Motion Control for Human Animation

THESE N^{o} 1601

PRESENTEE AU DEPARTEMENT D'INFORMATIQUE

POUR L'OBTENTION DU GRADE DE DOCTEUR ES SCIENCES

Par

Zhiyong Huang

Msc in Computer Science, Tsinghua University, Beijing, P. R. China

Director

Prof. Daniel Thalmann

Swiss Federal Institute of Technology, Lausanne (EPFL), 1996

Résumé

Nous avons développé et implementé un logiciel de contrôle de mouvement complet qui offre des fonctions innovatrices pour l'animation du corps humain.

La méthode de “dynamique en boucle fermée” est proposée afin d'avoir un contrôle interactif du squelette humain. La dynamique inverse permet de calculer la force ainsi que la torsion à chaque itération de la simulation dynamique à l'aide de l'algorithme d'Armstrong-Green.

La méthode basée sur plusieurs senseurs est utilisée pour le contrôle de la saisie. Une méthode heuristique est utilisée pour définir la stratégie de saisie. La cinématique inverse peut être utilisée pour définir la position finale des bras de manière à amener les mains autour de l'objet. Pour calculer l'interpolation entre la position de départ et la position finale des bras un système polynomial est utilisé. La dernière étape consiste à calculer les points de contact entre les doigts de la main et l'objet grâce a une méthode de détection de collision entre objets basée sur de multiples senseurs.

Nous élargissons par ailleurs la méthode de saisie à la manipulation d'objets et l'interaction entre humains avec les mains. Une méthode pour détecter et corriger l'auto-collision est proposée. Elle est basée sur une extension de la configuration a plusieurs senseurs et l'utilisation du deuxième aspect de la cinématique inverse. Une très bonne efficacité est obtenue et on peut directement l'utiliser pour un mouvement sans auto-collision. Enfin, la structure à plusieurs senseurs est appliquée à un modèle virtuel du CyberGlove de Virtual Technologies. En utilisant une méthode de saisie automatisée enrichie d'une détection de collision de la posture de la main nous pouvons simuler plus précisément la saisie par des acteurs réels munis du CyberGlove. Cette dernière méthode fonctionne en liaison avec la saisie par des acteurs synthétiques.

Pour manipuler les séquences de positions-clés, deux méthodes sont proposées: mélange fréquentiel de séquences de mouvement et compression de données de mouvement à l'aide de splines cubiques. D'autres techniques de manipulation sont implémentées: édition interactive de splines et de structures pyramidales multirésolution, déformation interactive du temps, compression de données basées sur la structure pyramidale.

Nous utilisons les capteurs 3-D Flock of Birds de Ascension Technology et un convertisseur anatomique temps réel développé dans notre laboratoire pour la conception des animations. Les mouvements saisis grâce au CyberGlove sont interprétés pour l'interaction tridimensionnelle.

Abstract

We have designed and implemented an integrated motion control software called Track which offers innovative functions for human animation.

The method of closed loop dynamics is proposed for interactive motion control of a human skeleton. Inverse dynamic functions calculate the required force and torque at each interval in the loop of the forward dynamic simulation based on the Armstrong-Green algorithm.

The multi-sensor based method is proposed for grasping motion control. A heuristic method is defined to decide among different grasping strategies. Inverse kinematics can derive the final posture of the arms to bring the hands around the object. Then, a group of polynomials is used to interpolate between the initial and final arm postures. Finally, Multi-sensor object collision detection chooses the finger contact points so that the hands can fully grasp the object.

We extend the grasping method to manual object manipulation and human interaction. The method of body self-collision detection and correction is proposed. It is based on the extension of multi-sensor configuration and the use of the secondary task of inverse kinematics. It is very efficient for self-collision detection and can directly result in the movement that is free of self collisions. We finally model the same multi-sensor structure on the 3-D hand model of the CyberGlove made by Virtual Technologies. Using a method based on grasping automata and hand posture collision response, we simulate more precisely the interactive grasping by a real person wearing the CyberGlove. It works along with the synthetic actor automatic grasping.

Two methods are proposed for the manipulation of keyframe sequences: motion blending on its frequency band and motion data compression using cubic splines. We also implement other manipulation techniques: interactive sequence editing on both spline curves and multiresolution pyramid structure, interactive time warping, and data compression based on the pyramid structure.

We have described the animation design using the Flock of Birds 3-D sensors made by Ascension Technology and a real-time Anatomical Converter developed in our lab. The gestures captured by the CyberGlove are recognized for 3-D interaction.

Acknowledgments

The most sincere thanks go to Prof. Daniel Thalmann, the director of EPFL-LIG, for accepting me as an assistant in his lab, allowing me to embark on a new path of my academic career. His enlightening support during the whole period as my thesis supervisor helped me to extend my understanding and continue progressing in my research. I wish to express my sincere gratitude to Prof. Nadia Magnenat Thalmann, the director of MIRALab - Centre Universitaire d'informatique at the university of Geneva, for her insight, direction and encouragement.

I would like to thank Dr. Ronan Boulic, the Scientific Adjunct of LIG, for his guidance, discussion, collaboration, and furthermore, the detailed review of the manuscript. I extend my thanks to other members of LIG for their collaboration, especially Srikanth Bandi for detailed checking of the manuscript, Tom Molet for his development of basic libraries and a motion capturing toolkit, Dr. Jianhua Shen and Eric Chauvineau for the library of Body Deformation, Serge Rezzonico for the use of the CyberGlove and head-mounted display, Mireille Clavien and Patrick Keller for making video demos, Luc Emering and Pascal Becheiraz for the collaboration of the grasping function in Agentlib, Hansrudi Noser for the framework of Agentlib, Tolga Çapin for the parallelism of the forward dynamic function, Selim Balcisoy for providing images produced by him, Christian Miccio for the translation of the abstract into French, Paolo Baerlocher and Walter Maurel for the help in dynamics, Christian Babski for helping to use the Macintosh, Oliver Paillet for improving the user interface, writing users manual and system testing of the Track software, and Dr. Russell Turner, Dr. Enrico Gobbetti and Dr. Francis Balaguer for the development of LIG Toolkit.

I would like to thank all staff members of MIRALab for their collaboration, especially Dr. Prem Kalra for the coordination of the project, Dr. Laurent Moccozet for the Hand Deformation, Karin Blanch, Mehdy Davary, Silvie Holowaty and Jean-Claude Moussaly for their nice suggestions and video sequences using the Track software.

I would like to thank Prof. Zesheng Tang, Prof. Jiaguang Sun and other former colleagues of Tsinghua University in China.

I would like to thank the collaboration from University of Karlsruhe in Germany, Prof. Alfred Schmitt, Kurt Saar and Bernd Lintermann, in the European ESPRIT HUMANOID Project.

I would like to thank Prof. Gerard Hégron of École des Mines de Nantes in France for his helpful suggestion and discussion of dynamics and grasping during the ESPRIT HUMANOID Project Review.

I would like to thank Dr. Ramon S. Mas for his previous work on grasping motion control. I would like to express my thanks to former EPFL student Yvan Greppin for the Augmented Vision project and the Multiresolution Sequence project, and former postgraduate student Solli Hege for making demos and system testing.

I would like to thank Josiane Bottarelli, the secretary of LIG, and Geneviève Rime, the secretary of DI, for their time and patience to me. I would like to thank all Chinese friends who helped me during this period, especially Dr. Xiaoqun Wang of IRRMA and Dr. Changyuan Hu of LSP.

I would like to thank Dr. Christopher P. Fuhrman of LIT for checking the final manuscript during Christmas.

This research is supported by the Swiss National Research Foundation and the Federal Office for Education and Science in the context of the European ESPRIT HUMANOID Project.

Contents

1. Introduction

In human animation, a fundamental problem is to generate and control the movement of the articulated structure representing the human skeleton. This control process is called motion control in human animation. The purpose of this thesis is to investigate the novel methods and construct a new software system for motion control in human animation. In this chapter, we first discuss the importance of this subject, raise the main problems in this field and then describe specific subjects investigated in this work. Finally, the organization of the thesis is outlined.

1.1 Importance of the subject

Human animation is still one of the most important research fields after a decade of development. With the rapid progress of virtual reality (VR) technology, the requirement for simulating realistic synthetic actors in virtual environment becomes a new direction for human animation. In the near future, we expect any human individual can be modeled and placed in a virtual environment in which any human behavior can be simulated. Moreover, the synthetic humans living in the virtual world can communicate with people in the real world. In multimedia applications, human animation is taking an increasingly important role in education, entertainment and telecommunication.

Many important results have been achieved, though there are only a few teams working in human animation. One of the most well known teams is from the University of Montreal and is directed by Magnenat-Thalmann and Thalmann. In May, 1987, Rendez-vous à Montréal, a 7-minute film directed by them brought Hollywood stars Marilyn Monroe and Humphrey Bogart back to life, marking a breakthrough of Human Animation. They also introduced new techniques in synthetic actor animation by using computers [Magnenat-Thalmann and Thalmann, 87].

The Montreal team has been working together as two teams in Switzerland since 1988. One is MIRALab in Geneva directed by Magnenat-Thalmann, and the other is LIG in Lausanne directed by Thalmann. The two teams continue to work in human animation but on different aspects. The research in MIRALab is mainly on facial animation [Magnenat-Thalmann et al., 88a, Kalra and Magnenat-Thalmann, 94], hand deformation [Magnenat-Thalmann et al., 88b, Gourret et al., 89, Moccozet, 96], clothes animation [Lafleur et al., 91, Carignan et al., 92, Volino et al., 96], while the LIG is mainly on body modeling and deformation [Shen and Thalmann, 95, Thalmann et al., 96], walking [Boulic et al., 91], grasping [Mas and Thalmann 94, Huang et al., 95], synthetic vision [Renault et al., 90, Noser et al., 95], synthetic audition [Noser and Thalmann, 95], motion control systems [Boulic et al., 94], motion capturing [Molet et al., 96], human animation toolkit [Turner et al., 90], camera control [Turner et al., 91], body balance [Boulic and Mas, 96], collision detection [Bandi and Thalmann, 95], and networked virtual human [Noser et al., 96].

Another well known team is from the University of Pennsylvania and is directed by Badler. Most of their work is reported in a recent book [Badler et al., 93b], including parametric keyframe for human articulated figure [Steketee and Badler, 85], inverse kinematics [Phillips et al., 90], body balance [Phillips and Badler, 91], spine modeling

[Monheit and Badler, 91], walking [Ko and Badler, 93], motion capturing [Badler et al., 93a], human collision detection and correction [Zhao and Badler, 95], and intelligent motion planning [Badler et al., 96].

Other teams also work on specific subjects of human animation such as a dynamic walking model [Bruderlin and Calvert, 89], human dynamics [Kunii and Sun, 90], dancing [Calvert et al., 91], physics-based facial animation [Terzopoulos and Waters, 90], vision-based facial animation [Essa and Pentland, 94], and human athletics simulation [Hodgins et al., 95].

All this research in human animation has brought new methods and techniques in modeling, motion control, interaction and visualization. Human animation is also stimulated from the development of other fields such as computational geometry, mechanics, medicine, optimal control, signal processing, robotics, etc., and the availability of new devices, hardware and software [Ascension, 94, Rohlf and Helman, 94].

Human animation is important for multimedia and virtual reality. The standard for human model has already been discussed in the proposal of MPEG-4 Synthetic/Natural Hybrid Coding Development of Media Model Standards. MPEG-4 is soliciting technology on the coding for storage and communication of 2-D and 3-D scenes involving Synthetic/Natural images, sounds and animated geometry. One of the challenges of the proposal is parameterized animated models [ISO 96]. MPEG-4 will apply to a VRML (Virtual Reality Modeling Language) server environment.

1.2 Main problems and our objectives

Though there has been much progress in human animation, there are only a few human animation systems integrating the different levels of motion control functions and different sequence manipulation tools. Even in these systems, the human models are often oversimplified and most of them are only 2-D. As the viewing humans are very familiar with them, their eyes are sensitive to even a tiny unrealistic feature, and such a detail can not escape notice.

It is important to have a clear objective, as it is not possible to solve all the problems in human animation in a single thesis topic. This thesis is specified in the framework of the ESPRIT HUMANOID Project that is on the research of a more complete set of fundamental problems in human animation. Its main topics are as follows:

- human modeling
- motion control of the human body
- deformation of a body in motion
- interaction of a virtual human with the 3-D scene
- grasping
- walking
- facial animation

The ESPRIT HUMANOID Project provides a more general framework, based on which, we select the following aspects for this thesis:

- *Investigation of dynamics for an human articulated figure*

Forward dynamics naturally produces physically realistic movement. We propose the use of inverse dynamics in closed loop with forward dynamics for interactive motion control of a human skeleton. An efficient recursive algorithm based on Newton-Euler formulae is adapted to calculate the force and torque produced by joint actuators in order to fulfill a desired motion. The resulting force and torque are used in forward dynamics to make the final motion with external force and torque. The Armstrong-Green algorithm is used for forward dynamic simulation. Inverse dynamic functions calculate the required force and torque at every small time interval in the process of forward dynamic simulation. In this way, it can correct errors at each time interval.

. *Investigation of hand motion control methods*

The use of the hands is one of the most significant aspects of a human being. The large number of degrees of freedom in the hands is one of the major problems for their motion control. We propose the following methods on hand-motion control. The multi-sensor heuristic for grasping combinatorially uses the basic motion control methods: direct, inverse kinematics and procedural method. Personalized results can be accomplished with different hand motion pattern profiles created interactively. The resulting arm-hand motion can be further improved by other approaches such as keyframe, signal-processing-based methods and optimal control.

We extend the method to object manipulation and human interaction with hands. A novel method is proposed for BODY self-collision detection and correction. This method is based on the extension of a multi-sensor configuration and the use of the secondary task of inverse kinematics. It is very efficient for self-collision detection and can directly result in the movement that is free of self collisions. We finally model the same multi-sensor structure on the 3-D hand model of the CyberGlove provided by Virtual Technologies, Ltd. Using a grasping automata-based method and hand posture collision response, we simulate more precisely the interactive grasping by a real person wearing the CyberGlove. It can work along with synthetic actor automatic grasping.

. *Investigation of key frame sequence manipulation methods*

Sequence manipulation methods are important to produce a longer sequence from several smaller ones or to customize an existing sequence. We propose the following methods on keyframe sequence manipulation: motion blending on its frequency band and motion data compression using cubic splines. We also implement other manipulation techniques: interactive sequence editing on both spline curves and a multiresolution pyramid structure, interactive time warping, data compression from the pyramid structure. All these techniques are important for reusing and post-processing the keyframe sequence.

. *Development of an integrated motion control system called Track.*

An integrated motion control system is important for animation production and other applications. We have designed and implemented an integrated motion control software called Track. The human model can be of different complexities: from the lowest segment model to the highest realistic deformable skin model. First, it integrates the common motion control methods: direct/inverse kinematics, direct/inverse dynamics and high level motion control: walking, grasping, and motion capture with 3-D sensors. Second, it provides a complete set of techniques for sequence manipulation. Third, the data structure and motion control function are

implemented generally for different types of articulated figures. Fourth, the collision detection including self-collision detection, and collision response are investigated in the system. Fifth, it supports multiple articulated figures and their interaction with complex environments. Finally, as the rendering is not a defined topic in the project, popular renderer file formats are supported.

1.3 Organization of the thesis

Let us outline the organization of the dissertation:

In Chapter 2, we make a brief review of the current research in articulated figure motion control. We concentrate on the methods and techniques we use in our work.

In Chapter 3, we describe the use of inverse dynamics in closed loop with forward dynamics for interactive motion control of a human skeleton.

In Chapter 4, we describe the methods on hand motion control: a multi-sensor heuristic for grasping, object manipulation, human interaction by hands, self-collision detection and response, and interactive grasping with the CyberGlove.

In Chapter 5, we propose the method of motion blending on its frequency band, and the data compression method by directly using the cubic splines. We implement other sequence manipulation methods based on the multiresolution filtering and cubic splines.

In Chapter 6, we present the integrated motion control system Track. We show the integration of different motion control methods and sequence manipulation techniques. We propose a method based on 3-D interaction techniques and motion capturing by using VR devices that provide a new functionality for Track. Finally, we briefly mention some of its applications.

2. Review of General Methods

In this chapter, we briefly present the current research in motion control in computer animation, and then survey the general motion control methods.

2.1 Background

Motion control is an essential part of computer animation. We do not give an exhaustive review but a brief mention of the works inside computer graphics that are most relevant to our work. Motion control includes various approaches from traditional rotoscopy [Maiocchi and Pernici, 90], parametric keyframe [Parke 82, Reynolds 82, Shelley and Greenberg 82, Gomez 84, Sturman 84, Steketee and Badler, 85, Fortin et al., 86, Calvert et al., 91], procedural modeling [Boulic 91], direct and inverse kinematics [Phillips 91, Boulic 92, Bezault 92], inverse kinetics [Boulic and Mas 94] to physics-based methods like direct and inverse dynamics [Wilhelms 85, Armstrong 86], spacetime optimization [Witkin 88, Cohen 92, Liu 94], dynamic control [Hégron et al., 96] and recent motion capturing with multiple sensors that is very popular in commercial production [Molet et al., 96, Badler et al., 93a, Ascension 94]. The reuse of the captured motion is a very active topics [Witkin and Popovic 95, Bruderlin and Williams 95].

Parametric keyframe animation, considered as a direct kinematics method, is the most common method in all animation systems. It is called parametric in order to distinguish from the 2-D keyframe techniques used in the cartoon industry. The motion parameters could be 3-D, e.g., the joint value or value of degree of freedom (DOF) for an articulated figure. It gives the animator the direct control on the motion parameters of the model without considering the physical properties of the model. The animator can define the key postures on a time base, with in-between frames being generated automatically by spline interpolation. The advantage lies in its ease of use and understanding. The disadvantage comes when handling a complex articulated figure with hundreds of DOFs whose values the animator has to specify. Practically, it is almost impossible to conform to physically realistic animation.

Inverse kinematics is another common method which differs from keyframe on its working space: inverse kinematics is carried out in Cartesian Space while keyframe in Configuration Space, or Joint Variation Space. For an articulated figure, the number of DOF is 6 in the Cartesian Space for each branch's ending point (or end effector), being smaller than the DOF in its Joint Space. So the inverse kinematics allows the user to specify much fewer values for a key frame. The formula of inverse kinematics can calculate the DOF value in the Joint Space for the end effector to achieve the goal using the minimum norm solution. While the other problem with this solution is the singularity for some configurations, in most cases, it works well.

The procedural method is very efficient for some types of motion, e.g., walking and running. The animator only needs to specify a small set of parameters, e.g., the velocity, step size for walking, and a specific procedure can calculate the posture for each time step. The difficulty of this method is for a general movement, it is hard to define the procedure. Only limited types of motion are practical for this method.

All the methods described above do not use force and torque directly for motion control. They belong to the same category: kinematics. Another category is dynamics. Dynamic methods incorporate physical laws and the resulting motion is physically realistic. There are many impressive results such as multi-object collisions, swinging chains, snake creeping, etc. The method allows the animator to define physical parameters of the model, e.g., the mass, the inertia, the constraint, the initial velocity, and finally the forces and torques on it, and not to bother about values of DOFs. The movement is automatically obtained from solving the equations. An important issue that arises for physics-based models is how to control them. Mathematically, a physics-based method is posed as an initial-value problem: the user has little control other than setting up the initial configuration.

The spacetime optimization is a technique for exercising better control on physics-based method. It formulates the animation problem as a constrained multipoint boundary-value problem, thus allowing the user to specify intermediate and final configurations as well as the initial. But this method leads to a non-linear constrained variation problem that typically does not have a closed-form solution. In practice, the solution is found by reducing the space of possible trajectories to those representable by a linear combination of basis functions, e.g., cubic B-splines or wavelets. Unfortunately, there are no efficient numerical methods available to apply to such nonlinear constrained optimization problems.

Motion capture by using multiple 3-D sensors is becoming more and more popular today as the new hardware devices are flooding the market. The applications can be classified in two categories: animation design and real-time motion generation. For the animation design, postures or motion sequences can be captured with sensors. They can later be reused and combined with other sequences produced from other motion generators or edited with standard editing tools. The techniques from signal processing are currently used in order to reuse the captured motion to meet the new geometric or time constraints. The generation of human movements for real-time applications is either based on the combination of pre-recorded sequences or directly controlled by live performance. The problem of motion capture is the sensor device limit: the existence of another magnetic field disturbs the measurement of sensors using magnetism. The distance limit and sampling rate is another problem.

All these methods are very important and can not replace each other. How to use them for human figure animation is a big challenge. To integrate them smoothly in one system is another challenge.

2.2 Direct kinematics: keyframe

Keyframe animation consists of the automatic generation of intermediate frames, called in-betweens, based on a set of keyframes supplied by the animator. There are two fundamental approaches to keyframe:

1. The in-betweens are obtained by shape interpolation. This technique is introduced by Burtnyk and Wein [Burtnyk and Wein, 71]. It plays a major role in film production and has been improved continuously by using more and more sophisticated mathematical tools [Catmull 78, Reeves 81]. There is a serious problem for the image-based keyframe: the motion can be distorted due to the interpolation.

2. A way of producing better images is to interpolate parameters of the model of the object itself. This technique is called parametric keyframe animation by Parke [Parke 82] and Steketee and Badler [Steketee and Badler, 85] and key-transformation animation by Zeltzer [Zeltzer, 82]. The parameters are normally spatial parameters, physical parameters and visualization parameters that decide the models' behavior.

Parametric keyframe animation is considered as a direct kinematics method in motion control when the interpolated parameters are defined in Joint Space of the articulated figure. Efficient and numerically well behaving methods exist for the transformation of position and velocity from Joint Space to Cartesian Space, see Figure 2-1.

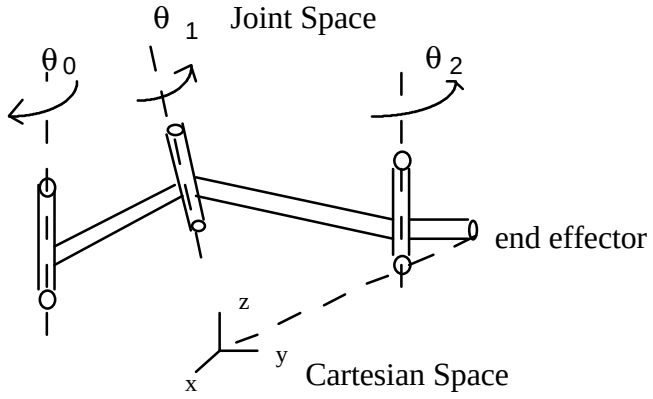


Figure 2-1 simple articulated figure

Spline curves are used for keyframe interpolation. The term spline comes from a familiar drafting tool used in several industries. It is a thin elastic lath used to draw a smooth curve through a set of given points as in interpolation. Splines can be described mathematically as piecewise approximations of cubic polynomial functions. For animation, the most interesting splines are cardinal splines, Catmull-Rom splines and Kochanek-Bartel splines [Kochanek and Bartel, 84]. More details of the techniques are described in [Magnenat-Thalmann and Thalmann, 90, Foley et al., 90]. We give more implementation details and examples in chapter 5.

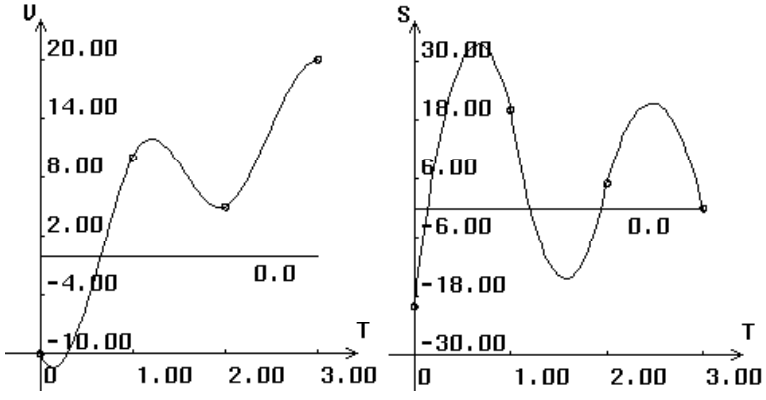


Figure 2-2 cubic Hermite splines to interpolate one degree of freedom (DOF) joint value and its first derivative

2.3 Inverse kinematics

Inverse kinematics is the opposite of the direct kinematics, see Figure 2-3. Its problem is the determination of the joint variables given the position and the orientation of the end of the manipulator, or end effector (Figure 2-1), with respect to the reference coordinate system.

Inverse kinematics is an important subject in Robotics. It can be solved by various methods, such as inverse transform [Paul et al., 81], screw algebra [Kohli and Soni, 75], dual matrices [Denavit, 56], dual quaternian [Yang and Freudenstein, 64], iterative [Uicker et al., 64] [Milenkovic and Huang, 83], and geometric approaches [Lee and Ziegler, 84]. More details of each method can also be found in [Fu et al., 87].

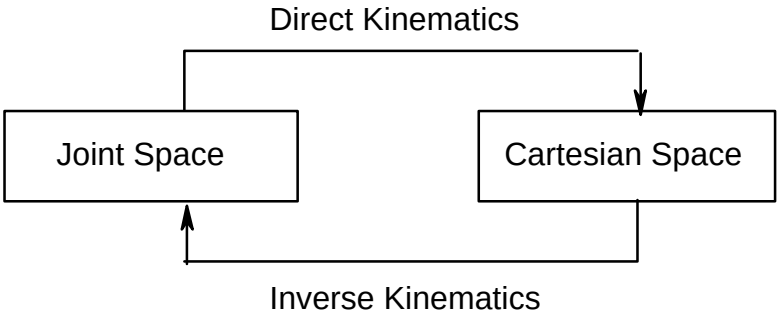


Figure 2-3 direct and inverse kinematics

An end effector location depends on the current state of the joint parameters. The set of non-linear equations establishing the end effector location as a function of the joint state is called the direct geometric model in Robotics. Inverting it is possible if the dimensions of Joint Space and Cartesian Space are the same. However, a general articulated structure may contain more DOF in Joint Space which are highly redundant in accomplishing tasks. The inversion is not always possible. The solution is the first order approximation of the system: to linearize the direct geometric model, as shown in Figure 2-4.

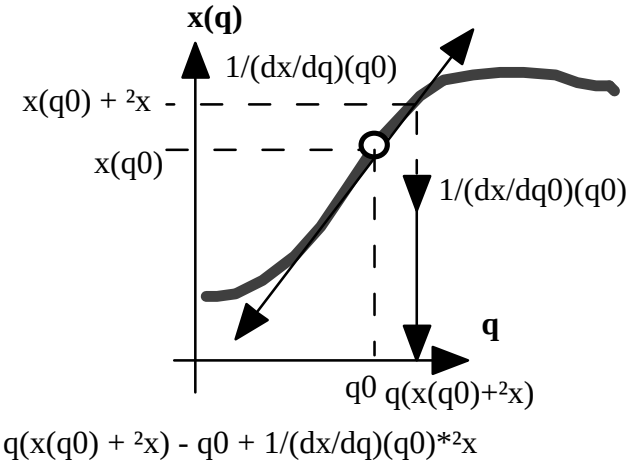


Figure 2-4 the linearization of the direct geometric model

As a consequence of the linearization, the solution's validity of inverse kinematics is limited to the neighborhood of the current state and, as such, any desired motion has to comply with the hypothesis of small movements.

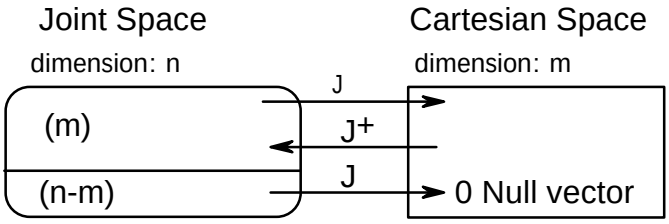


Figure 2-5 illustration of the main task and secondary task

The position and orientation vector of the end effector in Cartesian space is called main task (or behavior). If its dimension m (usually six: three rotations and three translations) is less than the dimension n of the Joint space, the (n-m) vectors in Joint space are projected to the null vector in Cartesian space by the linear transformation J. They do not modify the achievement of the main task. The (n-m) vector in Cartesian space is called secondary task (or behavior), see Figure 2-5.

The discrete form of the general solution provided by inverse kinematics is:

$$\Delta \mathbf{q} = \mathbf{J}^{+}\Delta \mathbf{x} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\Delta \mathbf{z} \qquad \textbf{Eq. 2-1}$$

where

$\Delta \mathbf{q}$ is the unknown vector in the joint variation space, of dimension **n**.

$\Delta \mathbf{x}$ describes the *main task* as a variation of the end effector position and orientation in Cartesian space. For example in figures 2-6 a, b and d the main task assigned to the end of the chain is to follow a curve or a line in the plane under the small movements hypothesis. The dimension **m** of the main task is usually less than or equal to the dimension **n** of the joint space.

J is the Jacobian matrix of the linear transformation, representing the differential behavior of the controlled system over the dimensions specified by the *main task*.

$\mathbf{J}^{+}$ is the unique pseudo-inverse of **J** providing the minimum norm solution which realizes the *main task* (Figure 2-5 a and b).

I is the identity matrix of the joint variation space (**n** x **n**)

$\mathbf{(I-J^{+}J)}$ is a projection operator on the *null space* of the linear transformation **J**. Any element belonging to this joint variation sub-space is mapped by **J** into the null vector in the Cartesian variation space.

$\Delta \mathbf{z}$ describes a *secondary task* in the joint variation space. This task is partially realized via the projection on the *null space*. In other words, the second part the of equation does not modify the achievement of the main task

for any value of $\Delta\mathbf{z}$. Figure 2-6 c has a null vector as its main task, so the displayed motion is always realized in the *null space*.

Usually $\Delta\mathbf{z}$ is calculated so as to minimize a cost function. If the main task belongs to the image space of $\mathbf{J}$ then the null space is (**n-m**) dimensional in the joint variation space. This information is fundamental to evaluate the potentiality of the secondary task and clearly leads to a tradeoff between realization of main and secondary tasks.

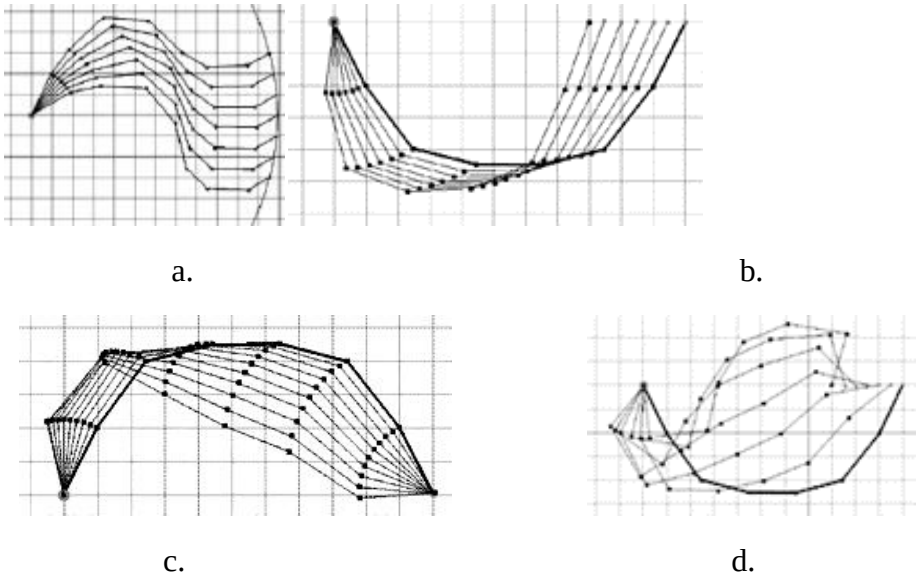


Figure 2-6 (a) and (b) *main task* only (minimum norm solution) (c) the null space is illustrated through a null vector *main task* (d) same *main task* as 2b with arbitrary *secondary task*

Here are the steps of using inverse kinematics:

1) Defining the Main Task;

2) Evaluating the potential of the Secondary Task

3) Constructing the Jacobian Matrix

4) Invert the Jacobian Matrix

5) Handling the Joint Limits

6) Managing Singularities

The so-called Jacobian of the system is the matrix gathering the first-order variations. It is inverted [Whitney, 69] [Liégeois, 77] in order to obtain the joint variation realizing a desired variation of the end effector. For a more detailed explanation of these steps, refer to [Boulic and Mas, 96]. We implement inverse kinematics as one of the motion generators in the Track software.

2.4 Forward dynamics

For the direct and inverse kinematics, the solution is to find the position and orientation of the structure without regard to the forces and torques that cause the motion. On the other hand, forward dynamics takes forces and torques directly into the mechanics equation to calculate the spatial parameters of the articulated figure:

position, velocity and acceleration, and uses them to update the structure at each time step (see Figure 2-7). The forward dynamics belongs to physics-based modeling which is extensively used in computer graphics [Barr, 87]. Using forward dynamics, the obtained motion is realistic and conforms to the laws of physics.

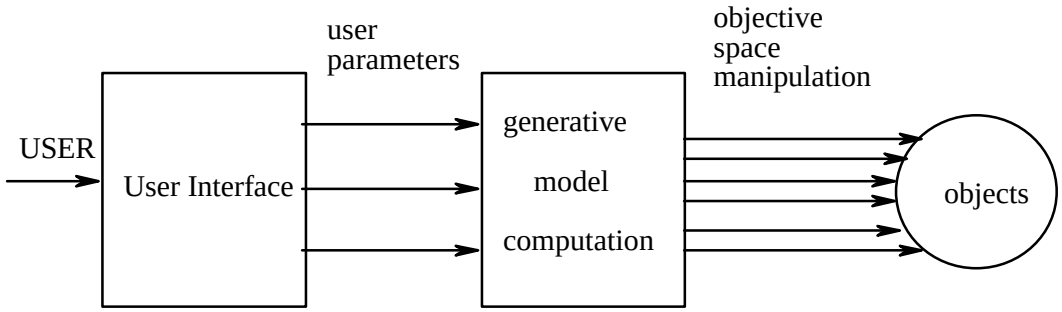


Figure 2-7 Forward dynamics animation

The common mechanic equation is Lagrange's equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = f_i \quad i = 1, 2, ..., n \qquad \textbf{Eq. 2-2}$$

where

$L = Lagrangian = T - V$,

T = kinetic energy and V = potential energy

q_i = generalized joint coordinate representing the internal coordinate of the i-th joint

$\dot{q}_i$ = the first derivative of q_i

f_i = generalized force applied to the i-th link; this generalized force can be thought of as the force (for sliding joints) or torque (for rotating joints) active at the joint of the i-th link.

The direct application of the Lagrange's equation, together with Denavit-Hartenberg link coordinate representation, results in a convenient and compact algorithmic description of the manipulator equations of motion. The algorithm is expressed by matrix operations and facilitates both analysis and computer implementation. For more details of this algorithm see [Fu et al., 87], and for its application in motion control of articulated figures see [Bruderlin and Calvert 89, Kunii and Sun, 90]. The problem is its computational inefficiency due to the use of 4 x 4 homogeneous transformation matrices.

