inverse kinematics, and then interpolate the posture of the skeleton between that frame and its control frames using quaternions.

The signal curve editor provides animators with more control over constraint handling but still avoid tedious editing work on the skeleton.

5.4.2 Constraint handling with displacement mapping

Editing a motion directly on a skeleton at many or all frames is painful work for an animator, and it is unacceptable for handling constraints in our problem. However, direct editing can be acceptable if the animator just needs to edit one or a few frames on the skeleton and a skeleton editing tool at a relatively high level is available which enables the animator to "pull" an end-effector to a desired position without setting up each joint angle along the link, e.g., by using inverse kinematics. The displacement mapping technique we have discussed in Section 3.4.2 can be used for this purpose [13].

To handle constraint violations resulting from motion transformations, the motion produced is displayed, the animator finds the violations, and then changes the pose of the skeleton at one or a few frames where the constraints are violated. After this, the changes made to these frames are realized as displacements, a spline curve is fitted through the displacements for each degree of freedom involved, and added to the original movement.

Handling constraints in this way gives the animator full control. The animator can adjust the skeleton to the desired posture at the adjustment frames, and control the adjustment of the whole movement by adjusting parameters for displacement mapping. Since displacement mapping works at interactive speeds, and setting the parameters is also convenient, it is possible to edit the skeleton at the adjustment frames once and apply the displacement mapping several times until getting a satisfactory result.

In this chapter, we have presented the constraint handling techniques we developed in this work, and in next chapter, more experiments will be carried out to evaluate their performance.

Chapter 6

Experiments and analyses

We have discussed the motion transformation system and its constraint handling in previous chapters, and some examples are given to illustrate its performance. In this chapter, we first briefly discuss how to analyze the frequency properties of a movement, and then present the results of more complex constraint handling experiments to illustrate the problems that this system handles successfully and identify those where this approach cannot do very well. The movements used in many of these experiments are generated using a keyframe animation system called LifeForms[31], and the results are displayed also using this animation system.

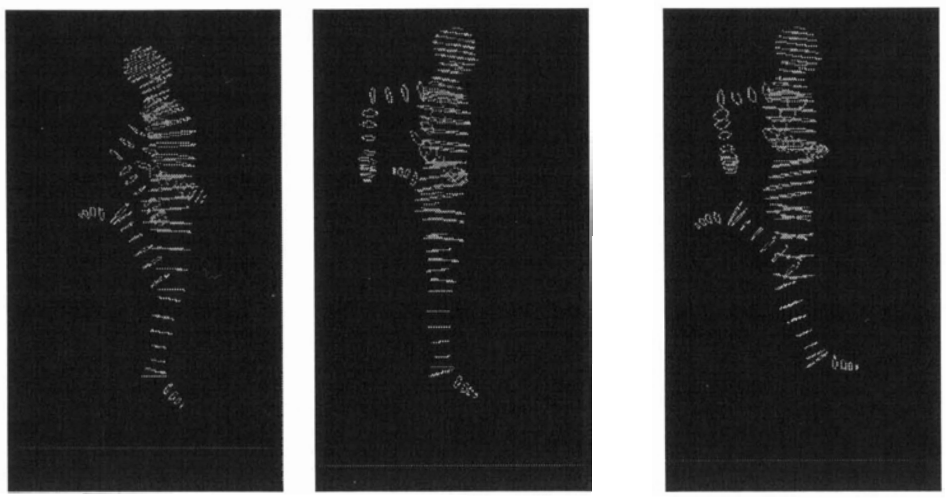
6.1 Frequency analysis

To generate a useful new movement from an existing movement using our motion transformation system, the animator usually has to know the frequency content of the movement, i.e., to know which frequency bands reflect which patterns of the movement. In Chapter 4, we have briefly discussed this using a walk movement as an example. Generally, the high frequency bands contain details and noise in the movement and low frequency bands determine the basis of the movement. This general knowledge is not enough to understand the details of the movement since the frequency property is different from movement to movement. For example, from the knock movement generated by motion capture, we have seen in Chapter 4 that the "knock " pattern is contained in the high frequency bands. The frequency property can be used to guide the generation of new movements and avoid some constraint violations.

To identify the movement patterns contained in a particular frequency band or a group of frequency bands, animators can set the gains of those frequency bands to zero and keep the gains of other bands unchanged, and thus we know that these particular frequency bands contain the movement patterns which are in the original movement but missing in the resulting movement. We can also keep the gains of the frequency bands we are interested in unchanged, and set the gains of all other frequency bands to zero, thus we know that the frequency bands we are interested in contain the remaining movement pattern in the result.

