Coding of virtual human motion

by

Herman van der Elst

Submitted in partial fulfillment of the requirements for the degree

Ph.D. (Electronic Engineering)

in the

Faculty of Engineering

UNIVERSITY OF PRETORIA

October 1999

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Coding of virtual human motion

by

Herman van der Elst

Supervisor: Prof. J.J.D. van Schalkwyk

Department: Electrical and Electronic Engineering

Degree: Ph.D. (Electronic Engineering)

Key terms: Virtual environments, virtual humans, waveform coding, model based coding, motion compression, MPEG-4, human modeling, human animation, motion capture

Abstract

The growing use of realistic virtual embodiments of human participants in virtual environments brings the difficulty of efficiently communicating and storing the actions of these virtual humans. It is only recently that researchers have started to bring together motion capture and other natural input technologies with realistic, real-time graphic representations of humans. A highly articulated and realistic virtual human can easily generate orders of magnitude more information than traditional synthetic objects such as vehicles and aircraft.

In this thesis, we approach the problem of dealing with virtual human motion data from a solid mathematical and engineering background. A basic introduction to virtual environments, networked environments and virtual humans is given. Analogous to the route taken by speech and video coding, we analyze the source data and investigate various implementations of coding techniques for this "new" class of data. A number of waveform coding and model based coding methods are implemented, and the results compared. We address the current disparity between facial expression coding methods and full body motion coding methods. This work contributes towards the MPEG-4 standardization process, and more specifically towards the work done by the SNHC (Synthetic/Natural Hybrid Coding) group.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Coding of virtual human motion

deur

Herman van der Elst

Promotor: Prof. J.J.D. van Schalkwyk
Departement: Elektriese en Elektronies Ingenieurswese
Graad: Ph.D. (Elektroniese Ingenieurswese)

Sleutelterme: Virtuele omgewings, virtuele mense, golfvormkodering, modelgebaseerde kodering, bewegingskompressie, MPEG-4, mensmodellering, mens-animasie, bewegingsmonstering

Samevatting

Die toenemende gebruik van realistiese virtuele voorstellings van menslike deelnemers in virtuele omgewings bring effektiewe kommunikasie- en bergingsprobleme mee. Navorsers het eers onlangs die gebruikmaak van intydse bewegingsmonsters en ander natuurlike beheertegnologië versoen met intydse grafiese voorstellings van mense. 'n Hoogs geartikuleerde en realistiese mensmodel kan ordes meer informasie genereer in vergelyking met tradisionele sintetiese voorwerpe soos voertuie en vliegtuie.

In hierdie tesis benader ons die probleem van virtuele mensdata vanuit 'n gevestigde wiskundige en ingenieursagtergrond. 'n Oorsig ten opsigte van virtuele omgewings, netwerkomgewings en virtuele mense word gegee. Soortgelyk aan die roete wat geneem is in die veld van spraak- en videokodering, word die brondata geanaliseer en 'n aantal koderingsimplementasies ondersoek vir hierdie "nuwe" klas van data. Ons spreek die huidige gebrek van vollyfkoderingsmetodes in vergelyking met die tans weldeurdagte gesigskoderingsmetodes aan. Hierdie werk dra by tot die MPEG-4 standaardiseringsproses, en meer spesifiek, tot die werk wat gedoen word deur die SNHC (Synthetic/Natural Hybrid Coding) groep.

Aan my ma

Table of contents

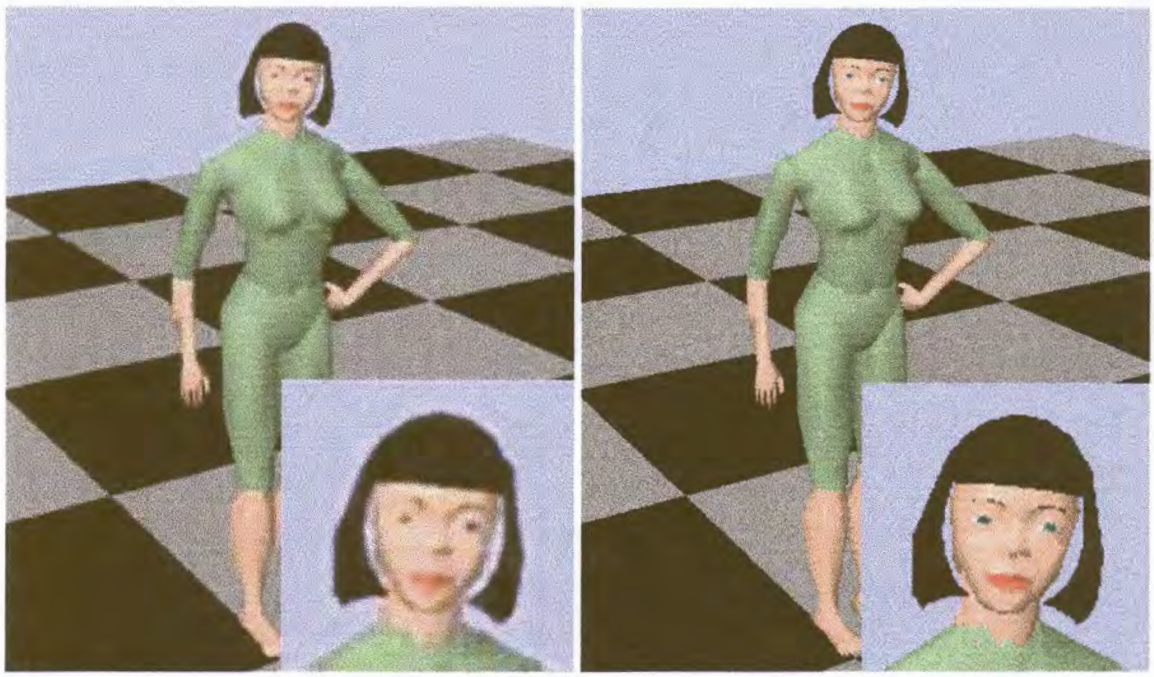
Chapter 1 Introduction

Recent advances in computing power and graphics capabilities have had a huge impact on realistic and interactive virtual environments. Representations previously only possible on high end graphics workstations are now possible on entry level personal computers. In the same sense, networked virtual environments (NVEs) are made possible by using common network technologies to link multiple participants to a single environment. Such networked systems demand a natural representation of participants, including credible visual embodiment and interaction. The use of simulated virtual human figures provides this functionality visually, as well as in the way we interact with our natural surroundings.

Graphical human representation has been developed to such a stage that realistic and credible renderings are at the order of the day. At the same time advanced modeling techniques such as human dynamics, inverse kinematics, real-time motion capture and combinations thereof, have been used increasingly in the animation industry to generate extremely realistic synthetic humans. Many NVE systems have been created using various types of network and computer architectures. The efficient fusion of modeling, animation, rendering and network technologies is one of the next steps in constructing realistic networked virtual environments. Inserting virtual humans into the NVE is a complex task, and one has to consider the scalability of the system, the complexity of the human model in terms of appearance and animation, and the impact on network requirements such as bandwidth and latency.

By displaying natural images (such as a human body) in a synthetic fashion (by means of a 3D computer rendering), an inherent compression gain is achieved. This concept is illustrated by the two figures below. On the left is a single MPEG-1 coded/decoded video

frame and on the right the same image, but regenerated in total by the receiver. The MPEG video stream requires ~1 Megabits/second, while the synthetic animation requires less than 4 Kilobits/second to convey the *same* information. The resolution of the MPEG frame is ~320x240 pixels, while the synthetic frame can be rendered independent of any bandwidth constraints in excess of 1024x768 pixels. The visual aspect of the synthetic image is free of any compression artifacts. The user also has complete control over camera movement, and no additional information for changes in viewpoint and orientation needs to be transmitted. The advantages of synthetic imagery are becoming more evident each day, and the display of human-like figures is no exception.



Pixel based video compression (left) versus synthetic imaging (right).

1.1 Thesis motivation, goal and contribution

The insertion of highly articulated and realistic virtual humans into networked virtual environments can easily generate orders of magnitude more information than traditional synthetic objects such as vehicles and aircraft. Current low cost dial-up network solutions can hardly accommodate two such humans, yet the demands are for a multitude of

simultaneous participants. To date very little attention has been given to the problem of efficiently *communicating* body animation parameters over a distance using limited bandwidth channels. The past few years saw much research dedicated to facial parameter estimation, compression and transmission. This is understandable, since the concept of ultra low bit-rate "talking-head" communication has very lucrative potential in the field of video conferencing, distance learning and video games. Very little attention has been given to a complete analysis of the rest of human motion in terms of bandwidth requirements and compression potential. There are no clear comparison results on the use of various compression techniques for body motion in general.

The goal of this study is to investigate and develop efficient compression methods for human motion data. These include the use of existing coding methods, but adapted for use with the "new" class of motion data. The research also seeks to develop new coding methods, especially in the field of ultra low bit-rate synthetic model based approaches.

This study contributes towards research in the field of networked virtual environments and specifically in the area of virtual human communication. The research addresses the bandwidth problems associated with the insertion of multiple articulated virtual humans into an NVE. The work supports the research of the SNHC group within the MPEG-4 standardization process, and narrows the gap between current facial parameter compression methods and full body motion parameter compression. Additionally, the study also contributes towards bringing together two major areas of research, namely the field of computer graphics and animation and the field of communication engineering and source coding.

1.2 Thesis overview and outline

With the inherent compression of synthetic 3D images over pixel-based approaches as a starting point, further reduction can be found by coding the parameters that define the animation of the synthetic image. In this study we concentrate on the nature of virtual

human motion and the bandwidth requirements of inserting virtual humans into an NVE system. The ultimate goal is to point out redundancy, *especially* for body motion, and to implement existing and new compression techniques in order to reduce the redundancy. As an example, there are a number of existing methods in the field of speech and video coding that can be readily applied to human motion. However, there is one fundamental difference between sampled human motion and sampled speech. Speech (or the information in speech) can be seen as a *modulated* signal, while human motion is strictly a *baseband* signal. Motion is sampled at a very low rate compared to speech, and there are very few samples available to process before coding delay becomes a problem.

The complexity of simulating and rendering virtual humans varies, but is generally a function of the surface detail and the number of joints modeled. With appropriate position and orientation sampling sensors and data gloves, we obtain the necessary joint angles either directly, or by using an inverse kinematics solution. Due to a limited number of sensors, we restrict ourselves to motion with both feet rooted to the ground. Software has been developed to map the joint data in real-time to an articulated figure. The data can be viewed graphically, edited, manipulated and stored. Stored human motion is analyzed in terms of spatial, temporal and frequency content. Various compression techniques ranging from joint angle quantization to model based coding is investigated. Techniques from the computer graphics animation and pattern recognition research fields are combined and put to good use as compression techniques. The use of appropriate error measurement techniques is also discussed.

The thesis addresses two main areas of knowledge, namely computer graphics/animation and communication engineering. In order accommodate readers of both disciplines, it was decided to include detailed background descriptions where appropriate. Readers from a computer science background who are familiar with the concepts of virtual environments, virtual humans, human modeling, animation and motion capture need not study the chapters on these topics in detail. Similarly, readers from an engineering background who

are familiar with the concepts of data analysis, error measurement and the basics of waveform coding may regard these chapters as background material.

Briefly, chapter 1 is a compact introduction to the study. Chapter 2 gives a general review of virtual environments, virtual humans and virtual human communication in terms of previous and current research. Chapter 3 explains the hierarchical and geometrical models that are used, and the relationship between them. Chapter 4 discusses the concept of motion capture, as well as real-time implementation using a minimal set of sensors. Chapter 5 presents a detailed data analysis of the resulting human motion as sampled by such motion capture devices. Chapter 6 investigates a number of objective error measurement techniques, addresses the shortcomings of these methods and introduces an improved error measurement. Chapter 7 deals with waveform compression techniques and chapter 8 develops a model based approach. Chapter 9 forms a comparison between the various compression methods. Chapter 10 concludes the study, and discusses potential future work. A convenient summary of the most important aspects, results and conclusions is presented at the end of each appropriate chapter. A number of appendices containing supporting information are included at the end of the document. Throughout the thesis occasional proofs and other qualifying discussions are omitted in order to maintain information flow. The reader is referred to the end of Appendix I for further information.

Chapter 2 Literature background

2.1 Virtual environments

A virtual environment (VE) can loosely be described as a computer generated three-dimensional world that mimics a real world which might exist or not. Steven Elles of NASA Ames defined virtualization as "the process by which a human viewer interprets a patterned sensory impression to be an extended object in an environment other than that in which it physically exists" [1]. The patterned sensory impressions are delivered to the senses of the human through computer generated output, and might include visual, auditory, tactile and kinesthetic output. An ideal *immersive* virtual environment is one where *all* of the user's senses are continually supplied by computer generated output. The immersivity of a virtual environment can therefore be defined as the degree to which some or all of the senses are stimulated. For example, the visual sense is stimulated by looking at a computer monitor, but the user is much more immersed when wearing a head mounted display (HMD) with head tracking. The computer or process controlling the virtual environment should also be able to obtain inputs and commands from the user in order to update the simulation appropriately. Various technologies such as keypads, mice, joysticks, body tracking and treadmills can be used to accomplish this. Again, the more natural the input device is and the more senses it incorporates, the more immersive the world becomes.