To derive more efficient equations of motion, several investigators turned to Newton's second law and developed various forms of Newton-Euler equations [Armstrong, 79, Orin et al., 79, Luh et al., 80, Walker and Orin, 82]. This formulation, when applied to a robot arm results in a set of forward and backward recursive equations. The Newton-Euler equations:

$$\vec{F} = m_i \ddot{\vec{r}}_i \qquad \textbf{Eq. 2-3}$$

$$\vec{N} = J_i \dot{\vec{w}}_i + \vec{w}_i \times (J_i \vec{w}_i) \qquad i = 1, 2, ..., n \qquad \textbf{Eq. 2-4}$$

where

$\vec{F}$ = total external force exerted on link i at the center of mass

m_i = total mass of link I

$\vec{r}_i$ = position of the center of mass of link i from the origin of the base reference frame

$\ddot{\vec{r}}_i$ = the second derivative of $\vec{r}_i$

$\vec{N}$ = total external moment exerted on link i at the center of mass

J_i = inertia matrix of link i about its center of mass with reference to the base coordinate system

w_i = the angular velocity of the coordinate system of the link i with respect to the base coordinate system

The most significant aspect of this formulation is that computation time of the applied torques can be reduced significantly to allow real-time control [Armstrong and Green, 85].

2.5 Dynamic control

An important issue that arises in forward dynamics is how to control the model. Mathematically, forward dynamics translates into differential equations, which are typically posed as an initial-value problem; the user has little control other than setting up the initial configuration. This is exactly opposite to keyframes where the animator has the full control. Control of physics-based models remains an open research issue. Figure 2-8 and 2-9 show the two general types of control from a mathematical point of view.

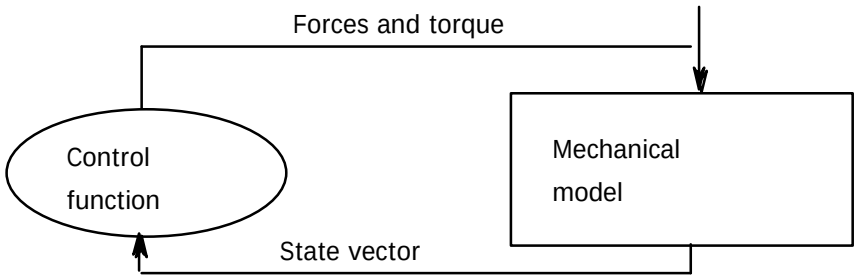


Figure 2-8 loosely coupled control

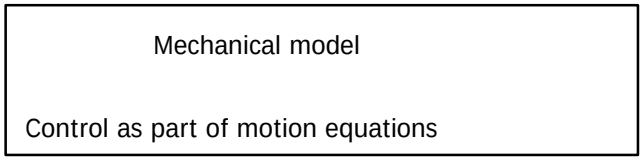


Figure 2-9 tightly coupled control

A common and effective approach is to use constraint-based techniques, which allow users to specify required values for various properties of the models. One typical achievement was proposed in [Isaacs and Cohen, 87]. The most interesting result is that the kinematic constraints permit traditional keyframe animation systems to be embedded within a dynamic analysis. Joint limit constraints are also handled correctly through kinematic constraints. A kinematic constraint consists of an explicit specification for the acceleration of some DOFs during the current time increment, thus removing an unknown degree of freedom from the system. Within the context of mathematical formulation, a kinematic constraint consists of removing a row and column from the system of equations. For more details of how the kinematic constraints are used see [Isaacs and Cohen 87, pp. 219].

Another well known work is the dynamic simulation and control scheme illustrated with a 3-D vehicle and a hopping one-legged robot [Hégron et al., 96]. The forward dynamics based on Lagrange's formalism is developed for the complex articulated rigid objects representing the vehicle and robot. The result is compared to real measurements. Two classes of techniques are proposed. The first class is based on Proportional Integral Derivative control by constraints. It is an on-line control technique that allows the animator to interact with the animation evolution. The second class is based on optimal control where animator interaction is impossible.

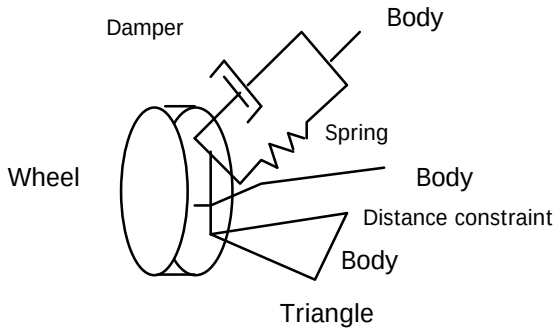


Figure 2-10 part of the vehicle model with rigid objects linked by mechanic joints

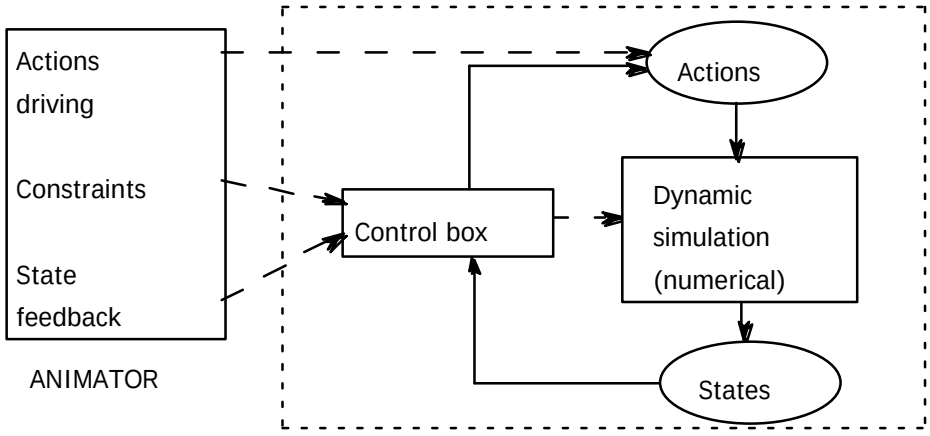


Figure 2-11 on-line simulation

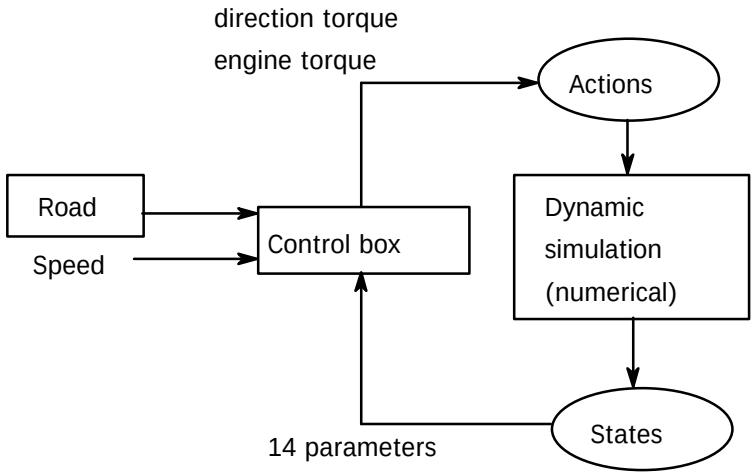


Figure 2-12 off-line control scheme for the vehicle

We adapt the Armstrong-Green algorithm to our articulated human figure with closed loop of inverse dynamic control [Huang et al., 94]. We describe it in chapter 3.

2.6 Procedural method

Procedural methods rely on the computer's ability to determine the kinematics based on implicit instructions rather than explicit positions. One class of procedural methods is "inverse kinematics" described above where the motion of end effector is specified by the animator, but the motion of interior links is computed algorithmically. Another typical work is walking [Boulic et al., 90] where a walking model is built from experimental data based on a wide range of normalized velocities. All spatial values of the model are normalized by the fundamental characteristic of the walk: the height of the thigh H t. It is the length between the flexing axis of the thigh and the foot sole whose average value is 53% of the total height of the human being [Kreighbaum and Barthels, 85].

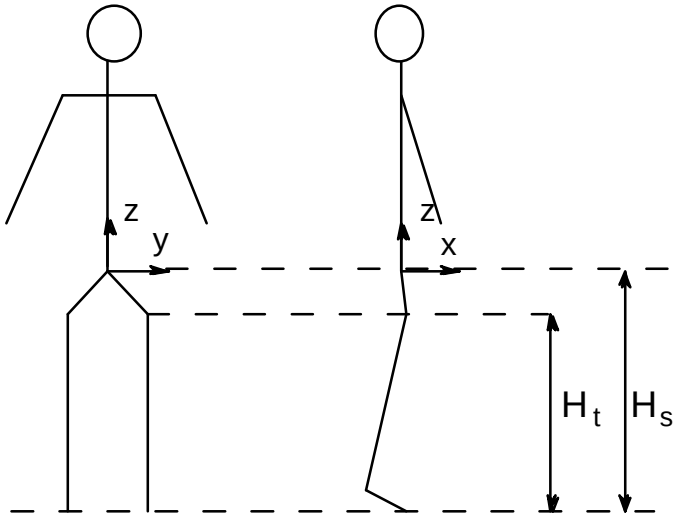


Figure 2-13 definition of the body coordinate system and initial position

The algorithm is based on the Spatial characteristics that are defined by the normalization formula of Inman [Inman et al., 81]:

$$RLc = 1.346\sqrt{RV} \qquad \textbf{Eq. 2-5}$$

where RLc is the relative length of cycle. RV is the relative velocity of walking that is defined as the average walking velocity normalized by H t.

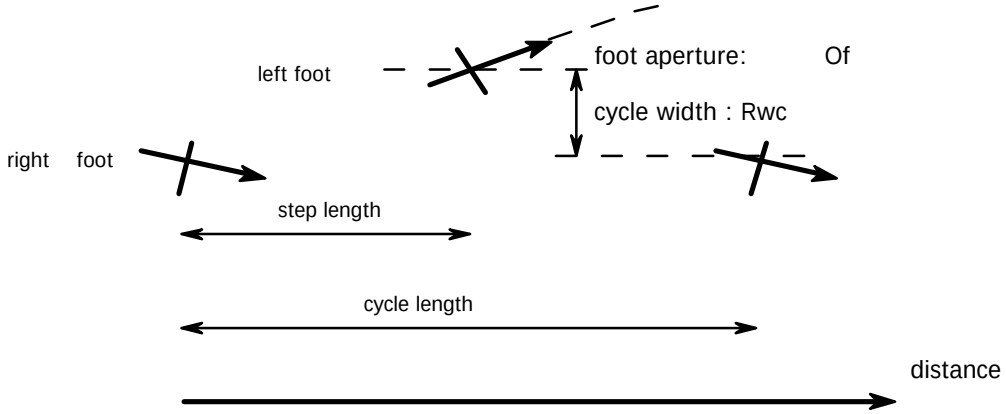


Figure 2-14 spatial structure of the walking cycle

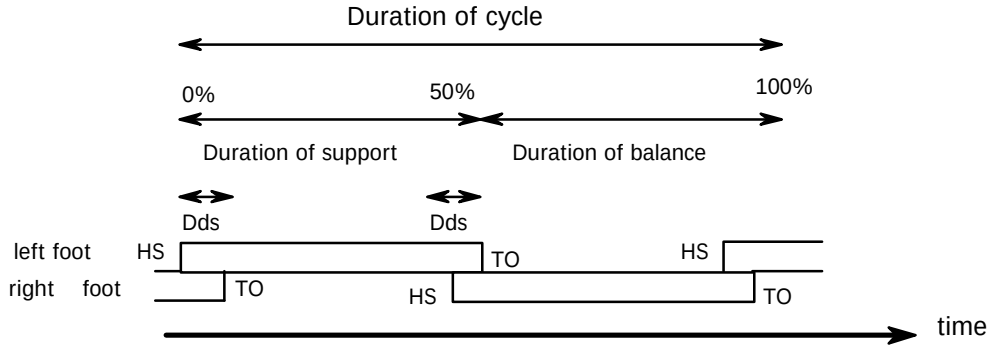


Figure 2-15 temporal structure of the walking cycle

The temporal characteristics are defined as follows:

$$Ds = 0.752Dc - 0.143$$

$$Db = 0.248Dc + 0.143$$

$$Dds = 0.252Dc - 0.143$$

Eq. 2-6

where Ds is the Duration of support, Dc the Duration of cycle, Db the Duration of balance and Dds the Duration of double support.

The animator only needs to define the general parameters, relative velocity, personification parameters or path curve. The walking sequence is algorithmically computed.

The advantage of the procedural method is that it allows animator to control some types of specialized motion easily by specifying a few high level parameters, e.g., velocity of walking. The procedure can automatically control each degree of freedom. It is also realistic because of the use of the empirical equations. We propose the multi-sensor-based grasping method that uses the procedural method to choose the grasping frame according to the object geometry [Huang et al., 95].

2.7 Motion capture

Motion capture is not a new motion control method compared to rotoscopy [Maiocchi and Pernici, 90]. It is the availability of the new VR devices that can more easily capture position and orientation in real time, e.g., Flock of Birds of Ascension Technology Ltd. and the CyberGlove of Virtual Technologies Ltd. (Figure 2-16). The Flock of Birds contains a Transmitter and many Receivers. The transmitter creates a magnetic field which is detectable by the surrounding environment. At the same time, the receivers, or sensors, receive the magnetic signal and send back to its electronic unit to calculate the receiver's position and orientation. Similarly, the glove is used to measure the flexion of the hand and relative positions of the fingers and inputs these data to the computer.

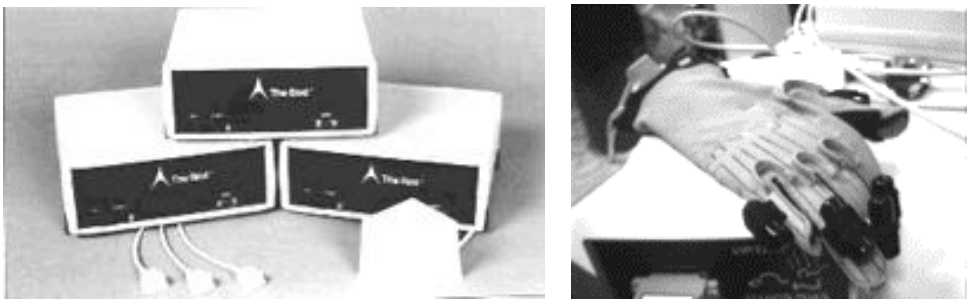


Figure 2-16 Flock of Birds of Ascension Technology Ltd., CyberGlove of Virtual Technologies Ltd.

[Badler et al., 93a] try to use a minimal number of 6-D sensors to capture full body standing postures. Four sensors are used to create a good approximation of a human operator's position and posture, and map it on to the articulated computer graphics of a human model. Other joints are positioned by a fast inverse kinematics algorithm. The goal is to realistically recreate human postures while minimally encumbering the operator with sensor attachments. However, difference between the postures of the performer attached with six sensors and that of the resulting synthetic is clearly noticed.

[Molet et al., 96] proposed an implementation of a real-time anatomical converter. The Anatomical Converter is based on a very efficient method of capturing human motion after a simple calibration. The sensor data are converted into the anatomical rotations of a hierarchical representation of a body. Such a choice facilitates a wider use of the motion for other human models with the same proportions. We present our animation design method based on this converter and the gesture recognition from CyberGlove input in Chapter 6.

2.8 Conclusion

We have briefly described the current research of motion control of articulated figures in computer animation that directly relates to our work. In human animation, there are also other types of objects besides articulated figures, e.g., facial animation [Kalra and Magnenat-Thalmann, 94], skin deformation [Thalmann et al., 96, Shen and Thalmann, 95], clothes animation [Volino et al., 96, Carignan et al., 92, Lafleur et al., 91]. These are secondary motion control techniques resulting from the movement of the articulated figure. Though similar techniques such as dynamics can be applied for this kind of animation, the techniques are different from what we describe in this chapter,

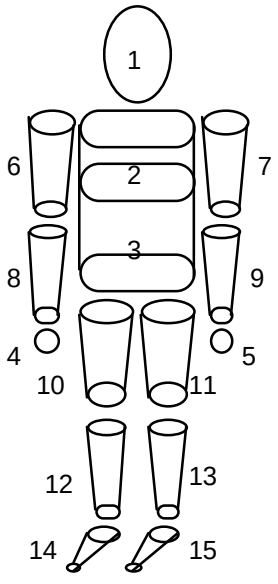
In the next chapter, we describe our work in this general order: background, presentation of the method, implementation, results and conclusions.

3. Close loop Dynamic Motion Control for an Human Articulated Figure

In this chapter, we propose a method of close loop forward/inverse dynamic control on our articulated figure. We first briefly describe our data structure and a model of human body approximated with simple geometric objects. The physical parameters of each part of the body are derived from the experimental results of biomechanics. As the background has been presented in the previous chapter (see section 2.4 and 2.5), we directly start by presenting our work.

3.1 Volume approximation

To efficiently use the dynamic equation for a body composed of multiple objects, we approximate it with 15 simple rigid geometric solids, cylinders, spherical ellipsoids, and truncated cones. It avoids the complexity of calculation on the general free-form surface body volume, but can give good results if we use proper geometry and assign correct physical parameters for the solid. The mass of each volume is derived from biomechanical experiments [Seireg and Arvikar, 89]. In Figure 3.1 (a), a skeleton is associated with solids, and the mass of each body part is listed in Figure 3.1 (b). They are linked by joints in a hierarchical structure.



(a) skeleton with solids

Name	Type	Mass (Segment / Total)
head and neck	ellipsoid cylinder	0.081
up_torso	cylinder	0.216
down_torso	cylinder	0.281
r_hand l_hand	sphere	0.006
r_upper_arm l_upper_arm	frustum	0.028
r_forearm l_forearm	frustum	0.016
r_thigh l_thigh	frustum	0.100
r_leg l_leg	frustum	0.0465
r_foot l_foot	frustum	0.0145

(b) the mass proportion of each body part

Figure 3-1 representation of body volume

3.2 Solid object attachment and mobility

Our skeleton is constructed by N3D [Boulic et al., 95a]. It is a general data structure to retain the geometric, topological and display information sufficient to set a frame in a 3-D space. We describe it in more detail later. One of the advantages of using N3D is that an additional functionality can be associated to a N3D through the binding

typed structures. The most common data in N3D for a human skeleton is Joint. For each Joint-N3D there is one translation or rotational DOF. The internal joint of our model can only rotate, thus comprising one to three Joint-N3Ds. The solid adds a new N3D type: Solid-N3D. It is attached to the last Joint-N3D as shown in Figure 3-2.

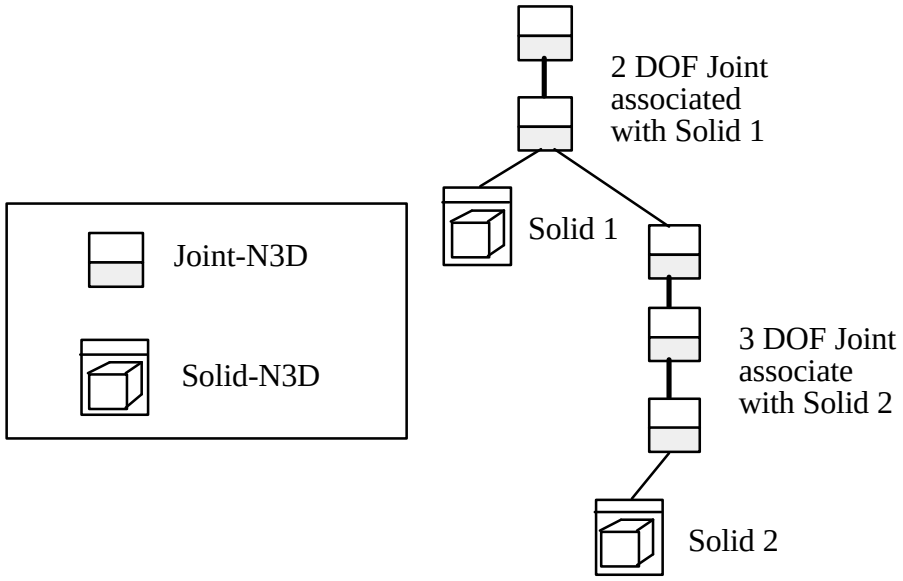


Figure 3-2 associating solids with joints

The position and orientation of each solid can not be changed by Solid-N3D itself because there is no DOF for the Solid-N3D. The mobility is implemented by the series of attached Joint-N3Ds. Now we discuss how to express an instantaneous rotation vector of Solid-N3D by the incremental value of each associate Joint-N3D. We first examine closely the Joint-N3D series. It is an Euler sequence as shown in Figure 3-3.

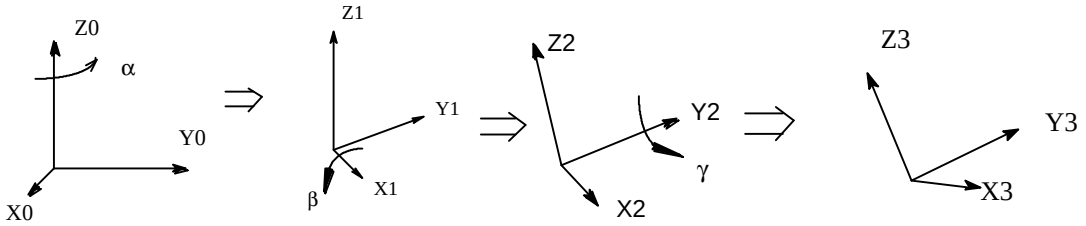


Figure 3-3 Euler sequence: (α, β, γ)

If the rotation always starts around Z-axis, there are two sequences, Z-Y-X and Z-X-Y, which result differently. In Figure 3-3, it is Z-X-Y. Now we show how to calculate a rotating vector, e.g., an orientation vector resulting from a dynamic equation, from this Euler sequence. Let the instantaneous rotation vector that is defined in coordinate system 0 be $(w_x, w_y, w_z)_0$:

$$(w_x, w_y, w_z)_0 = \Delta\alpha Z_0 + \Delta\beta X_1 + \Delta\gamma Y_2 \qquad \textbf{Eq. 3-1}$$

where Z_0, X_1, Y_2 are three axes shown in Figure 3-3 with $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$ are the variation of Euler sequence. Next step is to express X_1 and Y_2 in coordinate system 0, which is the same as the left side of equation:

$$X_1 = X_0 \cos(\alpha) + Y_0 \sin(\alpha) \quad \textbf{Eq. 3-2}$$

where α is the rotation around Z_0.

and

$$Y_2 = Y_1 \cos(\beta) + Z_1 \sin(\beta) = (-X_0 \sin(\alpha) + Y_0 \cos(\alpha))\cos(\beta) + Z_0 \sin(\beta) \quad \textbf{Eq. 3-3}$$

where β is the rotation around Z_1.

if we replace X_1 and Y_2 in Eq. 3-1 with 3-2 and 3-3 and write the result in matrix form:

$$(w_x, w_y, w_z)_0 = (\Delta\alpha, \Delta\beta, \Delta\gamma) \begin{bmatrix} 0 & 0 & 1 \\ \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha)\cos(\beta) & \cos(\alpha)\cos(\beta) & \sin(\beta) \end{bmatrix} \quad \textbf{Eq. 3-4}$$

Given the $(w_x, w_y, w_z)_0$, the $(\Delta\alpha, \Delta\beta, \Delta\gamma)$ can be obtained from solving Eq. 3-4. The new Euler sequence can be calculated:

$$(\alpha, \beta, \gamma)_{new} = (\alpha, \beta, \gamma) + (\Delta\alpha, \Delta\beta, \Delta\gamma) \quad \textbf{Eq. 3-5}$$

3.3 Forward dynamics equations

The strategy of forward dynamics is to calculate the kinematics parameters of a system from the input force and torque on the previous kinematics status. It results in the realistic movement because it is based on physics. One of the most popular algorithms in computer graphics literature for articulated figure animation is Armstrong-Green algorithm [Armstrong and Green, 85]. It is based on Newton-Euler formulations described in Section 2.4.

For the full list of equations for articulated figure, see Appendix A.

3.4 Forward dynamics solution

To get a simplified form by avoiding the inversion of matrices larger than three by three, two hypotheses are made in Armstrong-Green algorithm [Armstrong and Green, 86]:

(1) for the acceleration of a link and the amount of angular acceleration it undergoes:

$$\dot{\omega}^r = K^r a^r + d^r \quad \textbf{Eq. 3-6}$$

where $\dot{\omega}^r$ is the angular acceleration and a^r is the linear acceleration at the hinge of link r. K^r and d^r and are the coefficients to be calculated.

(2) for the acceleration and the reactive force on the parent.

$$f^r = M^r a^r + f'^r \quad \textbf{Eq. 3-7}$$

where f^r is the force that link r exerts on its parent link at the hinge and a^r is the same as in Eq. 3-6. M^r and f'^r are coefficients to be calculated.

The algorithm computes the linear system coefficients in two processes, inbound and outbound. The inbound process calculates some matrices and vectors from the geometric and physical structure of the system, and propagates force and torque along each link from leaves to the root of the hierarchy. The outbound process calculates kinematics quantities, the linear and angular acceleration of each link, then obtains the linear and angular velocities by numerical integration for updating the whole structure. It is a recursive algorithm along the whole time of dynamic simulation. The kinematic results of the former time step are used as the initial values for the next step calculation.

(1) Inbound process: from the leaves to the root, for each link, compute the linear system's coefficients K^r, M^r, d^r and f'^r in Eq. 3-6 and Eq. 3-7. For the detailed solution see Appendix A.

(2) Output process: from the root to the leaves, compute $\dot{\omega}^r$ and f^r by Eq. 3-6 and Eq. 3-7 for each link, perform a time integration and frame transformation to obtain angular velocity ω^r and linear velocity v^r to update the articulated figure.

3.5 Inverse dynamics

Forward dynamics can produce physically realistic movement for the articulated figure. However, it is difficult to specify the input force and torque required for a desired movement. The force and torque are constantly changing in the simulation process. The inverse dynamics problem is to find at each joint the force and torque that generate the desired motion of the structure. It is a fundamental problem of robotics to get the force and torque for each DOF motor that drives the robot arm to perform a defined task [Luh et al, 80]. Various forms of motion equations for robot arm are derived mainly from Lagrange and Newton-Euler formulations. The motion equations are equivalent to each other in the sense that they describe the dynamic behavior of the same physical robot manipulator. However, the structure of these equations may differ as they are obtained for various reasons and purposes. Among many formulations, we select a kind of recursive formulation based on Newton-Euler equations for their computational efficiency. It is described in [Fu et al., 87] [Luh et al, 80] and is successfully used in robotics control. The greatest advantage is that the computation time is linearly proportional to the number of joints of the robot arm and independent of the robot arm configuration.

To construct the formulations for the model of the human body described in part 2, we write a series of equations for each link, using constrained forces and torques to guarantee their connection. The details of this formulation for inverse dynamics are listed in the Appendix B. It contains two opposing processes. The forward recursion process starts from the inertial coordinate frame to the end-effector coordinate frame to forward propagate kinematics information. The backward recursion process propagates the forces and moments exerted on each link from the opposite direction.

Using inverse dynamics in a closed loop with forward dynamic simulation is the key for getting the desired motion. In Figure 3-4, we illustrate the control process. The desired motion is represented by joint variables q, $\dot{q}$ and $\ddot{q}$. Here we use the same notation as in robotics. The `time_step` is the period to use inverse dynamics. From experiments it is set to be about 10 times the period `small_time_step` in the Armstrong-Green algorithm.

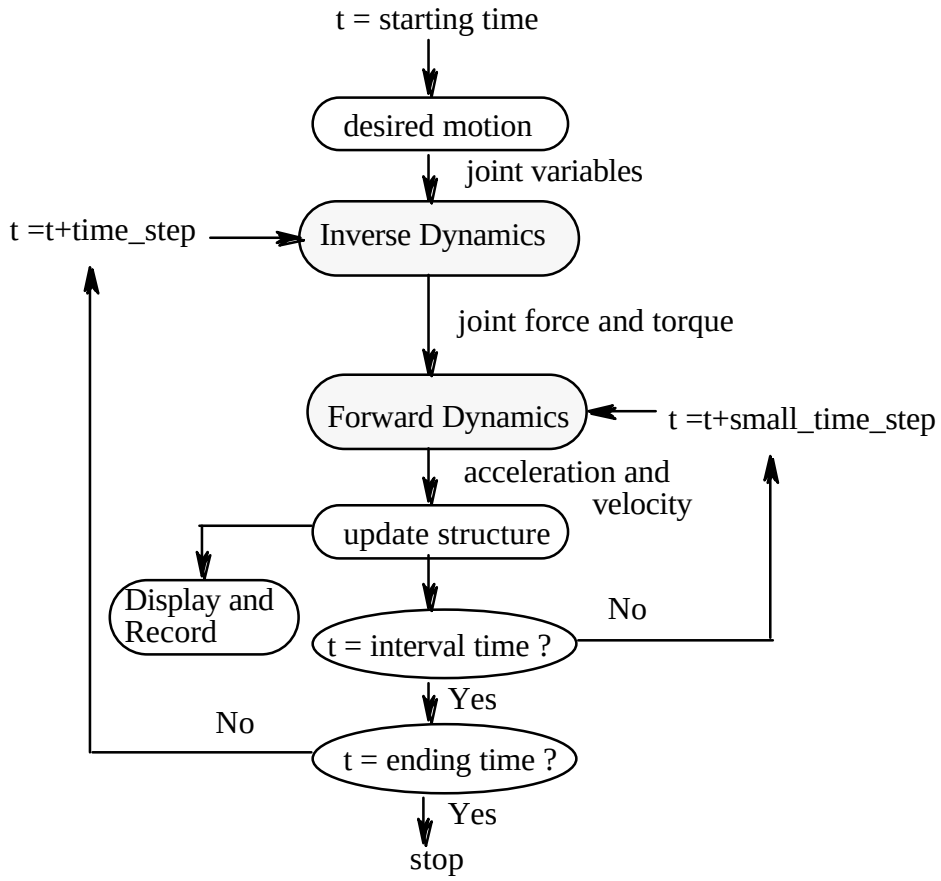


Figure 3-4 the close loop control with inverse dynamics

3.6 Results

The first example uses forward dynamics with supporting force, torque and gravity. The supporting force acts on the center of gravity of the whole body located at the abdomen. A torque is also acting on the abdomen that makes the body turning.

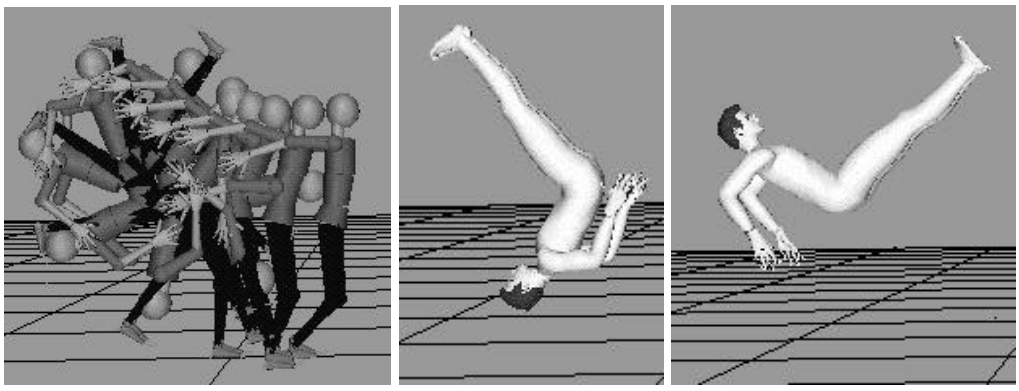


Figure 3-5 the jumping motion under torque, gravity and supporting force

In the second example, we simulate the human falling down using forward dynamics. A force field is defined for the floor. Collision detection is carried out for each body part and the resulting penetration distance is used to calculate the response forces.

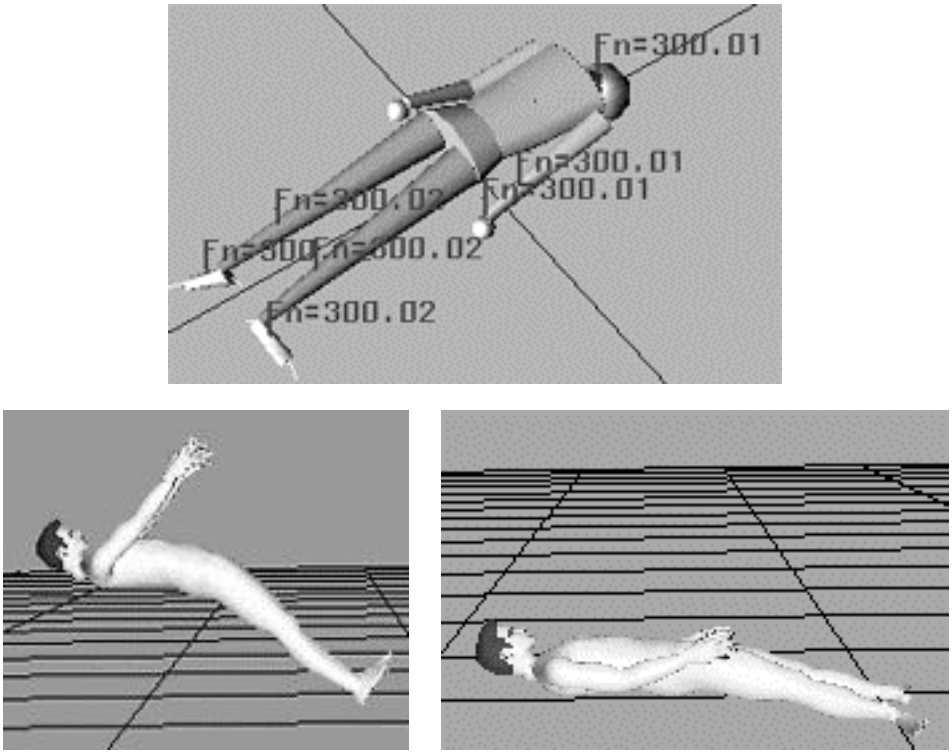


Figure 3-6 simulation of the human falling down

In the third example, we use proportional-derivative servo to control one actor to reach the posture of another actor. The resulting torque for each of the body part is displayed.

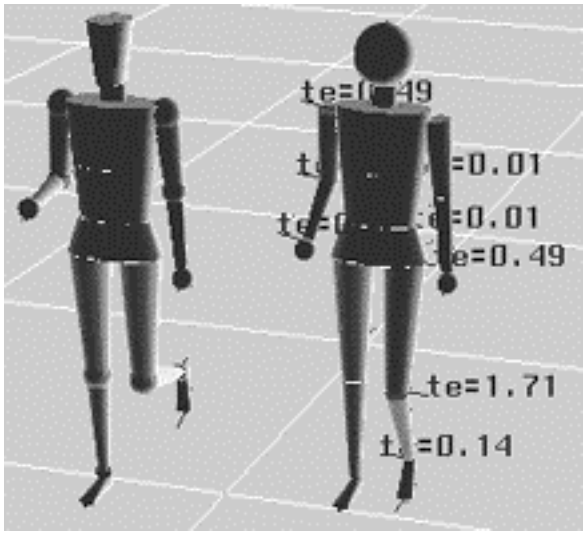


Figure 3-7 using proportional-derivative servo to control the actor at right side

In the fourth example, we use inverse dynamics to get the joint torque of the upper body with the root of the thorax to keep it balanced under gravity. Motion control is described in Figure 3-4. In order to show the value and direction of joint force and torque clearly, we do not display the solids representing the volume. `G` is the gravitational force of each solid measured in [Newton], and `ti` is the internal torque from inverse dynamics measured in [Newton] [Meter].