The frequency properties for different motion sequences can be very different. For some motion sequences, specific frequency bands contain different parts or phases of the movement; for example, for a running and then spiking-valley-ball sequence generated using LifeForms, we find that most of the running movement is in the low frequency bands while most of the spiking-valley-ball movement is in the middle frequency bands. This is because the spiking-vallay-ball movement is much faster than the running movement. For some other motion sequences, different frequency bands may contain the movement of different body parts, for example, in a swimming movement with which we have experimented, low frequency bands contain the movements of hands, and middle frequency bands contains the movements of the legs. If the animator is interested in only one movement in the motion sequence, then she/he can apply the motion transformation technique to only one segment of the motion sequence, or to only selected parts of the body.

We analyzed the frequency property of various movements, and found that the motion transformation technique works better for periodic movements such as walks than aperiodic movements, and this is to be expected since frequency analysis works better on periodic signals.



original jump transformed jump checked transformed jump

(a) (b) (c)

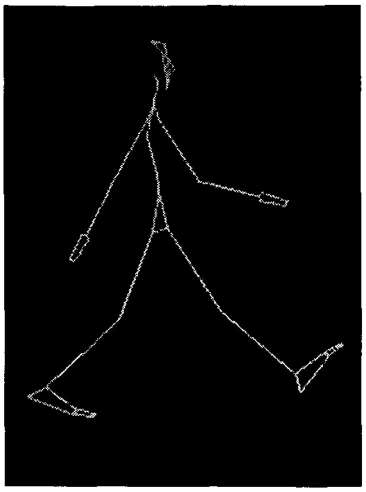
Figure 6.1: Impose linear constraint on jump

6.2 Experimental evaluation of the constraint handling

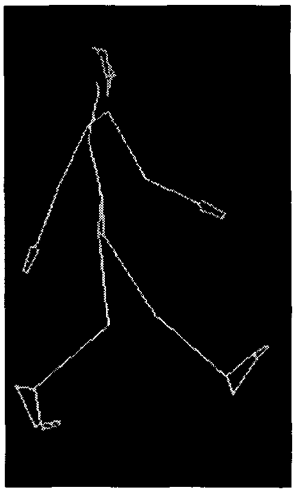
In this section, we will present some experiments we have carried out to evaluate the constraint handling in motion transformations, and we will provide analysis and discussion of the results of the experiments.

The linear constraint handler is simple, and it works very well in all experiments we have carried out. In the example of Figure 6.1, we emphasize the middle frequency bands of a jump movement and get an exaggerated jump with large movements in both the hands and the legs. In the resulting jump, the right leg gets into an unusual position, and we adjust this physically impossible result by imposing a linear constraint on the joint angle of the knee: it must be less than 150 degrees. Snapshots of the original jump, the jump after transformation without a constraint check and the jump with a constraint check are displayed in Figure 6.1 (a), (b) and (c) respectively.

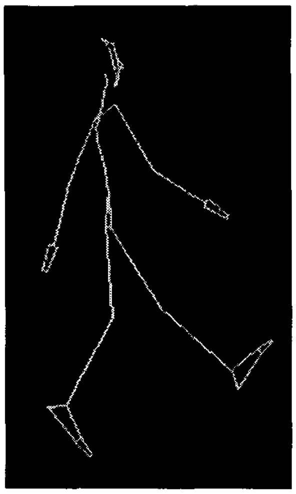
The figures in Figure 6.2 illustrate the handling of the *hit-the-floor* constraint for


walking above floor

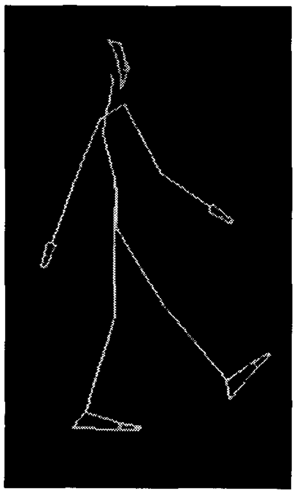
(a)


check with toes

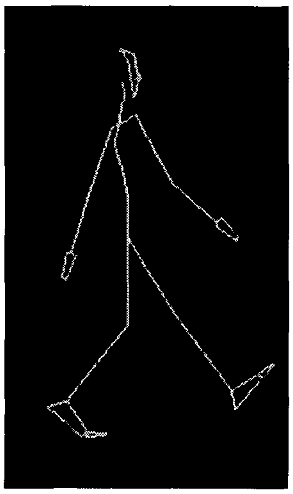
(b)


check with ankle

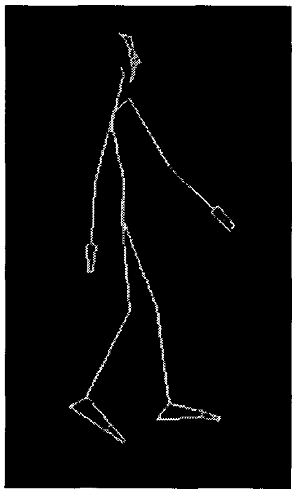
(c)


orientation constraint

(d)


joint angle limit

(e)


check two legs

(f)

Figure 6.2: The "hit-the-floor" constraint for walking

walking.