2.2 Virtual humans

One of the fundamentals of effective virtual environments is the "sense" of presence and degree of immersion [2]. In order to utilize the maximum potential of the virtual world, the user must suspend belief in the physical real world *outside* in order to allow him/her to be

immersed *inside* the virtual world. This sense of being somewhere else other than where the real physical body is located is heightened by the concept of having a *virtual body*, not only for oneself, but also for other creatures possibly inhabiting the virtual world. For example, when looking down while wearing a HMD with head tracking, the user should see a computer generated body that responds to their own. A *virtual human* can therefore be defined as the visual embodiment of a human participant in a virtual environment, also sometimes referred to as an *actor* [3].

There can be distinguished between two types of virtual humans, an *agent* and an *avatar* [3,4]. An agent is a virtual human representation that is created and controlled by computer programs. An avatar is a virtual human controlled by a live participant. Most of virtual human technology and concepts apply to both agents and avatars, except in the underlying control and behaviour strategies. There will not be an explicit distinction between the two in the rest of this document, except where necessary.

According to [4], synthetic actors can be classified into four groups:

- Pure avatars or clones
- Guided actors
- Autonomous actors
- Interactive-perceptive actors

Pure avatars or clones

Pure avatars are always fully controlled by a human participant and can therefore not be agents. Interactive or direct control of the virtual human involves measurement of the full body of the live participant, or the reduced tracking of end effectors. In the latter case, the use of inverse kinematics techniques can be used to find or interpolate the missing data. Electromagnetic six degree of freedom sensors are a popular way of animating a virtual human. Facial animation can be accomplished by continuously texture mapping live video onto the avatar's face, or by facial expression recognition and synthetic regeneration.

Guided actors

Guided actors are driven by users, but do not correspond directly to the user's motion. These actors are also controlled by live participants and can therefore not be agents. Participants can use various input devices to control or steer the actors. For example, a joystick supplies incremental position and orientation changes to the computer, which can then calculate and update the position and orientation of the actor in the VE. Similarly the user can select from a menu of predefined actions and facial expressions, which can then be mapped in real-time onto the actor.

Autonomous actors

Autonomous actors are computer-controlled entities that should be able to demonstrate behaviour and conduct themselves. These actors are mostly agents, and therefore not driven by human control. These virtual humans should have visual, tactile and auditory senses to determine their behaviour and actions in the VE. For example, rendering the environment through the agent's point of view can provide visual information. The positional and semantic information of audible sound events can provide auditory information. Multisensor collision detection attached to the articulated figure could trigger actions when the actor touches other actors or objects. Facial animation and expressions should be controlled by perception of the environment and by emotional state.

Interactive-perceptive actors

Lastly interactive-perceptive actors are similar to autonomous actors, but are aware of, and can communicate with other actors and real people.

2.3 Networked environments

The concept of virtual humans come to the fore when the same virtual environment is shared by multiple participants connected from different hosts across a network. The participant's local host stores the whole or a subset of the virtual world, and the participants use their own representations or avatars to move around in the scene. The

avatar representation has the following functions in addition to those in single-user environments [5]:

- Perception (see who's around)
- Localization (where are the other inhabitants)
- Identification (recognize the other inhabitants)
- Visualization of the other inhabitants' focus of interest
- Visualization of the other inhabitants' actions (what the others are doing and gesturing)
- Social status of self and others through decoration of the avatar

Using virtual humans in this manner, fulfills these functions realistically and provides a natural way of controlling and behaving ourselves in the VE, as well as interacting with other participants. Even a rudimentary guided actor with limited sensor information can successfully represent a user's activities and intentions.

In [5] it is noted that virtual human research has developed to such an extent that extremely realistic looking synthetic humans can be animated with credible behaviours and controlled on multiple levels. At the same time many networked VE (NVE) systems have been created using various types of network and computer architectures. The efficient fusion of these technologies is one of the next steps in constructing realistic networked virtual environments. Inserting virtual humans into the NVE is a complex task, and one has to consider the scalability of the system, the complexity of the human model with regard to appearance and animation, and the impact on network requirements such as bandwidth and latency.

2.4 Virtual human modeling

Humans and human motion have been studied for many years, and it is a natural step to extend the concepts of these physiological studies of humans to a biomechanical format that is mathematically tractable for computer simulation. Computer-based modeling, simulation and animation of humans have attracted considerable attention in computer

graphics for many years. Recent developments have resulted in highly detailed computer generated characters portraying real actors in films like *Titanic* and *Star Wars*. Virtual humans share the same set of facets as other representation and animation activities, namely *modeling, animation, motion control, appearance and rendering*. Different applications require different levels of each. In [3], the state of virtual human modeling is characterized along five dimensions:

i. **Appearance:** wireframe, polygons, freeform deformations, clothing, equipment, etc.
ii. **Function:** cartoon, jointed skeleton, strength limits, skills, roles, etc.
iii. **Time:** off-line animation, interactive manipulation, parameterized motion synthesis, crowds, coordinated teams, etc.
iv. **Autonomy:** drawing, scripting, interacting, communicating, leading, etc.
v. **Individuality:** generic character, cultural distinctions, personality, gender and age, specific individual, etc.

Table 2-1 (adapted from [8]) summarizes the basic capabilities and requirements for a number of applications in terms of these dimensions, with a higher value indicating a higher complexity. Naturally, as computer hardware and software evolve, these values will also change.

Table 2-1: Capabilities and requirements of virtual human applications.

Application	Appearance	Function	Time	Autonomy	Individuality
Cartoons	*High*	*Low*	*High*	*Low*	*High*
Games	*High*	*Low*	*Low*	*Medium*	*Medium*
Special effects	*High*	*Low*	*High*	*Low*	*Medium*
Medical	*High*	*High*	*Medium*	*Medium*	*Medium*
Ergonomics	*Medium*	*High*	*Medium*	*Medium*	*Low*
Education	*Medium*	*Low*	*Low*	*Medium*	*Medium*
Tutoring	*Medium*	*Low*	*Medium*	*High*	*Low*
Military	*Medium*	*Medium*	*Low*	*Medium*	*Low*

2.4.1 Motion and animation

The most effective way to represent motion and animation in a virtual human is to model the actual physical human skeleton as closely as possible. The human skeleton is quite a complex structure and it is usually necessary to use a simplified model, depending on the application and scope of the virtual human [3,7,8]. The skeletal model is a hierarchy of joint rotation transformations. The body is animated and moved by changing these angles as well as the global position of the body. The joints have either one, two or three degrees of freedom, depending on the accuracy and particular physical joint being modeled. Being a hierarchical representation, the joint locations and angles are specified relative to its parent.

There exist a number of methods to specify the posture of a skeleton figure at a given point in time. The most basic form is forward kinematics, where all the joint angles are specified, starting at the top of the hierarchy and working down. This is a non-intuitive method as it is extremely difficult to control and visualize the total effect. Inverse kinematics is a technique that has been studied extensively in the robotics field [9,10], and enables the animator to start at the bottom of the hierarchical chain and specify the position and/or orientation of the end-effector. This is a much more natural interface, but it allows non-realistic and impossible position and orientations, since it is an underspecified optimization problem. Much recent research has concentrated on methods to naturally constrain inverse kinematics methods [10,11], while still keeping the intuitive interface it provides. Animation over time is generally accomplished by key framing [12,13]. Once a set of key frames or postures has been specified on certain time intervals, the computer interpolates the postures to obtain the in-between frames. The interpolation algorithm can vary from simple linear interpolation to smooth spline-based interpolation. However, kinematic methods, both forward and inverse, require considerable effort to produce the realistic movement we have come to expect from our experience with the physical laws of the real world.

Physically-based animation techniques make use of the laws of physics to generate motion. Applying time varying forces and torques to the model controls the simulation and animation. Techniques for dynamic motion control can be categorized as either forward dynamic methods or inverse dynamic methods. Similar to forward kinematics, forward dynamic simulation involves the explicit application of forces and torques to objects, and then solving the equations of motion for small time steps. Articulated skeletons usually requires a large set of simultaneous equations, since there will be one equation of motion for each degree of freedom. A number of approaches have been proposed to solve this set of equations, such as the Gibbs-Appell formulation [9] and the recursive Armstrong formulation [9,14]. The latter can be implemented in real-time for reasonably simple articulated skeletons, but highly detailed articulated structures generally require high-speed processors. The interaction between body parts also complicates matters, and introduces numerical instabilities due to stiff sets of equations. Forward dynamics are usually used when a set of initial conditions (or initial postures) is available, and the goal is to compute the resulting motion or animation.

Inverse dynamic methods automatically determine the force and torque functions necessary to accomplish a stated goal. In the simplest case, the complete description of the motion is available, and the aim is to determine the forces and torques that reproduce the motion under forward dynamic simulation. More recent use of forward dynamic techniques involves the specification of high-level goals and constraints, such as "pick up the cup" or "do not touch the table", from which the dynamic simulation calculates the forces and torques required to meet the goal. The use of constrained optimization plays an important part in these techniques [15,16,17]. Appendix II presents a formulation for a practical recursive dynamic simulation algorithm.

A further animation technique that is currently very popular, is direct recording of human motion, referred to as *motion capture*. Various types of sensors or trackers are attached to the body parts to be sampled, and the joint angles can either be directly measured or indirectly calculated through the use of inverse kinematics. The recorded data can then be edited, modified and blended to generate the motion for the animation. We use motion

capture as the primary source of human motion data, and the subject is discussed in more detail in chapter 3.

2.4.2 Control

Although virtual human control shares some of the same aspects discussed in the previous section on motion and animation, it also encompasses higher level mechanisms such as locomotion, facial expressions and natural language interfaces. Pure avatars are directly controlled by a human and the term *control* is not really applicable, but rather mapping. The other types of virtual humans all to some degree share high level control mechanisms. The dynamic simulation methods all require an additional control algorithm in order to produce useful motion and animation. Control of virtual humans is classified in [3,8] as interactive, autonomous, gesture control, attention control, locomotion and multi-agent task allocation.

Motion generators or *motor skills* are defined by [8] as the procedures to change and animate the virtual human. These include replaying a stored motion sequence, posture and balance adjustments, reaching, grasping and other gestures, locomotion such as walking, running and climbing, looking and other head gestures, facial expressions, physical force or torque-induced movements and blending of movements. An important fact is noted that all of these activities may be executed simultaneously, which leads to the Parallel Transition Network structure. Parallel Transition Networks or PaT-Nets is a model for a parallel virtual machine that animates graphical models. Diagrammatically it consists of a network of nodes and arcs where nodes represent processes and arcs represent predicates, conditions, rules or other transition functions. PaT-Nets provide a non-linear time approach to animation and is a step towards autonomous behaviour. In [7,8], a natural language interface called a Parametric Action Representation (PAR) is defined, which is a conceptual representation of actions, objects and agents that is simultaneously suitable for execution as well as natural language expression. The PAR technique was implemented on the *Jack* animation system (developed at the University of Pennsylvania) with positive results. *Jack* also possesses other high level control mechanisms such as guided path

generation, obstacle avoidance and footstep placement. *Jack* is currently being used in a wide variety of ergonomic, medical and military virtual environment applications [7].

Perlin and Goldberg developed the Improv system [18,19], in which synthetic animation techniques were used to create autonomous character behaviour. Improv uses a layered architecture with a behaviour engine for selecting among a set of higher level behaviours. Improv allows control at several levels to construct a virtual environment with interactive characters exhibiting distinct personalities. The Miralab at the University of Geneva and the Computer Graphics Laboratory at the Swiss Federal Institute of Technology have been researching human animation, motion and control for many years. Recent focus on real-time virtual humans for synthetic environments has resulted in the development of control methods for avatars with high degrees of freedom, autonomous walking and grasping motions and the animation of crowd behaviour. They are also investigating the use of natural language and speech as virtual human control mechanisms [4,5,6]. The Georgia Institute of Technology is researching the field of multi-agent behavioural control [20-23]. Dynamically simulated characters are animated using various control methods. Steady state motion, turning and obstacle avoidance algorithms have been demonstrated using a herd of hopping robots and a group of eighteen cyclists as examples.

2.4.3 Appearance

Graphical human figures started out as just a collection of 3D points or lines. Surface representation was limited to a dense collection of points or lines. Early hardware limitations made acceptable update speeds and detail very difficult. Polygon meshes soon replaced points and lines. Polygons are sized, shaped and tiled to completely cover the surface at some resolution. True curved surfaces are used in animation packages, but current display hardware technology makes them awkward for real-time manipulation. Virtual humans or avatars can be portrayed in a number of ways, such as 2D icons, cartoons, composited video, 3D sprites or full 3D bodies. Full control over articulation together with realistic surface detail is only possible in full 3D bodies, and in this study we will concentrate on the latter.