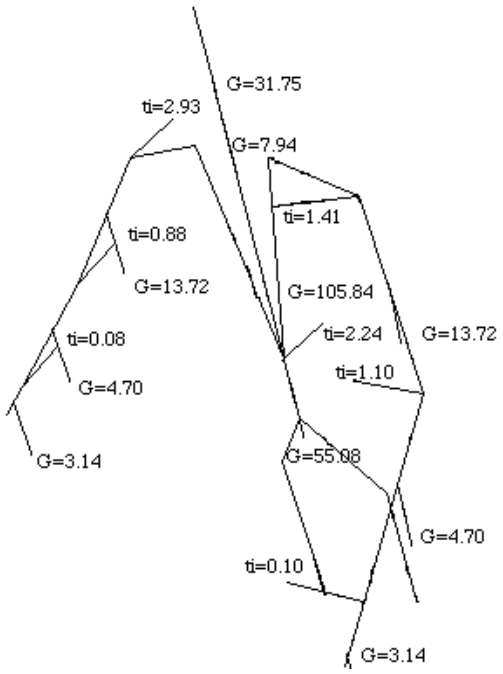
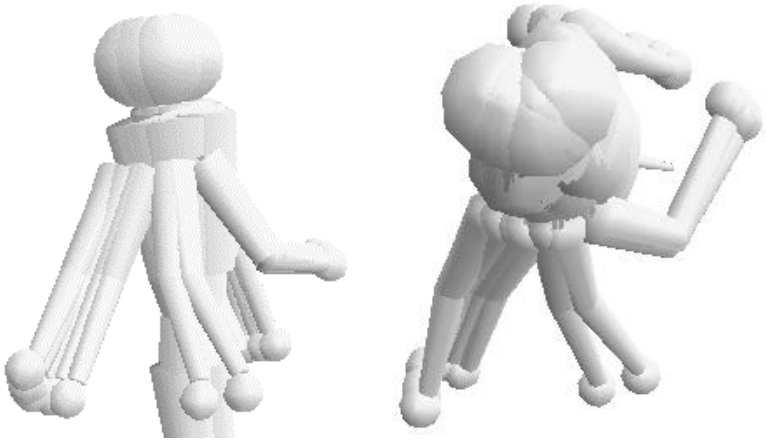


Figure 3-8 Keep upper body still in gravity field

We want to show a swing motion in the final example. It is similar to the motion in a forehand swing that hits a tennis ball. In Figure 3-8, we show a series of postures of the motion in two different views. The time step for simulation is 0.001 seconds, and the total time is 1.8 seconds.



(a) side view (b) top view

Figure 3-9 swing arm by dynamics with 0.04 second step

The joint force and torque is calculated from inverse dynamics to drive the motion. In Figure 3-8, the joint torque of shoulder is drawn with curves.

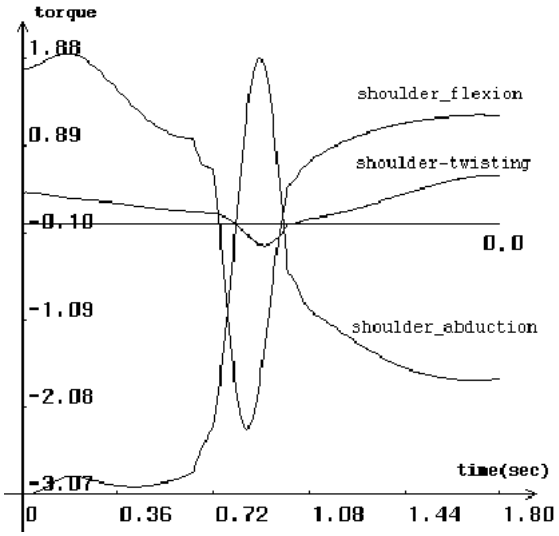


Figure 3-9 the joint torque from inverse dynamics

3.7 Conclusion

In this chapter we have proposed the use of inverse dynamics in a closed loop with forward dynamics for interactive motion control of a human skeleton. An efficient recursive algorithm based on Newton-Euler formulae is used to calculate the force and torque produced by joint actuators to fulfill a desired motion. The resulting force and torque are then used in forward dynamics to make the final motion with external force and torque. The Armstrong-Green algorithm is used for forward dynamic simulation. Inverse dynamic functions calculate the required force and torque at every small time interval in the process of forward dynamic simulation. In this way, it can correct errors at each time interval.

The future work can be continued in forward dynamics and dynamic control. Lagrange equations can be used for forward dynamics and the result can be compared with the Newton-Euler recursive formulation. The optimal control method can be used in dynamic control process. Besides close loop control, open loop control is also interesting because the animator can interactively control the dynamic simulation.

4. Multi-sensor Heuristic for Grasping

In this chapter, we propose a novel method for motion control: a multi-sensor-based approach of automatic grasping motion control for multiple synthetic actors. Despite the fact that it is described and implemented in our specific model, the method is general and can be applicable to other models. A heuristic is defined to decide the different grasping strategies from object geometry, hand geometry and observation of real grasping. Inverse kinematics can derive the final posture of the arms in order to bring the hands around the object. Multi-sensor object detection chooses the finger contact points on the object and determine their position and orientation.

4.1 Background

As we have not yet described the current research in human grasping, we describe some key contributions now. Several researchers in Computer Graphics worked on grasping simulations. [Magnenat-Thalmann et al., 88] proposed a semi-automatic way of grasping for a synthetic actor interacting with the environment. They also described hand deformations based on the concept of joint-dependent local deformation operators. [Rijpkema and Girard 91] used a knowledge-based approach suitable for simulating human grasping, implementing an expert system. [Kunii et al., 93] presented a model of hands and arms based on manifold mappings. They also considered dependencies between joints and produced some realistic animation of hand gestures. [Mas and Thalmann 94] presented the implementation of a hand control and automatic grasping system. [Koga et al., 94] applied manipulation planning to grasping. A new path planner is presented that automatically computes the collision-free trajectories for several cooperating arms to manipulate a movable object between two configurations. However, their method can not use the redundant degrees of freedom of the arms to avoid obstacles. It also does not consider the control of the finger movements.

Our work is based on this earlier work and extends it in the following aspects:

1) A generalization of the concept of multi-sensors for hand joints is proposed to allow synthetic actors to grasp common objects, share them and exchange them with each other.

2) An empirical set of grasping patterns is interactively specified by a 2-D interface or the CyberGlove to control the finger movement.

3) The multi-sensors are applicable to grasping using the VR device CyberGlove that provides a way for real humans to interact with synthetic actors.

4) The Secondary Task of Inverse Kinematics is used to solve the problem of self-collision of the body parts during grasping, so we use the redundant degrees of freedom of the arms to avoid self collisions.

5) A novel inverse method is proposed to simulate object manipulation and synthetic actor interaction by hands.

6) The multi-sensor grasping method is implemented on a very realistic free-form-surface hand model that is deformed in the grasping process by an extended geometric Free-Form Deformations (FFDs) tool based on a scattered data interpolant defined over Delaunay and Dirichlet / Voronoi diagram [Moccozet et al., 96]. We use the deformation to show the motion control results.

We focus on the above aspects in the following section but first, we briefly present our hand model to describe the grasping method.

4.2 Hand model

Our skeletal model of the hand is a part of our complete human skeleton structure with rigid articulated figures connected by one to three DOFs at each joint. Special care should be given to the mobility of the hand model, because of the great complexity of the wrist-palm region on which the fingers articulate themselves. First, at the level of the metacarpus, a small flexing mobility is provided to model the deformations of the palm. Then a standard structure is used for the four following fingers: pinky, ring, middle and index. It is organized as:

- the first flexing DOF for closing the first knuckle;

- a pivoting rotation for the lateral mobility of the finger;

- the two final flexing DOFs for closing the corresponding knuckles.

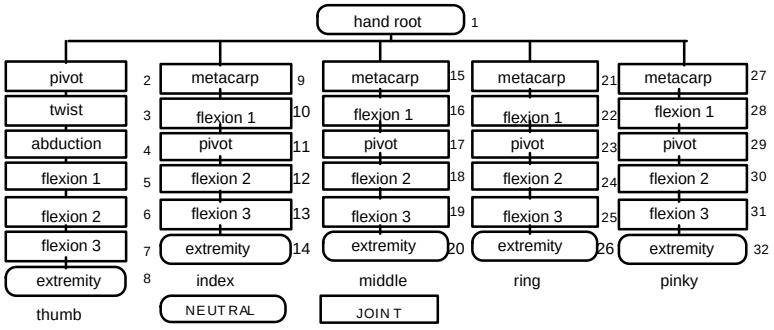


Figure 4-1 hierarchical hand modeling

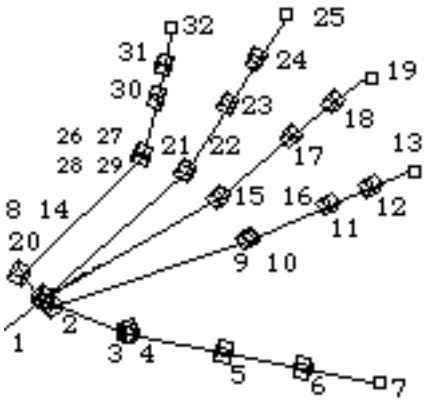


Figure 4-2 hand skeleton (the integer values refer the Figure 4-1)

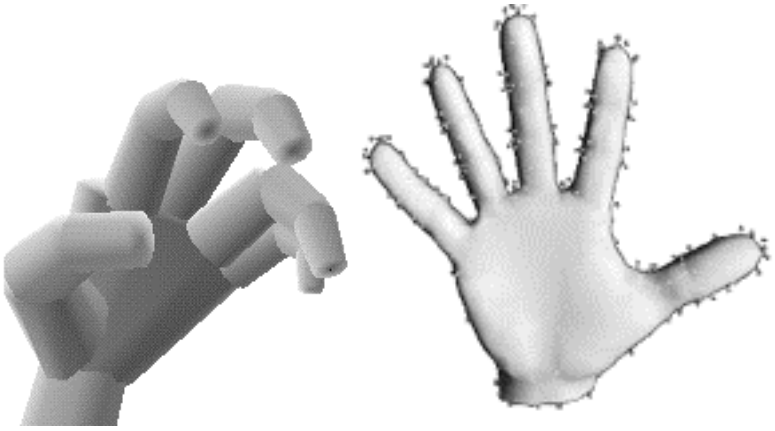


Figure 4-3 hand modeled with geometric primitives and free-form surface

4.3 Multi-sensor method

4.3.1 Multi-sensors modeling: highlight

The multi-sensor is one of the key data structures in our method. Sensors were already used by Boulic [Boulic, 86] in teleoperation, Zhao and Badler [Zhao and Badler, 94] for self-collision detection, and van de Panne and Fiume [van de Panne and Fiume, 93] for their sensor-actuator networks. We will discuss more about Zhao and Badler's work in our self-collision detection section. Here we propose the sphere multi-sensors that have both touch and length sensor properties with simple geometry for collision detection. We want to use the multi-sensors for the synthetic actor grasping motion control.

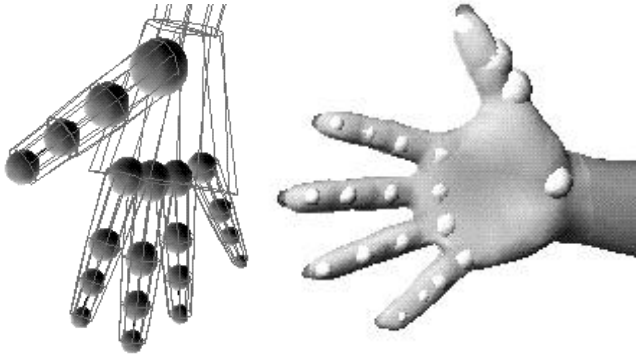


Figure 4-4 the hand with sphere sensors at each joint

Each sphere sensor is fitted to its associated joint shape with different radii. This configuration is important in our method because when a sensor is activated in a finger, only the articulations above it stop moving, while others can still move. By doing this, all the fingers are finally positioned naturally around the object, as shown in Figure 4-5.

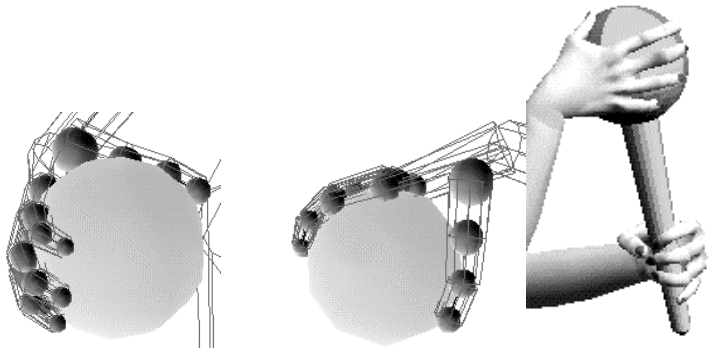


Figure 4-5 the final grasping posture resulting from sensor-object collision detection

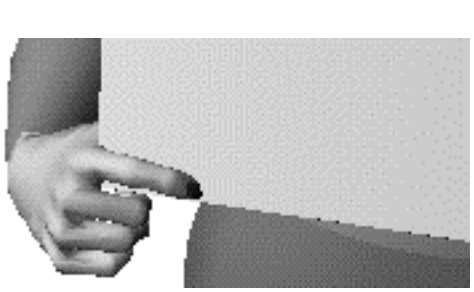
4.3.2 Heuristic grasping decision

From observing both real grasping and robotics methods [Rijpkema and Girard, 91], we see that there are many ways of grasping that involve one or two hands, and two to five fingers of each hand according to the type, geometry and weight of the object

to be grasped. Mas and Thalmann [Mas and Thalmann, 94] summarize this in Table 1 for solid primitives. If the primitive is too small, pinch is applied. If it is too large, two hands are used. Small and large are relative to hand size. We continue to use these results, but for a more general object, the decision should be made according to its multiple bounding volumes. The bounding volumes can be spheres, cubes, cylinders, etc. For example, for the surface model of a human body, the head is bounded by a sphere, and the neck by a cylinder**.**

Type	Size	Grasping method
Cube	thin	lateral
	thick	pinch
Sphere	small diameter	tripod
	large diameter	wrap
Cylinder and Frustum	small diameter	pinch
	large diameter	wrap
	small height	disk

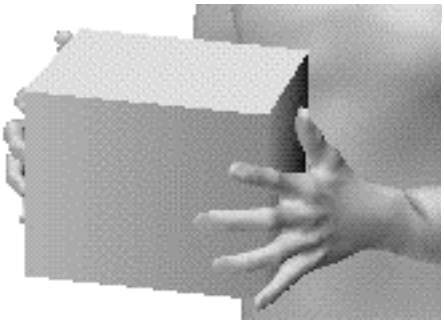
Figure 4-6 one heuristic grasping decision way



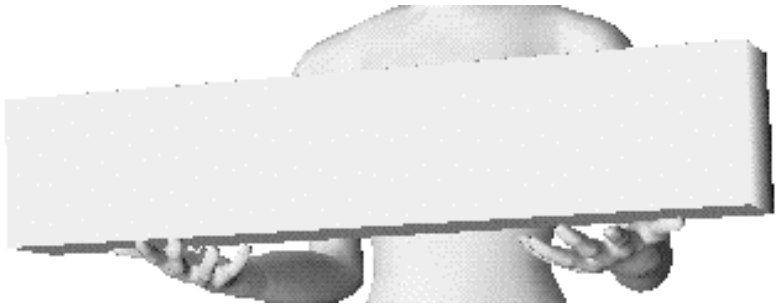
cube lateral

cube pinch

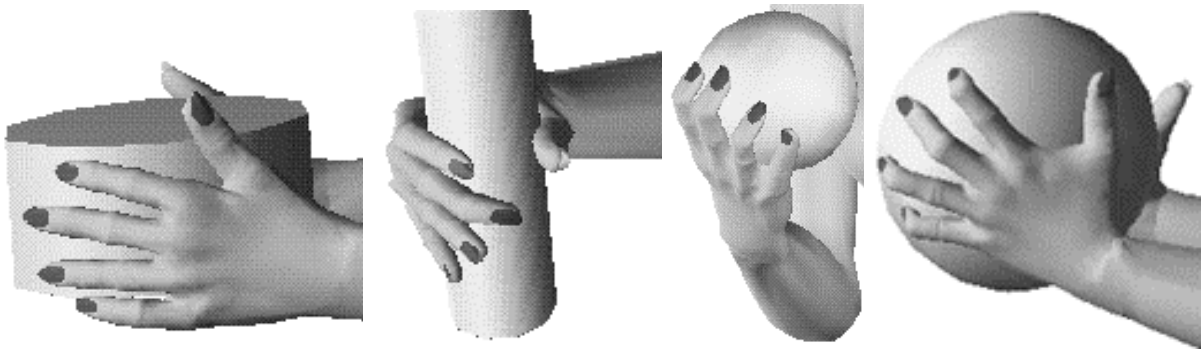
sphere tripod



cube pinch



cube hold



cylinder wrap cylinder wrap sphere wrap two hands sphere wrap

Figure 4-7 heuristic grasping for different solid primitives

Now, we come back to the problem of finding our final object frame. We use the similar heuristic consideration of [Mas and Thalmann 1994]. If we use a cube as an example, the steps can be as follows:

- we state that the thumb always remains on a visible surface;

- we prevent the palm from facing the exterior side of the body because it results in unnatural grasping;

- the contact faces can be chosen by minimizing a weighted sum of translation and rotation between the initial and final hand postures.

The desired cube frame is then determined according to the following procedure:

- set the center of the cube as the initial origin of the axes;

- transform the cube to the local frame of the hand to check whether it collides with the hand at initial posture;

- if no collision is detected, continue; otherwise move the origin closer to the border of the cube and repeat collision detection;

- the orientation of each axis is the same as the local frame of the cube;

Sphere, cylinder and frustum frames are similar to a cube. More details can be found in [Mas and Thalmann 1994].

For a general free-form surface object, the origin of the object frame can start from a point inside the object, which may be the center of its bounding object (either a sphere, cube or cylinder, according to its shape). In Figure 4-8, some final frames resulting from this method are displayed. For the head, it starts from the bounding cylinder of the neck. The collision detection should be performed on the object.

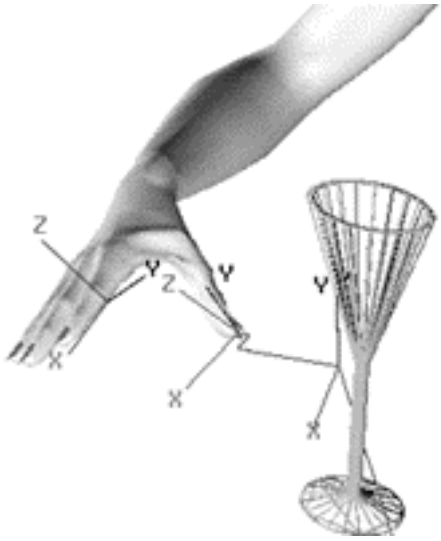


Figure 4-8 The final grasping frames from heuristic decision

4.3.3 Hand reaching

The result from the previous section gives the goal frame where the hand center should be positioned before performing the grasping task. Inverse kinematics is used to compute the final arm posture. The simplest way to move the arm from the initial posture to the final posture is the linear interpolation for all DOFs of the arm. It works well in some cases, but in order to have dynamically plausible results, more sophistication is necessary. Proportional-derivative servos are used to compute torques to move the arm toward the desired values. We approximate the arm and hand model with two truncated cones and one cylinder. Then for each DOF the control equation is

$$\tau = k(q_d - q) + k_v(\dot{q}_d - \dot{q}) \qquad \textbf{Eq. 4-1}$$

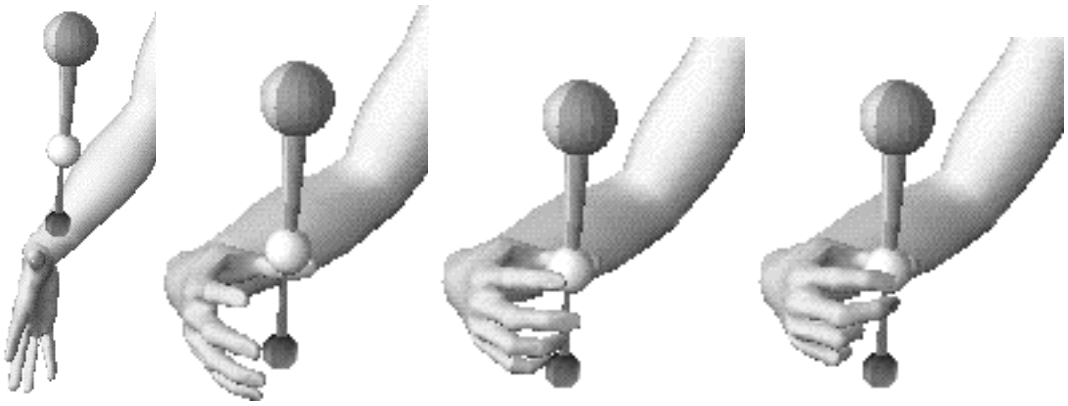
where q and $\dot{q}$ are the DOF value and its derivative; q_d and $\dot{q}_d$ are the desired values for q and $\dot{q}$; k is the proportional gain on DOF value, and k_v is the proportional gain for the derivative.

4.3.4 Sensor with hand motion pattern

Once the hand is close enough to the object for grasping, the fingers start moving with a one-hand motion pattern that is decided by the animator with the help of the above heuristic. Hand motion patterns are key frames derived from interactive design or motion capture devices. We reuse them at this step for general hand posture and object geometry. At each time step, collision detection is carried out between the object and hand sensor hierarchy. The sensor hierarchy is modeled according to the hand skeleton hierarchy. If no collisions are detected, the fingers move by the interpolation of the key frames toward the final frame. When a sensor is activated by colliding with an object, all DOFs associated with the ancestor's sensors stop moving, but other DOFs continue moving. When all DOFs stop moving, the fingers are closed around the object. The different personalized results can be derived with different hand motion patterns, object geometry and initial hand postures.

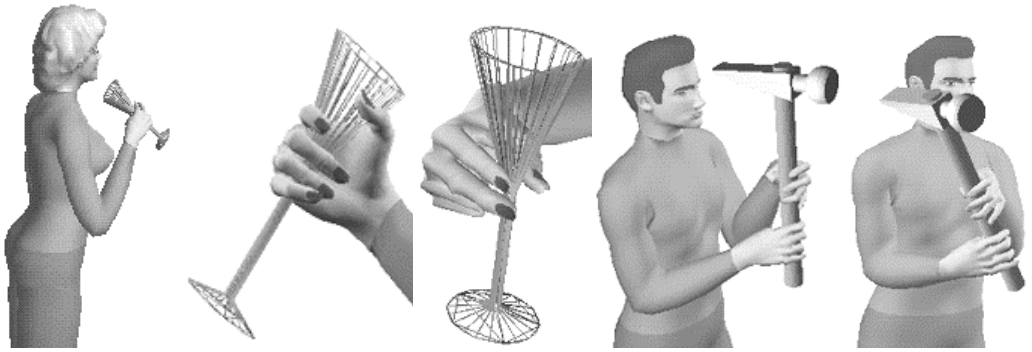
a b c d e f

Figure 4-9 some grasping patterns



a. b. c. d.

Figure 4-10 a resulted sequence using the pattern c of the Figure 4-9



a. b. c. d. e

Figure 4-11 some examples of grasping different types of objects

4.4 Object manipulation

To move the object to a specified place is one of the problems of object manipulation. This action often follows grasping. Usually, the movement is a global locomotion like walking, to reach a new place before moving objects by hands. During the locomotion period, the object's location follows the movement of the hand center. The remaining problem is to place the object to its destination by hands. We propose a method based on inverse kinematics for motion control.

Naturally, humans manipulate the object directly. It is a trivial task even for a child to move a grasped object to the desired place. However, directly controlling the joints of

an arm of the articulated figure is very complex because of the large number of the DOFs. Alternatively, to control the object directly is easier. For a rigid object, the number of DOFs is only 6. In our method, the object moves actively while the joints of an arm move passively following the object movement. Here the object motion is controlled interactively by keyframe, but generally it could be controlled using robotic methods. Figure 4-12 shows 3 key frames of the racket.

The next problem is to calculate the joint values of the arm following the object movement. It falls exactly into the inverse kinematics solution category described in section 2.3. Because the solution is based on the first order approximation of the system, we interpolate the key frames of object with a small time step, e.g., 0.04 sec. The intermediate position and orientation of the object are used as the guidance for inverse kinematics. So the joint values of arm are calculated at each step. The result is shown in Figure 4-13.

Figure 4-12 object is controlled by keyframe: three keyframes are specified for the racket

Figure 4-13 the arm movement is resulted from keyframes of the racket

Two more examples are shown in Figure 4-14 and 5-15.

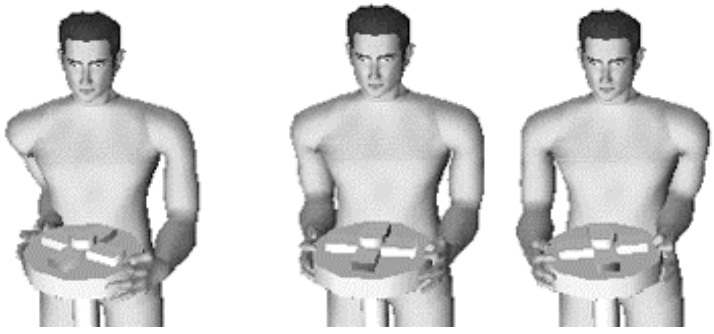


Figure 4-14 turning a wheel: the wheel is turned first by keyframe

Figure 4-15 hitting a nail: the hammer is moved first by keyframe

In this section, we proposed an object manipulation motion control method with an inverse process: The object is animated by keyframe or other motion control methods, then the inverse kinematics is used in multi-steps for the arm movement to follow the constraint of the object at each step.

4.5 Actor interaction

The method of object manipulation based on inverse kinematics can be further extended to actor-actor interaction. It is a key aspect in the natural social behavior of humans. Becheiraz and Thalmann [Becheiraz and Thalmann, 96] present a model of nonverbal communication and interpersonal relationship between virtual actors. Nonverbal communication improves actors' believability. They respond not only to the presence of the other actors but also to their postures. Furthermore, their interpersonal relationships are influenced by their social interactions. To avoid homogenous group behaviors, each actor is provided a distinct character profile. Their model successfully simulates the communication of humans in a virtual park garden. Tu and Terzopoulos [Tu and Terzopoulos, 94] work out a system simulating interactive behavior among fish with impressive results. Here we do not try to simulate the actor-actor interaction on the higher behavioral level as they did, but on motion control level, concentrating on the interaction resulting from actors using hands, e.g., hand shaking, holding, and pushing. The inverse method we use is an extension of [Lee et al., 90] which only treats holding an object according to the strength model of the actor.

This process can be implemented in two steps: the first step where one actor actively reaches, touches and holds another actor's hand, arm or body; and the second step where two actors move but keep in contact, e.g., shaking hands, holding closer, and pushing away. We found our heuristic multi-sensor method can be applied to this problem in the first step for arm movement and hand grasping. Different types of body-part bounding volumes, a cylinder for arms, a frustum for the thorax, and an ellipsoid for the head, are used for heuristic decisions regarding hand-target position and orientation. This resulting position and orientation can be further adjusted for a more flexible animation result. The sensor-body collision detection takes place only if a sensor-bounding volume collision occurs.

The second step of the process, which animates the motion of actors holding each other, uses a combination of direct and inverse kinematics. Our main idea is that only one actor actively moves, while the other follows. Direct kinematics and inverse kinematics fit this case exactly. The method is illustrated in Figure 4-16.

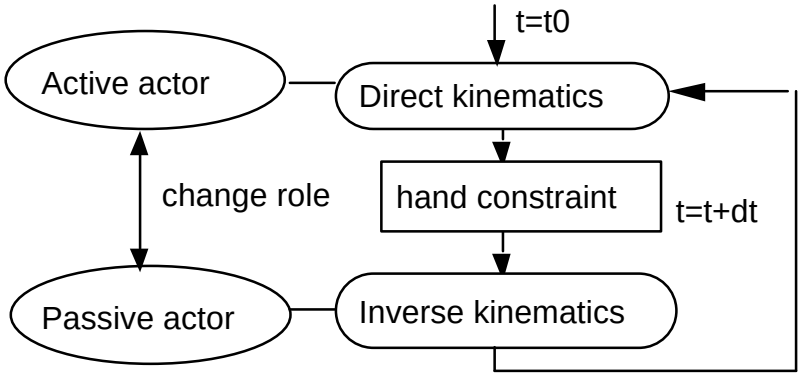


Figure 4-16 the combination motion control method

At each time step, the active actor is moved by direct kinematics. If this movement involves holding hands with the passive actor, the passive actor is moved by inverse kinematics in order to maintain the hand-holding. The hand of the active actor can be used as guidance for the inverse kinematics method.

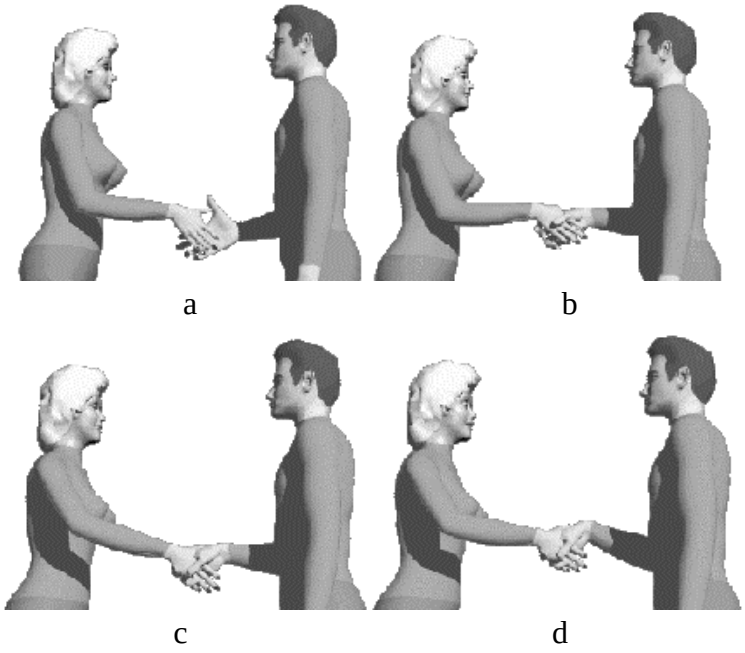


Figure 4-17 in a and b the grasping is applied, in c and d, the woman actor actively moves with direct kinematics while the man follows with inverse kinematics

The activity of actors can change in this process producing a more general interactive movement. This approach can be also used in an interactive system. When two actors hold each other, one of them can be animated by keyframes, while the other actor's movement can be automatically calculated.

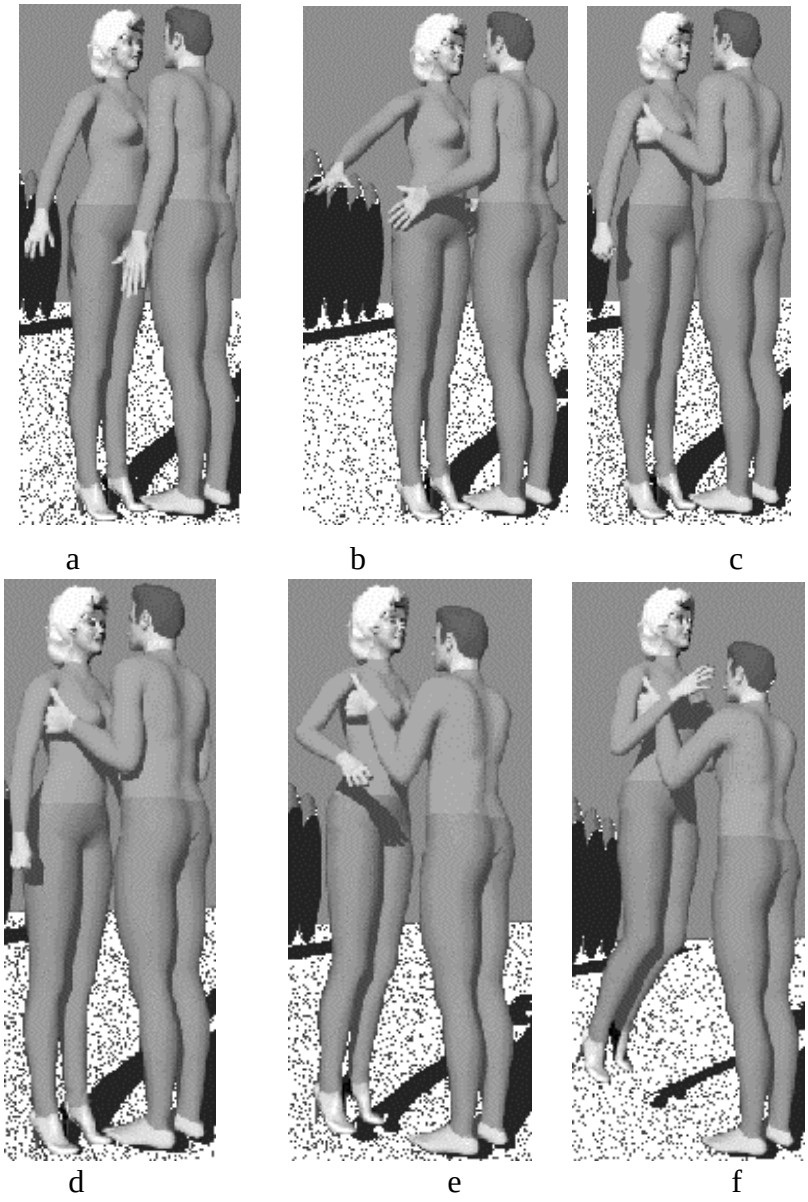


Figure 4-18 in a, b, and c, grasping is shown, in d, e and f, the actor on the left actively moves with direct kinematics while the actor on the right follows with inverse kinematics

4.6 Self-collision detection and response

When multiple body parts or several objects are animated, there is the chance that they collide and interpenetrate. This is often an undesired state. Two issues are involved: detecting that a collision occurs, and responding to it.

4.6.1 Background

The collision detection and response can be found from [Moore and Wilhelms, 88], [Zhao and Badler, 94], [Volino and Magnenat-Thalmann, 94] and [Bandi and Thalmann, 95]. Each work is on the different aspects. Moore and Wilhelms discuss collision detection in general, present two collision detection algorithms, describe modeling collisions of arbitrary bodies using springs, and present an analytical collision response algorithm for articulated rigid bodies that conserves linear and angular

momentum. The purpose of collision detection and response by Volino and Magnenat-Thalmann is to dynamically control the clothes animation, thus the efficiency of the algorithm is most important. The algorithm is based on hierarchisation and takes advantage of the adjacency which is combined with a surface curvature criteria to skip large regular regions from the self-collision detection. The work of Bandi and Thalmann is to simulate the rigid object collision, moreover, the moving human articulated figure collision with the rigid objects in the environment. The algorithm is based on an adaptive subdivision of the object space based on octree structure.