- (a) is the result of dramatically emphasizing the middle frequency bands of a normal walk; the feet do not hit the floor during the walk, and also, the joint angle of knees are out of normal limits. In this experiment, we specify the constraint in this way: when one foot is off the floor, the other must be on the floor. In (b),(c),(d) and (e), we just put the hit-the-floor constraint on right foot.

- (b) is the result when we specify the constraint based on right toes: when the position of left ankle is 7cm higher than the floor, the position of right toes must not be 2cm higher than the floor. The position of right toes will not be lower than 2cm above the floor after the adjustment because the position of right toes at the smoothing frames are no lower than 2cm above the floor.

- (c) is the result when we specify the constraint in a similar way, but the constraint body is based on right ankle: the position of right ankle must not be 7cm higher than the floor. In the resulting animation, the constraint on right ankle is satisfied, but the toes intersect the floor. From the results of (b) and (c), we can see it is important to choose an appropriate goal point for a constraint.

- (d) is the result of imposing an orientation constraint on the resulting movement of (c); now the right foot is directed forward instead of into the floor.

- (e) is the result of imposing the non-linear constraint of (b) and a linear constraint on the knees at the same time. Both of the constraints are satisfied in the resulting animation.

- When we specify the "hit-the-floor" constraint on one leg, we use the state of the other. The check and adjustment of the constraint on one leg will change the movement of that leg, but this change is unknown to the check and adjustment of the constraints on the other leg, which may lead to undesirable results. When we try to impose the "hit-the-floor" constraint on both legs at the same time, we get an awkward walk as shown in (f). To get a reasonable result, the constraint

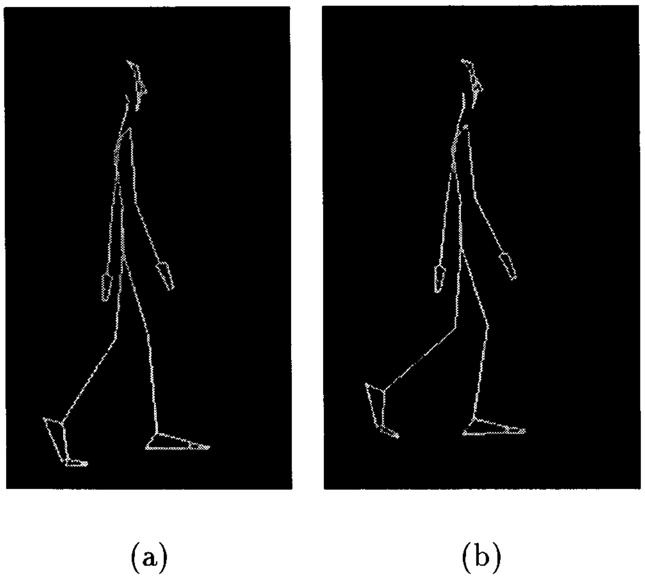


Figure 6.3: The no-intersection-with-floor constraint for walking

must be imposed on the legs sequentially, i.e., imposing the constraint on one leg first, then imposing the constraint on the other leg and checking it on the resulting movement of checking the constraint on the first leg.

Another constraint we imposed on a walk is that the feet cannot intersect the floor. In Figure 6.3, (a) is the result of setting the low frequency gains to a low value; we can see that the feet intersect the floor. This constraint is easy to specify: just keep the feet above the floor at all times; (b) is the resulting walk after applying the constraint.

As we have discussed before, locking a body part to a position during the entire motion sequence is not well supported in our constraint handling system in the case of Figure 5.12, but usually, the lock constraint is not continuously violated from the beginning to end of the motion sequence, and the violation is the case of Figure 5.16. In that case, we can use the EQUAL, ABOVE or BELOW unit types to impose lock constraints. In the "hit-the-floor" example we have given above, the feet are required to be locked to the floor under certain conditions, and this constraint is specified using a *BELOW-GIVEN* unit type.