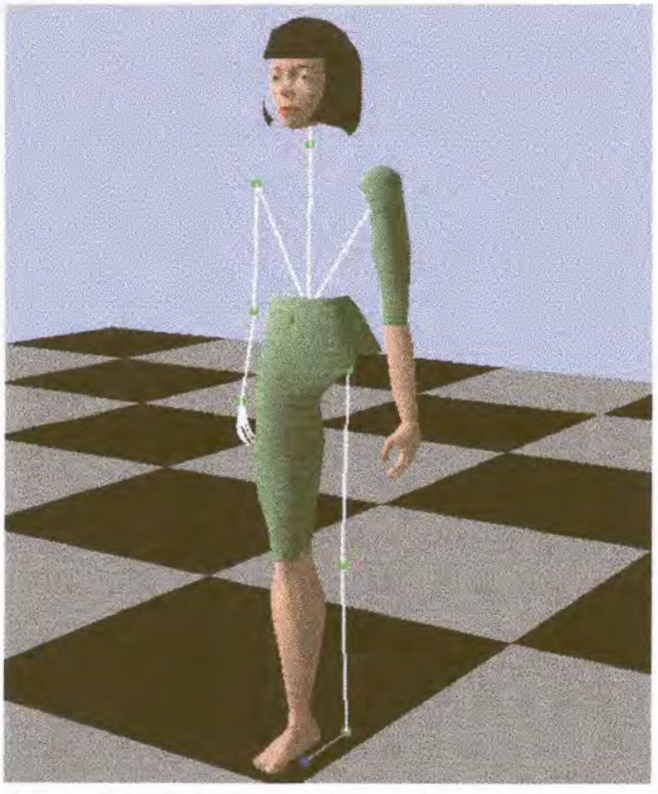


Figure 2-1: Hierarchical skeleton with partial solid surfaces.

A full three-dimensional human representation generally revolves around an articulated hierarchical skeleton (such as discussed in the previous section) with some form of solid surface wrapped around it. Such a skeleton is shown in figure 2-1, with solid surfaces added for the head, hips, right leg and left arm. Skin detail can be added by colouring or texture mapping the surface. Clothing can be added either as a separate surface layer or by modifying the skin surface directly. Facial expression is usually a combination of texture mapping and mesh animation. The surface layer can be implemented as either a rigid polygonal mesh or a smoothly deformable mesh.

Examples of current virtual human figure and appearance technology are the *Jack* figure that originated from the University of Pennsylvania, and the figures developed at the Miralab in Switzerland. The *Jack* figure evolved from a polygonal model with rigid segments and joint limits accurate enough for ergonomic evaluations [7]. The smooth body [24] was developed to aid in the portrayal of visually appealing virtual humans using free-

form deformation techniques [25]. The Miralab at the University of Geneva and the Computer Graphics Laboratory at the Swiss Federal Institute of Technology have invested considerable effort in developing and integrating several different human modeling techniques [6]. The human figure is divided into head/face, body and hand sections, each using a different method for modeling, animation and display. The head is modeled with polygon meshes that are deformable using rational free-form deformations (FFDs) [26]. The body is modeled using a multilayered metaball approach [27], and animated using deformable cross sectional contours, which results in a visually appealing figure at real-time speeds. The hands are modeled as a three layer structural approach; skeleton, muscle and skin. The muscle layer deforms the skin polygon mesh by a geometrical mapping of the skeleton layer using Dirichlet FFDs [25,28].

2.5 Virtual human communication

The concept of a shared or networked virtual environment (NVE) has already been introduced. One of the key elements in a NVE is communication among the various entities inhabiting the world. There are different forms and formats of communication, but a NVE generally implies geographically separated hosts connected across a bandwidth-limited channel. By introducing highly articulated virtual humans into the NVE, network traffic can increase dramatically, especially as the number of participants increases. For example, representing the posture of a single virtual human requires as many as 175 parameters [5,29]. If these parameters are transmitted as floating point values, then a staggering 168 Kb/s is needed at an update rate of 30Hz. Even by discretizing the parameters to two or one byte quantities still exceeds the capabilities of current dial-up network hardware. Additionally, the very nature of a NVE requires *multiple* virtual humans to be present and there is clearly a communications bottleneck. Body postures and actions, similar to traditional speech and video compression, can be communicated in more compact forms by accepting some loss in quality or accuracy. Inserting multiple virtual humans into a NVE is a relatively new research field. Important research in this area has been done at the University of Geneva and the Computer Graphics Laboratory and the

Swiss Federal Institute of Technology using the Virtual Life Network (VLNET) system [5], and at the Naval Postgraduate School using the NPSNET system [30-33].

The VLNET system aims to integrate artificial life techniques with virtual reality techniques to create convincing virtual environments that can be shared by real and autonomous actors. The VLNET system is comprised of cooperative processes, each responsible for a certain task. The two main processes are the core VLNET process and the external driver processes. The core process executes the main simulation, interfaces with the driver processes, does external communication and maintains the display, cull and database traversal functions. The main simulation consists of four logical units or engines, namely *the object behaviour engine, the navigation engine, the body engine and the face engine*. The external driver processes provide a simple and flexible means to access and control all the complex functionalities of VLNET. The *facial expression driver* controls the expression of a user's face, and uses either predefined expressions, or maps real-time expressions using MPAs [6]. The *body posture driver* controls the motion and animation of the user's body, using either direct motion control with motion capture devices, or higher-level control techniques. The *navigation driver* is used for basic navigation, hand movement, head movement and basic object and system control. The *object behaviour driver* controls the dynamic behaviour, such as motion and scaling of objects within the virtual world.

In [5] it is noted that natural human communication is based on speech, facial expression and gestures. All of these should be supported in a NVE, and should be captured, transmitted and faithfully reproduced for the other participants on remote sites. The work done on VLNET concentrates mainly on facial expression and body posture communication. Facial expression is done by either video texturing the face or using model based coding of facial parameters. Body movements are divided into three groups, namely *instantaneous gestures*, *gesture commands* and *rule-based sign language*. All these movements can be generated either by direct tracking or by predefined postures and gestures. In [34] a technique for interacting with virtual humans is presented using a

gesture/posture recognition system, based on a top-down refinement of the characteristic levels of an action.

The NPSNET system has investigated integrating virtual human figures into a Distributed Interactive Environment (DIS) compliant environment. The system includes level of detail representation, body tracking technology and a set of postures. The level of detail is selected depending on view volume, range and maximum distance of interest. The upper body is animated using a tracker attached to each hand and one to the head. The lower body animation is accomplished by a set of predefined motions such as upright, walking, kneeling and crawling. The *Jack* model [7] has been used in this application. The body posture is sent through the standard DIS protocol data units. Additional provision is made for group behaviour.

2.5.1 Networking

It has already been noted that complex representation of virtual humans can create significant loads on network resources when compared to traditional, low degree of freedom objects such as vehicles and aircraft. Network tasks can be separated [29] into two stages, namely *setup* and *simulation*. Setup refers to the stage when a new participant joins the virtual world, sends its embodiment information to all the other participants and loads the scene (including the embodiment of the other inhabitants). The simulation stage refers to the continuous transmission and reception of update information. During simulation, the communication can be decomposed into:

- Transformation of body parameters to a network message
- Compression of the message
- Decompression at the receiver site
- Inverse transformation to a list of body parameters

The transmission lag or delay is defined as the sum of the lag for each step. Two types of data compression can be considered. *Lossless* compression guarantees an exact replica of

the original data at the receiving end. *Lossy* compression introduces a controlled amount of distortion into the original data in exchange for better compression. Virtual human data is inherently a candidate for lossy compression, since both the body tracking of the participant, and the animation and graphical appearance of the human already introduce a loss in resolution. In [29] a number of compression schemes are investigated, such as multiple levels of joint resolution, transmission of a subset of joints, arithmetic coding and predictive coding, the latter being the proposed approach taken in the MPEG-4 standard.

In [35] a *dead reckoning* algorithm is investigated as a method to reduce the amount of network transmissions. Dead reckoning uses velocity and acceleration information to extrapolate a position or orientation variable, and has been used extensively for non-articulated objects in the DIS system [30,31]. A Kalman filter is used to estimate the joint value, velocity and acceleration. This information is then used to update the dead reckoning algorithm. The results in [35] are not very encouraging, as only a 50% decrease in messages is reported. Clearly more than this is required for acceptable multiple virtual human communication.

In [36] a number of methods are investigated specifically for the purpose of facial parameter compression. Motion interpolation is investigated as a method to reduce, on a high level, the amount of parameters that need to be sent. A technique used in large scientific databases [37] is used as a spatial compression method to decompose the original facial parameter space to a reduced subspace. Temporal redundancy is investigated using predictive coding techniques. The discrete cosine transform (DCT) is used as a transform based compression method. A combination of the above methods is also investigated. Promising results for facial parameter compression were obtained, but the results for body motion parameters are unclear.

2.6 Virtual humans and MPEG-4

MPEG-4 [38] addresses networked multimedia applications such as video telephony, networked virtual environments and games, distance learning, remote presentation and

other applications that require interaction, transmission and the combination of natural audio and video streams with synthetic 2D and 3D computer graphics models. The objective of this synthetic/natural hybrid coding (SNHC) scheme is to facilitate content-based manipulation, interoperability and wider user access in the delivery of animated mixed media. Standard compressible A/V objects included are audio, video and 2D/3D computer graphics. MPEG-4 defines the *standard* for mesh-segmented video coding, compression of geometry, synchronization between A/V objects, multiplexing of streamed A/V objects and the spatial-temporal integration of mixed media types, but does not necessarily define the exact *method* for doing so.

Within the MPEG-4 effort is a subgroup (SNHC) that works on efficient coding of graphics models and compressed transmission of the animation parameters specific to the model type. Initial information has been released [39] on the A/V object that specifies face and body definition and animation (referred to as the face and body animation (FBA) object). Detailed body and face shape and texture is provided, as well as parameters to control expression, posture and animation. The FBA object is divided into separate sections for the head and body. The head is described by the facial definition parameters (FDPs) and the facial animation parameters (FAPs). In a similar fashion, the body definition parameters (BDPs) and body animation parameters (BAPs) describe the virtual human body, excluding the head.

Due to extensive research in the field of face modeling prior to the existence of MPEG-4, the FDPs and FAPs are well defined. It is based primarily upon the well-known facial action coding system (FACS) developed by [40]. MPEG-4 adopts a model based approach that allows user-defined face models to communicate with each other without the standardization of a common face model. The result is the definition of 68 facial animation parameters that must be supported by all MPEG-4 decoders. Of these, two are high-level parameters and the rest are low-level parameters such as actual jaw or eye movement. The high-level parameters are visual phonemes (derived from the phoneme concept in speech analysis) called *visemes*, and expression parameters.

The body animation parameter set contains a set of angles that specify for every joint in the figure its possible motions. For example, the parameters for the arm include the wrist, elbow, shoulder and clavicle joints. BAPs can be used in grouped form (such as the whole leg) or in non-grouped form (individual joints). Currently the BDPs and BAPs are not as well defined as the FDPs and FAPs. Furthermore, the use of BAP compression techniques is still a relatively new concept compared to the ongoing research in model based facial compression and communication.

2.7 Summary

In this chapter some of the basic definitions and concepts of virtual environments, virtual humans and virtual human motion communication were reviewed. Current and previous work done in this field were investigated and, as has been concluded by others, it became clear that fully articulated virtual human communication could present a serious bandwidth problem to existing networked virtual environments. There is a definite need for the use of compression techniques when inserting virtual humans into virtual environments. MPEG-4 addresses the problem directly, but there is a distinct gap between the current state of research for facial parameter compression and full body parameter compression.

Chapter 3 The human model

Human modeling can be described as the embodiment of all human characteristics within the context of computer databases and programs. There are a number of different model types [7]:

- **Mathematical formulations:** physical equations of motion, limb strength, measuring workload and fatigue.
- **Geometric and topological models:** object structures, body segments, joints and joint limits, reach and constraints.
- **Conceptual models:** names, attributes, flexibility, materials, functions and relationships.

Usually only a few of these dimensions are used, depending on the application of the model. In this study, the interest does not lie in the specific application of a virtual human, but rather in the minimum amount of capabilities required to fulfil such a possible application. Of the model types shown above, the first two is of most interest.

3.1 Physical model

Humans are physically comprised of bones, tissue and skin. These are all intricately connected together to form a highly complex and highly functional hierarchical structure. In order to obtain quantitative human data suitable for computer modeling and animation, accurate estimation of body segment parameters such as size, volume, mass, center of mass and moments of inertia is required. A number of techniques have been used to measure these quantities, including gamma-ray scanning [41], magnetic resonance imaging [43] and

on a more somber note, cadaver dissection. Muscles are the primary source of human motion, and extensive models have been developed in the biomechanics community using empirical measurements. Due to skeletal simplifications that will be discussed below, these muscle models are overly complex and inappropriate for real-time virtual human implementation. We use techniques from the robotics literature to approximate complex muscle functionality by applying forces and torques directly to the degrees of freedom of a joint.