The topic of Zhao and Badler is closed to our work: the self-collision problem in the animation of human-like agents. Instead of using the traditional motion planning approach, they propose a method using potential functions as sensors for the self-collision detection. Inverse kinematics is used for collision response.

We propose a novel method for the same topic. For the collision detection, we extend the multi-sensor model used in grasping to the body. We also use the inverse kinematics for self-collision response. However, we use the secondary task of inverse kinematics, so we avoid giving the end effector a temporary goal in order to satisfy the potential function as in Zhao and Badler's method. Moreover, giving the end effector a temporary goal can not solve the problem of self collisions of which the end effector is not involved, e.g., elbow entering body while hand is not. We describe more details in the next section.

4.6.2 Motion control that is free of self collisions

The body articulations are highly constrained by joints the adjacent segment intersection can be eliminated. However, avoiding collisions of non-adjacent segments is much more difficult when the response time is constrained in an interactive environment.

Our extension of the hand sensor model to body is based on the fact that some key parts of the body collide: elbow, hand, knee, ankle, and foot. Thus, the multi-sphere sensors can be attached at these places. In Figure 4-19, ten sphere sensors are used. In grasping and object manipulation, only four sensors of the arms are necessary. We simplify the general collision detection to the very small number of the sphere-surface collision detection. It is more efficient than using the potential functions.

There is a special case to check, where sensors are not directly colliding with the body, but self collisions happen as in Figure 4-20. To detect it, two sensors are used along with the body.

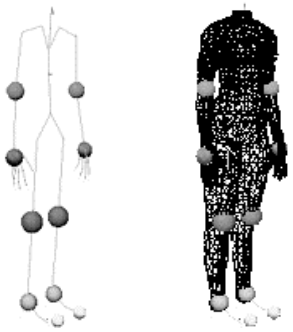


Figure 4-19 the body sensor model for self-collision detection, similar to the

Figure 4-20 special collision: two related sensors should be used for

hand sensor model **further test**

The method we use for the collision response is based on the secondary task of inverse kinematics. The secondary task is also used in the Coach-Trainee method [Boulic and Thalmann, 92] which allows the kinematical correction of Joint space based motion with respect to Cartesian constraints. The basic idea is to consider the joint motion delivered by a motion generator as a reference model whose tracking is enforced through the secondary task while the main task insures the realization of desired Cartesian constraints over some specified end effector(s). The goal here is similar to the Coach-Trainee in that we want the end effector (hand) to reach the target, while at the same time preventing the chain (arm) from penetrating the body. We illustrate the method in the case of right arm while keeping in mind its generality. It can be divided to the following steps:

---

(1) collision detection using sensor-body interference test. This step tests for intersection of each sphere sensor on the moving body part with other body volume;

(2) if collision is detected, calculate the response vector **v** opposite to the direction of collision to avoid penetration (see Figure 4-21 a), go to step 3; otherwise, the current posture is accepted and body is updated with the posture of the next time step, go to step (1).

(3) set up three reference coordination system and one guidance coordination system on the chain to prepare for the inverse kinematics: **1**,**2** and **3** on the chain root, sensor center and end effector respectively while guidance **g2** is the offset from **2** along the response vector **v** (see Figure 4-21 b).

(4) calculate the secondary task, i.e., the joint space vector $\Delta\mathbf{z}$ in Eq. 2-7, from the delta vector $\Delta\mathbf{z}'$ in Cartesian space. The delta vector $\Delta\mathbf{z}'$ transforms the coordinate system **2** to **g2** and can be calculated from **2** and **g2.** The inverse kinematics is used on chain **1-2**:

$$\Delta\mathbf{z} = \mathbf{J}'^{+}\Delta\mathbf{z}' \quad \textbf{Eq. 4-2}$$

where $\mathbf{J}^{+}$ is the pseudo inverse of the Jacobian matrix $\mathbf{J}$ of the shorter chain.

(5) calculating the new joint value $\Delta\mathbf{q}$ using kinematics (Eq. 2-9):

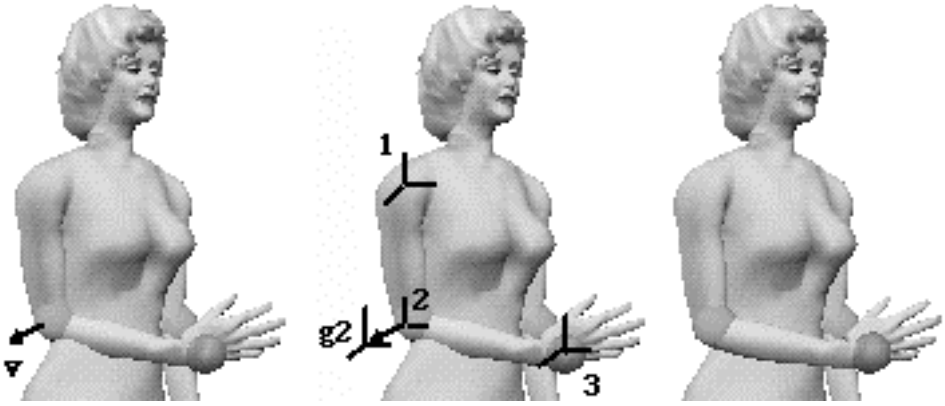
$$\Delta\mathbf{q} = \mathbf{J}^{+}\Delta\mathbf{x} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\Delta\mathbf{z}$$

where $\Delta\mathbf{x}$ represents the main task that is a vector from end effector (hand) to the target (object) in Cartesian space.

(6) stop if the collision is avoided (see 5-21 c), or reaches maximum step due to the limit of number or the value ranger of DOFs; otherwise go to step 2.

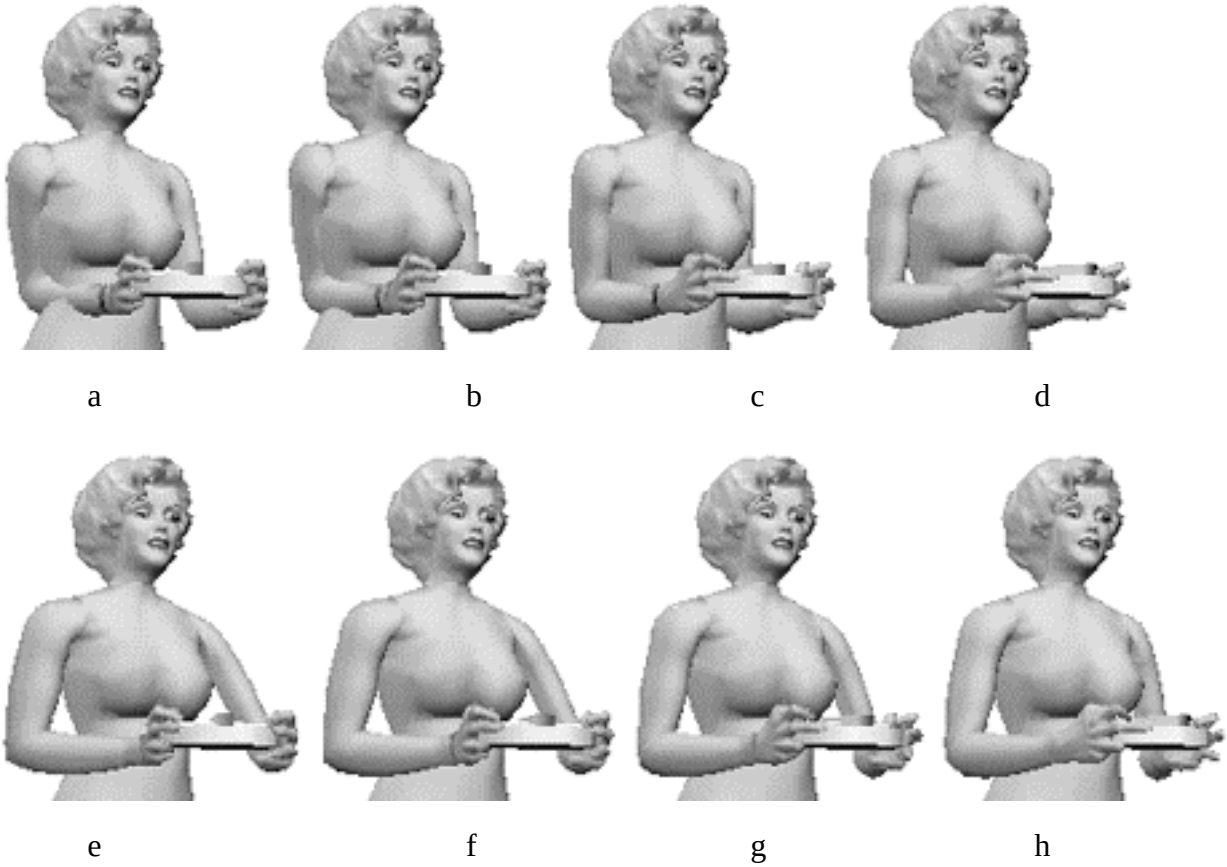
Table 4-1 flow chart for generating the movement that is free of self collisions

We have combined inverse kinematics as a full chain and sub-chain. One more example is shown in Figure 4-22.



a b c

Figure 4-21 self-collision correction with secondary task of inverse kinematics

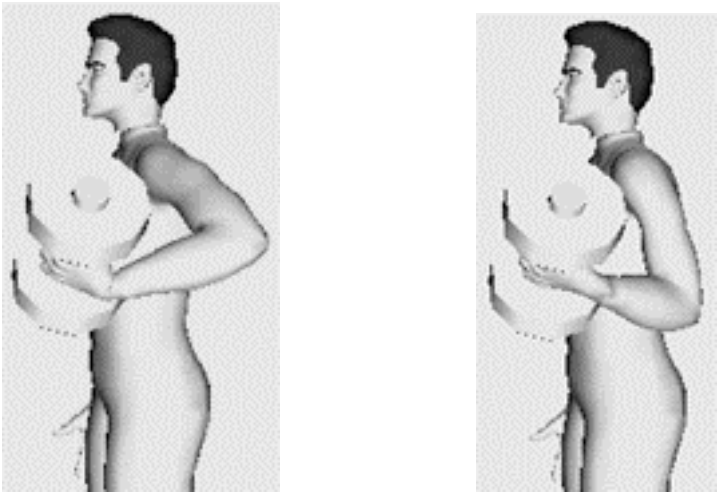


a b c d

e f g h

Figure 4-22 snapshots of object manipulation sequence: in a, b, c and d, with self collisions; in e, f, g, and h, they are automatically corrected

This method can be extended for simulating a more natural way of grasping if the weight of object is considered. We use the secondary task again, but this time the coordinate system **g2** is not only translated along the response vector but also towards the floor if the elbow sensor happens to be too high. Other steps are same as in the self-collision correction algorithm. One simple example is shown in the figure below.



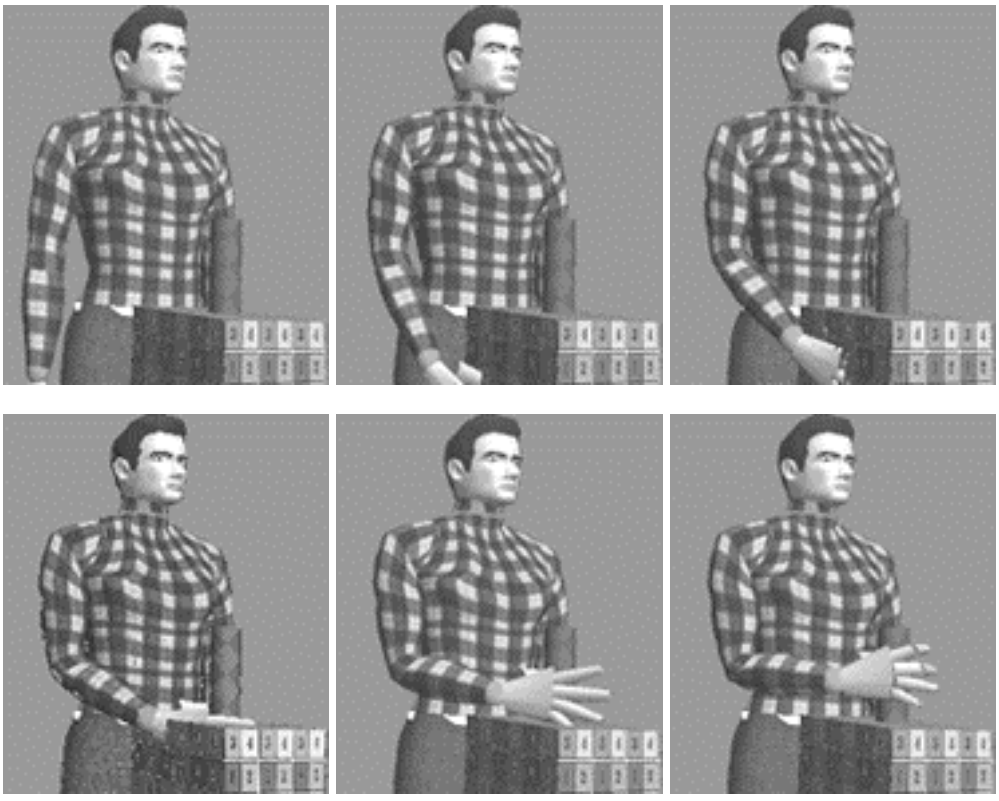
a. without correction b. with correction

Figure 4-23 using secondary task to simulate the effect of holding a heavy object

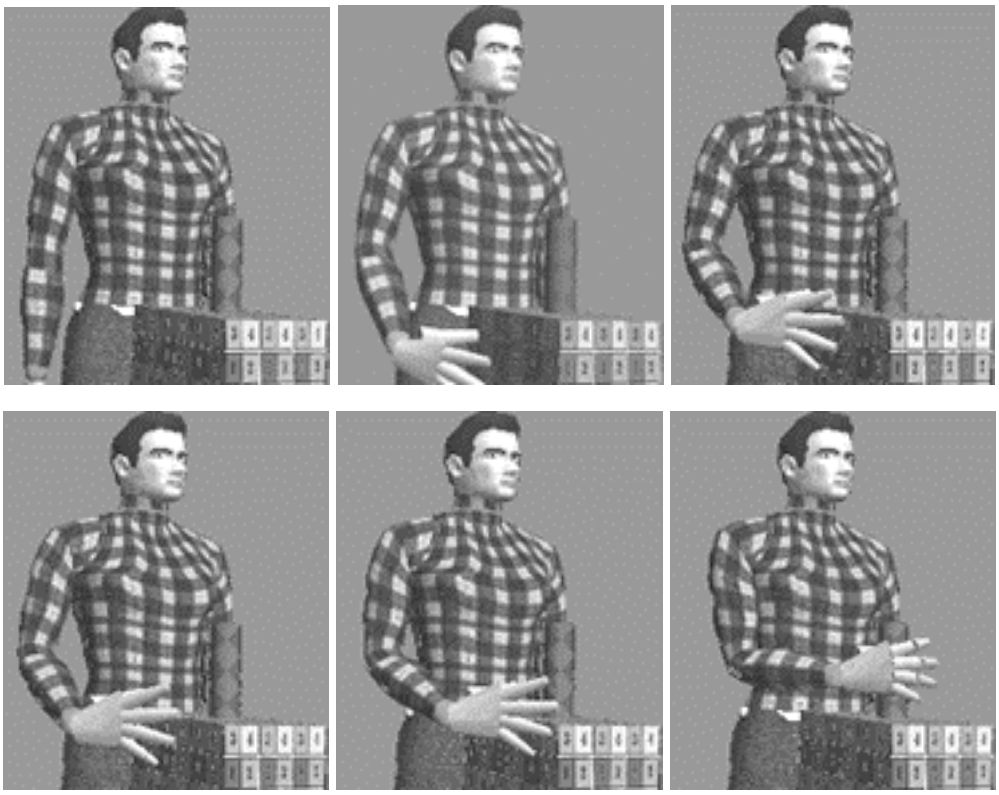
In b of the above figure, we correct the height of the elbow using the secondary task. A downward vector in Cartesian space is used along with the self-collision free vector. The result is close to the real human's movement.

4.6.3 Collision avoidance with temporary goals

We also implemented the collision avoidance using the temporary goal for end effector as Zhao and Badler proposed, but we still use our simple geometric sensors for collision detection. A temporary goal is defined for the end effector. We show the result in the following figure.



a. without collision detection and correction: collision happens



b. with collision detection and temporary goal for the end effector: collision is avoided

Figure 4-24 using the temporary goal for the end effector to avoid collision in hand reaching. The temporary goal is displayed as a coordinate system in the first snapshot of b. together with the final goal

In the above example, only one temporary goal is used for simplicity. For a more complicated object, more temporary goals are needed. They can be interactively set by the user in order to avoid a more complicated automatic planning.

4.7 Consistent grasping interactions using CyberGlove

In this section, we propose interactive grasping of virtual objects while wearing a digital glove. Such an approach is also interesting for Virtual Reality where more elaborate hand interaction is now possible with recent introduction of digital glove. In this context, we map the real posture of the digital glove on a sensor-based virtual hand in order to ensure a consistent collision-free grasping of virtual objects and provide a consistent visual feedback.

When an operator is wearing a digital glove, the joint values acquired with the device are normally used to update the visualization of the hand posture. Such an approach is pertinent as long as the device is used to specify commands through posture recognition [Stunman et al., 89]. Among these commands there is usually a symbolic grasping of virtual objects where the relative position and orientation of the object is maintained fixed in the hand coordinate system as long as the grasp posture is recognized. It is suitable for pick-and-place tasks of rigid objects, However, as the virtual environment is becoming more and more complex, especially with the advent of virtual humans, hand-based interactions also evolve in complexity. The limitation of

the current approach mostly comes from the rough relative positioning of the hand and object which does not convey a clear understanding of the action to perform with the object. Thus it is necessary to evaluate precisely the hand (fingers) object contact. We model the multi-sensor structure in the model of the CyberGlove, as shown in Figure 4-25.

a b c

Figure 4-25 CyberGlove worn by a person; b. its 3-D model provided with the digital glove of Virtual Technologies; c. multi-sensors are associated with the 3-D model to detect object contact for the grasping automata

Based on this model, we propose a novel approach for the interactive and consistent grasping of virtual entities with the interactive grasping automata, as shown in Figure 4-26. We also implement hand posture correction from the finger-sensor-object contact. More details can be found in [Rezzonico et al., 95].

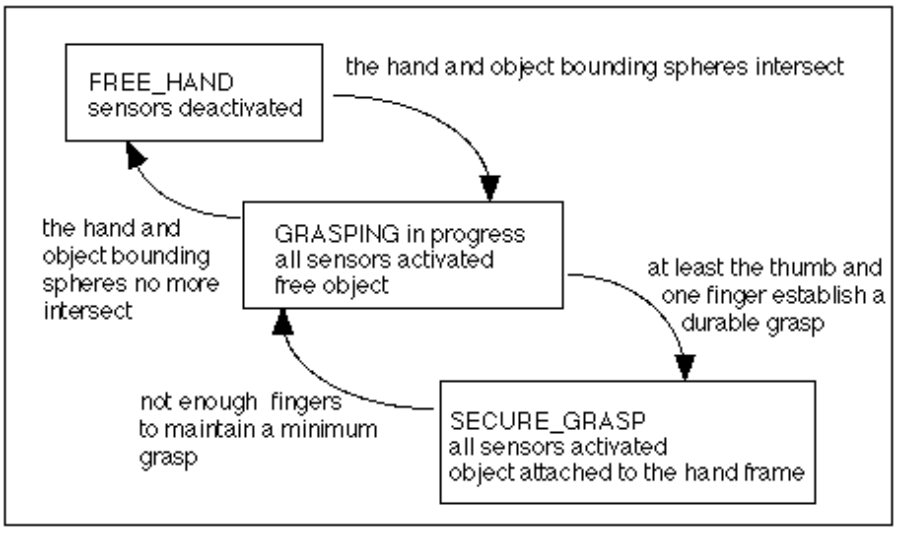


Figure 4-26 the interactive grasping automaton: the purpose is to display a posture of the hand consistent with the on-going manipulation of the virtual object

Here is an example of automatic grasping by a synthetic actor and interactive grasping by a real human wearing the CyberGlove. In Figure 4-27 a, the model of the CyberGlove controlled by a performer is moving towards the sphere; In b, it touches the sphere and its color changes to blue; In c, it grasps the object and its color changes to green. It can move the sphere now; In d, it moves the sphere to the new location and releases the sphere. Its color changes back to normal skin color; From e to f, the

synthetic actor reaches the sphere and grasps it. The synthetic actor can also move the sphere for the performer to grasp again with CyberGlove by repeating a to d.

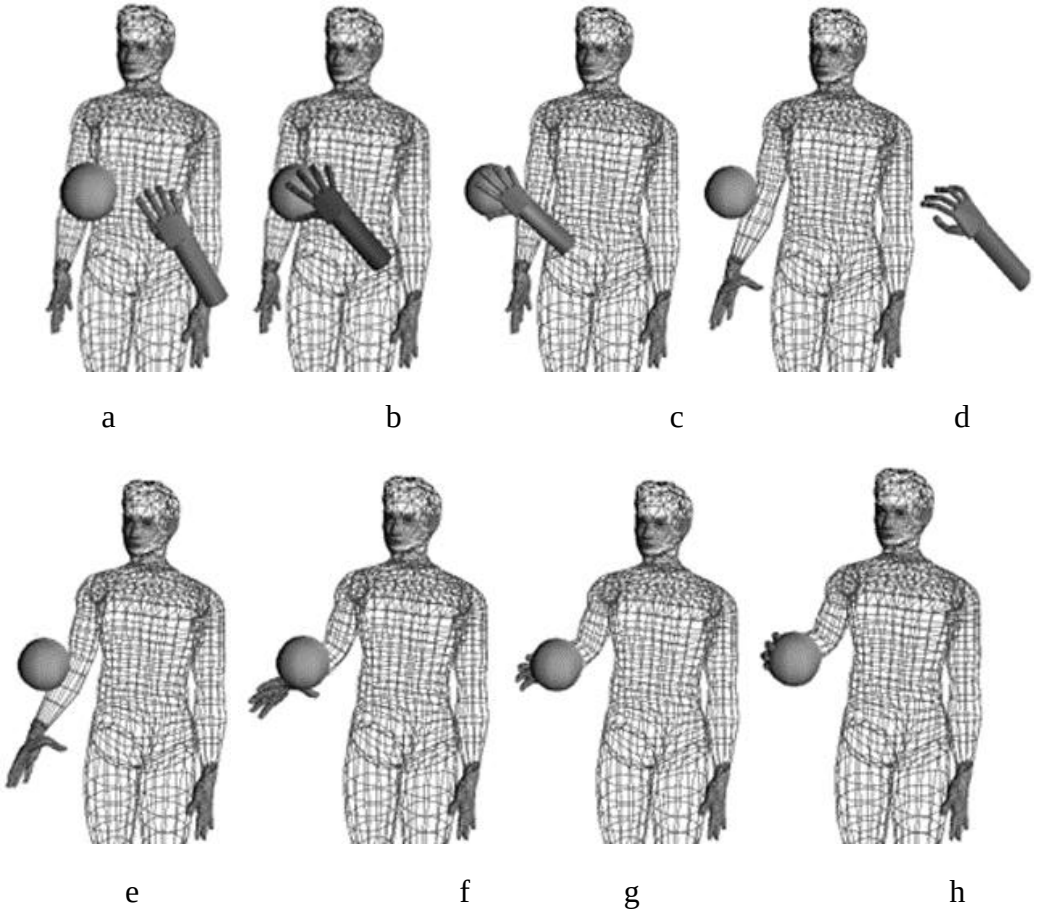


Figure 4-27 frames of snapshot of interactive grasping by a real person (a-d) and automatic grasping by a synthetic actor (e-f)

The interactive grasping experiments were realized at an interactive rate of approximately 10 images per second on a workstation screen with the management of up to 20 spherical sensors and the display of a high resolution hand (2600 polygons) and the object. Although a small delay was perceptible between performance and display of the corrected hand posture, it is not a decisive aspect.

Interactive grasping proved to be a valuable tool for the interactive design of realistic grasping posture for human animation system. From the visual feedback of the corrected grasp, an operator is able to achieve a good perception-action control loop in order to continuously adjust the grasp into a realistic posture. In our default setting the display is completed with color coding information as described now. Each state of the interactive grasping automata is associated with a different color of the whole hand : pale skin color for FREE_HAND, blue for GRASPING_IN_PROGRESS and green for SECURE_GRASP. Moreover, whenever a sensor is colliding, its color changes from dark gray to light gray. This additional symbolic information significantly enhances the operator feedback.

4.8 Conclusion

To conclude, we proposed an efficient set of techniques which allow us to animate hands. With our heuristic grasping decision and multi-sensor based approach, the animator can avoid the tedious work of controlling each hand-arm joint. This method uses the basic motion control methods: direct, inverse kinematics and procedural method. Personalized results can be accomplished with different hand motion pattern profiles. The resulting arm-hand motion can be further improved by other approaches such as keyframe, signal-processing-based methods and optimal control.

We presented a new application to simulate actor interaction with hands by the method of combining direct and inverse kinematics.

We proposed a method of motion control that is free of self collisions. It is done by extending the multi-sensors from hand to body model for the fast self-collision detection. The collision correction is done with a novel method that combines inverse kinematics for the full chain and the sub-chain in multiple steps. The full chain inverse kinematics includes the secondary task that is calculated from the collision response vector and the sub-chain inverse kinematics. The main task of full chain inverse kinematics is derived from the grasping planing, keyframe and other specifications. A sequence that is free of self collisions can be produced directly.

We finally modeled the same multi-sensor structure on the 3-D hand model of the CyberGlove provided from Virtual Technologies. Using a method based on grasping automata and hand posture collision response, we are able to simulate the interactive grasping by a real person wearing the CyberGlove. This method can work along with automatic grasping by a synthetic actor in the scene.

Our future work is to extend this method to induce more human-human, human-environment interaction. We also continue to work on a more general problem of the collision detection and response during grasping.

5. Sequence Manipulation

In this chapter, we describe our work on sequence manipulation. A sequence is a set of trajectories that represents values for each degree of freedom (DOF) within a limited time. We present a method of multiresolution filtering on motion blending. Other important manipulation techniques such as sequence editing on the coefficients of the frequency bands are implemented. Finally, we present the sequence manipulation techniques by directly using cubic splines.

A sequence can result from motion generators. First, we treat the sequence as signals and use the multiresolution filtering from the signals for motion editing and blending, and later, apply parametric splines directly to compress the data of the sequence.

5.1 Background

5.1.1 Current research

Keyframe animation is usually edited by adding, deleting and modifying key frames, the same process used to initially create the animation. Motion editing is more attractive recently as the motion capturing becomes more popular [Bruderlin and Williams, Witkin and Popovic, 95, Unuma et al., 95]. Bruderlin and Williams assemble a simple library of signal processing techniques applicable to animated motion. They introduce multiresolution filtering, multitarget motion interpolation with dynamic time warping, waveshaping and motion displacement mapping. They also illustrate an interactive editing technique on the coefficients of the frequency bands. Witkin and Popovic describe a simple technique for editing captured key frames based on warping of the motion parameter curves. They show that a whole family of realistic motions can be derived from a single captured motion sequence using only a few key frames to specify the motion warp. Unuma et al. apply Fourier transformations to data on human walking for animation purposes. Based on frequency analysis of the joint angles, a basic ‘walking’ factor and a ‘qualitative’ factor like “brisk” or “fast” are extracted. These factors are used to generate new movements by interpolation and extrapolation in the frequency domain, such that a walk can be changed continuously from normal to brisk walking.

First, we propose a sequence blending method based on multiresolution filtering. In this method, we construct the layered lowpass (Gaussian) and bandpass (Laplacian) pyramids for the two sequences to be blended, then we blend each layer of two Laplacian pyramids related to the same DOF at the transition period of the two sequences. This process continues for all DOFs and finally results in a new set of Laplacian pyramids. They are finally used to reconstruct a new sequence with a smooth transition between two sequences. We compare it with the straightforward weighted sum of the two motion curves.

Second, we propose a sequence data compression directly using cubic splines.

Finally, we implement other important methods such as interactive time warping, editing on the coefficients of the frequency bands of a sequence, and data compression from Laplacian pyramids.

In the following section, we briefly describe the multiresolution filtering before presenting our work.

5.1.2 Multiresolution filtering

Multiresolution filtering is a fundamental technique of signal processing. It contains a range of digital filtering bank techniques which pass a signal through a cascade of lowpass filters to produce a set of short-time bandpass or lowpass signal components. Figure 5-1 is a graphical representation of the iterative filtering procedure in one dimension [Burt and Adelson, 83]. Each row of dots represents the samples, or pixels, of one of the filtered images. The lowest row, G_0, is the original image. The value of each node in the next row, G_1, is computed as a weighted average of a 5 x 5 subarray of G_0 nodes as shown. Nodes of array G_2 are then computed from G_1 using the same pattern of weights. The process is iterated to obtain G_2 from G_1, G_3 from G_2 and so on. If we imagine these arrays stacked one above the other, the result is the tapering data structure known as a pyramid [Tanimoto and Pavlidis, 75].

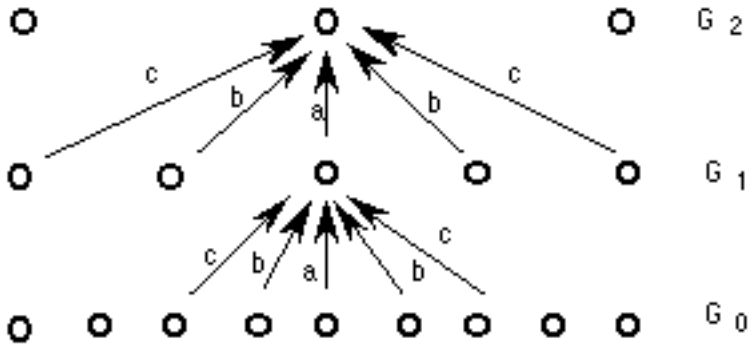


Figure 5-1 one dimensioned graphic representation of iterative REDUCE operation used in pyramid construction

Another method for pyramid construction directly uses wavelet functions [Stollnitz et al., 95]. The concept of a vector space is used in order to define the wavelet basis functions. A one-pixel image is just a function that is constant over the entire interval [0, 1). Let V^0 be the vector space of all these functions. Similarly, a two-pixel image has two constant pieces over the intervals [0, 1/2) and [1/2, 1) and V^1 be the vector space. Continuously, the space V^j includes all piecewise constant functions defined on the interval [0, 1) with constant pieces over each of 2^j equal subintervals. Thus, the space V^j are nested: $V^0 \subset V^1 \subset V^2 \subset \cdots$. A simple basis for the space V^j is given by the set of scaled and translated "box" functions:

$$\phi_i^j(x) := \phi(2^j x - i), \quad i = 0,\ldots,2^j - 1 \quad \textbf{Eq. 5-1}$$

where $\phi(x) := \begin{cases} 1 & \text{for } 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$.

The wavelets corresponding to the box bases are known as the Haar wavelets, given by

$$\psi_i^j(x) := \psi(2^j x - i), \quad i = 0,\ldots,2^j - 1 \quad \textbf{Eq. 5-2}$$

$$\text{where } \psi(x) := \begin{cases} 1 & \text{for } 0 \le x < 1/2 \\ -1 & \text{for } 1/2 \le x < 1 \\ 0 & \text{otherwise} \end{cases}.$$

Here is an example of pyramid construction:

Resolution	Averages	Detailed Coefficients
1	6	2
2	8 4	1 -1
4	9 7 3 5	

Table 5-1 decomposition of four-pixel image

Here is an example using the equations:

$$I(x) = c_0^2 \phi_0^2(x) + c_1^2 \phi_1^2(x) + c_2^2 \phi_2^2(x) + c_3^2 \phi_3^2(x)$$

$$= 9 \times \phi_0^2(x) + 7 \times \phi_1^2(x) + 3 \times \phi_2^2(x) + 5 \times \phi_3^2(x)$$

$$= c_0^1 \phi_0^1(x) + c_1^1 \phi_1^1(x) + d_0^1 \psi_0^1(x) + d_1^1 \psi_1^1(x)$$

$$= 8 \times \phi_0^1(x) + 4 \times \phi_1^1(x) + 1 \times \psi_0^1(x) - 1 \times \psi_1^1(x)$$

$$= c_0^0 \phi_0^0(x) + d_0^0 \psi_0^0(x) + d_0^1 \psi_0^1(x) + d_1^1 \psi_1^1(x)$$

$$= 6 \times \phi_0^0(x) + 2 \times \psi_0^0(x) + 1 \times \psi_0^1(x) - 1 \times \psi_1^1(x)$$

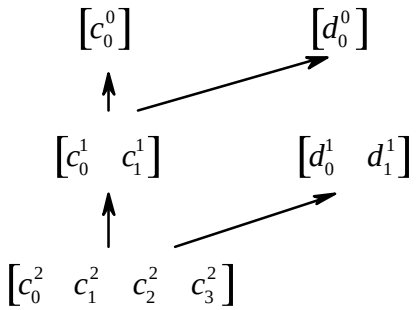


Table 5-2 the filter bank with Haar wavelets: each higher level of the left pyramid contains less detail, or "lowpass pyramid"; the right pyramid captures the lost detail of the same level in the lowpass pyramid, "bandpass pyramid"

Here are two general procedures to accomplish the above construction [Stollnitz et al., 95]:

proc **DecompositionStep** (C: array [1..h] of reals)

```
    for i<-1 to h/2 do
            C' [i]<- (C [2i-1] + C [2i]) / √2
            C' [h/2 + i]<- (C [2i-1] - C [2i]) / √2
    end for
    C<-C'
end proc
proc Decomposition (C: array [1..h] of reals)
    C<-C / √h       // normalize input coefficients
    while h > 0 do
            DecompositionStep (C [1..h])
            h<-h / 2
    end while
end proc
```

Table 5-3 one dimension Haar wavelet transforms

5.2 Multiresolution filtering on key frames

In this section, we describe the construction of the lowpass pyramids and bandpass pyramids for key frame sequence. We use the same method as in [Burt and Adelson, 83]. It is a multiresolution analysis in terms of a cubic B-spline scaling function in the currently popular wavelet parlance [Chui, 92]. This method obtains the lowpass pyramid by successively convolving the signal with a B-spline filter kernel while the signal sampling is subsampled by a factor of 2 at each iteration as shown in Figure 5-1. The local averaging process is called REDUCE and generates each pyramid level from its predecessor:

$$G_l = REDUCE(G_{l-1}),$$

$$G_l[i] = \sum_{m=1}^{5} w(m) G_{l-1}(2i+m), \quad (i = 0, \cdots, n-5) \qquad \textbf{Eq. 5-3}$$

where n is the number of samples in layer l - 1. If the original signal, i.e., G_0, contains $2^N + 1$ samples, there are N layers for the pyramid. In Figure 5-1, N = 3.