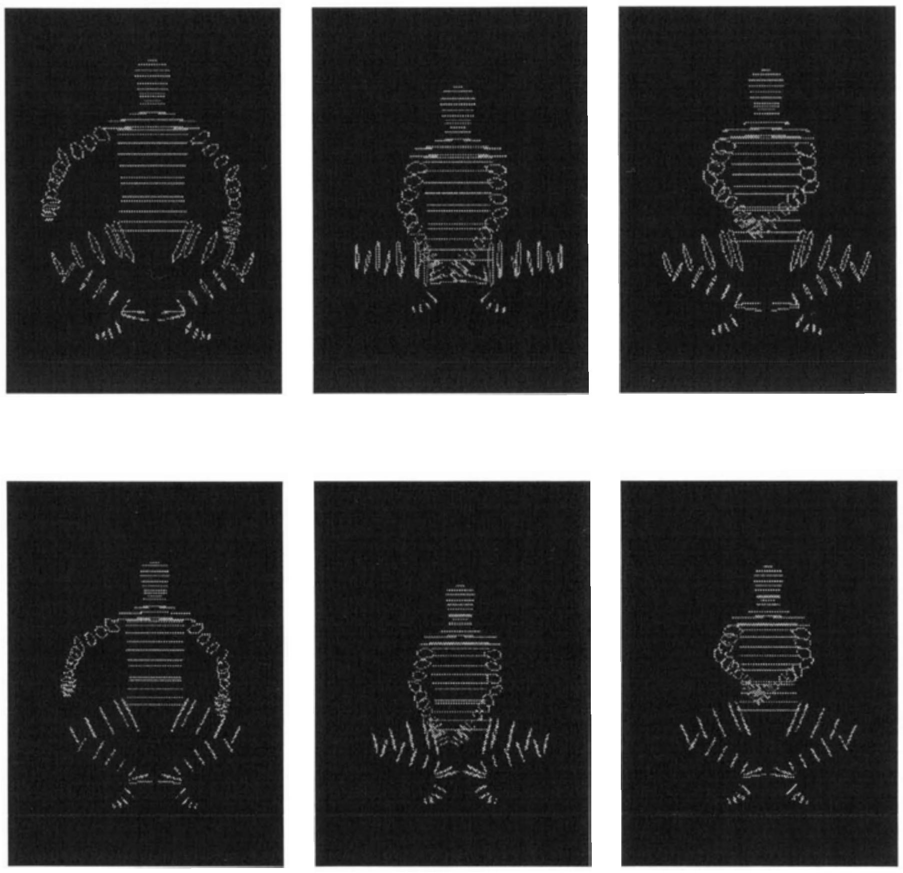
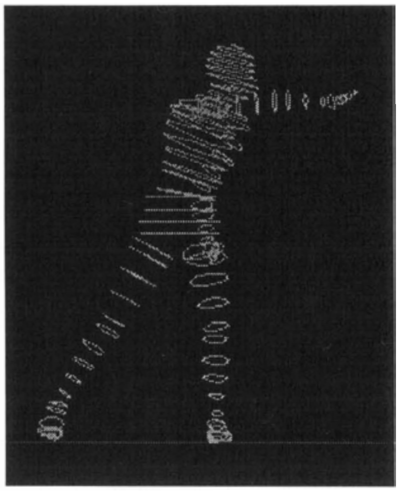


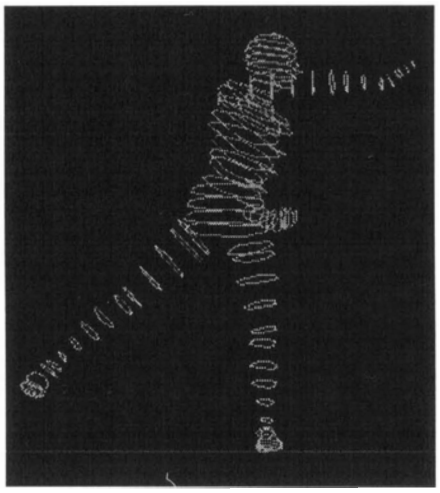
Figure 6.4: The result of imposing lock constraint

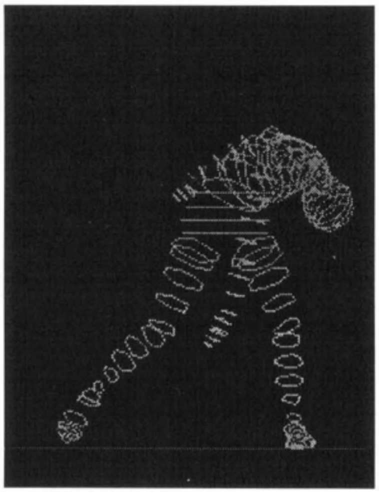
"Lock" is an important type of nonlinear constraint; for example, in one dance sequence which we experimented, the feet never get off the floor, after motion transformation, both feet get off the floor, and thus an unnatural result is produced. The figures in Figure 6.4 show the result of imposing lock on the feet. The first three figures are the snapshots of the uncorrected movement at different frames, and the second three are the snapshot of the corrected movement at those frames. During the whole movement, the toes are locked to the floor.