3.2 Hierarchical model

In order to obtain a mathematically tractable human model, we first need to define a hierarchical, articulated structure similar to the physical human skeleton. The actual human skeleton is highly intricate, but we can reduce the complexity by approximations suitable for our needs, such as for interactivity and for external motion requirements. In general, we need a fully linked body model specification that includes at least a spine, neck and head section, leg and foot sections and arm and hand sections. Secondly, we need to define all the joints in the body. Individual joints can either have one, two or three degrees of freedom, and the limits of each degree of freedom must be specified. Thirdly, we need to specify the size of the skeleton according to permissible human dimensions. This also includes specification of body segment weight and shape. Effectively there is no such thing as an "average" human. We must prescribe a target population in terms of percentiles of size, weight and stature. For example, statistically speaking 5^{th} percentile legs can not be found on a 95^{th} percentile body. The statistics of the skeleton should be scalable without too much effort to suit the required task. Lastly, we should add a human-like appearance to the model (this is discussed in the next section). Obviously the skeleton on its own cannot be effectively visualized, or taken seriously, when rendered as a wire frame image. Body segments such as the torso, arms and legs are added according the size, weight and shape description of the skeleton.

An example of a quite highly articulated human skeleton is shown in figure 3-2a and figure 3-2b. This skeleton model is similar to the one that has been proposed by the SNHC group

within the MPEG-4 effort. Figure 3-2a shows the main body hierarchy, while figure 3-2b shows the hand section in more detail. Another example, which is the standard for VRML 2.0 articulated humanoid objects (referred to as H-Anim), is shown in figure 3-3. Apart from the fact that some of the joints are specified as multiple 1 DOF joints, the figure is very similar to the MPEG-4 standard.

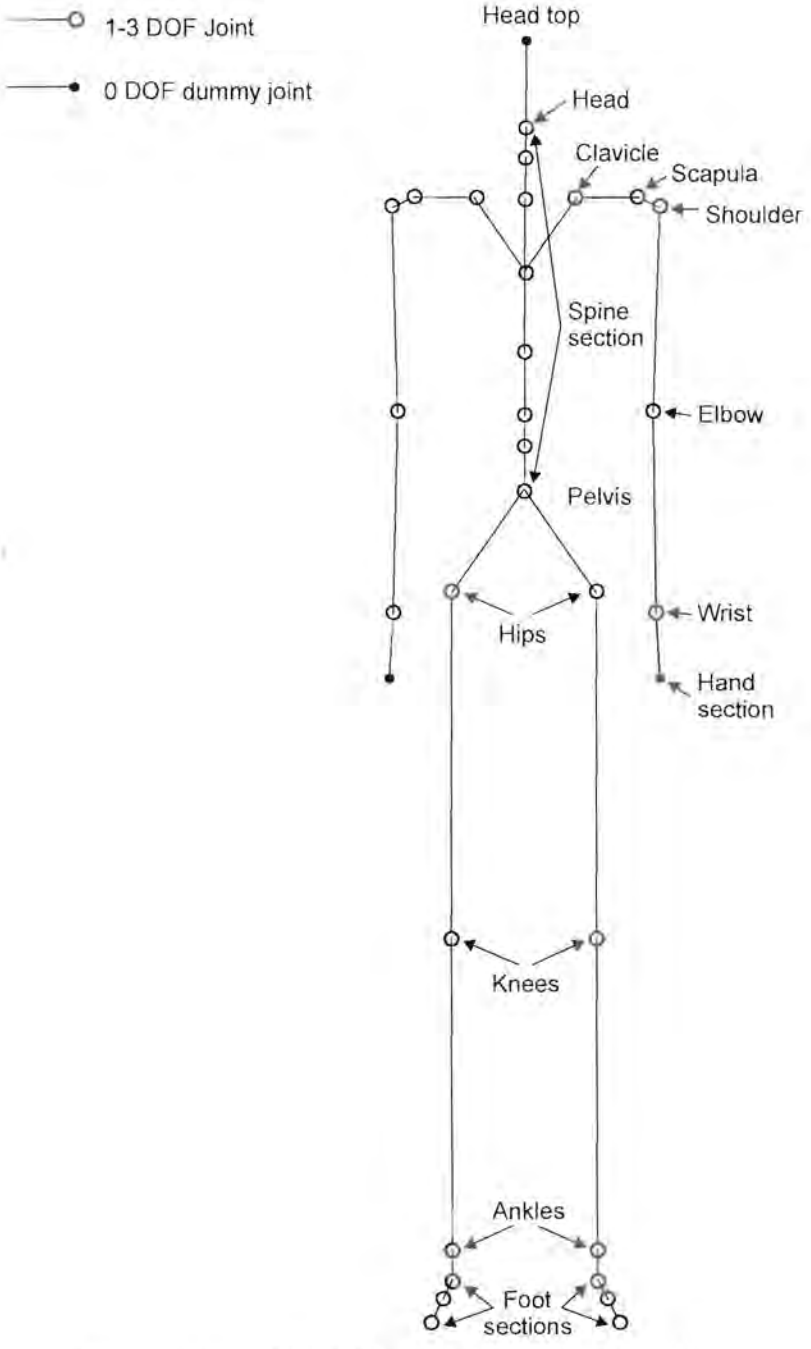


Figure 3-2a: MPEG-4 main body specification.

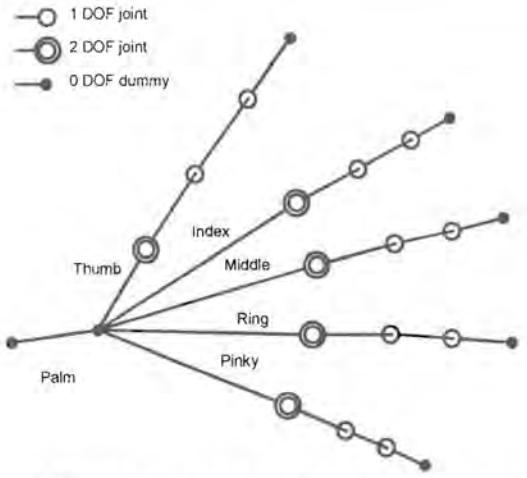


Figure 3-2b: MPEG-4 hand section specification.

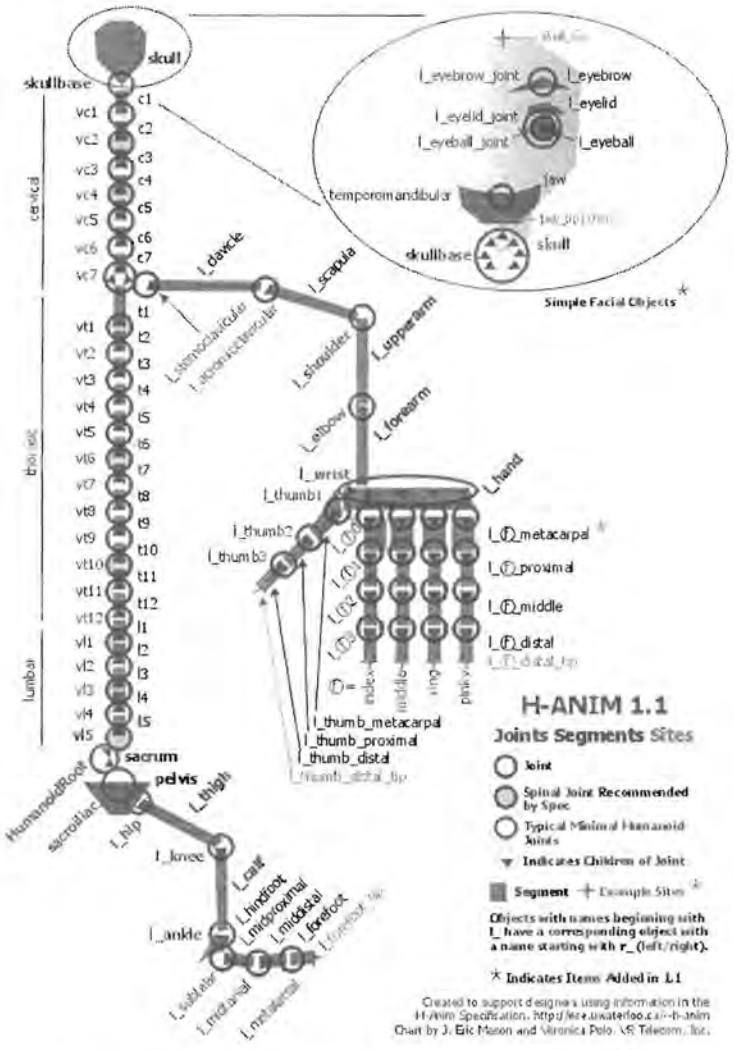


Figure 3-3: VRML 2.0 H-Anim specification.

3.2.1 Simplified human model

For the purpose of this study, a slightly simplified skeletal model was chosen, which is shown in figure 3-4a and 3-4b. The reasons for doing so are as follows:

- The interest lies in obtaining a useful set of core parameters for research purposes, and not in animating and rendering a highly detailed human figure;
- There is a limit to the amount of joint parameters that can be measured due to hardware constraints.

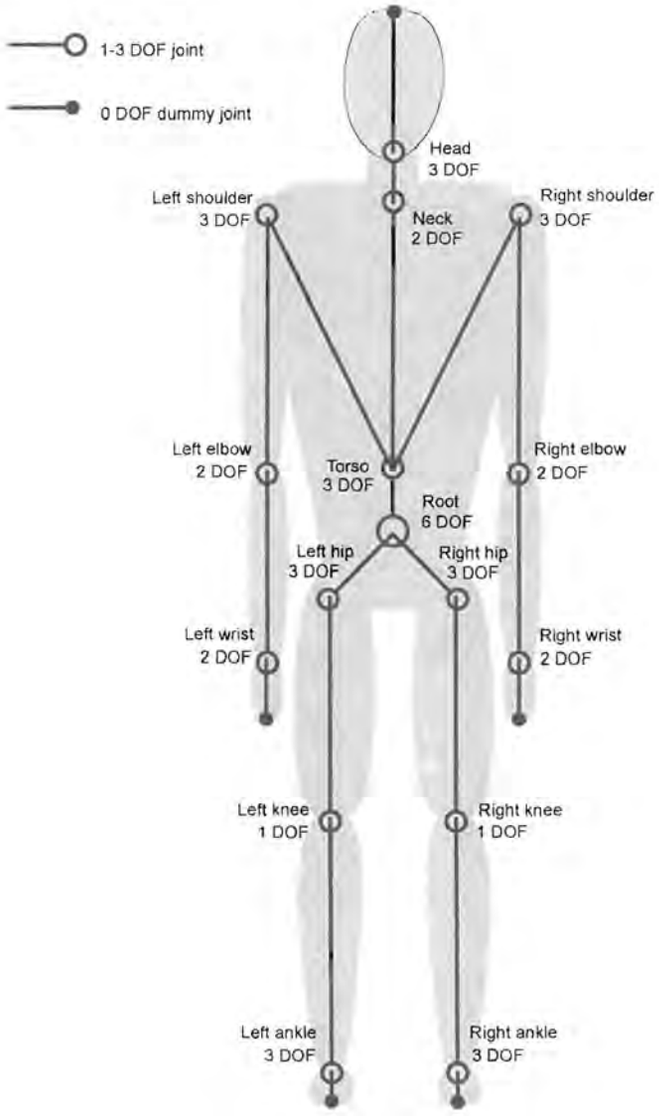


Figure 3-4a: Simplified human body model.

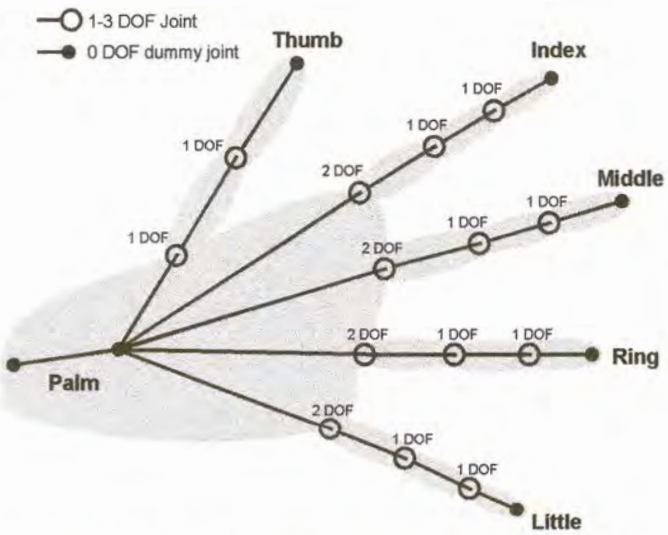


Figure 3-4b: Simplified human hand model.