The weights *w(m),* called the generating kernel, are chosen first subject to be symmetric, $w(0) = a$, $w(-1) = w(1) = b$ and $w(-2) = w(2) = c$. Second, to be normalized, $a+2b+2c = 1$. Third, from each level l each node must contribute the same total weight to level l +1 nodes: thus, $a + 2c = 2b$. If a is considered as a free variable, we have $b = 1/4$ and $c = 1/4 - a/2$. In Figure 5-2, the curves G_0, G_1 and G_2 are lowpass data constructed by set $a = 0.4$. G_0 containing 9 samples, 5 for G_1, and 3 for G_2. The two first and final samples are copied to the next layer. Burt and Adelson refer this sequence G_0, G_1, G_2 ..... as the *Gaussian pyramid* because the equivalent weighting functions resemble the Gaussian probability density function when $a = 0.4$.

The next step is to construct the bandpass pyramid that can be subtracted at each level of the low-pass Gaussian pyramid from the next lower level. Because these arrays differ in sample density, it is necessary to interpolate new samples between those of a given array before it is subtracted from the next lower array. This is done by the EXPAND operation:

$G_{l,k} = EXPAND\ (G_{l,k-1})$,

$G_{l,0} = G_l$, **Eq. 5-4**

$G_{l,k}(i) = 4\sum_{m=-2}^{2} G_{l,k-1}(\frac{2i+m}{2})$, for k > 0, (i = 0, ..., n - 5)

where n is the number of samples in layer l -1.

Now the bandpass pyramid L_0, L_1, ..., L_{N-1} can be calculated:

$L_l = G_l - EXPAND(G_{l+1}) = G_l - G_{l+1,1}$, (l = 0, ..., N-1)**Eq. 5-5**

thus each node of L_l can be obtained directly by convolving the weighting function W_l with the signal. This difference of Gaussian-like functions resembles the Laplacian operators commonly used in the signal processing, the sequence L_0, L_1, ..., L_{N-1} is refereed as the *Laplacian pyramid*. Each L_l represents the signal component with the same frequency. The higher of the value *l*, the higher the frequency. The frequency of L_l is 2 times as high as the frequency of its next layer L_{l-1}.

The steps used to construct the Laplacian pyramid may be reversed to recover the original image G_0. The top pyramid level, G_N, is first expanded and added to L_{N-1} to recover G_{N-1}; this array is then expanded and added to L_{N-2} to recover G_{N-2}, and so on. We get:

$G_0 = G_N + \sum_{l=0}^{N-1} L_l$ **Eq. 5-6**

It describes the re-constructing process. From another point of view, it represents the original signal in terms of its frequency.

As we have used the cubic Hermite spine to interpolate key frames, we use it again in the EXPAND process to get more samples on each layer. Suppose T_0 is time step of samples in G_0 (it is also the same in L_0), then the step in layer I for both G_i and L_i is calculated from the REDUCE process:

$T_l = 2^l T_0$ **Eq. 5-7**

Now we can list the reconstructing process:

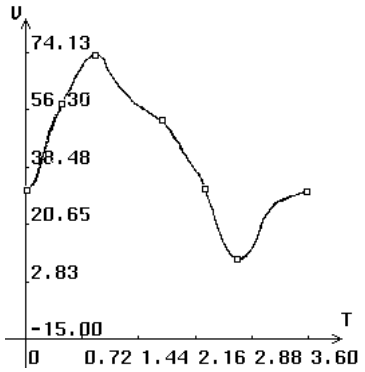
---

(1) set i <- N;
(2) if i equals N, set L_i <- G_N;
(3) get m - 1 more samples of L_i at $\frac{1}{2}T_i$, $\frac{3}{2}T_i$, ..., on its curve, so the number of samples in L_i is the same as in L_{i-1}, where m is the former number of samples in L_i, thus $m = 2^{N-i} + 1$;
(4) add the sample values of L_i to the sample values of L_{i-1} at the same time step, and use the new samples to construct L_{i-1};

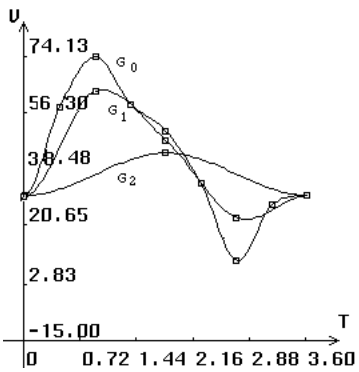
(5) if i-1 is 0, finish the process by setting G_0 <- L_{i-1}; otherwise set I <- I - 1, go to step (2).

Table 5-4 reconstruction process

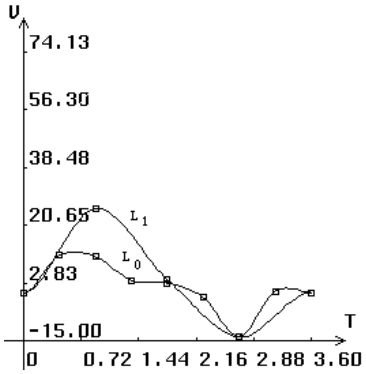
The following figure illustrates the Gaussian and Laplacian pyramid construction, see Figure 5-2 b and c. The pyramids are used to reconstruct the signal, see Figure 5-2 d and e.



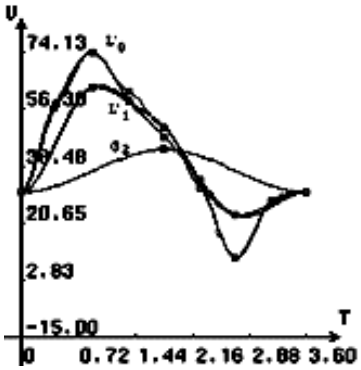
a. original trajectory



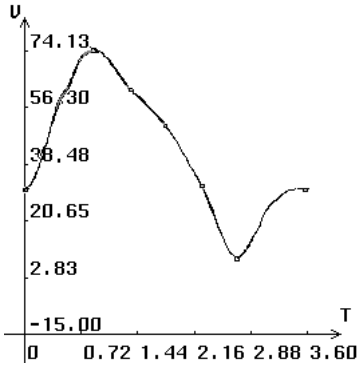
b. construction of Gaussian pyramid lowpass signal: G_0, G_1 and G_2



c. construction of Laplacian pyramid bandpass signal: L_0 and L_1



d. reconstructed process: $L'_2 = G_2$, L'_0 $L'_1 = L'_2 + L_1$ and $L'_0 = L'_1 + L_0$



e. the resulting trajectory:

Figure 5-2 the construction pyramids of the trajectory of the wrist flexion DOF and the trajectory reconstruction from the pyramids

5.3 Sequence blending on frequency band

In this section, we apply the multiresolution filtering for combining two key frame sequences into a longer sequence. The problem is how to link two sequences to have a smooth transition. The most common method is to use a straightforward weighted sum of the two motion curves, in the transition period:

$$q_{blend}(t) = w(t)q_1(t) + (1 - w(t))q_2(t) \qquad \textbf{Eq. 5-8}$$

where $t_1 \le t \le t_2$ defining the transition period; $w(t)$ is a normalized ease-in/ease-out weight function.

The weighted sum method works well. However, using the multiresolution filter, we can improve it in the same way as Peter and Adelson have done on image blending.

The method we use for the motion signal blending is similar to the method for the image. It contains two more steps than the weighted sum method: pyramid construction and signal reconstruction. Suppose we have two sequences SA and SB, with SA in period [t1, t2] and SB in [t3, t4], t3 > t1 and t4 > t2, we carry out following steps:

---

(1) constructing Laplacian pyramids LA and LB for SA and SB respectively;

(2) constructing a third Laplacian pyramid LS as following:

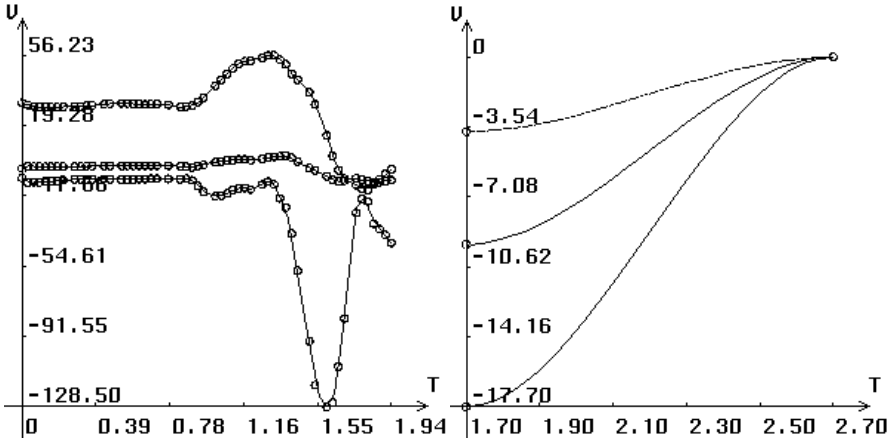
2.1 copying nodes from the layers of LA for [t1, t3);
2.2 copying nodes from the layers of LB for (t2, t4];
2.3 using the weighted sum method for $[t2, t3]$
$LS_l(t) = w(t)LA_l(t) + (1 - w(t))LB_l(t)$ for each layer l;

(3) reconstructing a new sequence S from LS.

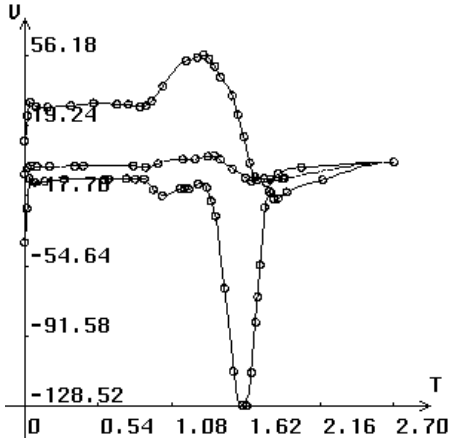
Table 5-5 multiresolution sequence blending flow chart

The method works on the transition period of the two incorporated sequences. It is [t3, t2] with t3<t2. More interpolation samples should be in [t3, t2] which is decided by the depth of pyramid layers input by the animator. Another interesting case is when t3 > t1 and t4 < t2, i.e., the second sequence SB takes place inside the first sequence SA. In this case step 2.3 is used in [t3, t4] but with the weighting function reaching 0 at middle point (t3+t4)/2.0 and 1.0 at t3 and t4. So the method is general by defining a general weighting function. From the point of view of image processing, the component signals within a transition zone are joined using a weighting average which is proportional in size to the wave lengths represented in the band. So the result can be seamless.

We show one example (more examples are shown in the video):



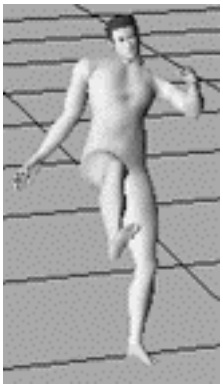
a. the first sequence SA b. the second sequence SB



c. result of blending sequence S

Figure 5-3 blend sequence SA and SB to S, the transition period is between 1.70 sec and 1.94 sec

a. a posture of SA b. another posture of SA c. a posture of transition



d. another posture of transition e. a posture of SB f. another posture of SB

Figure 5-4 some snapshots of the postures resulting from the example of the previous figure, the smooth transition is shown in c. and d. (the same example is shown in video for better illustration)

5.4 Interactive editing on bandpass pyramid

We describe the implementation of interactive editing of the frequency band coefficients in this section. We start from the Eq. 5-6. It can be generalized by defining weights on each term:

$$G_0 = w_N G_N + \sum_{l=0}^{N-1} w_l L_l \qquad \textbf{Eq. 5-9}$$

where w_l are weights. If each one is a unity, the resulting key frames are approximately same as the original ones. However, by interactively setting different values, we obtain a new sequence. In Figure 5-5, a, b, c, d are 4 original key frames, and e shows trajectories of three DOFs: body_turn, r_hip_abduct and r_ankle_flexion. By pyramid constructing with 4 layers and setting weights 2.0, 1.5, 0.0 and 1.0 respectively, we get a new series of keyframes shown in f, g, h, i and j.

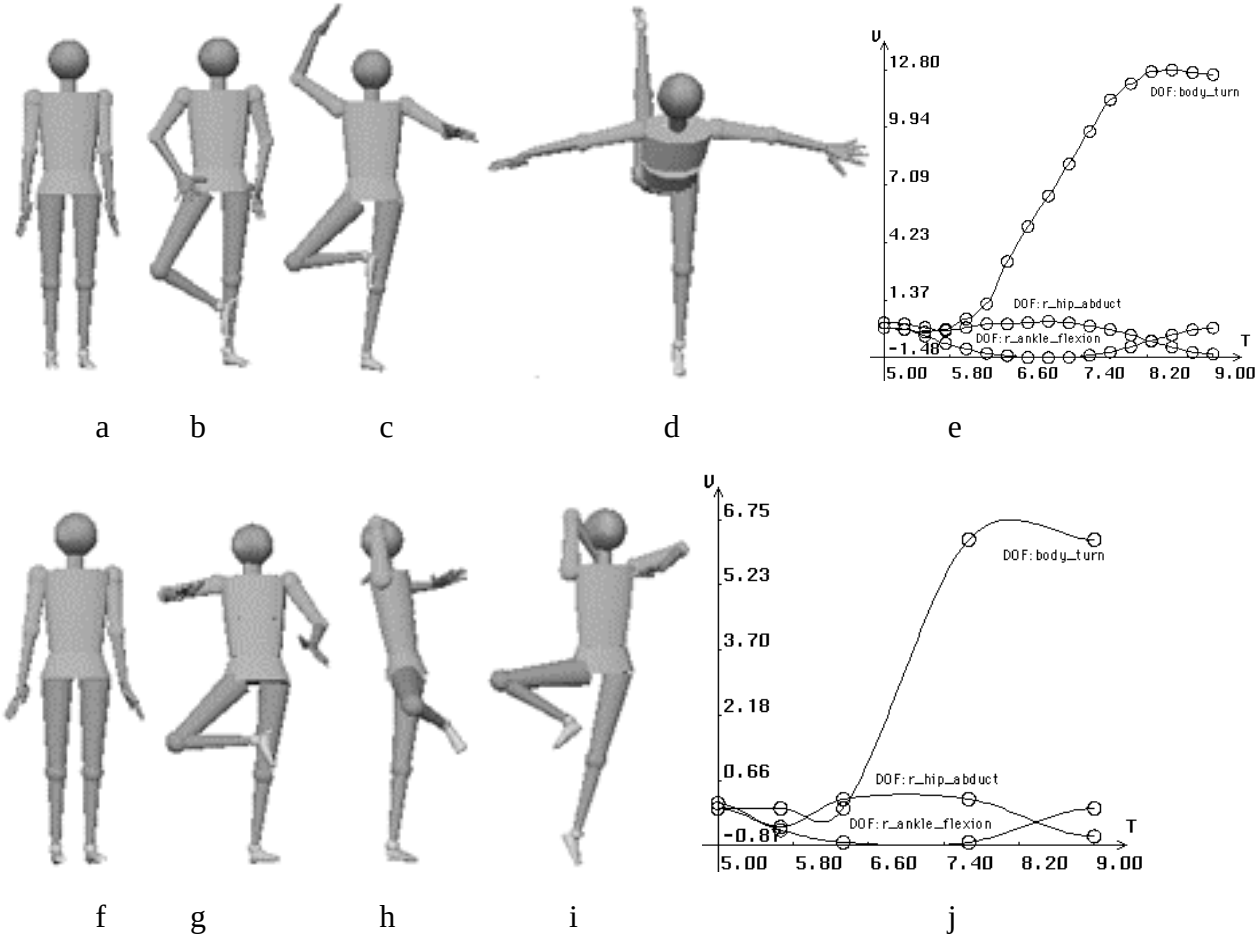
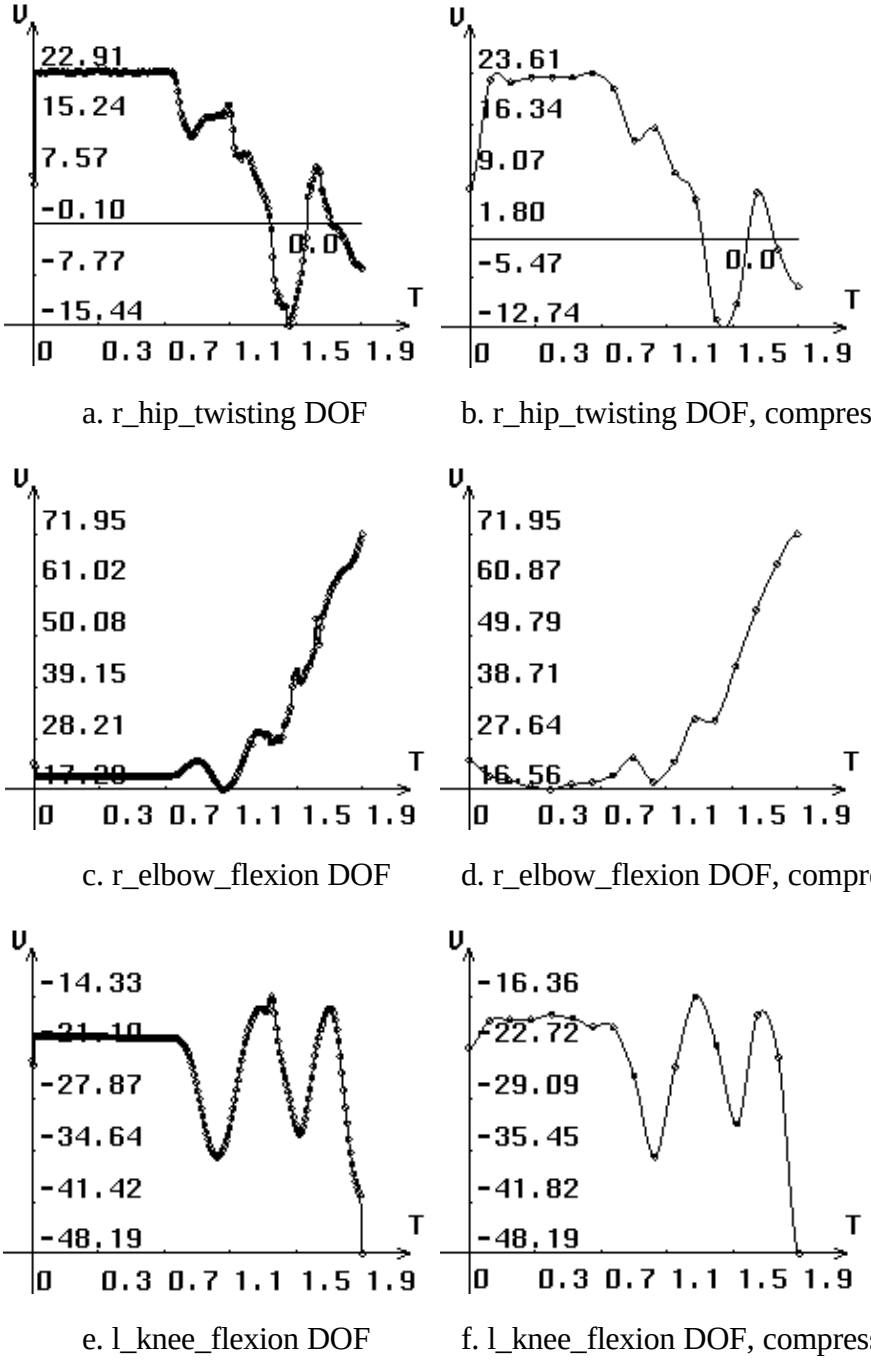


Figure 5-5 example of setting different weights on layers in keyframe re-construction and deriving a new sequence

5.5 Data compression from multiresolution pyramid structure

Next we present the application of multiresolution filtering on sequence data compression. We illustrate it with an example: to compress the key frame data captured using VR device: Flock of Birds of Virtual Technologies (We describe the motion capture technique in a later chapter). It samples 198 frames in 1.97 seconds.

We first construct the Gaussian and Laplacian pyramids with only four layers so that the G_0 and L_0 contain $17(2^4+1)$ samples. We reconstruct the sequence from the four layered pyramids, so the final sequence contains only 17 samples. By this manipulation, the higher frequency components are filtered out. They often represent noise or over detailed movement of the sequence. The result is shown in Figure 5-6.



a. r_hip_twisting DOF b. r_hip_twisting DOF, compressed

c. r_elbow_flexion DOF d. r_elbow_flexion DOF, compressed

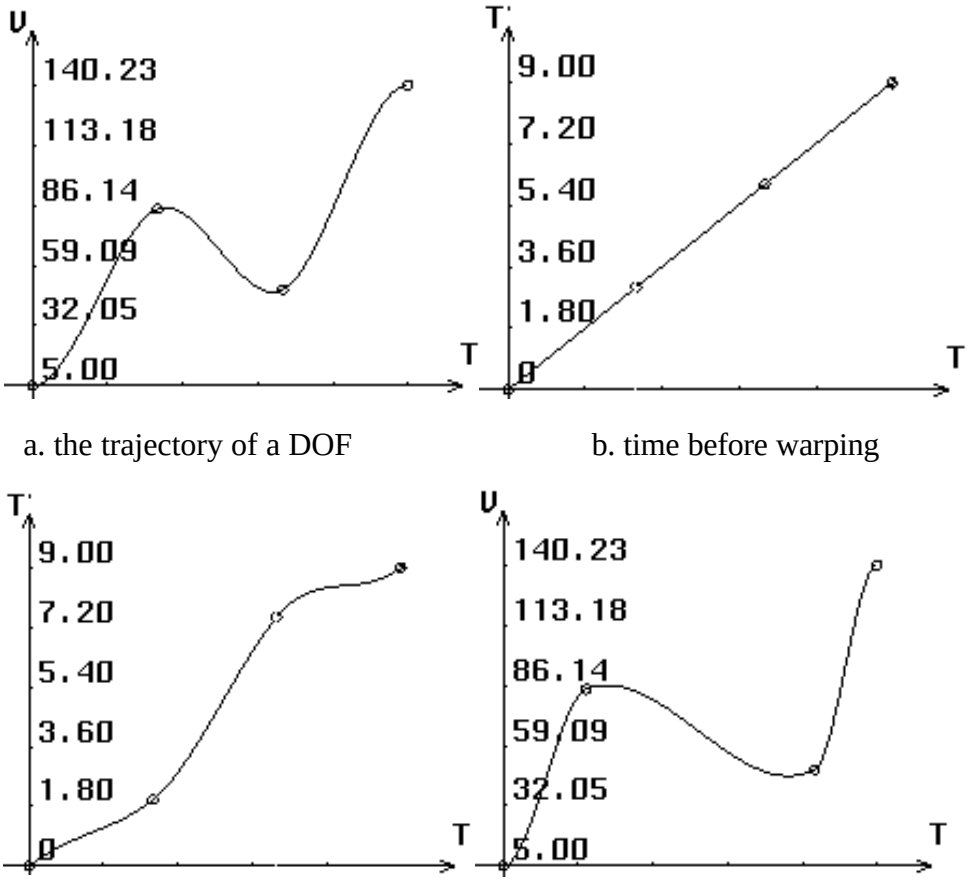
e. l_knee_flexion DOF f. l_knee_flexion DOF, compressed

Figure 5-6 data compression from Gaussian and Laplacian pyramids. The trajectories of only 3 DOFs are shown

This method can compress data significantly while retaining the major features of the original motion signal. The greater the number of layers used to construct the pyramids, the better the approximation with less compression.

5.6 Interactive time warping

We present interactive time warping for key frame sequence. Time warping is a technique used in the field of speech recognition to compare templates, e.g., phonemes, syllables or words, with input utterances [Demori and Probst, 86]. As we represent the motion signal with key frames (q_i, t_i) for each DOF i, the time warping problem can be defined as to allow a set of (t'_j, t_j) acting as time warp constraints, each giving the time t'_j to which the value of original associated with time t_j should be displaced. In [Bruderlin and Williams, 95], the time warping problem becomes the vertex correspondence problem and solved with Sederberg's shape blending algorithm [Sederberg and Greenwood, 92]. We implement the interactive time warping by using cubic Hermite splines to represent (t'_j, t_j) as shown in Figure 5-7 b and c. The animator can interactively select a key point on the curve and modify it. Finally, an acceptable result can be incorporated in the former sequence. In Figure 5-7 a, we show one trajectory of a DOF before warping. The result of warping is shown in Figure 5-7 d. The interactive time warping can be used before sequence merging so that the two sequences have the correct time steps in the merging period.



a. the trajectory of a DOF b. time before warping

c. interactive time warping d. the trajectory after time warping

Figure 5-7 an example of interactive time warping

Another operation on time is "squeeze and stretch". For the current key time T there are two limiting key times TL and TR with TL<T<TR. If T is decreased, all the key times between TL and T are squeezed and all the key times between T and TR are stretched.

This operation is enforced by maintaining the appropriate initial ratio: for Ti < T: (T - Ti) / (T - TL), or for Ti > T : (Ti - T) / (TR - T).

5.7 Keyframe: cubic Hermite splines

We now leave the multiresolution filtering for the direct use of cubic parametric splines. The mathematical splines are used for key frame interpolation. We choose cubic Hermite splines instead of cubic B-Splines to take advantage of Hermite splines that directly use of endpoints and their tangent vectors:

$$Q(t) = (t^3\ t^2\ t\ 1)\cdot M_H \cdot G_H = (t^3\ t^2\ t\ 1)\cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{pmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{pmatrix}$$

Eq. 5-10

where t is the parameter, M_H is Hermite basis matrix, G_H is Hermit geometric matrix, P_1 and P_2 are endpoints with R_1 and R_2 as their tangent vectors [Foley et al, 90].

Key frames of one DOF are represented in pairs (q_i, t_i), $i = 1,2,\cdots,n$ and n is the total number of frames. Thus it can be piecewise represented by 1-D cubic Hermite spline with time t as the parameter. The tangent vector of end points q_1 and q_n can be set default as 0 for start slowly / end slowly. The intermediate tangent vector can be calculated by default from its two neighboring value points. See Figure 5-8.

The axis **V** represents the key values (in degree) and **T** the time (expressed in seconds). The small circles represent the key values for their interpolation. The user can easily modify the motion by editing the spline as shown in Figure 5-9.

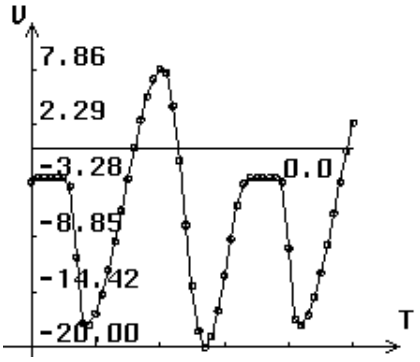


Figure 5-8 key frame interpolation with cubic Hermite spline

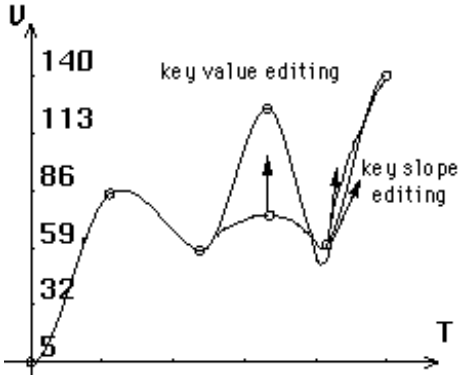


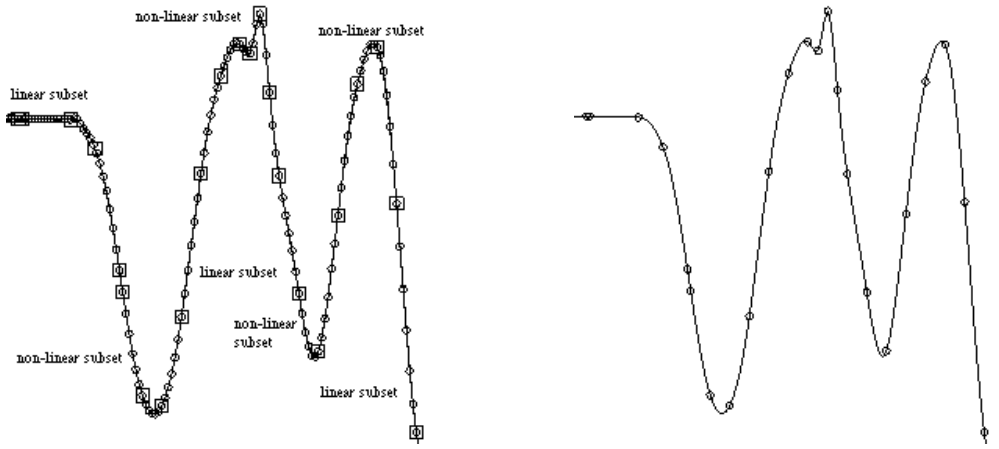
Figure 5-9 interactive editing on the spline

Hermite splines are not completely contained in the convex hull like B-splines and Bézier splines. Thus the interpolated result could be far from control points in some cases. This problem can be tackled by interactively editing the tangent (slope) of the curve.

5.8 Data compression by directly using spline

In this section, we propose the data compression method [Boulic et al., 94, Boulic and Thalmann 92] by directly using the cubic splines. As stipulated by the Shannon Theorem to guarantee no loss of information when sampling a signal over time, the sampling time step can be very small. For most of the motions performed by a human being, a value of 0.01 seconds fulfills this condition and for a large variety of every day motions the video sampling rate is still sufficient (depending on standards: around 0.08 seconds). As a direct consequence, the number of sampled key values for each trajectory is often penalized by further manipulations of the sequence. For this reason, we provide a compression filter to minimize the number of key values using the Hermite spline interpolation.

The basic principle is to discard any linear subset from the range of approximations and then apply a recursive Hermite spline approximation over the remaining non-linear subsets. The underlying idea is that a linear subset conveys a higher level information which structures the motion, as shown in Figure 5-10.



a. illustration of the linear approximation on the linear subset and spline approximation on the non-linear subset. The circles represent the original samples while the cubes represent samples from the approximation

b. replacing the original samples with the samples from the approximation when the animator is satisfied with the results

Figure 5-10 illustration of the data compression by using spline

Let's list the flow chart of the algorithm. Suppose S0 is the set of all points of the trajectory and S1 is the initially empty compressed trajectory. Here is the outline of the algorithm:

---

(1) S1 is initialized with the first control point of S0;

(2) Scan S0 to establish the longest linear subset of key points;

(3) If its last key point is the end of S0 then stop, otherwise add it to S1;

(4) Scan S0 to establish the longest non-linear subset of key points;

(5) Add its last key point to S1 and apply the recursive Hermite spline approximation on it if it contains at least one in-between key point;

(6) Go to step 2. Continue until the end of S0 is reached.

Table 5-1 data compression flow chart

The recursive Hermite spline approximation evaluates an error criteria over the non-linear subset. S1 is validated if the normalized integral error value is smaller than a predefined constant. Otherwise it is divided into two parts and the algorithm is recursively applied.

There are two approximation criteria in this algorithm. The first is the linear criteria, used to separate the initial point set into linear and nonlinear subsets. A key point is in the linear case if: its slope, the slope of the previous key point and the slope of the cord linking the key point to the previous key point are within a normalized neighborhood. The second criterion is used to evaluate the quality of the cubic approximation on the current non linear subset. It first measures the area between the sampled curve and the cubic approximation with a Simpson integral and finally normalizes it by the subset duration.

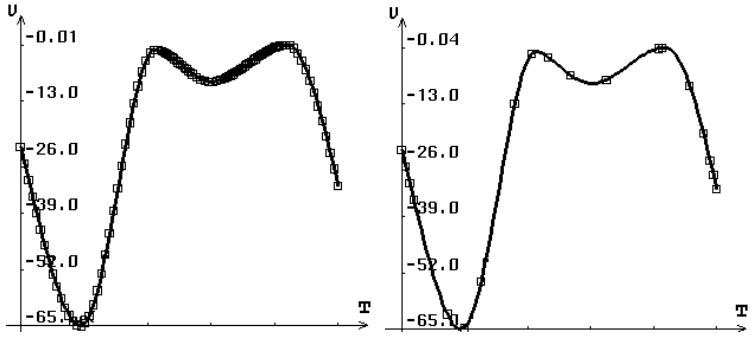
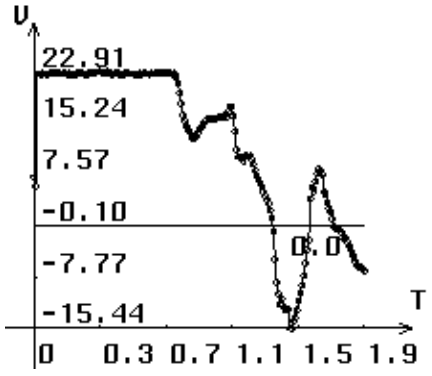
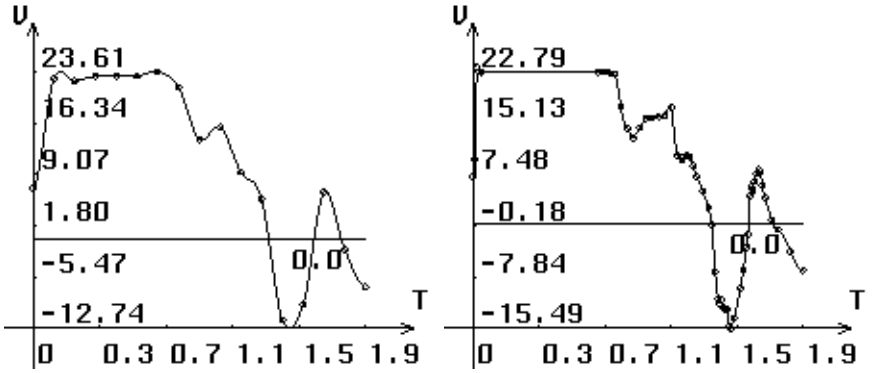


Figure 5-11 example of compression of a sampled curve with default criterion values



a. same as Figure 5-6 a



b. same as Figure 5-6 b (resulting from multiresolution filtering)

c. compressed by cubic splines

Figure 5-12 using the same example in Figure 5-11 to compare the compression by two methods: the cubic spline method results in better approximation

5.9 Conclusion

In this chapter, we proposed a motion-blending method by using the multiresolution pyramid representation, and a sequence-data-compression method by directly using the cubic splines. We have implemented other important sequence manipulation methods: frequency band editing, data compression from pyramids and interactive time warping. We also described the direct interactive editing techniques on the key frames.

This method could be continued in the reuse of sequences. One of the interesting problems is how to adapt a sequence to new spatial and temporal constraints, or how to customize a sequence. A more general subject is motion recognition that is interesting for virtual reality applications.

6. Integrated Human Animation System: Track

So far we are still in the discussion of special motion control techniques for an articulated human figure. In this chapter, we describe a motion control software called Track. We start by a system overview, then continue the discussion on topics of user interface, input / output, system configuration, major functions, 2-D / 3-D interaction with VR devices, and the software integration.