Whether a generated movement violates some constraints might need to be determined by the animators. For example, for a shot-put movement, the original shot-put figure never lifts either foot off the ground, and we get a more powerful shot-put by emphasizing the gains of low and middle frequency bands. In the resulting movement, the right foot lifts up. Figure 6.5 (a) and (c) are two snapshots of the original shot-put, (b) and (d) are two snapshots of the resulting shot-put. Lifting the foot is desirable for pushing out the ball in those frames. However, when teh body moves to the right side, the right foot should not be lifted to keep the body in balance, but in the resulting shot-put, when the body moves to the right side, the right foot is lifted, as shown in Figure 6.6(a), and this is not desirable. We specify the constraint like this: when the ball is on the right side of the body, the right foot must be locked onto the floor. The result of handling this constraint is shown in Figure 6.6 (b). The animator can also just let the constraint handler check one selected segment of the movement, and no constraint condition is required; the result is shown in Figure 6.6 (c), which has no big difference from Figure 6.6 (b). With an animation tools which can display the key frames of a motion sequence, such as LifeForms, it is easy for animators to determine which part of the motion sequence needs to be checked for a particular constraint. This example also shows that some non-geometric constraints like the balance of body can be achieved using geometric constraints.

Applying motion transformation to only a part of the skeleton can avoid some constraint violations. When a lock constraint needs to be imposed, masking related body parts might be the best way in many cases. In our motion signal processing system, data for position and orientation of the whole body are processed in the same way as joint angles, but sometimes, transformation of those data can lead to undesirable results; for example, when the motion transformation technique is applied to a gymnastic routine, if the orientation of the whole body is also transformed, we get a mess, and one snapshot is shown in Figure 6.7 (a); when it is masked and therefore not transformed, we get a reasonable result, and the snapshot of the same frame is shown in Figure 6.7 (b).


(a): original


(b): low/middle frequency emphasized


(c):original

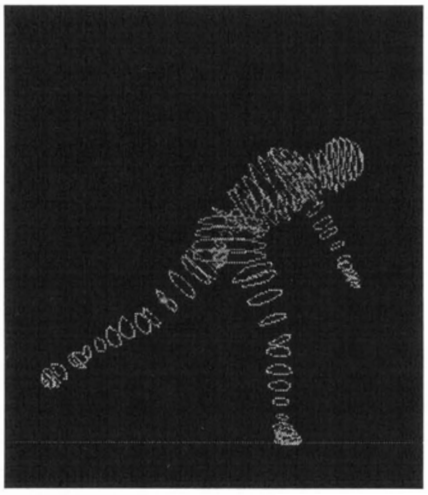

(d): low/middle frequency emphasized

Figure 6.5: Lift right foot for shotput

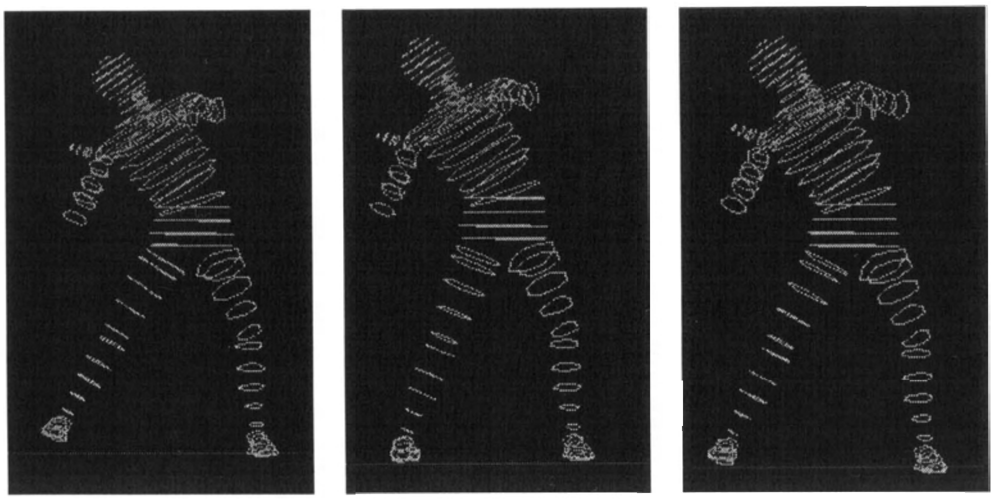

(a) (b) (c)