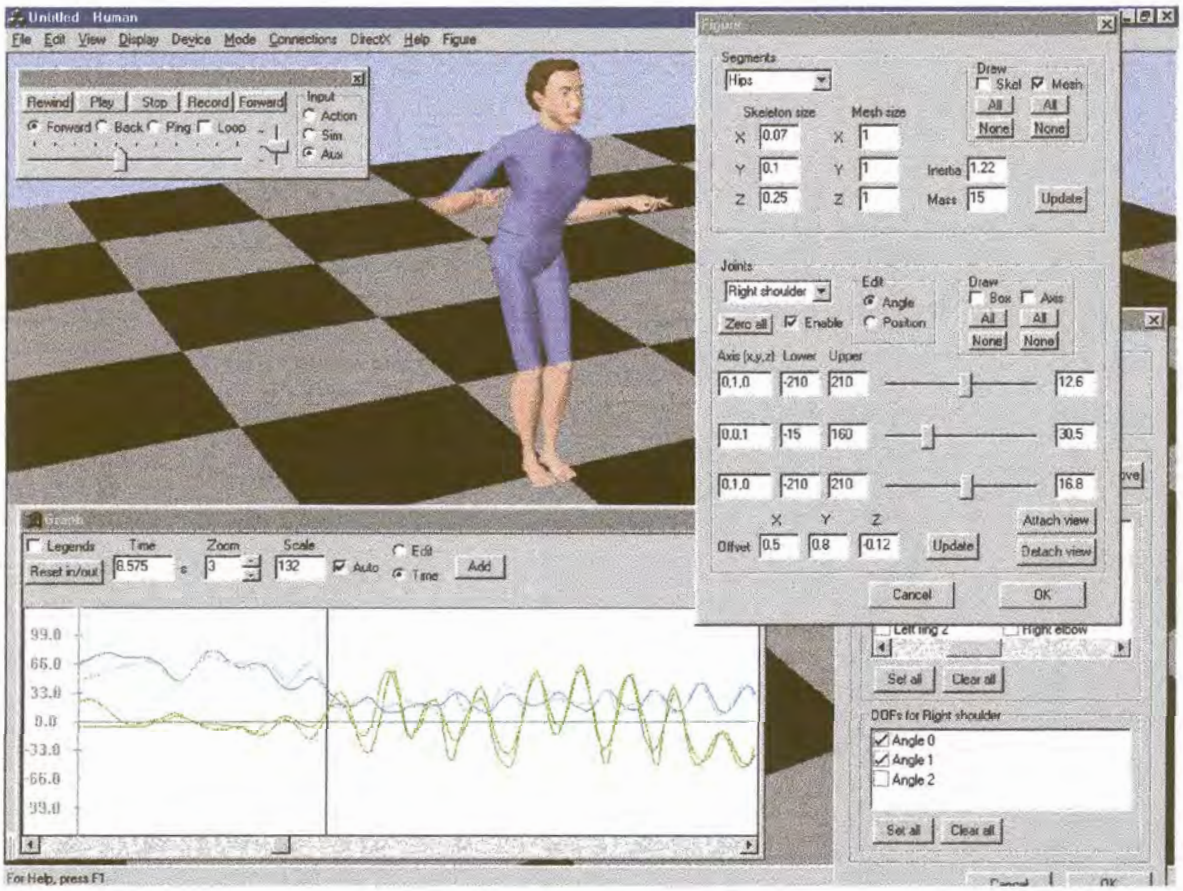


Figure 3-5: Graphical user interface for the human model.

The basic building blocks for the figure are *segments* and *joints*. Segments are body parts such as the head, torso and arm or leg limbs and are approximated by the shaded areas in figures 3-4a and 3-4b. Segments are connected to each other through joints. Segments are defined by parameters such as size and weight, while joints are defined by parameters such as degrees of freedom (DOF) and joint limits. The next section describes the joint layout and the following one the segment layout. All the quantities and directions are specified in a left handed coordinate system, as described in Appendix I. Figure 3-5 shows an example of the software that was developed to enable interactive manipulation of segment and joint parameters.

3.2.2 Joints

Joints are specified by the offset, DOF, limits and rotation axis or rotation order parameters. The offset parameter is a vector that indicates the offset from the parent joint to the current joint. In our model, we use a normalized joint offset scheme together with a size quantity (that is part of the segment specification). The joint offset value is normalized so that each of the x, y and z components lies between –1 and 1. This enables us to scale the human model or figure with ease using only segment size, which can vary considerably among physical humans. The other parameters such as DOF and joint limits depend on the location and type of joint, and are reasonably common across most humans. Each DOF in the human figure is denoted by $\theta_{i,j}$, where i is a zero-based integer indicating the joint number and j is a zero-based integer indicating the DOF number. Table 3-1 below shows the functional assignment for each i and j. It is convenient to separately discuss the spine and head section, the arm sections, the leg sections and hand sections. These sections are inherently independent of each other, and we also want to avoid duplicity due to left/right symmetry.

The spine and head

The spine and head section consists of 4 separate joints. The root has a 3 DOF prismatic joint and a 3 DOF revolute joint, and controls the global position and orientation of the

figure. These 6 DOFs are specified in Cartesian coordinates, and with respect to a left-handed coordinate system (Appendix I), the motion can be described by the cascaded transformations

$$\mathbf{R}_z(\theta_{0,2})\mathbf{R}_x(\theta_{0,1})\mathbf{R}_y(\theta_{0,0})\mathbf{T}(\theta_{0,3},\theta_{0,4},\theta_{0,5}), \tag{3-1}$$

where $\theta_{0,3}$, $\theta_{0,4}$ and $\theta_{0,5}$ are the prismatic DOFs (in that order) and $\theta_{0,0}$, $\theta_{0,1}$ and $\theta_{0,2}$ are the revolute DOFs. The torso joint is a 3 DOF revolute joint that controls the orientation of the base of the spine or torso section. The 3 DOFs are specified in Cartesian coordinates, and with respect to the root reference frame, the motion can be described by the cascaded rotations

$$\mathbf{R}_z(\theta_{1,2})\mathbf{R}_x(\theta_{1,1})\mathbf{R}_y(\theta_{1,0}), \tag{3-2}$$

where $\theta_{1,0}$, $\theta_{1,1}$ and $\theta_{1,2}$ are the revolute DOFs. The neck joint is a 2 DOF revolute joint that controls the orientation of the neck section. The 2 DOFs are specified in Cartesian coordinates, and with respect to the torso reference frame, the motion can be described by the cascaded rotations

$$\mathbf{R}_z(\theta_{2,1})\mathbf{R}_x(\theta_{2,0}), \tag{3-3}$$

where $\theta_{2,0}$ and $\theta_{2,1}$ are the revolute DOFs. The head joint is a 3 DOF revolute joint that controls the orientation of the head. The 3 DOFs are specified in Cartesian coordinates, and with respect to the neck frame, the motion can be described by the cascaded rotations

$$\mathbf{R}_z(\theta_{3,2})\mathbf{R}_x(\theta_{3,1})\mathbf{R}_y(\theta_{3,0}), \tag{3-4}$$

where $\theta_{3,0}$, $\theta_{3,1}$ and $\theta_{3,2}$ are the revolute DOFs.

The arms

The arms are each modeled as crude 5 DOF manipulators, with two separate joints, one at the shoulder and one at the elbow (the wrist joint will be discussed with the hands). The physical shoulder in itself is quite complex. Movement is accomplished by separate articulation of the glenohumeral, claviscapular and sternoclavicular joints. Modeling it only with a single joint introduces visual errors as well as measurement errors, especially when the arms are lifted high. Most highly articulated human shoulders are modeled with two or three joints. However, the limitations discussed above force us to model the shoulder as a single joint. It is convenient to describe the shoulder joint in terms of spherical coordinates, with an additional angle DOF specifying the upper arm "twist", which results in 3 DOFs for the shoulder. The motion relative to the torso reference frame can be described by the cascaded rotations

$$\mathbf{R}_y(\theta_{5,2})\mathbf{R}_z(\theta_{5,1})\mathbf{R}_y(\theta_{5,0}), \tag{3-5}$$

where $\theta_{5,0}$, $\theta_{5,1}$ and $\theta_{5,2}$ are the revolute DOFs for the left shoulder. The right shoulder is similar, with $i = 22$. There are two singularities when dealing with spherical coordinates, namely when the elevation is 0° or 180°. In this case, one should take care as $\theta_{5,0}$ and $\theta_{5,2}$ are poorly defined, and are susceptible to round-off errors and noisy input hardware. The elbow joint is a 2 DOF revolute joint that specifies the "hinge" and "twist" angles for the lower arm. The 2 DOFs are specified in Cartesian coordinates, and with respect to the shoulder reference frame, the motion can be described by the cascaded rotations

$$\mathbf{R}_y(\theta_{6,1})\mathbf{R}_x(\theta_{6,0}), \tag{3-6}$$

where $\theta_{6,0}$ and $\theta_{6,1}$ are the revolute DOFs for the left elbow. Another singularity occurs when $\theta_{6,0} = 0°$ or $\theta_{6,0} = 180°$. In this case, the y or twist-axis of the elbow and shoulder line up and the value of $\theta_{6,1}$ loses meaning.

The legs

The legs consist of 3 joints each, one at the hip, one at the knee and one at the ankle. Toe movement is disregarded. The hip joint is a 3 DOF revolute joint that controls the orientation of the upper leg. The 3 DOFs are specified in Cartesian coordinates, and with respect to the root reference frame, the motion can be described by the cascaded rotations

$$\mathbf{R}_z(\theta_{40,2})\mathbf{R}_y(\theta_{40,1})\mathbf{R}_x(\theta_{40,0}), \tag{3-7}$$

where $\theta_{40,0}$, $\theta_{40,1}$ and $\theta_{40,2}$ are the revolute DOFs for the left leg. The right leg uses similar notation with $i = 44$. Strictly speaking the hip joint should also be specified in spherical coordinates. However, due to the limited abduction and twist movement of the upper leg, and the fact that we cannot measure those movements directly, we left it as a Cartesian system. The knee is modeled as a simple 1 DOF hinge joint, and the movement relative to the hip reference frame is given by the rotation

$$\mathbf{R}_x(\theta_{41,0}), \tag{3-8}$$

where $\theta_{41,0}$ is the single revolute DOF variable specified in Cartesian coordinates. The ankle is modeled as a 3 DOF revolute joint that controls the orientation of the foot. The 3 DOFs are specified in Cartesian coordinates, and with respect to the knee reference frame, the motion can be described by the cascaded rotations

$$\mathbf{R}_z(\theta_{42,2})\mathbf{R}_y(\theta_{42,1})\mathbf{R}_x(\theta_{42,0}), \tag{3-9}$$

where $\theta_{42,0}$, $\theta_{42,1}$ and $\theta_{42,2}$ are the revolute DOFs.

The hands

The hands consist of a 3 DOF joint at the wrist, 2 DOF joints for the first segment of each finger and a 1 DOF joint for the rest. The thumb is modeled with 2 joints, and the other fingers with 3 joints each. The palm joint is a 2 DOF revolute joint specified in Cartesian

coordinates. The motion relative to the elbow reference frame can be described by the cascaded rotations

$$\mathbf{R}_x(\theta_{7,1})\mathbf{R}_z(\theta_{7,0}), \tag{3-10}$$

where $\theta_{7,0}$ and $\theta_{7,1}$ are the revolute DOFs for the left hand. The right hand uses similar notation with $i = 24$.The joint of the first segment of the thumb is a 2 DOF revolute joint specified in Cartesian coordinates. The rotation of the thumb is easiest described in two perpendicular planes that are not aligned with the orthogonal x-, y- and z-axis of the reference frame. The motion relative to the wrist reference frame can be described by the cascaded rotations

$$\mathbf{R}(0.5,0,1,\theta_{8,1})\mathbf{R}(1,0,-0.5,\theta_{8,0}), \tag{3-11}$$

where $\theta_{8,0}$ and $\theta_{8,1}$ are the revolute DOFs. The joint of the second segment of the thumb is a 1DOF revolute joint, and the motion with respect to the first joint is given by the rotation

$$\mathbf{R}(1,0,-0.5,\theta_{9,0}), \tag{3-12}$$

where $\theta_{9,0}$ is the single revolute DOF. Each first segment joint of the remaining fingers are 2 DOF joints specified in Cartesian coordinates, and the motion relative to the wrist reference frame can be described by the cascaded rotations

$$\mathbf{R}_z(\theta_{i,1})\mathbf{R}_x(\theta_{i,0}), \tag{3-13}$$

were $\theta_{i,0}$ is the abduction DOF and $\theta_{i,1}$ is the flexion DOF for the left hand fingers, with $i = \{10, 13, 16, 19\}$. The last two joints of the remaining fingers are simple 1 DOF hinge joints, and the motion of each with respect to its parent reference frame is given by

$$\mathbf{R}_z(\theta_{i,0}), \tag{3-14}$$

where $\theta_{i,0}$ is the single flexion DOF, with $i = \{11, 12, 14, 15, 17, 18, 20, 21\}$.

Table 3-1 summarizes the basic parameters for the male model. Shown are the internal name, normalized offset, degrees of freedom and limits for each joint.