6.1 System overview

Track is an integrated motion control software for articulated human figures. The main features of Track are:

. integration of six motion generators for multiple actors: keyframe, inverse kinematics, dynamics, walking, grasping and capturing. These six motion control methods are most popular for human animation which can be used separately for multiple actors, one actor walking, another grasping object, etc. or for a single actor during different periods. A set of keyframe sequence manipulation techniques can be applied to the resulting motion that produces a more complex sequence, e.g., blending of walking and jumping to have a smooth transition between two movements.

. integration of the deformation module: the skin of human model can be deformed in motion control process that allows animator to see and control both motion and shape at the same time. Integrating complex shape models with motion control in the same platform is one of the most specific features of Track.

. handling the collision detection and response including self-collision detection. The correction is automatically done in real time.

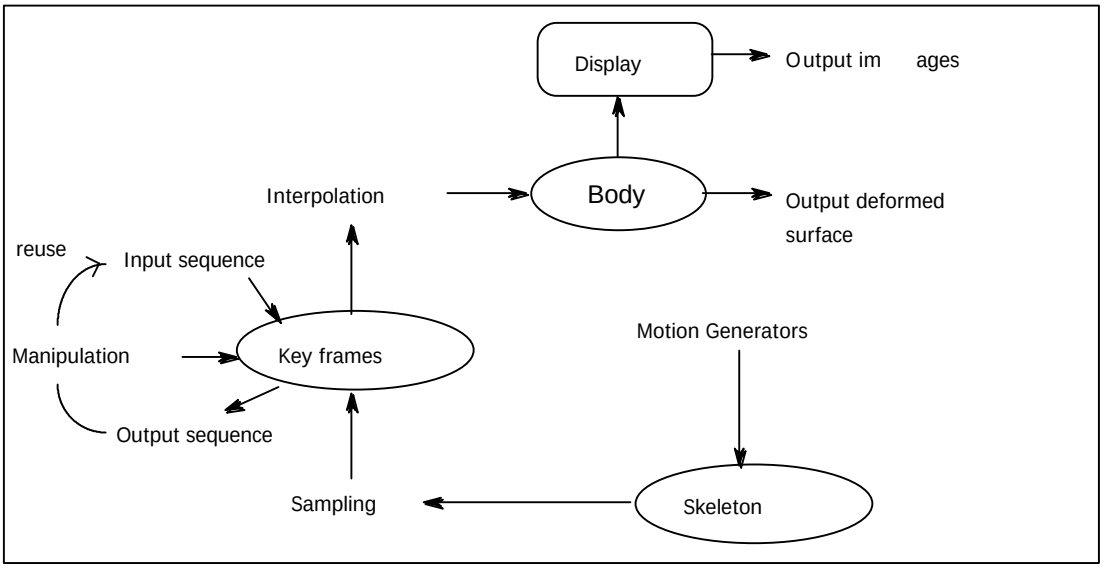


Figure 6-1 the Track working circle

6.2 User interface

First we show the layout of Track. It contains four major parts,

graphic viewer: displays the animation of human figure with different models (segment, solid, envelope, skin) and environment; allows the mouse interaction such as changing view point and picking a DOF from the model;

curve viewer: displays one or more key frame trajectories of DOFs; display the pyramids of multiresolution filtering; allows the mouse interaction like picking and dragging;

H3D viewer: displays the topology of each articulated figure; indicates types of each node with different colors; allows interaction of using mouse;

control panel: allows user to work on motion control through widget operation; displays the status of each operation, e.g., error message, prompt.

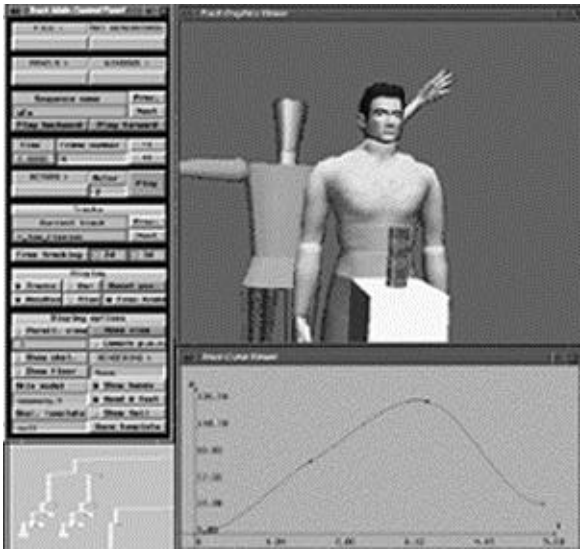


Figure 6-2 Track layout: graphic viewer, curve viewer, H3D viewer and control panel

The user interface is configured with 5D LIG Toolkit [Turner et al., 90]. Figure 6-3 shows the relationship of classes. We use the 2-D class subset of this toolkit to design interface and manage the interaction by handling events in TRACK. It is facilitated by the use of the dedicated interface builder First_step. The interface layout can be built out of multiple windows, hierarchical panels, and many input-output items such as buttons, radios, 1-D and 2-D sliders, toggles, text and browser widgets, menus, etc.

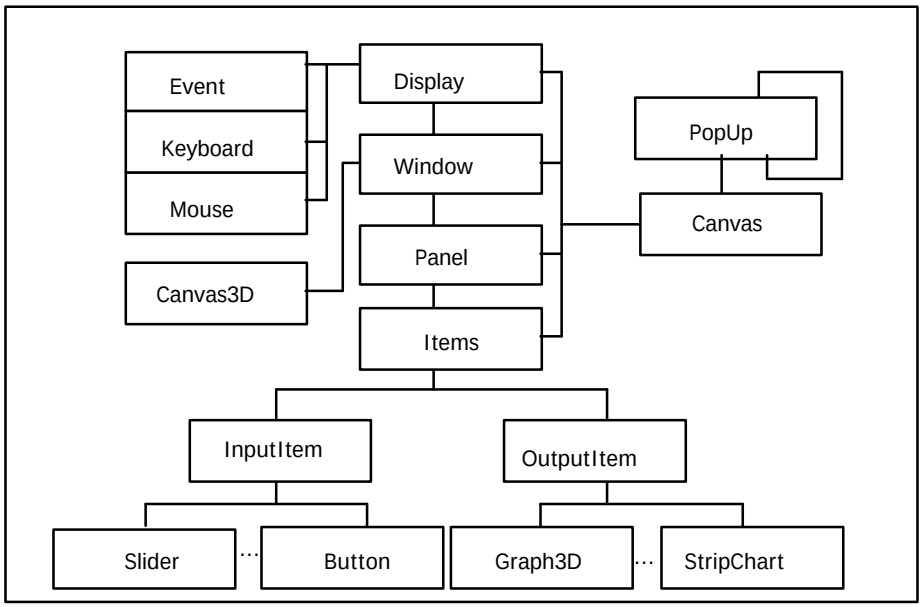


Figure 6-3 object-relation diagram of the 5D LIG Toolkit

Object instances in the Toolkit communicate with each other by sending messages. The event message is a formal type of message in order to respond to user-generated input and implement the dynamic behavior of objects. All event messages have two parameters which specify the source and destination objects. This mechanism has several advantages: first, one can query the source to get additional information about the event. Secondly, this strict form of event messages allows events to be manipulated as objects, and the distribution of events to be specified at run-time. The 5D Toolkit also integrates recent 3-D input devices such as the Spaceball that are very important for human animation.

6.3 Input / Output

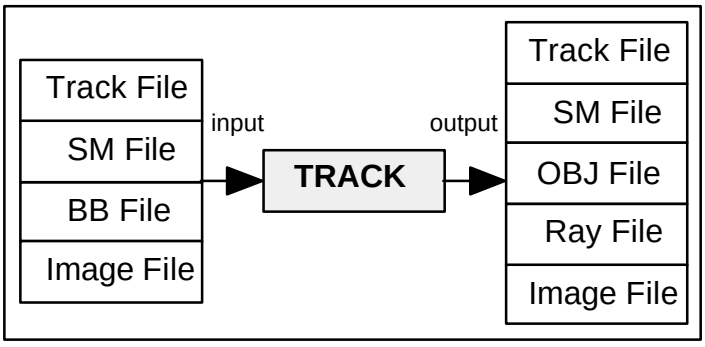


Figure 6-4 Track input / output

There is only one intrinsic file format with the suffix .TRK after the file name. The Track file represents the key frames of each DOF of the articulated figure. Track outputs it as a result or inputs it for further refinement.

There are other types of files which are of other software but which Track supports. The first one is the body model file of Body Builder (BB) [Shen and Thalmann, 95]. Based on the Skinlib of BB, Track supports input body model file that allows motion control directly on the free form surface human model. The second one is the surface model file of Surface Modeler (SM) [LeBlanc et al., 91]. Based on the Smlib of SM and Skinlib of BB, Track inputs the file to model the object for grasping and the environment, and outputs one for each body at each frame as the animation result. Other two types are the object file of Alias | Wavefront [Ramey et al., 95] and the ray file of Rayshade [Kolb, 92]. Track outputs these two types of files one for each body at each frame for the final rendering. The final one is an image file in SGI image format that can be input for texture mapping or output as a result, one file for each frame. We mention all these different systems again in the integration section of this thesis.

6.4 Main functions

We now discuss system functions. We can see that Track is an interactive human animation software with both basic and high level motion control for articulated figures. It is also an interactive tool for the visualization, editing and manipulation of multiple sequences. A sequence is associated with an articulated figure. The system provides a set of tools for motion control and manipulation. This approach allows an incremental refinement design combining information and constraints from both the configuration space (usually joints) and the Cartesian space. We dedicate this system to the design and evaluation of human motions for the purpose of animation. For this reason, we also ensure the real-time display of the 3-D figure motion with a simultaneous scan of the 2-D trajectories. Figure 6-5 lists the main functions. Most of them have been presented in the previous chapters. Thus we only focus on the remaining topics.

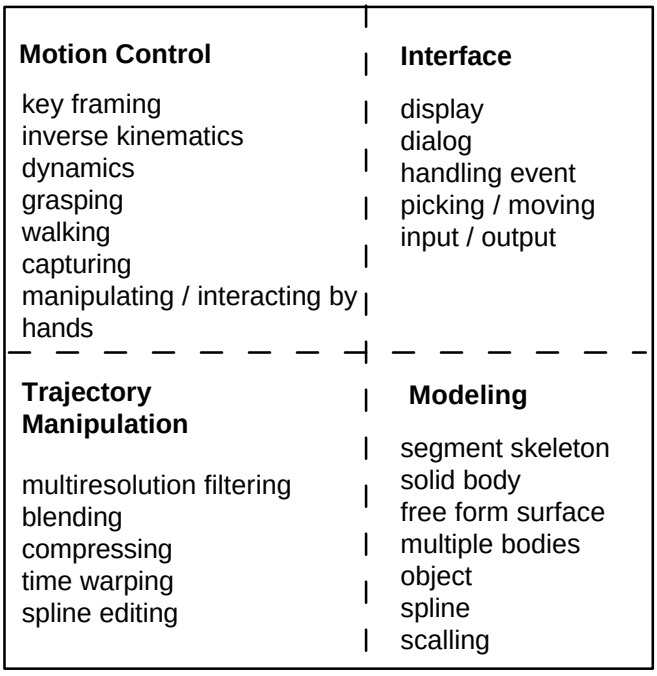


Figure 6-5 main functions of Track

6.4.1 Motion capturing

In this section, we present the VR devices to capture motion directly from a real human to whom the 6D magnetic sensors are attached. We use the Ascension Flock of Birds device that we describe in section 2.7. Moreover, the development of the real-time Anatomical Converter [Molet et al., 96] provides a good interface to Flock of Birds that prevents the application from directly using the low level device driver, as shown in Figure 6-6. The Anatomical Converter is based on a very efficient method to capture human motion after a simple calibration. The sensor data are converted into the anatomical rotations of a hierarchical body representation. Such a choice facilitates a wider use of the motion for other human models with the same proportions. The topology of the body instances of Track and converter are the same, thus Track can get exactly the data it needs.

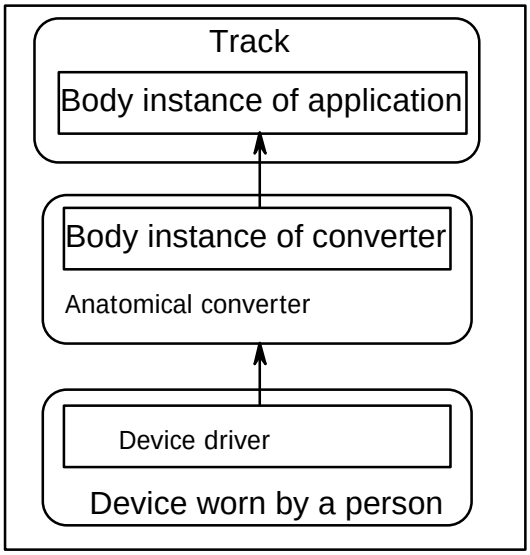
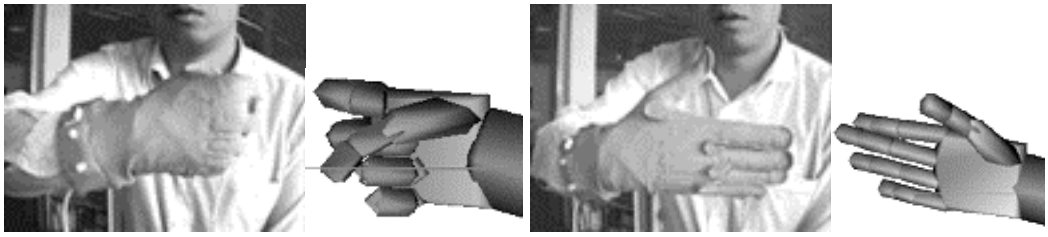
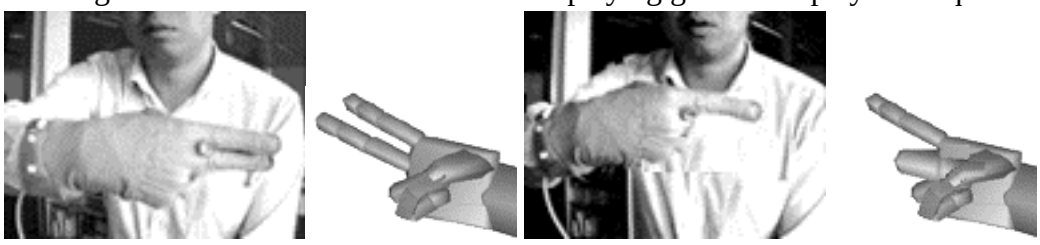


Figure 6-6 Track: derive data from the Anatomical converter of 6D sensors

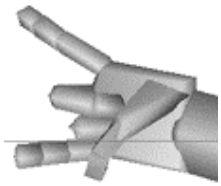
The second device to use is the Virtual Technologies CyberGlove. This is used to input the gestures that replace the 2-D mouse / keyboard interaction. Thus the performers attached with Flock of Birds can capture their movements, which is impossible using only a 2-D interface. There are three basic command gestures, b, c, d and e of the Figure below. The default gesture is shown in a. This is used as a neutral gesture.



a. default gesture b. playing gesture: to play the sequence



c. sampling gesture d. stop gesture for sampling



e. FOB calibration gesture

Figure 6-7 gesture commands from using the CyberGlove

The Figure below shows the 3-D interaction with the CyberGlove. In the data sampling loop, we turn off the display in order to use the maximum sampling rate of the device (100 HZ for the Flock of Birds). We use the change of hand posture instead of the posture itself to trigger a command to avoid redundant reset and calibration process.

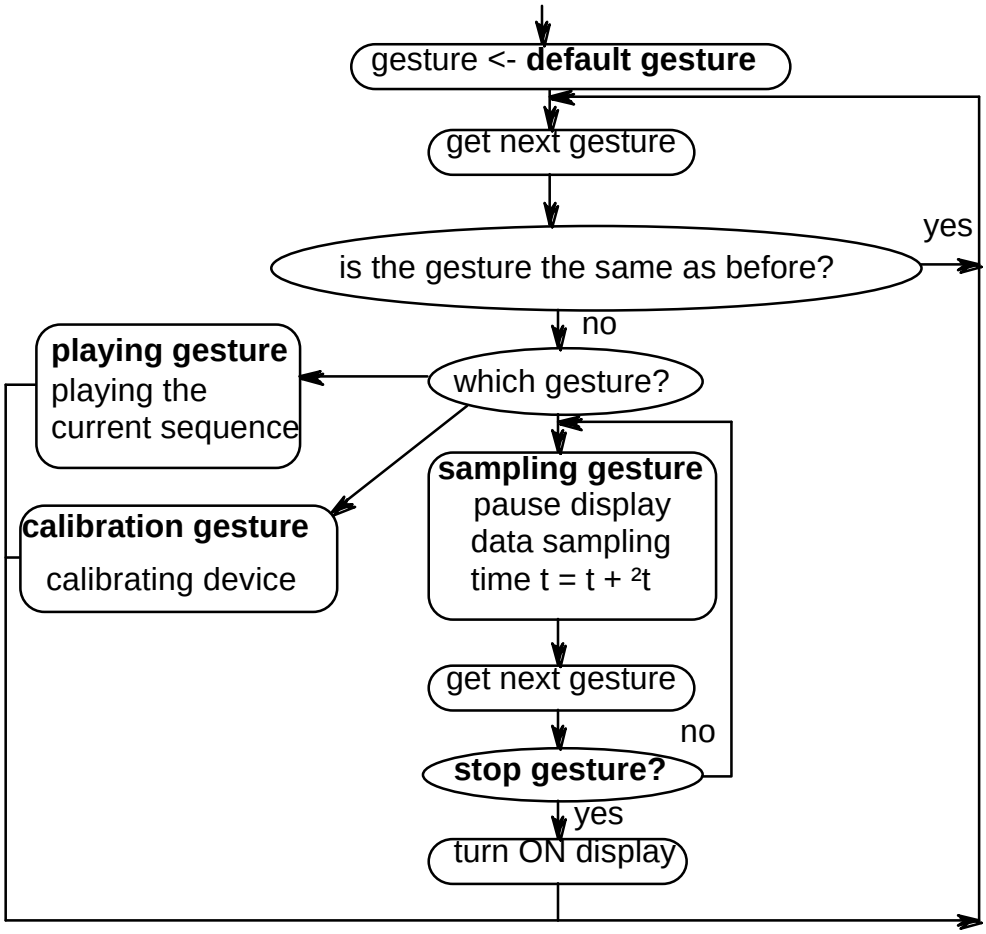
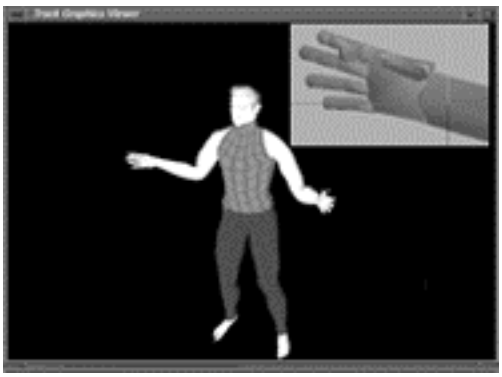


Figure 6-8 basic 3-D interaction with gesture commands

a. snapshot 1

b. snapshot 2

c. snapshot 3

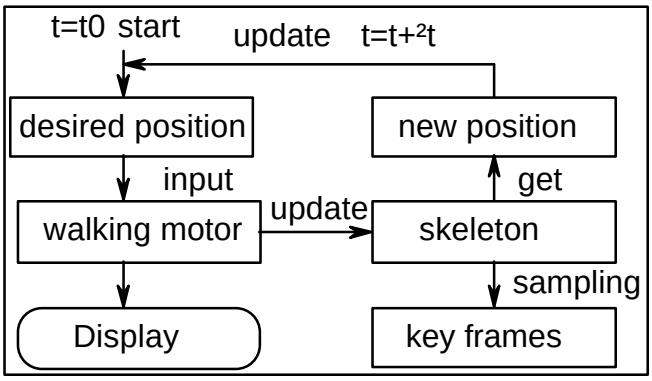


d. the display

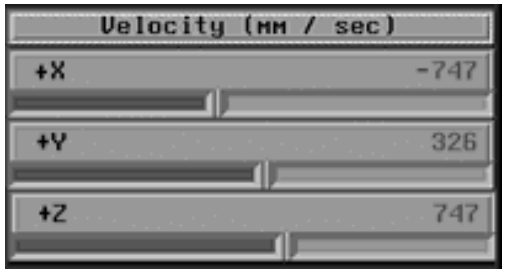
Figure 6-9 motion control using the VR devices, the graphic display in d can also use big video projection screen for better viewing

6.4.2 Walking

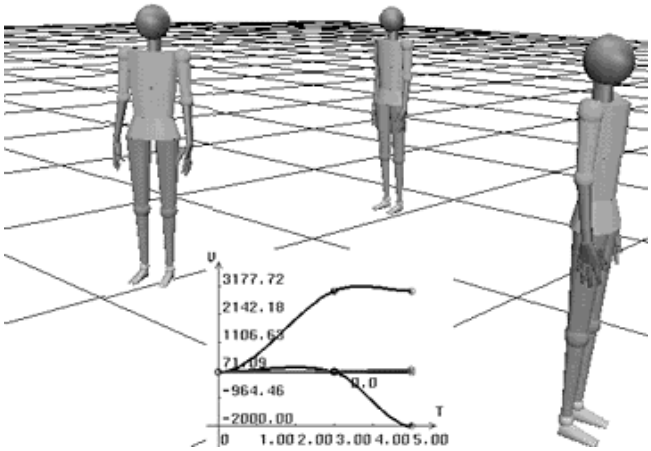
Track integrates the walking motor [Boulic et al., 90] as another motion generator (see also section 3.6 for a brief description). Here we stress on its integration in Track. The walking function is driven by continuously providing the new position vector of the skeleton as shown in Figure 6-10 a. Track supports two ways: to interactively define walking velocity (see Figure 6-10 b), or to use an existing keyframe sequence of one 3-D DOF. By interpolating its key frames, the position vector can be derived for the walking function. Figure 6-10 c shows an example in which the user first sets three key frames for DOF *Global_motion*, that is, a 6-D DOF describing the skeleton position and orientation. The curve shows the interpolation result that is used to provide the desired position for the walking motor. The final result is shown in Figure 6-10 d.



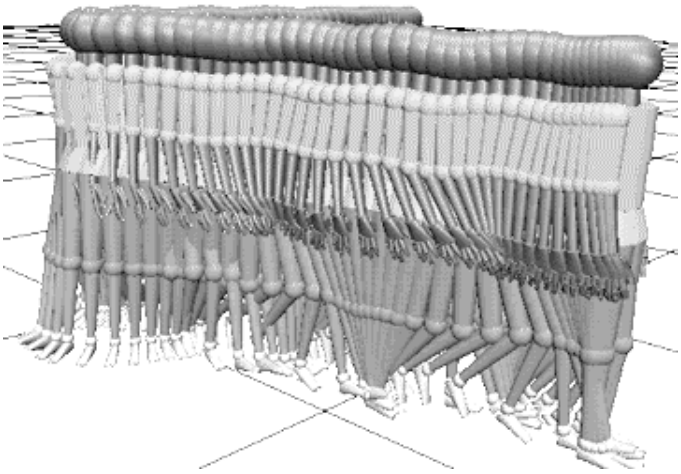
a. walking motion control



b. interactively set walking velocity

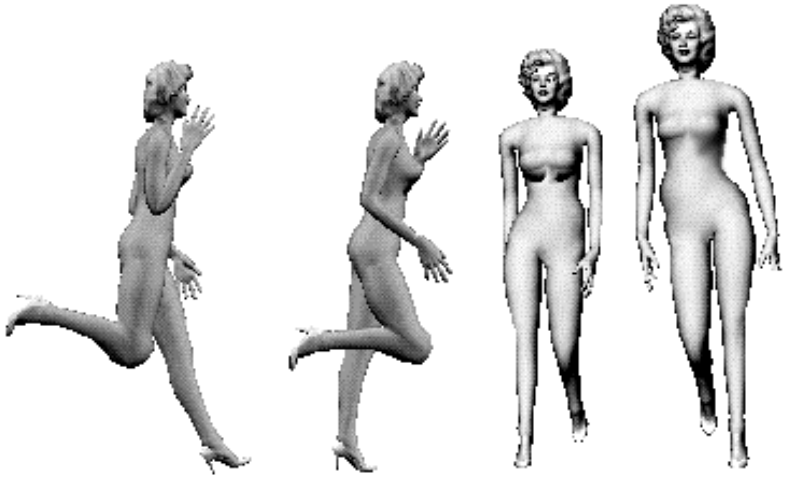


c. three keyframes on DOF Global_motion and their interpolation



d. walking sequence resulting from updating position by interpolating key frames of DOF *Global_motion*

Figure 6-10 walking motion control in Track



a **b** **c** **d**

Figure 6-11 motion resulting from modifying the personalization parameters of walking function in Track. In a and b, it is close to running. In c and d, the movement of pelvic is excessive

6.4.3 Grasplib

The implementation of the Grasplib is in the framework of AGENTlib [Boulic et al., 96]. AGENTlib provides autonomous virtual humans with the skills necessary to perform stand-alone role in films, games and interactive television. In such a system, the autonomous humans can move, react in an environment based on their visual and touching sensors.

A Grasplib is an element (motion generator) of ACTION that provides one type of the action calls in the animation loop of the AGENTlib application. An ACTION is defined as an entity where the goal is clearly identified from the local knowledge at its initial state. Figure 6-12 shows a standard grasping ACTION template.

GRASPING

identificator: a unique string

status: initiating, active, terminating, completed

entry point to AGENT

set of standard function entry points: create, delete, initialize, perform, get, set, stop, etc.

set of special functions: for object, hands, step, etc. specially for grasping

time: begin time, end time

transition management: full initial, full termination

type, typed data entry point: SA_GRASP

SCOPE: right / left arms, hands

Figure 6-12 a grasping ACTION template

An action can be produced from more than one motion generator running in parallel. Thus each generator can not directly modify the DOFs of the articulated figure. It works on a SCOPE (see Figure 6-13). A SCOPE is defined as a subset of DOFs constructed from the full set DOFs articulated figure. The biggest advantage of using the SCOPE is that different motion generators can run in parallel. It is also efficient as motion generators usually influence only a few DOFs.

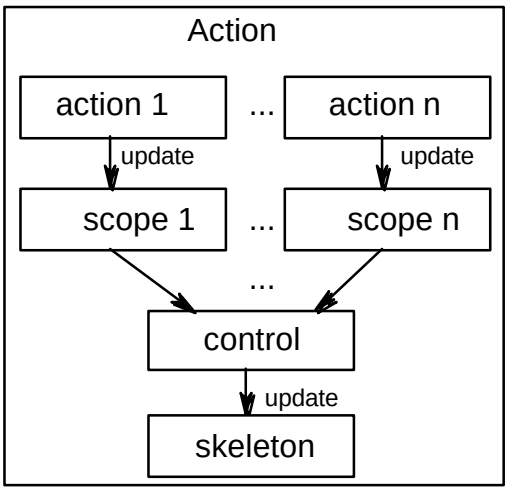


Figure 6-13 ACTION and GENERATOR in animation

The control updates the skeleton according to the priority definition of a generator:

default: it only can start once the previous one that shares the same SCOPE totally/partially stops;

weighted: it can perform with others that share the same SCOPE, weight being between 0 and 1;

exclusive: it can start at its planned time and force others performing on the same SCOPE to stop.

clearing: similar to *exclusive* except that it stop all, even those on different SCOPEs.

In this framework, different actions can be performed in parallel if they are of different SCOPEs or if they are on the same SCOPE but with non-exclusive weights.

6.4.4 Multi-actor support

Track directly supports motion control on multiple actors (see Figure 6-14). It is done by defining an actor data structure and functions over the body that manage all the information of the actor, besides its hierarchical 3-D (H3D) data.

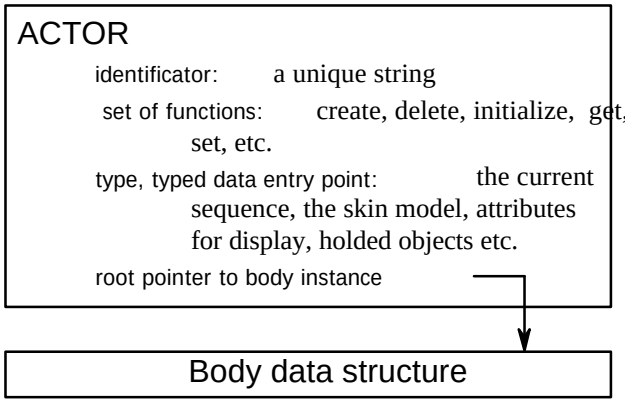


Figure 6-14 actor data type to support multiple actors

At any time, only one actor is active, and control and manipulation are applied to this actor. The interaction of actors with their hands described before works on two actors at the same time. During motion playing, either only the active actor's sequence or all actors' sequences can be played.

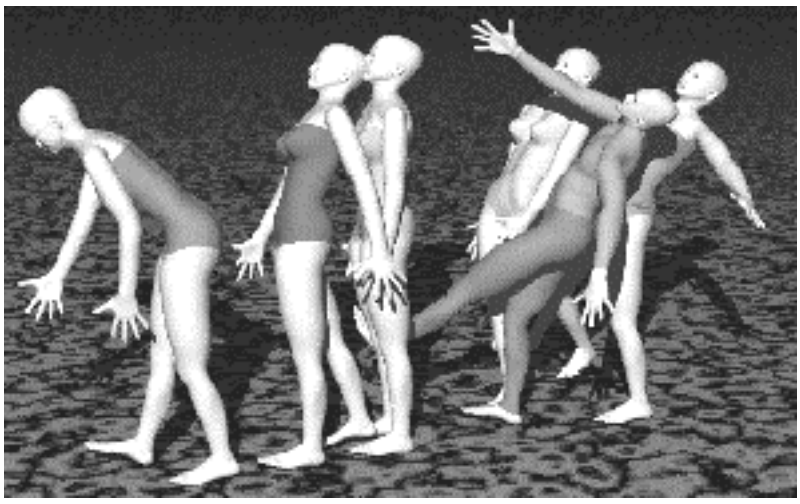


Figure 6-15 one example of animating multiple actors

6.4.5 Human motion representation

Previous work using tracks in computer animation are in [Fortin et al., 86], the MUTAN system. MUTAN (MUltiple Track ANimator) is an interactive system for independently animating 3-D graphical objects with synchronization of different tracks.

The same principle is used for our multiple track sequence where all the key values belonging to the tracks point to a common time-base. Basically, a sequence is associated with all or part of the 3-D hierarchy, but it can also work on a subset of special pre-defined variables. JOINT `node_3D` and FREE `node_3D` mobility information can be accessed to define key values or modified when interpolating them.

A key value list is maintained within a track. It stores the value information and the first derivative with respect to time. The user can always modify the value, the associated time and the slope for any key value, resulting in great freedom of motion design.

Moreover, the track curve display provides a clear kinematic feeling of the motion, with time expressed in seconds. Finally, from the integration point of view, any sampled motion can be recorded, manipulated and easily played back with standard cubic interpolation. The general structure of the multiple track sequence is presented on Figure 6-16. All the structure definitions are implemented in Keyframelib, which also provides a set of functions to manipulate them, e.g., construct, destroy, get, set, interpolate, read and save.

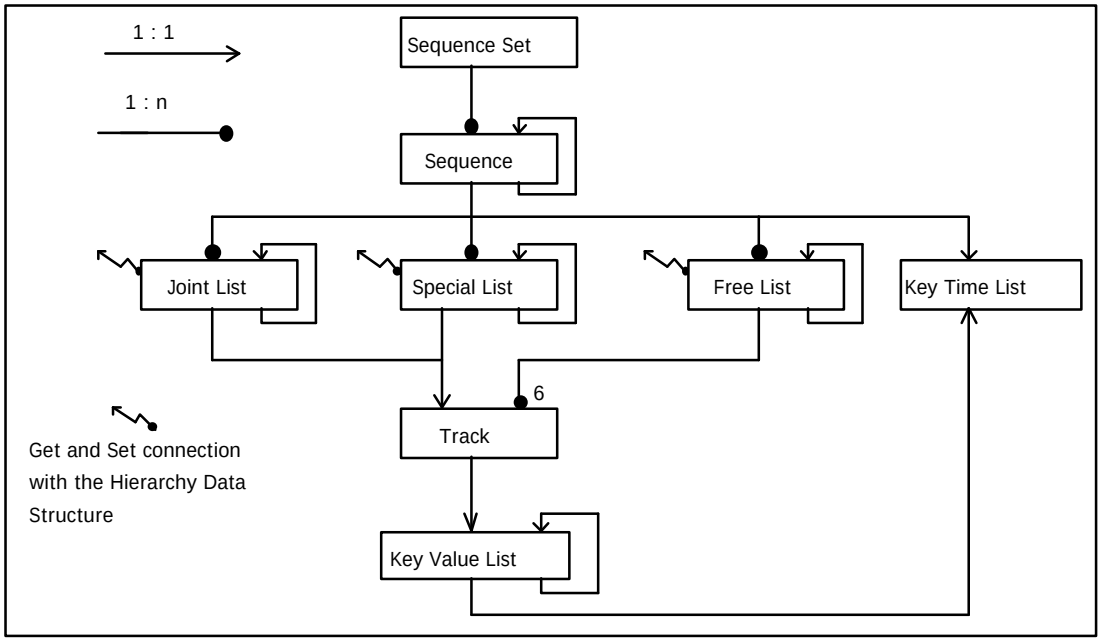


Figure 6-16 general structure of a multiple trajectory sequence

The Sequence Set holds sequences that represent different motions. The current sequence can be manipulated directly and outputs the interpolated motion to the 3-D skeleton hierarchy. Different sequences can be combined to produce new sequences. Whatever method is used, the motion value and velocity can be recorded as a multiple trajectory sequence, and this constitutes the base of the TRACK system for complex motion design. Then, a large set of tools can manipulate the resulting multiple track sequence. The first is usually a compression filter used to reduce the number of key values to the minimum within a pre-defined error rate, as described in section 5.8.

6.5 System implementation

In this section, we describe the system implementation of Track. The Figure below illustrates different software libraries. Each library is dependent on the libraries under it in the drawing. Amongst them, the Glovelib for the CyberGlove, the Anatomic Converter for the Flock of Birds device and the LIG Toolkit for GUI are described in the previous sections. We continue to describe the remaining libraries.