Figure 6.6: Lift right foot for shotput

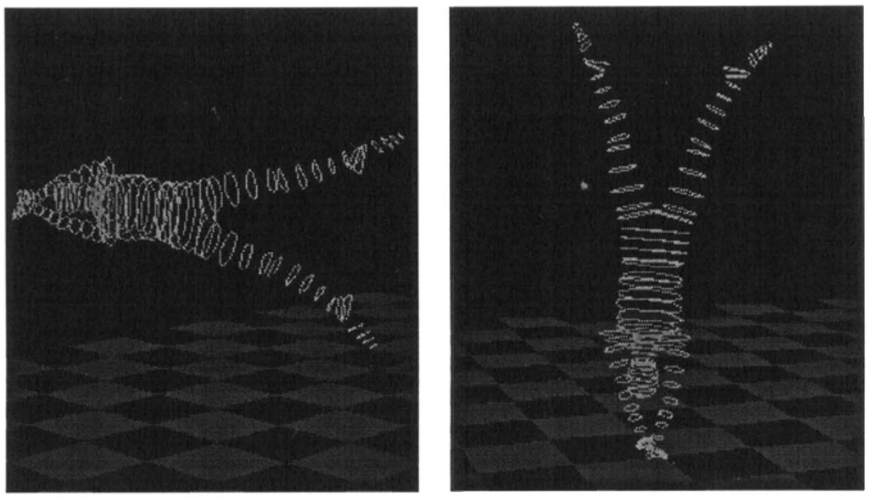

(a) (b)

Figure 6.7: Mask the orientation of the body

Chapter 7

Conclusion and future work

In this thesis, we provide animators with several convenient ways to handle the constraints for motion signal processing. The constraint violation problem is inherent to motion signal processing techniques and cannot be avoided. The constraint handling system described in this work can successfully handle most of the geometric constraint violations resulting from motion signal processing at interactive speeds, and animators do not have to do tedious editing work at each key frame, thus retaining the major advantage of motion signal processing. These constraint handling methods work for all kinds of motions on all kinds of articulated figures.

We first generally reduce the chance and degree of constraint violations by offsetting motion data and applying consistent dynamic timewarping (when applicable) before any motion transformation is applied. Linear constraint violations are successfully adjusted with bound mapping and motion signal compression techniques. The major contribution of this thesis is the design of a novel way for animators to specify non-linear geometric constraints, and for the constraint handling system to check the constraints, to adjust their violations with inverse kinematics technique, and to smooth the adjusted motion using quaternion interpolation. We also provide two complementary constraint handling methods, using a signal curve editor and displacement mapping techniques, to give animators relatively low level control over the constraint handling.

Although the constraint handling techniques presented in this thesis are intended to be used with motion signal processing and handle the constraint violations resulting from motion transformations, they can also be used for more general purposes. For example, they can be used to impose constraints on existing movements which are not necessarily the result of motion transformations.

The constraint handling system is intended to be used for all kinds of movements of all kinds of articulated figures, therefore, every constraint must be specified by the animator. When it is used for a particular skeleton and for a particular type of movement, some constraints should always be satisfied by default. For example, if it is used for human walking, walking "above floor" and walking "on the floor" should be default constraints. Since constraints can be specified in text, this is not a problem; those default constraints can be attached in the skeleton description file, and can be read to the constraint handling system and applied. The animator just needs to specify those constraints once. A constraint base may be built in this way for any particular skeleton.

7.1 Future work

This work provides a novel way to specify non-linear geometric constraints. The system should be used, examinated, and evaluated by animators, and a more friendly user interface and more convenient constraint specification methods can then be designed based on their feedback.

Editing motion sequences with the constraint handling system described here is much easier than the painful editing at each key frame which otherwise would be necessary. However, the animator still needs to be familiar with the skeleton, and has use quantities to specify a point. An alternative approach would be to let the animator specify the points on a display of a skeleton. The challenge lies in the fact that the specification must be 3 dimensional, and after specifying the points, it is still very hard to specify the constraints. This should be investigated.

In our constraint handling system, if a constraint is violated too much, or the violation segment is too long, the motion in the segment which is adjusted might be too fast or too slow. In Figure 5.13 (c), if the current state is too far away from its goal at the adjustment frame, the movement in the adjustment segment may be fast. The animator may want to lengthen the smoothing segments; this works but a longer smoothing segment means losing more characteristics of the movement being processed. On the other hand, in Figure 5.12 and 5.13 (a) (b), when the violation segment is too long, the movement in the adjustment segment will be slow, and it may even look like as if the figure is not moving. It seems that one solution is to change the length of the checked motion sequence, e.g., by adding or cutting some frames. The problem with this solution is to find a way to process the motion sequence of other body parts.

Although quaternion interpolation guarantees that there is no overshoot or undershoot problem for quaternions, it does not guarantee that there is no overshoot or undershoot for positions or orientations because position and orientation are non-linear functions of the interpolated quaternions. Therefore, when the violation segment is very long, there may be an overshoot or undershoot problem; we encountered this problem in one experiment, where the feet touch the floor at the smoothing frames and the adjustment frame, but do not touch the floor at some frames between them. Adding more control frames in the violation segment might be a solution to this problem.

In summary, the constraint handling system described here is complete and provides extensive functionality. It could be improved by including user friendly tools to handle a number of difficult situations.

Bibliography

[1] Joel Auslander, Alex Fukunaga, Hadi Partovi, Jon Christensen, et al. Further experience with controller-based automatic motion systhesis for articulated figures. *ACM Transaction on Graphics*, 14(4):311–336, October 1995.