Table 3-1: Summary of the male model joint parameters

Joint name : Parent	Normalized offset	Degrees of freedom	Lower limit [m] [deg]	Upper limit [m] [deg]	Description
Root : None		6			
		$\theta_{0,0}$	-180	180	Figure x angle
		$\theta_{0,1}$	-40	40	Figure y angle
		$\theta_{0,2}$	-30	30	Figure z angle
		$\theta_{0,3}$	$-\infty$	$+\infty$	Figure x position
		$\theta_{0,4}$	$-\infty$	$+\infty$	Figure y position
		$\theta_{0,5}$	$-\infty$	$+\infty$	Figure z position
Torso : Root		3			
		$\theta_{1,0}$	-30	30	Torso twist
		$\theta_{1,1}$	-40	40	Torso bend
		$\theta_{1,2}$	-30	30	Torso flexion
Neck : Torso		2			
		$\theta_{2,0}$	-30	30	Neck bend
		$\theta_{2,1}$	-15	15	Neck flexion
Head : Neck		3			
		$\theta_{3,0}$	-90	90	Head yaw
		$\theta_{3,1}$	-50	50	Head pitch
		$\theta_{3,2}$	-15	15	Head roll
Left shoulder : Torso		3			
		$\theta_{5,0}$	-180	180	Upper arm flexion
		$\theta_{5,1}$	-160	0	Upper arm abduction
		$\theta_{5,2}$	-180	180	Upper arm twist
Left elbow : Left shoulder		2			
		$\theta_{6,0}$	-180	0	Elbow hinge
		$\theta_{6,1}$	-90	90	Lower arm twist
Left wrist : Left elbow		2			
		$\theta_{7,0}$	-60	70	Wrist yaw
		$\theta_{7,1}$	-30	30	Wrist pitch
Left thumb 1 : Left wrist		2			
		$\theta_{8,0}$	0	60	Thumb flexion
		$\theta_{8,1}$	-20	60	Thumb abduction
Left thumb 2 : Left thumb 1		1			
		$\theta_{9,0}$	0	60	Thumb segment 2 flexion
Left index 1 : Left wrist		2			
		$\theta_{10,0}$	-5	5	Index finger segment 1 abduction
		$\theta_{10,1}$	0	60	Index finger segment 1 flexion
Left index 2 :		1			

Left index 1		$\theta_{11,0}$	0	75	Index finger segment 2 flexion
Left index 3 : Left index 2		1 $\theta_{12,0}$	0	45	Index finger segment 3 flexion
Left middle 1 : Left wrist		2 $\theta_{13,0}$ $\theta_{13,1}$	-5 0	5 60	Middle finger segment 1 abduction Middle finger segment 1 flexion
Left middle 2 : Left middle 1		1 $\theta_{14,0}$	0	75	Middle finger segment 2 flexion
Left middle 3 : Left middle 2		1 $\theta_{15,0}$	0	45	Middle finger segment 3 flexion
Left ring 1 : Left wrist		2 $\theta_{16,0}$ $\theta_{16,1}$	-5 0	5 60	Ring finger segment 1 abduction Ring finger segment 1 flexion
Left ring 2 : Left ring 1		1 $\theta_{17,0}$	0	75	Ring finger segment 2 flexion
Left ring 3 : Left ring 2		1 $\theta_{18,0}$	0	45	Ring finger segment 3 flexion
Left little 1 : Left wrist		2 $\theta_{19,0}$ $\theta_{19,1}$	-5 0	5 60	Little finger segment 1 abduction Little finger segment 1 flexion
Left little 2 : Left little 1		1 $\theta_{20,0}$	0	75	Little finger segment 2 flexion
Left little 3 : Left little 2		1 $\theta_{21,0}$	0	45	Little finger segment 3 flexion
Right shoulder : Torso		3 $\theta_{22,0}$ $\theta_{22,1}$ $\theta_{22,2}$	-180 -160 -180	180 0 180	Upper arm flexion Upper arm abduction Upper arm twist
Right elbow : Right shoulder		2 $\theta_{23,0}$ $\theta_{23,1}$	-180 -90	0 90	Elbow hinge Lower arm twist
Right wrist : Right elbow		2 $\theta_{24,0}$ $\theta_{24,1}$	-60 -30	70 30	Wrist yaw Wrist pitch
Right thumb 1 : Right wrist		2 $\theta_{25,0}$ $\theta_{25,1}$	0 -20	60 60	Thumb flexion Thumb abduction
Right thumb 2 : Right thumb 1		1 $\theta_{26,0}$	0	60	Thumb segment 2 flexion
Right index 1 : Right wrist		2 $\theta_{27,0}$ $\theta_{27,1}$	-5 0	5 60	Index finger segment 1 abduction Index finger segment 1 flexion
Right index 2 : Right index 1		1 $\theta_{28,0}$	0	75	Index finger segment 2 flexion
Right index 3 : Right index 2		1 $\theta_{29,0}$	0	45	Index finger segment 3 flexion
Right middle 1 : Right wrist		2 $\theta_{30,0}$ $\theta_{30,1}$	-5 0	5 60	Middle finger segment 1 abduction Middle finger segment 1 flexion
Right middle 2 : Right middle 1		1 $\theta_{31,0}$	0	75	Middle finger segment 2 flexion
Right middle 3 : Right middle 2		1 $\theta_{32,0}$	0	45	Middle finger segment 3 flexion

Right ring 1 : Right wrist		2 $\theta_{33,0}$ $\theta_{33,1}$	 -5 0	 5 60	 Ring finger segment 1 abduction Ring finger segment 1 flexion
Right ring 2 : Right ring 1		1 $\theta_{34,0}$	 0	 75	 Ring finger segment 2 flexion
Right ring 3 : Right ring 2		1 $\theta_{35,0}$	 0	 45	 Ring finger segment 3 flexion
Right little 1 : Right wrist		2 $\theta_{36,0}$ $\theta_{36,1}$	 -5 0	 5 60	 Little finger segment 1 abduction Little finger segment 1 flexion
Right little 2 : Right little 1		1 $\theta_{37,0}$	 0	 75	 Little finger segment 2 flexion
Right little 3 : Right little 2		1 $\theta_{38,0}$	 0	 45	 Little finger segment 3 flexion
Left hip : Root		3 $\theta_{39,0}$ $\theta_{39,1}$ $\theta_{39,2}$	 -90 -15 -30	 90 15 15	 Hip flexion Hip twist Hip abduction
Left knee : Left hip		1 $\theta_{40,0}$	 0	 180	 Knee flexion
Left ankle : Left knee		3 $\theta_{41,0}$ $\theta_{41,1}$ $\theta_{41,2}$	 -30 -30 -15	 30 30 15	 Ankle flexion Ankle twist Ankle abduction
Right hip : Root		3 $\theta_{43,0}$ $\theta_{43,1}$ $\theta_{43,2}$	 -90 -15 -30	 90 15 15	 Hip flexion Hip twist Hip abduction
Right knee : Right hip		1 $\theta_{44,0}$	 0	 180	 Knee flexion
Right ankle : Right knee		3 $\theta_{45,0}$ $\theta_{45,1}$ $\theta_{45,2}$	 -30 -30 -15	 30 30 15	 Ankle flexion Ankle twist Ankle abduction

From table 3-1 it can be seen that there is a total of 46 joints and 80 degrees of freedom for our simplified human figure. This is more than adequate for academic purposes, and in most cases we will work only with a subset of these.

3.2.3 Segments

The model shown in figure 3-4 consists of 14 segments excluding the hands. Each hand consists of a palm segment and 14 finger segments. The whole figure therefore has 44 segments. Each segment is defined by parameters such as size, weight, moment of inertia and appearance. The size parameter is used in conjunction with the normalized joint offset

value to obtain exact location of the joint relative to its parent. The size is a vector that specifies the largest distance from the segment pivot point to all the child pivot points. Figure 3-6 illustrates this concept, using the torso segment as an example. Shown are the torso segment x and y sizes (z size not shown), the torso joint, the neck joint, and two shoulder joints.

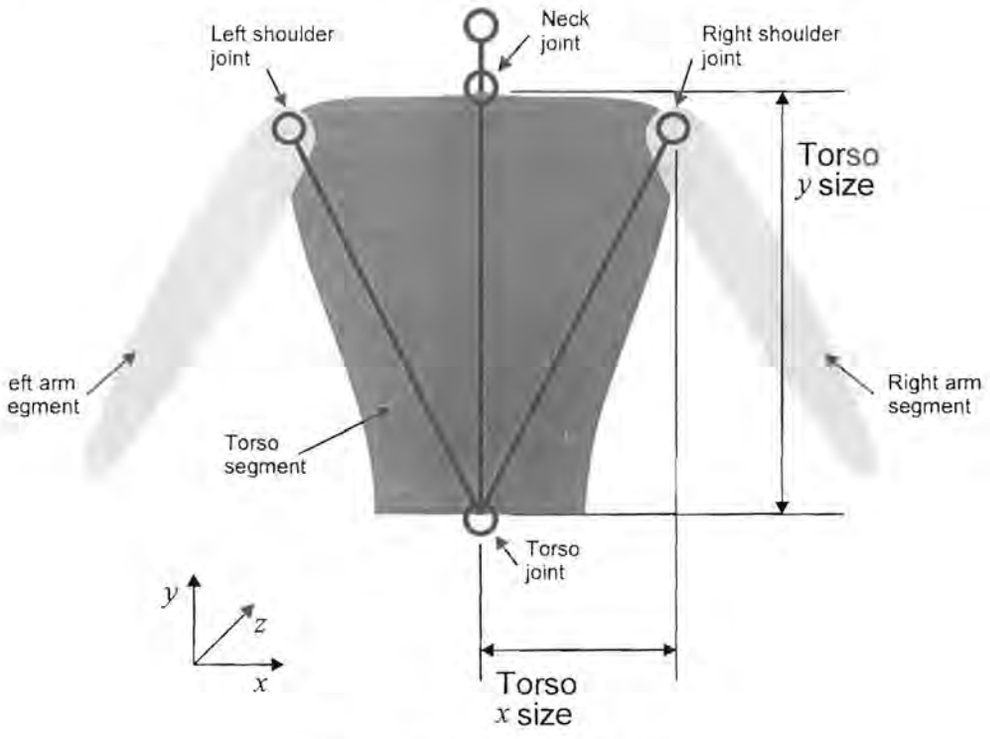


Figure 3-6: Segment sizes.

The moment of inertia is a function of mass (or density) and shape, and can be quite difficult to calculate for complex structures. In this case, we approximate the shape of each segment with a known geometrical shape such as a box, a cylinder, a cone or a sphere, as shown in figure 3-7. The density of the segment is assumed to be constant. Using this approximation, the inertia is described by a few parameters such as length, width or radius and axis of rotation, and can be easily calculated.

Figure 3-7: Approximated segment shapes, excluding fingers.

The appearance of the segment is defined by a polygon mesh structure, and is totally independent of the actual hierarchical model. We can therefore use any shape, method or texture to describe the appearance of the figure. This is discussed in the next section. As is the case with joints, the segment parameters can vary dramatically from human to human. Table 3-2 summarizes the basic parameters for our male model, adapted from [42]. Indicated are the internal segment name, size, weight and approximate shape for each segment. The moments of inertia (or rather the inertia tensor, which is a matrix containing the moments of inertia of all the possible axis and combinations) can be calculated from these parameters using standard formulas. We take the average weight of a male figure to be 75 kg, and of a female figure to be 60 kg.

Table 3-2: Summary of the basic male model segment parameters.