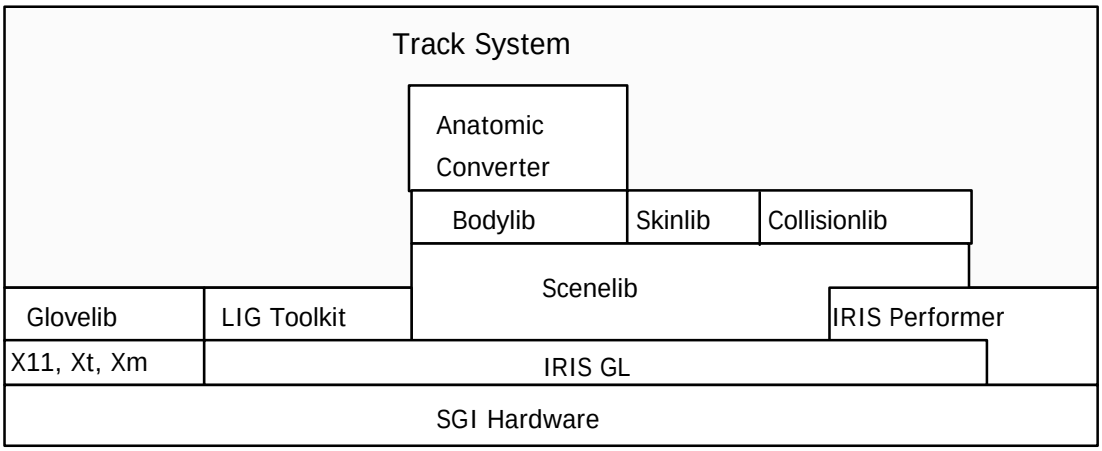


Figure 6-17 Track system library layering

6.5.1 Graphic library: IRIS GL and IRIS Performer

Track is implemented on SGI workstations mainly because of the availability of IRIS Graphics Library (IRIS GL) and IRIS Performer. IRIS GL is a library of subroutines for creating 2-D and 3-D color graphics and animation. Here are some of the IRIS GL features that are used in Track:

·draw graphics primitives such as points, lines, polygons

·draw characters and define fonts

·use color modes and color maps to control the way colors are displayed

·use double buffering to create animated graphics

·perform coordinate transformations

·define and manipulate light sources to create lighted scenes

·use texture mapping to add surface characteristics to geometry

IRIS Performer [Rohlf and Helman, 94] is a software development environment compatible with IRIS GL. It provides high-level support for visual simulation, interactive entertainment, virtual reality, and graphics-intensive tasks. Applications that require real-time visualization and high-performance rendering benefit from using IRIS Performer.

The main components of IRIS Performer are the two libraries libpr and libpf.

·libpr is a low-level library that provides optimized rendering functions, state control, and other functions fundamental to real-time graphics. It provides highly optimized rendering loops for rendering a wide variety of geometric primitives.

·libpf is a visual simulation development environment that adds a multiprocessing database traversal and rendering system on top of libpr. It supports multiprocessing, hierarchical scene construction, multiple channels, culling to each channel's field-of-view, and frame-rate control.

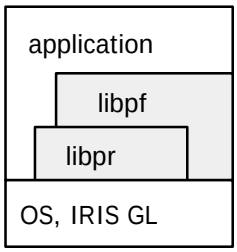


Figure 6-18 software layering: the grayed rectangles represent IRIS Performer

IRIS Performer drastically reduces the work required to tune an application's performance. General optimizations include the use of highly-tuned routines for all performance critical operations and the reorganization of graphics data and operations for faster rendering. IRIS Performer also handles tuning issues specific to Silicon Graphics architecture by selecting the best rendering and multiprocessing modes at run time based on the system configuration.

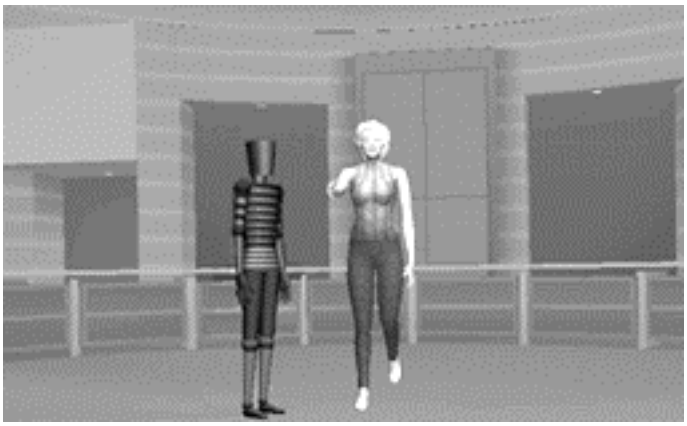


Figure 6-19 real time graphics from IRIS GL and IRIS Performer in Track system

6.5.2 Human modeling: Skinlib

Most of motion control software uses segment or simple geometric volume models. A more realistic model is only used in the final rendering. The animation sequence that looks realistic on a simplified model can be different when a more complex model is used.

One of the important effects for the human body is that the body is deformed during motion. Shen and Thalmann [Shen and Thalmann, 95] present a new layered approach for modeling and deforming human bodies. Volume primitives are employed to mimic the gross behavior of bones, muscles, and fat tissues. An implicitly defined surface by volume primitives is sampled using ray-casting on semi-regular cylindrical grids. These sample points are used directly as cubic B-spline control points to smoothen out the skin surface. Individual B-spline patches are triangulated, and these triangular meshes are stitched together to connect different parts of the human body for final rendering and output. This method is not real time, however, 5 seconds are needed to produce one frame for the whole body. Yet, their recent work [Thalmann et al., 96] simulates body deformation in real time. The new approach is based on contour deformations. First the contour points are calculated in a given position of the body (it is called ‘initial position’). At this point, the B-spline patches need not be calculated for each

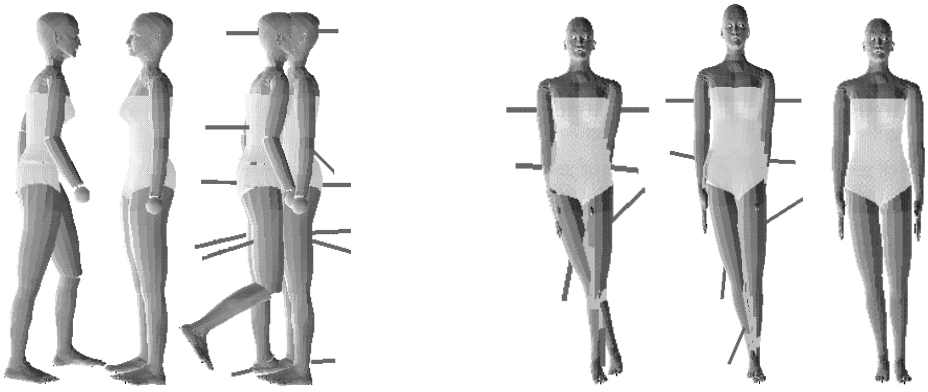
frame in order to triangulate these surfaces. Instead, the contour points are used directly for body triangulation mesh. Furthermore, the direct use of the IRIS Performer in the Skinlib accelerates the visualization process.

Track integrates the Skinlib based on the body deformation described above. Thus the animator can directly visualize and control the motion of a realistically deformed body. The difference from the final video is only on factors such as lighting, decoration and color.

Figure 6-20 realistic human model and deformation in Track

6.5.3 Collisionlib

In this section, we present the integration of Collisionlib developed in European Humanoid Project by our partners of University of Karlsruhe in Germany. The basic ideas of Collisionlib are the usage of a cascade of tests, each computing faster than the next, the exploitation of spatial and temporal consistency, and the possibility for the user to advise the system where to expend its resources: on accuracy or on efficiency. For details of the Collisionlib see [Boulic et al., 95c]. Here we show some examples of using Collisionlib in Track.



(a) (b)

Figure 6-21 Collision detection (the lines shown are response vectors). In context (a) a walking actor is colliding with a standing actor. In context (b) self collisions are detected and corrected according to the direction of correction (evaluated by the collision analysis).

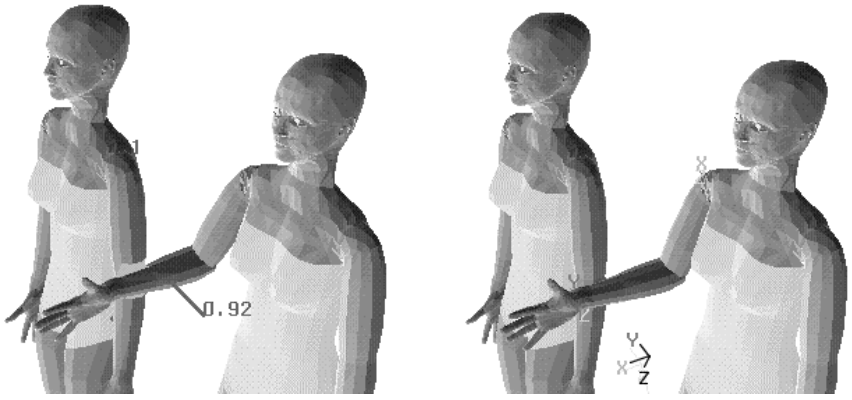


Figure 6-22 Multi-actor collision detection and correction with inverse kinematics, the guiding frame displayed in the right figure is derived from the response vector.

6.5.4 Bodylib

Bodylib [Boulic et al., 95a] provides skeleton templates from one pre-defined topological tree structure for a vertebrate body. From one template, a set of similar body instances can be created with the same topology. It also supports volume and envelopes surrounding the skeleton. For the template, a general mechanism allows the customization of the skeleton structure at two levels, either at a high level with a small set of scaling parameters or at the low level of the position and orientation of the various articulations defined in the SKELETON data structure. In either case the modifications are propagated to the lower level structure of the volume and envelope. In some working context some optional part of the SKELETON may be switched on or off to improve the performance. For example, the hands can be switched off if motion control is on other parts of body.

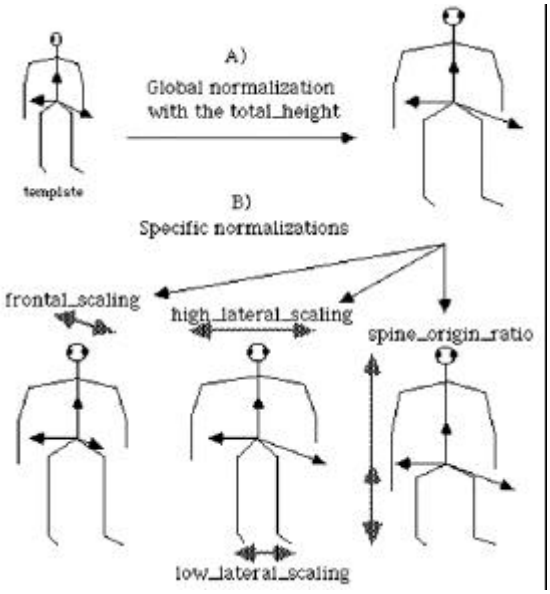


Figure 6-23 illustration of the skeleton templates

Based on Bodylib, Track can rapidly start from the skeleton templates. It also prevents Track from directly managing the low-level configuration. The advantage is the motion control functions can still work when the low-level change happens. Furthermore, Bodylib is also a common foundation for other software that Track can share data with.

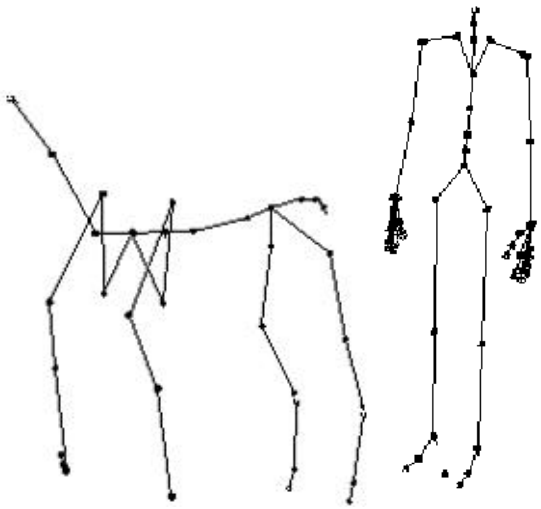


Figure 6-24 two different skeleton templates

The lowest level support of articulated figure modeling and manipulating is Scenelib [Boulic et al, 95b]. We include a detailed illustration in Appendix C to explain Scenelib.

6.6 Examples of applications

The first example is the project to animate Xian terra-cotta soldiers [Magnenat-Thalmann et al., 95]. A short film is made on the Xian terra-cotta soldiers using our integrated HUMANOID software including Track system. First, the soldiers' faces are created by the sculpturing software SM. Then, Track imports these face models and

associated with body models supported by the metaballs and spline surface Skinlib [Shen and Thalmann, 95]. The terrain is also imported as a reference for soldiers' locomotion. Now, the full set of motion control tools and sequence manipulation tools are used to produce the sequence for each soldier. Finally, clothes [Volino et al., 96] for the soldiers are then described as well, and horses, decor design have been added. All of them are rendered by the strategy using parallel machines [Pandzic et al., 95].

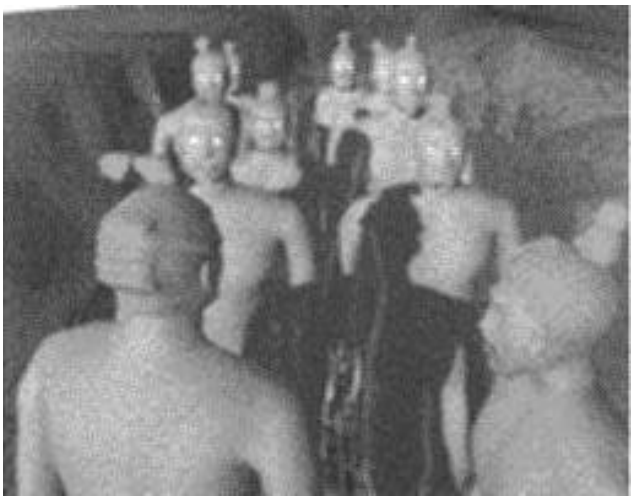


Figure 6-25 one snapshot from the Xian terra-cotta film (MIRALab)

The second example is the application of Augmented Vision to animate the virtual actor in real environment [Magnenat-Thalmann and Thalmann, 95]. The goal of Augmented Vision [Ahlers et al., 94] is to enhance the capabilities of the human visual system through the combination of computer generated graphics, computer vision and advanced user interaction technology. It shares similar techniques with Virtual Reality. In this application, the calibration techniques of Augmented Vision for the video camera, 3-D tracker and environments are used to calculate the virtual camera parameters for the rendering software to render the virtual actors/objects. Finally the resulting images are merged with video signal to have virtual actors/objects in a real world. This technique is applicable in film production, design, construction and manufacturing. Track is used again to animate the virtual actor. The following image is from the short film made in a student's diploma project in our lab.

Figure 6-26 virtual actor in a real environment

There is another similar example with real-time camera capturing, character animation and image blending.

Figure 6-27 virtual actor in a real environment

The next example is the sequence made for the "Golden Camera Award" ceremony which is broadcast by German TV Channel ZDF. MIRALab designers used the full set of Humanoid software as well as the Wavefront | Alias rendering package for rendering.

Figure 6-28 a snapshot from the MIRALab short film for ZDF

The following image is from the short film showing the interaction of two synthetic actors by hands. The lighting, color and decoration are made by the designer of our lab.

Figure 6-28 a snapshot from the short film to show interaction by hands

6.7 Conclusion

In this chapter, we have described a layered motion control system Track that has the following characteristics:

. integration of motion control methods: keyframe, inverse kinematics, dynamics, grasping, walking, motion capture, object manipulation and actor interaction by hands;
. integration of sequence manipulation methods: using signal processing techniques and cubic spline techniques;
. support of different complexities of human models: low level segment model, simple geometric shape and highly realistic skin model;
. support of multiple human models;
. support of other types of articulated figures;

Table 6-1 main characteristics of Track system

We have described a motion control method using the real-time anatomic converter to capture the animator movement in real time. Hand gesture commands are used for 3-D interaction. The captured sequence can be manipulated further.

We have described its user interface and input / output. It proves to be very helpful in its application in animation film production.

For the future work, we can improve the system in different aspects: to run on other platforms besides SGI workstations, the use of IRIS GL should be replaced by OSF OpenGL which is a widely accepted standard for graphics API. Considering the rapid development of PC technology, changing to OpenGL is very urgent. For the same reason, the LIG Toolkit needs to be implemented again by OSF Motif, or Track's GUI needs to be changed to some other OSF Motif-based toolkit. More new VR devices and 3-D interaction techniques need to be further investigated in the animation. More sophisticated motion control methods need to be studied and implemented in the system. Finally, the on going development in the Humanoid 2 Project provides the framework for different motion control working on one model at the same time. This framework needs to be implemented in Track software.

7. Conclusion

We have proposed the use of inverse dynamics in a closed loop with forward dynamics for interactive motion control of a human skeleton. An efficient recursive algorithm based on Newton-Euler formulae is adapted to calculate the force and torque produced by joint actuator in order to accomplish a desired motion. The resulting force and torque are used in forward dynamics to make the final motion with external force and torque. The Armstrong-Green algorithm is used for forward dynamic simulation. Inverse dynamic functions calculate the required force and torque at each small time interval in the process of forward dynamic simulation. In this way, it can correct errors at each time interval.

We have proposed methods for hand motion control: multi-sensor heuristic for grasping, human interaction by hands, object manipulation, self-collision detection and response and CyberGlove grasping. The multi-sensor heuristic for grasping combinatorially uses the basic motion control methods: direct, inverse kinematics and procedural method. Personalized results can be accomplished with different profiles of hand motion patterns created interactively. The resulting arm-hand motion can be further improved by other approaches such as keyframe, signal-processing-based methods and optimal control. We have extended the method to object manipulation and human interaction with hands. A novel method is proposed for the BODY self-collision detection and correction. This method is based on the extension of multi-sensor configuration and the use of the secondary task of inverse kinematics. It is very efficient for the movement free of self collisions. We finally modeled the same multi-sensor structure on the 3-D hand model of the CyberGlove provided from Virtual Technologies, Ltd. Using a method based on grasping automata and hand posture collision response, we have simulated more precisely the interactive grasping by a real person wearing a CyberGlove. It works along with automatic grasping by synthetic actors.

We have proposed methods on keyframe sequence manipulation: motion blending on frequency bands and motion data compression using the cubic splines. We have also implemented other manipulation techniques: interactive sequence editing on both spline curves and the multiresolution pyramid structure, interactive time warping, data compression from the pyramid structure. All these techniques are important for reusing and post-processing the keyframe sequence.

We have designed and implemented an integrated motion control system called Track. It integrates the motion control methods: keyframe, inverse kinematics, dynamics, grasping, walking, motion capture. It can provide the users with a set of methods to manipulate sequences. It supports multiple human models and different complexities of human models: low level segment model, simple geometric shape and a highly realistic skin model. It also supports other types of articulated figures. We described its user interface, functionality, integration and configuration. Finally, we have briefly mentioned some of its applications.

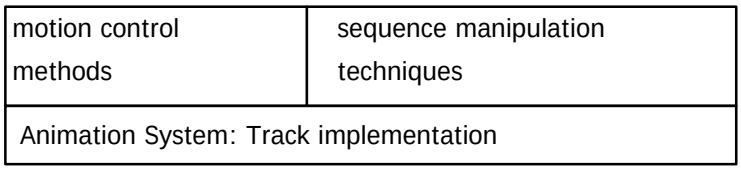


Figure 7-1 overview of the contributions of the thesis: two layers with three main aspects

8. References

[Ahlers et al., 94] Ahlers K. H., Crampton C., Greer D., Rose E. and Tuceryan M., Augmented Vision: A Technical Introduction to the Grasp 1.2 System, Technical report ECRC-94-14.

[Arai, 93] Arai K., Keyframe Animation of Articulated Figures Using Partial Dynamics, Models and Techniques in Computer Animation, edited by Magnenat-Thalmann N. and Thalmann D., Springer-Verlag 1993.

[Armstrong and Green, 85] Armstrong W. W. and Green M. W., The dynamics of Articulated Rigid Bodies for Purposes of Animation, Computer 1 (4), pp. 231-240.

[Armstrong, 79] Armstrong W. M., Recursive Solution to the Equations of Motion of an N-link Manipulator, Proc. of the 5th World Congr., Theory of Machines, Machanisms, vol. 2, pp. 1343-1346.

[Ascension, 94] Ascension, The Flock of Birds Installation and Operation Guide, Ascension Technology Corporation POB 527 Burlington, Vermont 05402 (802) 860-6440, 1994.

[Badler et al., 96] Badler N. I., Webber B. L., Becket W., Geib C., Moore M., Pelachaud C., Reich B. D. and Stone M., Planning for Animation, in Magnenat-Thalmann, N., and Thalmann D. eds., Interactive Computer Animation, Prentice Hall, 1996, pp. 235-261.

[Badler et al., 93a] Badler N., Hollick M. J., Granieri J. P., Real-Time Control of a Virtual Human Using Minimal Sensors, a Forum short paper in Presence, 2 (1), MIT, 1993, pp. 82-86.

[Badler et al., 93b] Badler N. I., Phillips C. B., and Webber B. L., Simulating humans: Computer Graphics, animation, and control, New York: Oxford University Press, 1993.

[Badler et al., 91] Badler N., Barsky B., and Zeltzer D. eds., Making Them Move, Morgan Kaufmann, 1991.

[Badler and Monheit, 91] Badler N. and Monheit G., A Kinematic Model of the Human Spine and Torso. IEEE CGA, 11 (2), March 1991.

[Bandi and Thalmann, 95] Bandi S. and Thalmann D., An Adaptive Spatial Subdivision of the Object Space for Fast Collision Detection of Animated Rigid Bodies, Proc. of Eurographics'95, Maastricht, pp. 259-270.

[Barr, 87] Barr A. H., chair, Topics in Physically-Based Modeling, Course Notes, vol. 16, ACM SIGGRAPH'87.

[Barzel, 92] Barzel R., Physically-Based Modeling for Computer Graphics: A Structured Approach, Academic Press, Inc, 1992.

[Basmajian, 80] Basmajian J. V., Grant's Method of Anatomy, Tenth Edition, The Williams & Wilkins Company, 1980.

[Becheiraz and Thalmann, 96] Becheiraz P. and Thalmann D., A Model of Nonverbal Communication and Interpersonal Relationship between Virtual Actors, Proc. of Computer Animation'96, Geneva.

[Boulic et al., 96] Boulic R., Becheraz P., Noser H., AGENT & ANIMA, An Architecture for the Integration of Low-Level Planification with Motion Control, Technique Report for ESPRIT HUMANOID 2, EPFL-LIG, 1996.

[Boulic and Mas, 96] Boulic R. and Mas R., Hierarchical Kinematic Behaviors for Complex Articulated Figures, Chapter 3 in "Advanced Interactive Animation", Thalmann, D. and Magnenat-Thalmann, N. eds., ISBN 0-13-518309-X, Prentice Hall Europe, 1996.

[Boulic et al., 95a] Boulic R., Capin T., Huang Z., Molet T., Shen J., Kalra P., Moccozet L., Human Data Structure & Parallel Integrated Motion Specialized Hierarchy for a General Vertebrate Body User Reference Manual, ESPRIT Project 6709 Humanoid, 1995.

[Boulic et al., 95b] Boulic R., Capin T., François J., Huang Z., Molet T., Shen J., Kalra P., Moccozet L., Human Data Structure & Parallel Integrated Motion Scenelib V4.0 with IRIS Performer 2, User Reference Manual, ESPRIT Project 6709 Humanoid, 1995.

[Boulic et al., 95c] Boulic R., Capin T., Huang Z., Moccozet L., Molet T., Kalra P., Lintermann B., Magnenat-Thalmann N., Pandzic I., Saar K., Schmitt A., Shen J., Thalmann D., The HUMANOID Environment for Interactive Animation of Multiple Deformable Human Characters, Proc. of Eurographics'95, Maastricht, pp. C337-C348.

[Boulic et al., 94] Boulic R., Huang Z., Magnenat-Thalmann N., and Thalmann D., Goal-Oriented Design and Correction of Articulated Figure Motion with the Track System, Journal of Comput. & Graphics, 18 (4), 1994, pp. 443-452.

[Boulic and Mas, 94] Boulic R. and Mas R., Inverse Kinetics for Center of Mass Position Control and Posture Optimization, EPFL Technical Report No 94/68, 1994.

[Boulic and Thalmann, 92] Boulic R. and Thalmann D., Combined Direct and Inverse Kinematic Control for Articulated Figures Motion Editing, Computer Graphics Forum, 2 (4), October 1992.

[Boulic and Renault, 91] Boulic R. and Renault O., 3D Hierarchies for Animation, New Trends in Animation and Visualization, edited by Magnenat-Thalmann N and Thalmann D, JOHN WILEY & SONS, 1991.

[Boulic et al., 90] Boulic R., Thalmann D. and Magnenat-Thalmann N., A Global Human Walking Model with Real-Time Kinematic Personification, The Visual Computer, 6 (6), December 1990.

[Boulic, 86] Boulic R., Conception Assistée par Ordinateur de Boucles de Commande avec Capteurs en Robotique et en Téléopération, Thèse présentée devant l'Universite De Rennes I, U.E.R Mathématiques et Informatique, 1986.

[Brotman and Netravali, 88] Brotman L. S. and Netravali A. N., Motion interpolation by optimal control, Proc. of ACM SIGGRAPH'88, pp. 309-316.

[Bruderlin and Williams, 95] Bruderlin A. and Williams L., Motion Signal Processing, Proc. of ACM SIGGRAPH'95. pp. 97-104.

[Bruderlin and Calvert, 89] Bruderlin A. and Calvert T. W., Goal-Directed, Dynamic Animation of Human Walking, Proc. of ACM SIGGRAPH'89. pp. 233-242.

[Burt and Adelson, 83] Burt J. P. and Adelson E., H., A Multiresolution Spline With Application to Image Mosaic, ACM Transaction on Graphics, 2 (4), Oct. 83, pp. 217-236.

[Burtnyk and Wein, 71] Burtnyk N. and Wein M., Computer-generated key-frame animation, J Soc Motion elevision Engineers 80, pp. 149-153.

[Calvert et al., 91] Calvert T.W., Welman C., Gaudet S., Schiphorst T. and Lee C., Composition of multiple figure sequences for dance and animation, The Visual Computer, 7 (2-3), 1991, pp. 114-121.

[Carignan, et al., 94] Carignan M. and Yang Y., Magnenat-Thalmann, N., Thalmann, D., Dressing Animated Complex Clothes, Proc. of ACM SIGGRAPH'92, pp. 99-104.

[Catmull, 78] Catmull E., The Problems of Computer-Assisted Animation, Computer Graphics 12 (3), August 1978, pp. 348-353.

[Chui, 92] Chui C. K., An Introduction to Wavelets, Series: Wavelet Analysis and its Applications. Academic Press, Inc., 1992.

[Demori and Probst, 86] Demori R. and Probst D., Handbook of Pattern Recognition and Image Processing, Academic Press, ch. Computer Recognition of Speech, 1986.

[Denavit, 56] Denavit J., Description and Displacement Analysis of Mechanisms Based 2x2 Dual Matrices, Ph.D. Thesis, Mechanical Engineering, Northwestern U., Evanston, Ill, 1956.

[Essa and Pentland, 94] Essa I. and Pentland A., A Vision System for Observing and Extracting Facial Action Parameters, IEE Conf. Computer Vision and Pattern Recognition, 1994, pp. 76-83.

[Foley et al, 90] Foley J. D., van Dam A., Feiner S. K. and Hughes J. F., Computer Graphics, Principles and Practice, Second Edition, Addison-Wesley Publishing Company, 1990, pp. 483-488.

[Fortin et al., 86] Fortin D., Lamy J. F.and Thalmann D., A Multiple Track Animator System For Motion Synchronization and Perception, Motion Representation and Perception, edited by N. I. Badler and J. K. Tsotsos, ACM 1986, pp. 180-186.

[Fu et al., 87] Fu K. S., Gonzalez, and Lee C. S. G., Robotics, Control, Sensing, Vision, and Intelligence, McGraw-Hill, inc. 1987, pp. 52-76, pp. 84 -102, pp. 111-112.

[Gomez, 84] Gomez J. E., Twixt : A 3D Animation System, Proc. of Eurographics'84, pp. 121-134.

[Gourret et al., 89] Gourret J. P., Magnenat-Thalmann N. and Thalmann D., Simulation of object and human skin deformations in a grasping task, Proc. of ACM SIGGRAPH'89, pp. 21-31.

[Grzeszczuk and Terzopoulos, 95] Grzeszczuk R. and Terzopoulos D., Automated Learning of Muscle-Based Locomotion Through Control Abstraction, Proc. of ACM SIGGRAPH'95, pp. 63-70.

[Hégron et al., 96] Hégron G., Arnaldi B. and Lecerf C., Dynamic Simulation and Animation, in Magnenat-Thalmann, N., and Thalmann D. eds., Interactive Computer Animation, Prentice Hall, 1996, pp. 71-98.

[Hemami and Farnsworth, 77] Hemami H. and Farnsworth R. L., Posture and gait stability of a planar five link biped by simulation, IEEE Trans. on Automatic Control, AC-22, 1977.

[Hodgins et al, 95] Hodgins J. K., Wooten W. L., Brogan D. C. and O'Brien J. F., Animating Human Athletics, Proc. of ACM SIGGRAPH'95, pp. 71-78.

[Huang et al., 95] Huang Z., Boulic R., Magnenat-Thalmann N., and Thalmann D., A Multi-sensor Approach for Grasping and 3D Interaction, Proc. of Computer Graphics International'95, Leeds UK, Academic Press, Jun. 1995, pp. 235-254.

[Huang et al., 94] Huang Z., Magnenat-Thalmann N. and Thalmann D., Interactive Human Motion Control Using a Closed-form of Direct and Inverse Dynamics, Proc. of Pacific Graphics'94, pp. 243-255.

[Inman et al., 81] Inman V. T., Ralston H. J. and Todd F., Human Walking, Baltimore, Williams & Wilkins, 1981.

[Isaacs and Cohen, 87] Isaacs P. M. and Cohen M. F., Controlling Dynamics Simulation with Kinematic Constraints, Proc. of ACM SIGGRAPH'87, pp. 215-224.

[ISO 96] Iso/Iec Jtc1/Sc29/Wg11 N1199, International Organization for Standardization Organisation Internationale Normalisation Iso/Iec Jtc1/Sc29/Wg11 Coding Of Moving Pictures And Audio Information, Mpeg96/ March 1996. (http://www.es.com/mpeg4-snhc/)

[Kalra and Magnenat-Thalmann, 94] Kalra P. and Magnenat-Thalmann N., Modeling of Vascular Expressions in Facial Animation, Proc. of Computer Animation'94, Geneva, pp. 50-58.

[Kochanek and Bartels, 84] Kochanek D. and Bartels R., Interpolating splines with local tension, continuity and bias tension, Proc. of ACM SIGGRAPH'84, pp. 33-41.

[Ko and Badler, 93] Ko H. and Badler N. I., Intermittent Non-Rhythmic Human Stepping and Locomotion, Proc. of Pacific Graphics Conference'93, Seoul, 1993.

[Koga et al., 94] Koga Y., Kondo K., Knuffner J. and Latombe J. C., Planning Motions With Intentions, Proc. of ACM SIGGRAPH'94, pp. 395-408.

[Kohli and Soni, 75] Kohli D. and Soni A. H., Kinematic Analysis of Spatial Mechanisms via Successive Screw Displacements, J. Engr. for Industry, Trans. ASME, vol. 2, series B., 1975, pp. 739-747.

[Kolb, C., 92] Kolb C., Rayshade version 4.0 User's Manual, 1992.

[Kunii.and Sun, 90] Kunii T. L. and Sun L., Dynamic Analysis-Based Human Animation, Proc. of CG International'90, pp. 3-16.

[Lafleur et al., 91] Lafleur B., Magnenat-Thalmann N. and Thalmann D., Cloth Animation with Self-Collision Detection, Proc. of the Conference on Modeling in Computer Graphics, Springer, Tokyo, 1991.

[Lake and Green, 91] Lake R. and Green M. W., Dynamic Motion Control an Articulated Figure Using Quaternion Curves, First Conf. on CAD and Graphics, Hangzhou, 1991.

[LeBlanc et al., 91] LeBlanc A., Kalra P., Magnenat-Thalmann N., and Thalmann D., Sculpting with the "Ball & Mouse" Metaphor, Proc. of Graphics Interface'91, Calgary, Canada, 1991.

[Lee et al., 90] Lee P., Wei S., Zhao J. and Badler N., Strength Guided Motion, Proc. of ACM SIGGRAPH'90, pp. 253-262, 1990.

[Lee and Ziegler, 84] Lee C. S. G. and Ziegler M., A Geometric Approach in Solving the Inverse Kinematics of PUMA Robots, IEEE Trans. Aerospace and Electronic Systems, AES-20 (6), pp. 695-706.

[Liégeois, 77] Liégeois A., Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms, IEEE Trans. SMC, 7 (12), 1977, pp. 868-871.

[Liu et al, 94] Liu Z., Gortler S. J., and Cohen M. F., Hierarchical Spacetime Control, Proc. of ACM SIGGRAPH'94, pp. 35-42.

[Luh et al., 80] Luh J. Y. S., Walker M. W. and Paul R. P. C., On-line computation control of mechanical manipulator. Trans. ASME J. Dynamic Systems, Measurement, and Control, 1980, pp. 69-76.

[Magnenat-Thalmann and Thalmann, 95] Magnenat-Thalmann N. and Thalmann D., Virtual Actors Living in a Real World, Proc. of Computer Animation'95, IEEE Computer Society Press, 1995.

[Magnenat-Thalmann et al., 95] Magnenat-Thalmann N., Thalmann D., Huang Z., Shen J., Pandzic I. S., The making of the Xian terra-cotta soldiers, Proc. of Computer Graphics International'95, Leeds UK, Jun. 1995, pp. 281-296.

[Magnenat-Thalmann and Thalmann, 90] Magnenat-Thalmann N. and Thalmann D., Computer Animation, Theory and Pratice, Second Revised Edition, Springer-Verlag, 1990.

[Magnenat-Thalmann et al., 88a] Magnenat-Thalmann N., Primeau N. E. and Thalmann D., Abstract Muscle Action Procedures for Human Face Animation, The Visual Computer 3 (5), 1988.

[Magnenat-Thalmann et al., 88b] Magnenat-Thalmann N., Laperrière N. R. and Thalmann D., Joint dependent Local Deformations for Hand Animation and Object Grasping, Proc. of Graphics Interface'88, 1988.

[Magnenat-Thalmann and Thalmann, 87] Magnenat-Thalmann N. and Thalmann D., The Direction of Synthetic Actors in the Film Rendez-vous a Montreal, IEEE CGA, 7 (12), 1987, pp. 9-19.

[Maiocchi and Pernici, 90] Maiocchi R. and Pernici B., Directing an Animated Scene with Autonomous Actors, Computer Animation 90, Springer Verlag Tokyo, Geneva, 1990, pp. 41-60.

[Mass and Thalmann, 94] Mass S. R. and Thalmann D., A Hand Control and Automatic Grasping System for Synthetic Actors, Proc. of Eurograph'94, pp. C167-C177.

[Milenkovic and Huang, 83] Milenkovic V. and Huang B., Kinematics of Major Robot Linkages, Proc. of the 13th Intl. Symp. Industrial Robos, Chicago, Ill, pp. 16-31.

[Moccozet, 96] Moccozet L., Hands Modeling and Animation for Virtual Humans, Thesis report, University of Geneva, 1996.

[Moore and Wilhelms, 88] Moore M. and Wilhelmes J., Collision Detection and Response for Computer Animation, Proc. of ACM SIGGRAPH'88, pp. 289-298.