[2] Norman I. Badler, Kamran H. Manoochehri, and Graham Walters. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications*, 7(6):28–38, June 1987.

[3] David Baraff. Linear-time dynamics using Lagrange multipliers. In *Computer Graphics (SIGGRAPH '96 proceedings)*, pages 137–146, August 1996.

[4] Alan H. Barr, Bana Currin, Steven Gabriel, and John F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *Computer Graphics (SIGGRAPH '92 proceedings)*, pages 313–320, 1992.

[5] Richard H. Bartels and Ines Hardtke. Speed adjustement for key-frame interpolation. In *Proceedings of Graphics Interface '89*, pages 14–19, June 1989.

[6] R. Barzel and A. Barr. A modelling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, 1988.

[7] Sumeet Bawa. *Interactive Creation of Animations: An Optimization Approach to Real-Time Inverse Kinematics.* M.Sc. Thesis, School of Computing Science, Simon Fraser University, 1995.

[8] P. Bergeron and P. Lanhapelle. Controlling facial expressions and body movements in the computer-generated animation short: Tony de Peltrie. In *Computer*

Graphics (SIGGRAPH '85), Course Notes: Techniques for Animating Characters, 1985.

[9] Lynne Shapiro Brotman and Arun N. Nelravali. Motion interpolation by optimal control. In *Computer Graphics (SIGGRAPH '88 proceedings)*, pages 309–315, August 1988.

[10] A. Bruderlin and T. Calvert. Goal directed dynamic animation of human walking. *Computer Graphics*, 23(3):233–242, 1989.

[11] Armin Bruderlin. *Procedural Motion Control Techniques for Interactive Animation of Human Figures.* Ph.D Thesis, School of Computing Science, Simon Fraser University, 1995.

[12] Armin Bruderlin and Tom Calvert. Interactive animation of personalized human locomotion. In *Proceedings of Graphics Interface '93*, pages 17–23, 1993.

[13] Armin Bruderlin and Lance Williams. Motion signal processing. In *Computer Graphics (SIGGRAPH '95 proceedings)*, pages 97–104, August 1995.

[14] P. Burt. A multiresolution spline with application to image merging. *ACM Transaction on Graphics*, 2(4):217–236, October 1983.

[15] P. Burt. Multiresolution method for image merging. In *Computer Graphics (SIGGRAPH '86 proceedings), Course Notes: Advanced Image Processing*, August 1986.

[16] Tom Calvert, Armin Bruderlin, John Dill, Thecla Schiphorst, and Chris Welman. Desktop animation of multiple human figures. *IEEE Computer Graphics and Applications*, 13(3):18–26, May 1993.

[17] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction for deformable animated characters. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 243–252, July 1989.

[18] Michael F. Cohen. Interactive spacetime control for animation. In *Computer Graphics (SIGGRAPH '92 proceedings)*, pages 293–302, 1992.

[19] R. Demori and D. Probst. *Handbook of Pattern Recognition and Image Processing*. Academic Press, 1986.

[20] David R. Forsey and Jane Wilhelms. Techniques for interactive manipulation of articulated bodies using dynamic analysis. In *Proceedings of Graphics Interface '88*, pages 8–15, June 1988.

[21] H. Fuchs, Z. Kedem, and S. Uselton. Optimal surface reconstruction from planar contours. *Communication of the ACM*, 10(10):693–702, 1977.

[22] Michael Girard. Constrained optimization of articulated animal movement in computer animation. In Norman I. Badler, Brian A. Barsky, and David Zeltzer, editors, *Making them move: mechanics, control, and animation of articulated figures*, pages 209–232. Morgan Kaufmann, 1991.

[23] J. E. Gomex. Twixt: a 3D animation system. *Computers and Graphics*, 9(3):291–298, 1985.

[24] S. Guo, J. Roberge, and T. Grace. Controlling movement using parametric frame space interpolation. In *Computer Animation '93, proceedings*, pages 216–227, 1993.

[25] James K. Hahn. Realistic animation of rigid bodies. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 299–308, August 1988.

[26] Gabriel Hanotaux and Bernard Peroche. Interactive control of interpolation for animation and modeling. In *Proceedings of Graphics Interface '93*, pages 201–208, 1993.

[27] Pat Hanrahan and David Sturman. Interactive animation of parametric models. *The Visual Computer*, 1(4):260–266, December 1985.

[28] John C. Hart, George K. Grancis, and Louis H. Kauffman. Visualizing quaternion rotation. *ACM Transaction on Graphics*, 13(3):256–276, 1994.

[29] C. Herr and B. Wyvill. Towards generalised motion dynamics for animation. In *Proceedings of Graphics Interface '90*, pages 49–59, 1990.