Segment name	Size x, y, z [m]	Weight [kg]	Shape
Hips	0.07, 0.1, 0.25	15	Box
Torso	0.32, 0.39, 0.3	22	Box
Neck	0.1, 0.07, 0.1	1	Cylinder
Head	0.15, 0.22, 0.2	5	Sphere
Left upper arm	0.15, 0.31, 0.15	2.1	Truncated cone
Left lower arm	0.1, 0.23, 0.1	1.2	Truncated cone
Left palm	0.03, 0.085, 0.036	0.25	Sphere
Left thumb 1	0.01, 0.025, 0.01	0.03	Cylinder

Left thumb 2	0.01, 0.02, 0.01	0.03	Cylinder
Left index 1	0.01, 0.025, 0.01	0.03	Cylinder
Left index 2	0.01, 0.022, 0.01	0.03	Cylinder
Left index 3	0.01, 0.02, 0.01	0.03	Cylinder
Left middle 1	0.01, 0.028, 0.01	0.03	Cylinder
Left middle 2	0.01, 0.026, 0.01	0.03	Cylinder
Left middle 3	0.01, 0.02, 0.01	0.03	Cylinder
Left ring 1	0.01, 0.024, 0.01	0.03	Cylinder
Left ring 2	0.01, 0.022, 0.01	0.03	Cylinder
Left ring 3	0.01, 0.02, 0.01	0.03	Cylinder
Left little 1	0.01, 0.019, 0.01	0.03	Cylinder
Left little 2	0.01, 0.017, 0.01	0.03	Cylinder
Left little 3	0.01, 0.01, 0.01	0.03	Cylinder
Right upper arm	0.15, 0.31, 0.15	2.1	Truncated cone
Right lower arm	0.1, 0.23, 0.1	1.2	Truncated cone
Right palm	0.03, 0.085, 0.036	0.25	Sphere
Right thumb 1	0.01, 0.025, 0.01	0.03	Cylinder
Right thumb 2	0.01, 0.02, 0.01	0.03	Cylinder
Right index 1	0.01, 0.025, 0.01	0.03	Cylinder
Right index 2	0.01, 0.022, 0.01	0.03	Cylinder
Right index 3	0.01, 0.02, 0.01	0.03	Cylinder
Right middle 1	0.01, 0.028, 0.01	0.03	Cylinder
Right middle 2	0.01, 0.026, 0.01	0.03	Cylinder
Right middle 3	0.01, 0.02, 0.01	0.03	Cylinder
Right ring 1	0.01, 0.024, 0.01	0.03	Cylinder
Right ring 2	0.01, 0.022, 0.01	0.03	Cylinder
Right ring 3	0.01, 0.02, 0.01	0.03	Cylinder
Right little 1	0.01, 0.019, 0.01	0.03	Cylinder
Right little 2	0.01, 0.017, 0.01	0.03	Cylinder
Right little 3	0.01, 0.01, 0.01	0.03	Cylinder
Left upper leg	0.2, 0.47, 0.2	7.5	Truncated cone
Left lower leg	0.15, 0.44, 0.15	3.5	Truncated cone
Left foot	0.1, 0.05, 0.15	1	Box
Right upper leg	0.2, 0.47, 0.2	7.5	Truncated cone
Right lower leg	0.15, 0.44, 0.15	3.5	Truncated cone
Right foot	0.1, 0.05, 0.15	1	Box

3.3 Surface modeling

The "surface" of the human model describes its visual appearance. Each segment in the skeletal model of figure 3-4 needs a surface description. There are a number of advanced surface or skin modeling techniques that can be used, some of which were discussed in the previous chapter. For academic purposes, we have found a simple rigid polygon model for every segment to be sufficient. Compared to mesh deformation techniques, this model introduces visual artifacts such as z-buffer poke-through, but when viewed from a reasonable distance, the results are satisfactory. The rigid polygon mesh is defined in world

or global Cartesian coordinates in such a way that the pivot point coincides with the origin. To render the mesh is simply a matter of transforming it using the joint location and orientation matrix. The effects of basic clothing, as well as body hair and nails, are provided by colouring the mesh appropriately. Figure 3-8a depicts our female figure, and figure 3-8b the male figure. Both consist in total of roughly 7000 polygons, and are reasonably detailed.

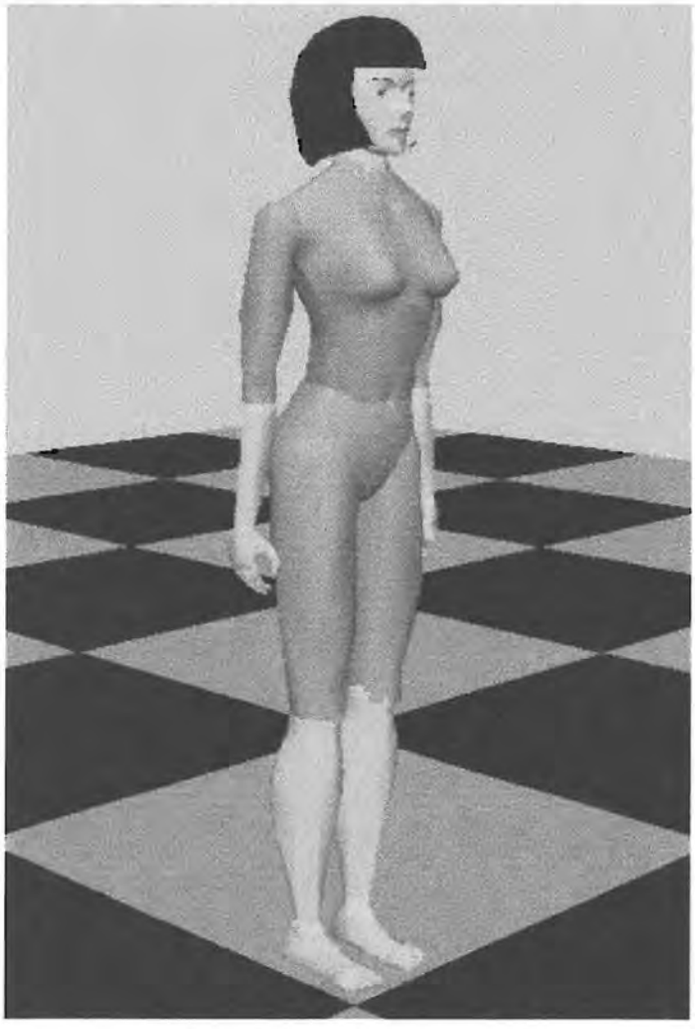


Figure 3-8a: Female model.

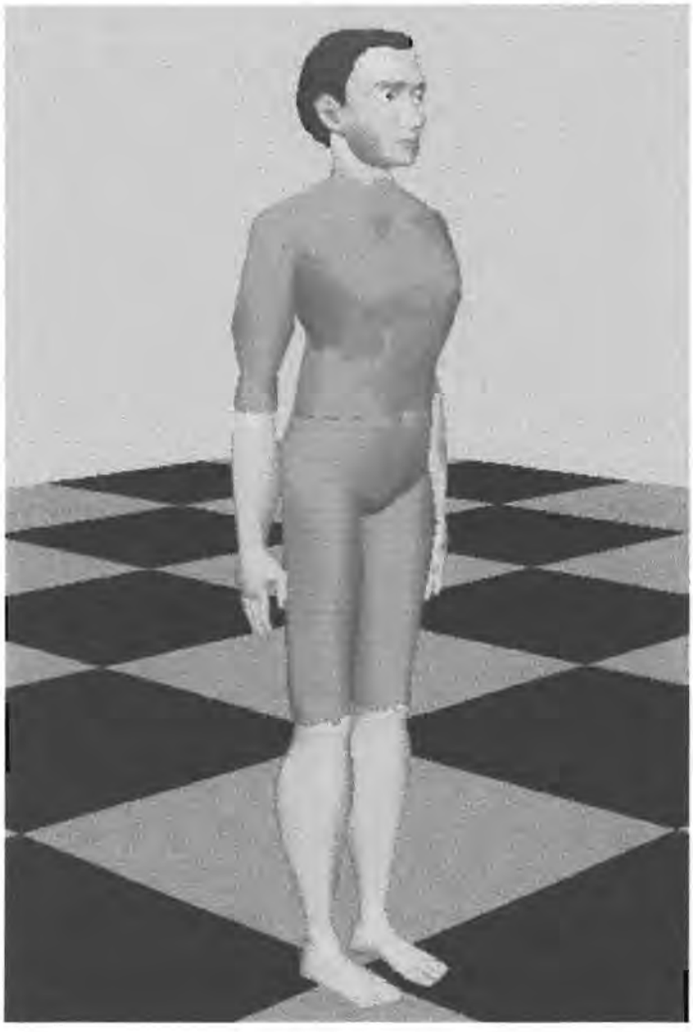


Figure 3-8b: Male model.

3.4 Dynamic modeling

Dynamic modeling or simulation refers to the process of numerically solving the equations of motion that govern human movement. Dynamic modeling consist of forward dynamics and inverse dynamics. Forward dynamics require the explicit definition of forces and torques to implement body motion. Inverse dynamics assume that the joint variables or certain goal postures are known, and the forces and torques required are calculated. Dynamic simulation of articulated structures is computationally expensive, and depending on the method and accuracy, cannot be implemented in real-time for large structures using current personal computers. However, using simplifications and efficient algorithms, dynamic modeling of a human figure is possible in real-time. The concept of dynamic

modeling application for human figure animation has already been discussed in the previous chapter. Appendix II summarizes a recursive formulation for a hierarchically linked body.

3.5 Summary

This chapter presented virtual human modeling in terms of the physical model, hierarchical model, surface or visual model and briefly the dynamic model. In terms of motion parameters and subsequent compression methods, the hierarchical model is the most important. The concept of *joints* and *segments* as well as the parameters that define each, was discussed. This chapter also introduced the simplified human model that will be used in the remainder of this document, as well as the associated quantitative parameters.

Chapter 4 Motion capture

4.1 Introduction

Motion capture in general, can be described as the methodology to provide a *natural* interface between the *real* human motion and the computer that controls the *virtual* human. Human body tracking is indeed a very important aspect of virtual environments. One of the most important motions, head motion, is frequently used to control the virtual camera and hence what the user sees. However, if this motion is not adequately captured, side effects can range from severe disorientation to simulator sickness. The more natural the interaction with a synthetic environment can be made, the better. The ideal motion capture device is one that the user perceives as natural, but does not interfere with the motion or encumber the body. Additionally, such a device must be capable of accurately capturing the user's motion with an update rate that warrants real-time interaction with the virtual environment. There are a number of quite different technologies that are currently being used. The information requirements for different body parts, as well as cost, encumbrance, accuracy and update speed are the general guidelines for identifying the appropriate technology. The most important motion capture technologies can be resorted under either mechanical, electromagnetic, acoustic, image-based, optical, inertial and spread-spectrum systems, or a combination thereof.

4.2 Sensor hardware

4.2.1 Body tracking

For this study the Polhemus InsideTRAK™ electromagnetic based sensor hardware is used for motion capture of body parts [74]. The InsideTRAK™ consists of a full sized PC

compatible expansion card, a transmitter frequency module, a single transmitter and up to two receivers. The specifications of the device are as follows:

- Update rate: 30 Hz or 60 Hz
- Latency: 12 ms
- Position accuracy: 13 mm RMS
- Orientation accuracy: 2 degrees RMS
- Position resolution: 0.003 mm / 10 mm of range
- Orientation resolution: 0.03 degrees

These specifications are valid when the receiver is within 762 mm from the transmitter. Operation up to 1524 mm is possible with reduced accuracy. It is prudent to make additional measurements within the operating environment to obtain baseline specifications for the raw captured data. Figure 4-1 shows the results for noise power against distance from the transmitter at a few discrete steps. Figure 4-2 shows the noise power spectral density (PSD) at a few discrete distances from the transmitter. The quantities have been normalized to the noise power at 0.3 m. It can be seen that the sensor noise is white, and that it increases exponentially with distance from the transmitter. When capturing motion, one should be aware of the *actual* resolution, accuracy and usable frequency content of the stored motion, especially if such information is to be used in postprocessing calculations.

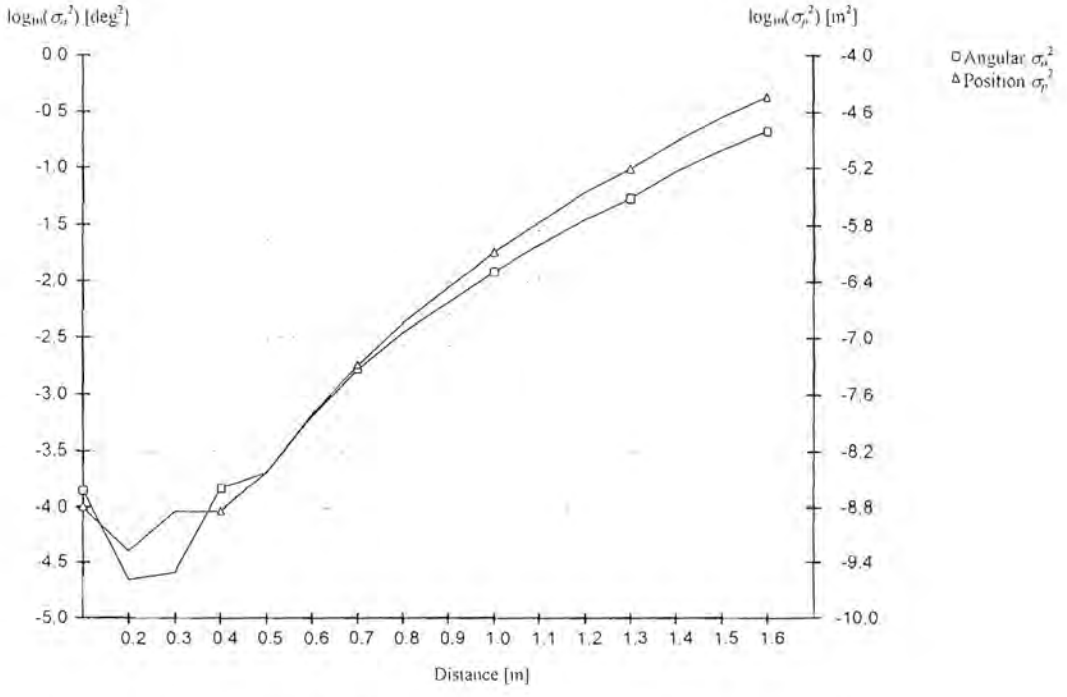


Figure 4-1: Noise power vs. distance from transmitter.