[Molet et al., 96] Molet T., Boulic R. and Thalmann D., A Real Time Anatomical Converter For Human Motion Capture, in Boulic R. and Hégron G. eds., Computer Animation and Simulation'96, SpringerComputerScience, SpringerWienNewYork, pp. 79-94.

[Monheit and Badler, 91] Monheit G. and Badler N. I., A kinematic model of the human spline and torso, IEEE CGA, 11 (2), 1991, pp. 29-38.

[Ngo and Marks, 93] Ngo J. T. and Marks J., Spacetime Constraints Revisited, Proc. of ACM SIGGRAPH'93, pp. 343-350.

[Noser et al., 96] Noser H., Capin T. K., Pandzic S. P., Magnenat-Thalmann N., Thalmann D., Playing Games Through the Virtual Life Network, Proc. of Artificial Life V, May 16-18, 1996, Nara-Ken New Public Hall, Nara, Japan (to appear).

[Noser et al., 95] Noser H., Renault O., Thalmann D., Navigation for Digital Actors Based on Synthetic Vision, Memory, and Learning, Comput. & Graphics, 19 (1), 1995, pp. 7-19.

[Noser and Thalmann, 95] Noser H. and Thalmann D., Synthetic Vision and Audition for Digital Actors, Computer Graphics forum, 1 (3), Conference Issue, Maastricht, The Netherlands, Aug. 28 - Sept. 1, 1995, pp. 325-336.

[Orin et al., 79] Orin D. E., McGhee R. B., Vukobratovic M. and Hartoch G., Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods, Math. Biosci., vol. 43, pp. 107-130.

[van de Panne and Fiume, 93] van de Panne M. and Fiume E., Sensor-Actuator Networks, Proc. of ACM SIGGRAPH'93, pp. 335-342.

[van de Panne et al., 90] van de Panne M., Fiume E., and Vranesic Z. G., Reusable Motion Synthesis Using State-Space Controller, Proc. of ACM SIGGRAPH'90, pp. 225-234.

[Pandzic et al., 95] Pandzic I. S., Magnenat-Thalmann N. and Roethlisberger M., Parallel Raytracing on the IBM SP2 and CRAY T3D, EPFL Supercomputing Review, No 7, 1995.

[Park et al., 92]. Park J., Fussell D., Pandy M. and Browne J. C., Realistic Animation using Musculotendon Skeletal Dynamics and Suboptimal Control, Eurographics Worshop on Computer Animation, 1992.

[Parke, 82] Parke F., Parameterized Models for Facial Animation, IEEE CGA, 2 (9), Nov. 1982, pp. 61-68.

[Paul et al., 81] Paul R.P., Shimano B. E. and Mayer G., Kinematic Control Equations for Simple Manipulators, IEEE Trans. System, Man, Cybern., SMC-11 (6), pp. 449-455.

[Phillips et al., 90] Phillips C. B., Zhao J. and Badler N. I., Interactive Real-Time Articulated Figure Manipulation Using Multiple Kinematic Constraints, Computer Graphics, 24 (2), 1990, pp. 245-250.

[Phillips and Badler, 91] Phillips C. B. and Badler N. I., Interactive Behaviors for Bipedal Articulated Figures, Computer Graphics, 25 (4), 1991, pp. 359-362.

[Press et al., 92] Press W. H., Teukolsky S. A., Vetterling W. T. and Flannery B. P., Numerical Recipes in C, The Art of Scientific Computing, Second Edition, Cambridge Press 1992.

[Raibert and Hodgins, 91] Raibert M. and Hodgins J., Animation of Dynamic Legged Locomotion. Proc. of ACM SIGGRAPH'91, pp. 349-358.

[Ramey et al., 95] Ramey D., Rose L. and Tyerman L., File Formats, Version 4.2, Alias | Wavefront, Inc., 1995.

[Reeves, 81] Reeves W., Inbetweening For Computer Animation Utilizing Moving Point Constraints, Computer Graphics 15 (3), August 1981, pp. 263-269.

[Renault et al., 90] Renault O., Magnenat-Thalmann N. and Thalmann D., A Vision-based Approach to Behavioral Animation, Journal of Visualization and Computer Animation, 1 (1), 1990, pp. 18-21.

[Reynolds, 82] Reynolds C., Computer Animation with Scripts and Actors, Computer Graphics 16 (3), July 1982, pp. 289-296.

[Rezzonico et al., 95] Rezzonico S., Boulic R., Huang Z., Magnenat-Thalmann N., Thalmann D., Consistent Grasping in Virtual Environments Based on the Interactive Grasping Automata, Virtual Environments'95, M. Gobel (ed.), Springer Computer Science, EG, SpringerWienNewYork, pp. 107-118.

[Rijpkema and Giard, 91] Rijpkema H. and Giard M., Computer Animation of Knowledge-Based Human Grasping, Proc. of ACM SIGGRAPH'91, pp. 339-348.

[Rohlf and Helman, 94] Rohlf J. and Helman J., IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics, Proc. of ACM SIGGRAPH'94, pp. 381-394.

[Sederberg and Greenwood, 92] Sederberg T. and Greenwood E., A physically-based approch to 2-D shape blending, Proc. of ACM SIGGRAPH'92, pp. 26-34.

[Seireg and Arvikar, 89] Seireg A. and Arvikar R., Biomechanical Analysis of the Musculoskeletal Structure for Medicine and Sports, Hemisphere Publishing Corporation, NY 1989.

[Shelley and Greenberg, 82] Shelley K. and Greenberg D., Path Specification and Path Coherence, Computer Graphics 16 (3), July 1982, pp. 289-296.

[Shen and Thalmann, 95] Shen J. and Thalmann D., Interactive shape design using metaballs and splines, Implicit Surfaces' 95, Grenoble, France, April, 1995.

[Sims, 94] Sims K., Evolving Virtual Creatures, Proc. of ACM SIGGRAPH'94, pp. 15-34.

[Stollnitz et al., 95] Stollnitz E. J., DeRose T. D. and Salesin D. H., Wavelets for Computer Graphics: A Primer, IEEE CGA, Part 1, May, 1995, pp. 76-84; Part 2, July, 1995, pp. 75-85.

[Steketee and Badler, 85] Steketee S. N. and Badler N., Parametric Keyframe Interpolation In Incorporating Kinetic Adjustment and Phrasing Control, Proc. of ACM SIGGRAPH'85, pp. 255-262.

[Sturman et al., 89] Sturman D. J., Zeltzer D. and Feiner S. (1989) Hands-on Interaction with Virtual Environments, Proc. of ACM SIGGRAPH Symposium on User Interface Software and Technologies, pp. 19-24

[Sturman, 84] Sturman D., Interactive Keyframe Animation of 3-D Articulated Motion, Proc. of Graphics Interface ‘84, Ottawa, Ontario, May 1984, pp. 35-40.

[Tanimoto and Pavlidis, 75] Tanimoto S. L. and Pavlidis T., A hierarchical data structure for picture processing, Comput. Gr. Image Process. 4, 1975, pp. 104-119.

[Terzopoulos and Waters, 90] Terzopoulos D. and Waters K, Physically-Based Facial Modeling and Animation, Journal of Visualization and Computer Animation, vol. 1, 1990, pp. 73-80.

[Thalmann et al., 96] Thalmann D., Shen J., Chauvineau E., Fast Realistic Human Body Deformations for Animation and VR Applications, Proc. of Computer Graphics International'96, Pohang, Korea, June, 1996.

[Tu and Terzopoulos, 94] Tu X. and Terzopoulos D., Artificial Fishes: Physics, Locomotion, Perception, Behavior, Proc. of ACM SIGGRAPH'94, pp. 43-50.

[Turner et al., 91] Turner R., Balaguer F., Gobbetti E. and Thalmann D., Physically-Based Interactive Camera Motion Control Using 3D Input Devices, Proc. of Computer Graphics International’91, MIT, Boston, 1991.

[Turner et al., 90] Turner R., Gobbetii E., Balaguer F., Mangili A., Magnenat-Thalmann N. and Thalmann D., An Object-Oriented Methodology using Dynamic Variables for Animation and Scientific Visualization, Proc. of Computer Graphics International'90, Springer-Verlag, pp. 317-327.

[Uicker et al., 64] Uicker J. J., Jr., Denavit J. and Hartenberg R. S., An Iterative Method for the Displacement Analysis of Spatial Mechanisms, Trans. ASME, J. Appl. Mech., vol. 31, Series E, pp. 309-314.

[Unuma et al., 95] Unuma M., Anjyo K. and Takeuchi R., Fourier Principle of Emotion-Based Human Figure Animation, Proc. of ACM SIGGRAPH'95, pp. 91-96.

[Volino et al., 96] Volino P., Magnenat-Thalmann N., Shen J., Thalmann D., The Evolution of a 3D System for Simulating Deformable Clothes on Virtual Actors, IEEE CGA, September, 1996, pp. 42-51.

[Volino and Magnenat-Thalmann, 94] Volino P. and Magnenat-Thalmann N., Efficient Self-Collision Detection on Smoothly Discretised Surface Animations using Geometrical Shape Regularity, Computer Graphics Forum (EuroGraphics Proc.) 13 (3), 1994, pp. 155-166.

[Walker and Orin, 92] Walker M. W. and Orin D. E., Efficient Dynamic Computer Simulation of Robotic Machanisms, Trans. ASME, J. Systems, Measurement and Control, vol. 104, pp. 205-211.

[Wilhelms and Barsky, 85] Wilhelms J. and Barsky B., Using Dynamic Analysis to Animate Articulated Bodies Such As Humans and Robots, in Magnenat-Thalmann N. and Thalmann D. eds., Computer-generated images, Springer, pp. 209-229.

[Witkin and Popovic, 95] Witkin A. and Popovic Z., Motion Warping, Proc. of ACM SIGGRAPH'95. pp. 105-108.

[Witkin and Kass, 85] Witkin A. and Kass M., Spacetime Constraints, Proc. of ACM SIGGRAPH'88, pp. 159-168.

[Whitney, 69] Whitney D. E., Resolved Motion Rate Control of Manipulators and Human Protheses, IEEE Trans. Man-Machine Systems, MMS-10 (2), pp. 47-53.

[Yang and Freudenstein, 64] Yang A. T. and Freudenstein R., Application of Dual Number Quaternian Algebra to the Analysis of Spatial Mechanisms, Trans. ASME, J. Appl. Mech., vol. 31, series E, pp. 152-157.

[Zeltzer, 82] Zeltzer D., Representation of complex animated figures, Proc. Graphics Interface'82, pp. 205-221.

[Zhao and Badler, 94] Zhao X. and Badler N. I., Interactive body awareness, Journal of Computer-Aided Design, 26 (12), Dec. 94, pp. 861-867.

9. Appendix A: Armstrong-Green Formulation

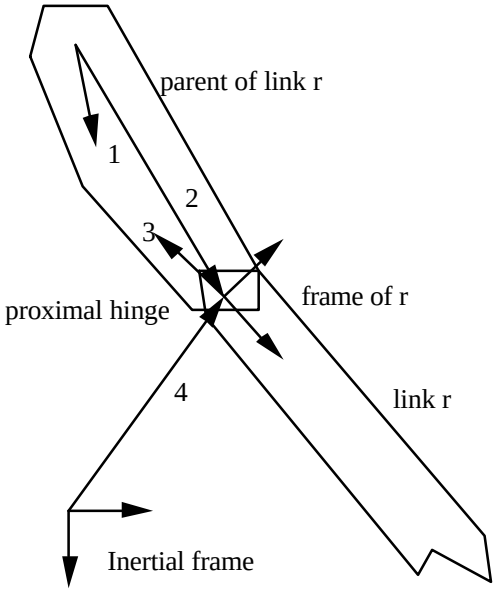


Figure A- 1 two link display to show the vectors: 1: c^p, 2: l^r, 3: f^r and 4: p^r in the algorithm

Now we only list the main equations of the algorithm without their deduction process:

$$J^r\dot{\omega}^r = g_\Sigma^r - m_r c^r \times a^r + \sum_{s\in S_r} l^s \times R^s f^s \qquad \textbf{Eq. A- 1}$$

where

$J^r\dot{\omega}^r$: the rate of change of angular momentum.

the right side:

g_Σ^r: the separated term to accelerate the solution.

$-m_r c^r \times a^r$: the coordination system of r is accelerating with respect to the root link coordination system, giving the effect of the force $-m_r a^r$.

$\sum_{s\in S_r} l^s \times R^s f^s$: the effect from the sons of r.

$$g_\Sigma^r = -\omega^r \times (J^r\omega^r) - g^r + \sum_{s\in S_r} R^s g^s + R_I^{rT} g_E^r + m_r c^r \times R_I^{rT} a_G + p_E^r \times R_I^{rT} f_E^r \qquad \textbf{Eq. A- 2}$$

where

$-\omega^r \times (J^r\omega^r)$: a torque from the rotation of coordination system r.

$-g^r$: a torque from parent's reaction.

$\sum_{s\in S_r} R^s g^s$: torque from children.

$R_I^{rT} g_E^r$: external toque.

$m_r c^r \times R_I^{rT} a_G$: a torque from the force of gravity.

$p_E^r \times R_I^{rT} f_E^r$: a torque from the external force acting on p_E^r.

$$f^r = f_\Sigma^r - m_r a^r + m_r c^r \times \dot{\omega}^r + \sum_{s\in S_r} R^s f^s \qquad \textbf{Eq. A- 3}$$

where

f^r: the force acting on the parent of r at the proximal hinge of r.

f_Σ^r: the separated term.

$-m_r a^r$: the coordination system of r is accelerating with respect to the root link coordination system, giving the effect of the force $-m_r a^r$.

$m_r c^r \times \dot{\omega}^r$: from the rotation of coordination system r.

$\sum_{s \in S_r} R^s f^s$: the force from children.

the right side:

$$f_\Sigma^r = -m_r \omega^r \times (\omega^r \times c^r) + R_I^{rT}(f_E^r + m_r a_G) \quad \textbf{Eq. A- 4}$$

where

$-m_r \omega^r \times (\omega^r \times c^r)$: a centrifugal force from the rotation of the coordination system.

$R_I^{rT}(f_E^r + m_r a_G)$: external force and force of gravity.

$$R^s a^s = a_c^s + a^r - l^s \times \dot{\omega}^r \quad \textbf{Eq. A- 5}$$

where

$R^s a^s$: the acceleration at the proximal hinge of a child s.

a_c^s: the separated term.

a^r: the acceleration at the proximal hinge of r.

$-l^s \times \dot{\omega}^r$: from the rotation.

and

$$a_c^s = \omega^r \times (\omega^r \times l^s) \quad \textbf{Eq. A- 6}$$

Inbound process: from the leaves to the root, for each link r:

$$K^r = T^r(\sum_{s \in S_r} \tilde{l}^s R^s M^s R^{sT} - m_r \tilde{c}^r) \quad \textbf{Eq. A- 1}$$

$$M^r = -m_r I + m_r \tilde{c}^r K^r + \sum_{s \in S_r} R^s M^s R^{sT}(I - \tilde{l}^s K^r) \quad \textbf{Eq. A- 2}$$

$$d^r = T^r(g_\Sigma^r + \sum_{s \in S_r} \tilde{l}^s R^s (f'^s + M^s R^{sT} a_c^s)) \quad \textbf{Eq. A- 3}$$

$$f'^r = f_\Sigma^r + m_r c^r \times d^r + \sum_{s \in S_r} (R^s f'^s + R^s M^s R^{sT}(a_c^s - l^s \times d^r)) \quad \textbf{Eq. A- 4}$$

where the single notations can be grouped by:

Scalars:

m_r the mass of link r;

scalars represented in the **root link system**:

a_G the acceleration of gravity;

p^r the position vector of the hinge of link r which joins it to its parent;

v^r the velocity of the proximal hinge of link r;

f_E^r an external force acting on link r at the point p_E^r;

g_E^r an external torque acting on link r;

scalars represented in the **coordination system of link r**:

a^r the acceleration of the proximal hinge of link r;

ω^r the angular velocity of link r;

$\dot{\omega}^r$ its rate of change;

c^r the vector from the proximal hinge to the center of mass of link r;

f^r the force which link r exerts on its parent at the proximal hinge;

g^r the torque which link r exerts on its parent at the proximal hinge;

p_E^r the vector from the proximal hinge of link r to the point of application of the external force f_E^r to link r;

scalars represented in the **coordination system of the parent of link r**:

l^r the vector from the proximal hinge of the parent of link r to the proximal hinge of link r (constant in the coordination system of the parent);

Rotation matrices:

R^r converts vector representations in the coordination system of link r to the representations in the coordination system of the parent link;

R^{rT} the inverse (= transpose) of R^r;

R_I^r converts from representations in coordination system r to representations in the root link coordination system;

R_I^{rT} the inverse (= transpose) of R_I^r;

Matrix representation in **coordination system r**:

J^r the moment of inertial matrix of link r about its proximal hinge;

10. Appendix B: Recursive Newton-Euler Formulation

Table B- 1. Summary of Notation

All vectors with sub index i are expressed in the body coordination frame of the ith link.
The following items except m_i are vectors.

$\dot{q}_{ij}$, $\ddot{q}_{ij}$: the first and second derivatives of generalized variable for the jth DOF of the ith joint.

ω^i : angular velocity of the ith link.

α^i : angular acceleration of the ith link.

a^i : linear acceleration of the ith link at the origin of its local coordination frame.

a_c^i : linear acceleration of the ith link at the center of mass.

g : gravitational acceleration vector in the inertial coordination frame.

g_i : gravitation of the ith link.

f_i : force exerted on the ith link by the (i-1)st link.

n_i : moment exerted on the ith link by the (i-1)st link.

τ_i : joint generalized actuator torque at the ith joint.

p_i : vector from the origin of ith link to the origin of (i+1)st link.

s_i : vector from the origin of ith link to its center of mass.

z_{ij} : rotation vector of the jth DOF of the ith joint.

m_i : mass of link i.

F_i : total external force exerted on the ith link.

N_i : total external torque exerted on the ith link.

The following items are 3 by 3 matrix.

J_i : inertia matrix of the ith link.

A_i^j : pure rotational matrix from the jth link coordination frame to the ith.

Integer

nd_i : number of DOF at joint i.

Table B- 2 Recursive Newton-Euler Algorithm

Initialization

$\omega^0 = \alpha^0 = v^0 = 0$

$f_{n+1} = \textit{force required at the end – effector}$

$n_{n+1} = \textit{moment required at the end – effector}$

Forward recursion

For $i = 1,\ldots,n$ *do*:

$$\omega^i = A_i^{i-1}\omega^{i-1} + \sum_{j=1}^{nd_i} z_{ij}\dot{q}_{ij}$$

$$a^i = A_i^{i-1}a^{i-1} + \sum_{j=1}^{nd_i} z_{ij}\ddot{q}_{ij} + A_i^{i-1}\omega^{i-1} \times \sum_{j=1}^{nd_i} z_{ij}\dot{q}_{ij}$$

$$a^i = A_i^{i-1}(a^{i-1} + a^{i-1} \times p_{i-1} + \omega^{i-1} \times (\omega^{i-1} \times p_{i-1}))$$

$$a_c^i = a^i + a^i \times s_i + \omega^i \times (\omega^i \times s_i)$$

$$g_i = A_i^0(m_i g) \quad /* \quad g = (0.0, 0.0, -9.8) \quad */$$

$$F_i = m_i a_c^i$$

$$N_i = J_i a^i + \omega^i \times J_i \omega^i$$

Backward recursion

For $i = n,\ldots,1$ *do*

$$f_i = F_i + A_i^{i+1} f_{i+1} - g_i$$

$$n_i = N_i + A_i^{i+1} n_{i+1} + s_i \times (F_i - g_i) + p_i \times A_i^{i+1} f_{i+1}$$

$$\tau_i = n_i \cdot z_i$$

end;

---

11. Appendix C: Scenelib

Scenelib defines the low-level data structures for articulated figures, e.g., joint and other elements, e.g., light and camera, and sets of manipulation functions. The node_3D, and noted N3D, is chosen to retain the geometric, topological and display information sufficient to set a frame in a 3D space that is the kernel structure of the Scenelib. It can bind one typed structure (Figure C-1). If a node_3D has no associated information it is said to be of NEUTRAL type. it can be useful as an intermediate frame to identify some functional location. The associated information includes:

JOINT: describes a single DOF joint either translation or rotation. The other usual joints can be constructed by a chain of such `JOINTs`.

FREE: specifies an arbitrary geometric transformation. It is useful for the animation of independent objects. For example the human skeleton is made of `JOINTs` except one `FREE` used to describe its general motion in the world coordinate system. This data structure also provides various modes to control a lookat transformation.

The terminal data is application-dependent and can be supplied by different applications. Scenelib provides these default types:

SM_FIG: This type is used to integrate information about geometrical modeling of surface information. The boundary representation is used to represent the human skin surface, and more specifically we use the *SurfMan* structure based on triangle facets. As an example, the whole human surface model is cut into multiple parts attached to their respective ***JOINT*** node_3D, such as "head attached to the last neck joint", "leg attached to the last thigh joint", etc.

CAMERA: This is one critical link for the scene visualization. Unless the CAMERA is to remain static in the scene this node 3D should be defined as the child of a FREE node 3D which provides modes to manage a lookat transformation.

SOLID: This is used to represent the human body volume with the simple geometric objects, e.g., cubes, spheres, cylinders, frusta, and ellipsoids, for dynamic simulation. More details are described in the dynamic motion control section.

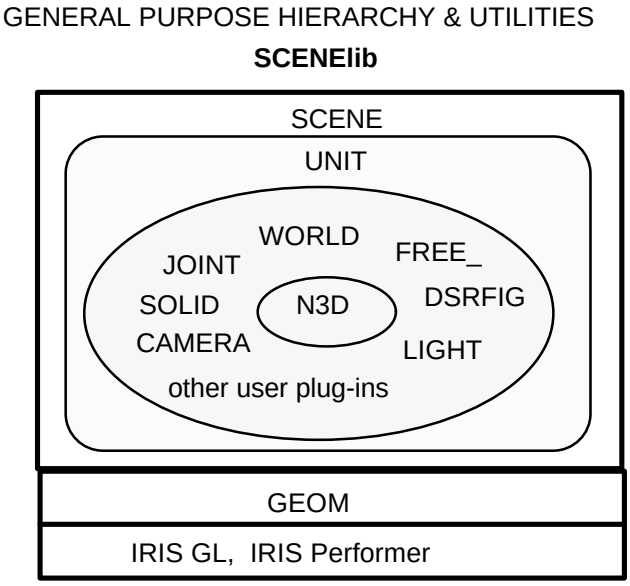


Figure C- 1 the hierarchical human data structure: the lowest level is N3D and highest is scene

The ***SCENE*** data structure is a two-layer structure in order to allow multiple positioning of complex functional units without duplication of data. The basic organization retained for a scene involving human models and their environment is a Direct Acyclic Graph (*noted DAG*) at the higher level of functional units, e.g., humans, and a tree 3D hierarchy (noted *H3D*) at the lower level of basic constituents, e.g., *JOINTs* .

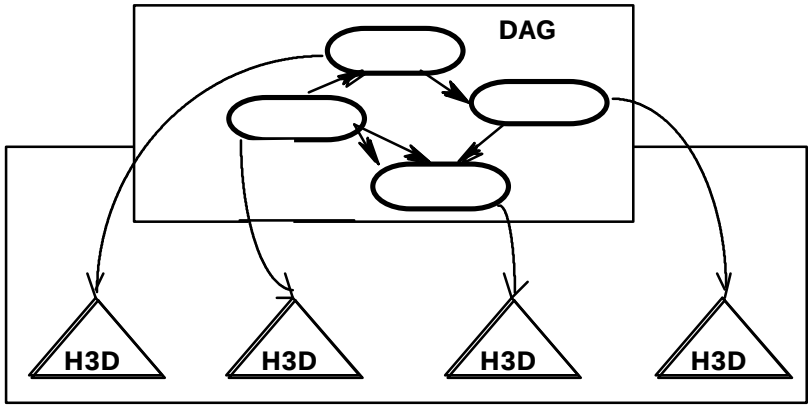


Figure C- 2 The two levels organization of a scene

Besides the SCENE data structure, there are other auxiliary data structures and functions in the Scenelib:

GEOM: matrix and vector calculation such as geometric transformation.

IRIS GL: the interface to use IRIS GL complex data structure, e.g., the Rectangle Mesh, and calling IRIS GL functions so that the application codes are IRIS GL independent.

IRIS Performer: the interface to use IRIS Performer and moreover, to construct and update the Performer data structure, i.e., libpf and libpr, from the Scenelib hierarchy.

In Figure C-3, each simple box represents a node_3D. The internal node_3Ds convey general purpose mobility information while terminal node_3Ds are more application oriented.

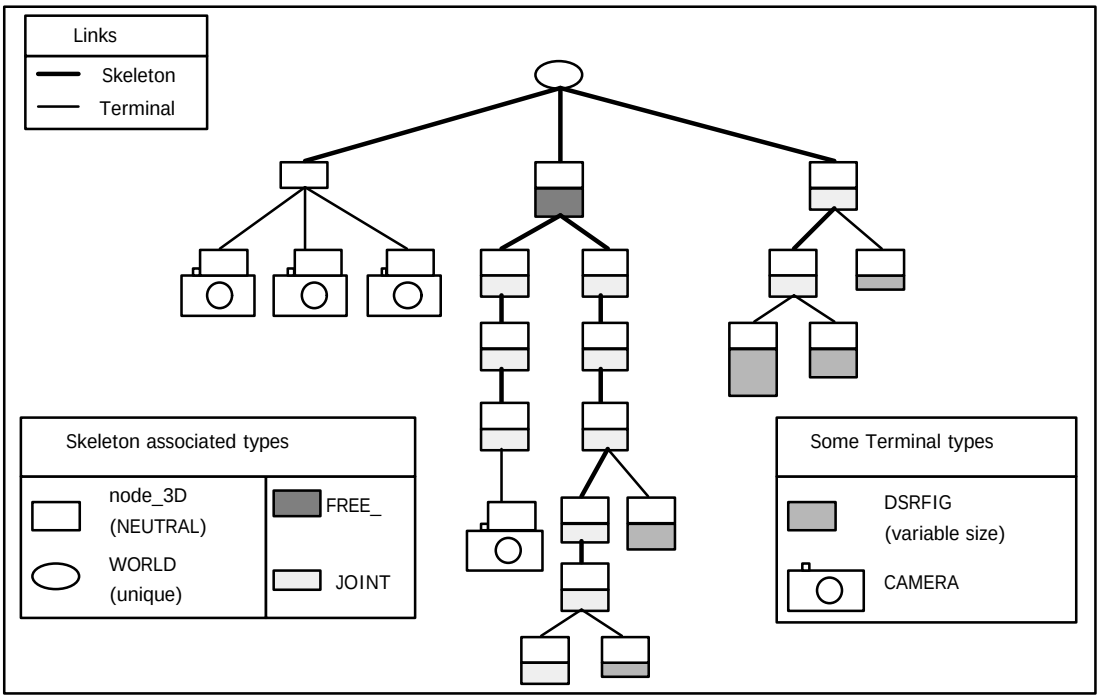


Figure C- 3 a symbolic representation of a H3D showing a simple skeleton and some typed data

The most important of the N3D information is its position and orientation that is represented by the Direct Inverse Transformation (DIT) Convention. For each N3D, there is a local coordinate system that maintains both the direct and inverse transformation matrices to its parent node_3D's local coordinate system. The homogeneous transformations are used to unify the coordinate system transformation by simple multiplication of such matrices. Moreover, SGI workstations can maintain a stack of such matrices to minimize the update cost of global transformations. The naming convention of the transformations is relative to the N3D possessing the DIT (Figure C-4):

dir: transforms from the proprietary N3D to external N3D.

inv: transforms from external N3D to the proprietary N3D.

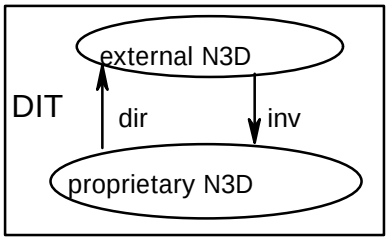


Figure C- 4 the DIT Convention for position and orientation relationship

There are three types of DIT: Initial (INI), Variable (VAR) and Reference (REF). The INI DIT represents the initial configuration of the N3D in the tree hierarchy while the VAR DIT represents the mobility of the N3D. In the previous section, the use of Bodylib automatically creates of the INI DIT for all N3D in a body hierarchy, and allows the application only can change the VAR DIT. The third REF DIT represents the global reference for each N3D. It is very efficient for the application to find the global information, i.e., the position and orientation in the global world coordinate system. Each change of VAR DIT needs to update the REF DIT.

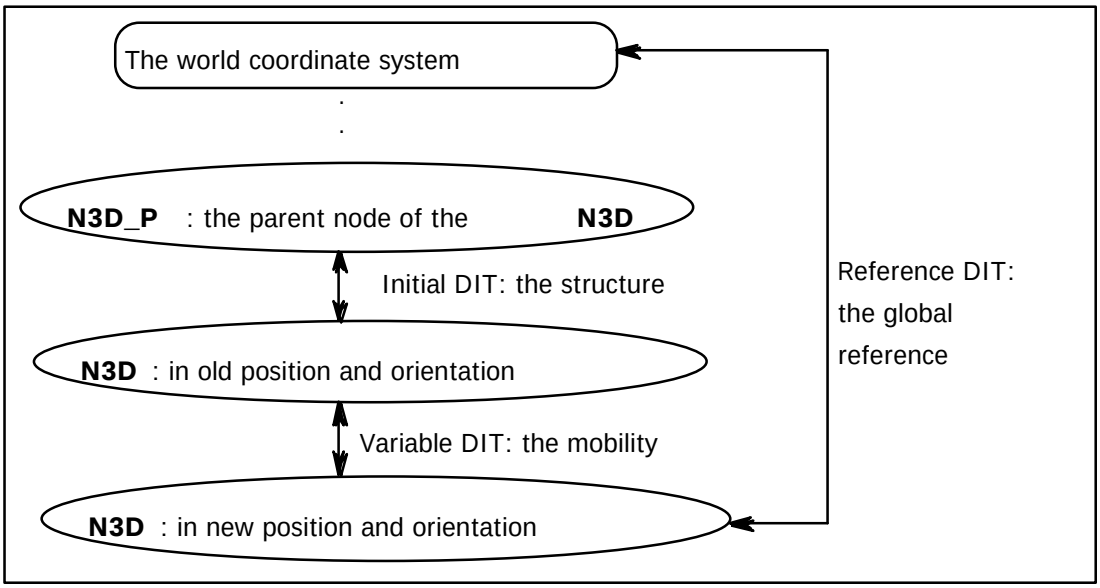


Figure C- 5 three types of DIT for the tree hierarchy (skeleton or articulated figure) model and animation

Zhiyong Huang

EPFL-DI-LIG
CH-1015 Ecublens
http://ligwww.epfl.ch/huang.html
Lausanne, Switzerland
Fax: +(41)(21)693 5328

e-mail: huang@lig.di.epfl.ch
WWW:
Tel: +(41)(21)691 5248 (lab)
+(41)(21)693 5216 (office)

Education:

Master of Engineering (Computer Science), Sept. 86 ~ Dec. 88

Tsinghua University, Beijing, P. R. China

Thesis: Rendering System for Computer Aided Design

Bachelor of Engineering (Computer Science), Sept. 81 ~Jul. 86

Tsinghua University, Beijing, P. R. China

Thesis: Solid Modeling with Analysis Method

Experience:

Research Assistant (2/92 ~present)

Computer Graphics Lab. (DI-LIG),

Swiss Federal Institute of Technology, Lausanne (EPFL)

Researcher and Lecturer (12/88 ~ 2/92)

Computer Graphics and Computer Aided Design Lab.,

Tsinghua University, Beijing, P. R. China.

Publications:

1. Daniel Thalmann, Hansrudi Noser and Zhiyong Huang, "How to Create a Virtual Life", INTERACTIVE COMPUTER ANIMATION, Nadia Magnenat-Thalmann and Daniel Thalmann (eds.), Prentice Hall 1996, pp. 263-291.
2. Daniel Thalmann, et al., "Virtual and Real Humans Interacting in the Virtual World", International Conference on Virtual Systems and Multimedia, Gifu, Japan, Sept 18-20, 1995, pp. 48-57.
3. Ronan Boulic, Zhiyong Huang, Daniel Thalmann, "A Comparison of Design Strategies for 3D Human Motions", Human Comfort and Security Workshop, Brussels, Oct. 26, 1995, Springer-Verlag(to appear).
4. Ronan Boulic, et al., "The HUMANOID Environment for Interactive Animation of Multiple Deformable Human Characters", Proc. of Eurographics'95, Maastricht, Sept. 1995, pp. C337-C348.
5. Nadia Magnenat-Thalmann, et al., "The making of the Xian terra-cotta soldiers", Proc. of Computer Graphics International'95, Leeds UK, Jun. 1995, pp. 281-296.
6. Zhiyong Huang, Ronan Boulic, Nadia Magnenat-Thalmann, Daniel Thalmann, "A Multi-sensor Approach for Grasping and 3D Interaction", Proc. of Computer Graphics International'95, Jun. 1995, Leeds UK, pp. 235-254.
7. Serge Rezzonico, Ronan Boulic, Zhiyong Huang, Nadia Magnenat-Thalmann, Daniel Thalmann, "Consistent Grasping in Virtual Environments Based on the Interactive Grasping Automata", Virtual Environments'95, M. Gobel (ed.), Springer Computer Science, EG, SpringerWienNewYork, pp. 107-118.

8. Ronan Boulic, Zhiyong Huang, Nadia Magnenat-Thalmann, Daniel Thalmann, "Goal-Oriented Design and Correction of Articulated Figure Motion with the Track System", Journal of Comput. & Graphics, 18 (4), 1994, pp. 443-452.

9. Ronan Boulic, Daniel Balmer, Zhiyong Huang, Daniel Thalmann (1994), A Stereoscopic Restitution Environment for 3D analysis of Gait, 3rd Symposium of 3D Analysis of Human Motion, Stockholm, July 1994.

10. Zhiyong Huang, Nadia Magnenat-Thalmann, Daniel Thalmann, "Motion Control with Closed Form Direct and Inverse Dynamics", Proc. of Pacific Graphics Conference'94, pp. 243-255.

11. Ronan Boulic, Zhiyong Huang, Nadia Magnenat-Thalmann, Daniel Thalmann, "A Unified Framework for the Motion Manipulation of Articulated Figures with the TRACK System", the Second Conference International Computer Graphics Conference, Beijing 1993, pp. 45-50.

12. Zhiyong Huang, "Review of Computer Realistic", Journal of China Computer Users, vol. 8, 1990.

13. Zhiyong Huang, Jiaguang Sun, "Research and Implement 3D definition in Geometric Modeling System", Proc. of the Sixth Conference of China Computer Graphics Society, 1989, Nanchang.

14. Zhiyong Huang, Zesheng Tang, "Fast Raytracing with Octree Based Hierarchy", Proc. of the Fifth Conference of China Computer Graphics Society, 1989, Shanghai.

15. Zhiyong Huang, Xinyou Li, "Rendering of Free Form Surface", Proc. of the fifth Conference of China Computer Graphics Society, 1988, Beijing.