[30] Jessica K. Hodgins, Paula K. Sweeney, and David G. Lawrence. Generating natural-looking motion for computer animation. In *Proceedings of Graphics Interface '92*, pages 265-272, May 1992.

[31] Kinetic Effects Inc. *LifeForms (1.3).* Burnaby, BC, 1993.

[32] Hyeongseok Ko and Norman I. Badler. Straight line walking animation based on kinematic generalization that preserves the original characteristics. In *Proceedings of Graphics Interface '93*, pages 9–14, 1993.

[33] Hyeongseok Ko and Norman I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(2):50–59, 1996.

[34] D. Kochanek and R. Bartels. Interpolating splines with local tension, continuity and bias control. In *Computer Graphics (SIGGRAPH '84 proceedings)*, pages 33–41, 1984.

[35] Yoshihito Koga, Koichi Kondo, James Kuffner, and Jean-Claude Latombe. Planning motion with intentions. In *Computer Graphics (SIGGRAPH '94 proceedings)*, pages 395–407, 1994.

[36] J. Lasseter. Principles of traditional animation applied to 3-D computer animation. *Computer Graphics*, 21(4):35–44, 1987.

[37] Marc Lebrun. Digital waveshaping synthesis. *Journal of the Audio Engineering Society*, 27(4):250–266, 1979.

[38] P. Inkwell Litwinowicz. A 2 1/2-D animation system. In *Computer Graphics (SIGGRAPH '91 proceedings)*, pages 113–122, 1991.

[39] Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. Hierarchical spacetime control. In *Computer Graphics (SIGGRAPH '94 proceedings)*, pages 35–43, 1994.

[40] Anthony A. Maciejewski. Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications*, 10(3):63–71, May 1990.

[41] Sang Mah, Thomas W. Calvert, and W. Havens. Nsail plan: An experience with constraint-based reasoning in planning and animation. In *Computer Animation '94*, pages 83–92, May 1994.

[42] J. Thomas. Ngo and Joe Marks. Physically realistic motion synthesis in animation. *Evolutional Computation*, 1(3):235–268, 1993.

[43] J. Thomas Ngo and Joe Marks. Spacetime constraints revisted. In *Computer Graphics (SIGGRAPH '93 proceedings)*, pages 343–350, August 1993.

[44] F. Parke et al. State of the art in facial animation. In *Computer Graphics (SIGGRAPH '90 proceedings), Course Notes*, August 1990.

[45] Cary B. Phillips and Norman I. Badler. Interactive behaviors for bipedal articulated figures. In *Computer Graphics (SIGGRAPH '91 proceedings)*, pages 359–363, 1991.

[46] Cary B. Phillips, Jianmin Zhao, and Norman I. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. In *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, pages 245–250, March 1990.

[47] T Sederberg and E Greenwood. A physically-based approach to 2-D shape blending. In *Computer Graphics (SIGGRAPH '92 proceedings)*, pages 26–34, 1992.

[48] Ken Shoemake. Animating rotation with quaternion curves. In *Computer Graphics (SIGGRAPH '85 proceedings)*, pages 245–254, 1985.

[49] Ken Shoemake. Quaternion calculus and fast animation, computer animation: 3-D motion specification and control. In *SIGGRAPH 1987 Tutorial*, pages 101–121, 1987.

[50] Scott N. Steketee and Norman I. Badler. Parametric keyframe interpolation incorporating kinetic adjustment and phasing control. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 255–262, 1985.

[51] G. Stern. Bbop - A program for 3-dimensional animation. In *Nicograph Proceedings*, pages 403–404, 1983.

[52] A. James Stewart and James F. Cremer. Beyond keyframing: An algorithmic approach to animation. In *Proceedings of Graphics Interface '92*, pages 273–281, 1992.

[53] D. Sturman. Interactive key frame animation of 3-D articulated models. In *Graphics Interface*, pages 35–40, 1984.

[54] M. Unuma and R. Takeuchi. Generation of human motion with emotion. In *Computer Animation '93, proceedings*, pages 77–88, 1993.

[55] Michiel van de Panne. Parameterized gait systhesis. *IEEE Computer Graphics and Applications*, 16(2):40–49, 1996.

[56] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 225–234, August 1990.

[57] Wenping Wang and Barry Joe. Orientation interpolation in quaternion space using spherical biarcs. In *Proceedings of Graphics Interface '93*, pages 24–31, 1993.

[58] Chris Welman. *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation.* M.Sc. Thesis, School of Computing Science, Simon Fraser University, 1993.

[59] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, 1988.

[60] D. Zeltzer. Motor control techniques for figure animation. *IEEE Computer Graphics and Applications*, 2(9):53–59, November 1982.

[61] D. Zeltzer. Representation of complex animated figures. In *Proceedings of Graphics Interface '82*, pages 205–211, 1982.