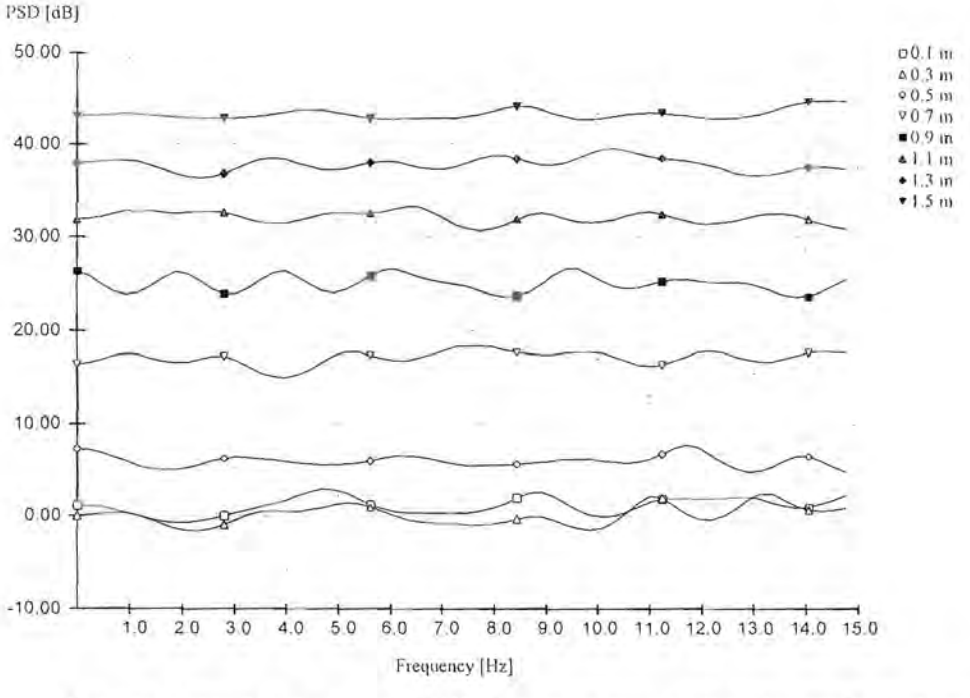


Figure 4-2a: Normalized PSD of sensor position noise with distance.

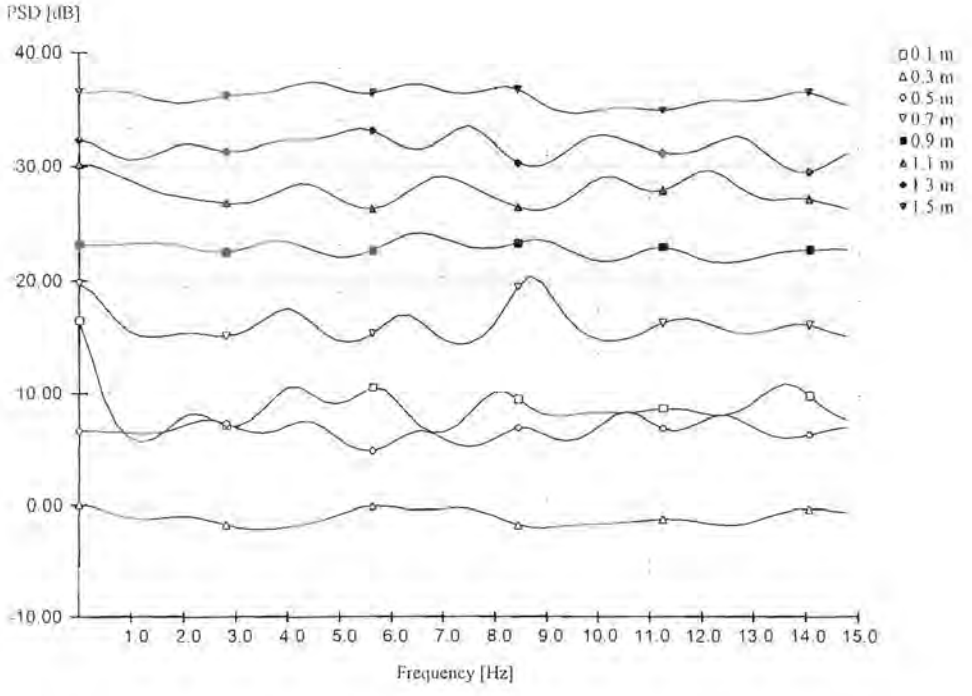


Figure 4-2b: Normalized PSD of sensor rotation noise with distance.

A single desktop electromagnetic transmitter can drive one or two receivers at an update rate of 60 Hz or 30 Hz respectively. More than one transmitter can be used in close proximity without serious side-effects, although there is a noise increase close to the transmitter that is not driving the particular receiver. Each receiver returns the x, y and z angles (pitch, yaw and roll) as well as the x, y and z position in 3D space. Driver software that provides initialization and update functionality was developed to interface with the InsideTRAK™ hardware. The software converts from the internal transmitter axis representation to a left-handed coordinate system (Appendix I, figure A-1), and provides the sensor output as a transformation matrix **S**

$$\mathbf{S} = \mathbf{R}_z(\theta_z)\mathbf{R}(\theta_x)\mathbf{R}(\theta_y)\mathbf{T}(x, y, z), \tag{4-1}$$

where x, y and z are the sensor positions and θ_x, θ_y and θ_z are the sensor angles respectively.

4.2.2 Gesture tracking

The motion capture device for the fingers is the 5DT Data Glove from Fifth Dimension Technologies [75]. This device differs from the electromagnetic sensor tracker in that it measures *joint angles* directly instead of the position and orientation of a segment, such as the wrist or head. It is basically a mechanical tracking device, but it uses a fiber optical approach to measure the curvature of the fingers. Communication with the host computer is via a RS232 serial link. The basic specifications for the device are:

- Number of sensors: Five, one for each finger
- Update rate: In excess of 200 Hz
- Latency: Less than 10 ms
- Resolution: 8 bits across full flexure
- Accuracy: At least 6 bits

As the glove is a fiber optical device, these values apply almost anywhere and under any conditions. The output from the glove is raw, uncalibrated data. Assuming that full output corresponds to an average flexure of 70°, the measured noise power is found to be approximately $\sigma_f^2 = 1.2$ deg^2. Figure 4-3 shows the normalized noise PSD up to 15 Hz, and it can be seen that it is more or less white. The reference value of 0 dB at 0 Hz is due to the unipolar nature of the glove output. Driver software that provides initialization and update functionality was developed to interface with the data glove. The software provides the average finger flexure in an automatic, linearly calibrated, normalized fashion. During every update, the raw value read from the sensor is compared to the current minimum and maximum raw values (raw_{min} and raw_{max}). The minimum and maximum values are overwritten if they are exceeded. The normalized output is given by the first order equation

$$out = \frac{raw_{val} - raw_{min}}{raw_{max} - raw_{min}}, \qquad (4\text{-}2)$$

which is in [0...1]. Doing a few flexing movements with the hand quickly sets the operating values for raw_{min} and raw_{max} and calibrates the glove.

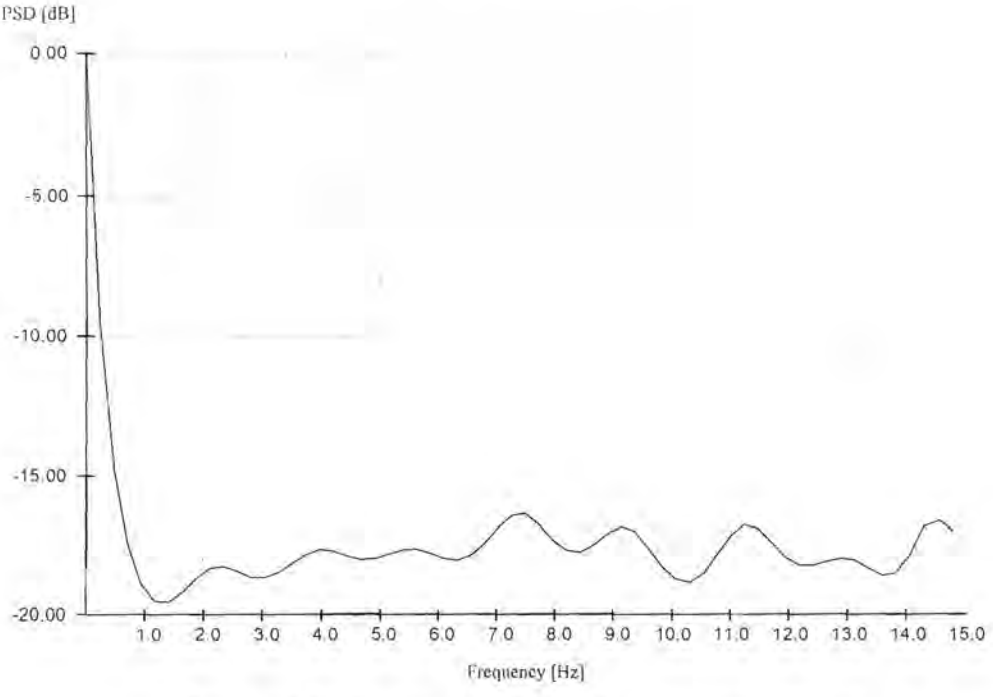


Figure 4-3: Normalized PSD of glove sensor noise.

4.3 Sensor arrangement

4.3.1 Full body motion

In general, if it is desired to capture motion for the entire human body, there are fifteen major parts to track independently. These major portions are the head, torso-clavicle region, abdomen-hips region, upper legs, lower legs, feet, upper arms, lower arms and hands. Such a solution is shown in figure 4-4. A position/orientation sensor is attached to every major body segment. This arrangement will directly supply all the major joint angle information, and in most cases it will not be necessary to use the position information from the sensors, except possibly for the root segment. No additional calculations (e.g. inverse kinematics) will be necessary and the result should be a stable configuration.

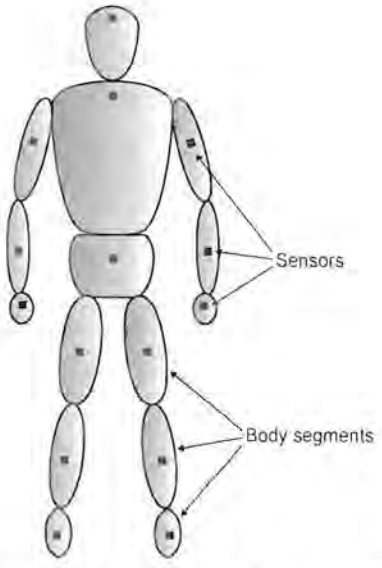


Figure 4-4: Full body sensors.

4.3.2 Minimal configuration

The cost of a full sensor solution as shown in figure 4-4 is quite prohibitive, and the set up is extremely cumbersome. It is possible to use fewer sensors and to estimate the missing information using inverse kinematics algorithms. For practical reasons, the motion capture capabilities of this research is limited to a maximum of four sensors. Such a minimal sensor configuration is shown in figure 4-5. One sensor is placed on the center of the hips, one on top of the head and one on top of each wrist. Some joint angles can be calculated directly, others can be deduced using an inverse kinematics solution, and the rest cannot be obtained at all. Obviously it is an underspecified system of variables, and there is not necessarily a single unique inverse kinematic solution. The best that one can do is to choose the solution that is expected to be the most stable. Classically, inverse kinematics is presented as a generalized and iterative solution. However, the interest lies in a sampling rate of at least 30 Hz, which means that computationally expensive iterative repetitions should be avoided. For some hierarchical systems [46], it can be shown that the inverse kinematics solution could be presented as a closed form analytical expression. By choosing a correctly reduced representation of the human skeleton and placing the sensors on the proper locations, all of the desired unknowns can be calculated using a closed form solution, which will be discussed in the next section.

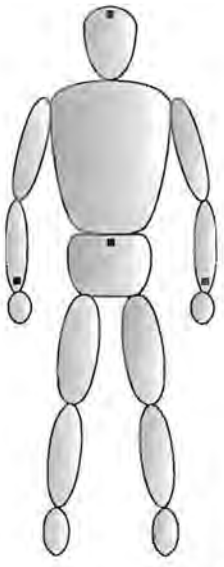


Figure 4-5: Minimal body sensors.

4.3.3 Sensor calibration

To calibrate the electromagnetic tracker sensor output, a known posture or stance is needed as well as the exact location and orientation of all the sensors. It is possible to compensate to a certain degree for either skew sensor placement, or for a skew calibration posture, but not both. In the following discussion it is assumed that the calibration posture is absolutely correct, and that the sensors are attached and aligned more or less correctly.

Figure 4-6: Calibration posture.

The sensor arrangement was shown in figure 4-5, and the calibration posture is shown in figure 4-6. It is assumed that all the segment lengths in question are known beforehand. For each sensor a calibration matrix **C** in the format of equation (4-1) is stored. The output of the calibration algorithm is given by

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \mathbf{I} & 0 \\ 0 & 0 \end{bmatrix} \mathbf{C}, \\ \mathbf{B} &= \begin{bmatrix} \mathbf{0} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{C}, \\ \mathbf{M} &= \mathbf{A}^{-1}\mathbf{S}\mathbf{B}^{-1}, \end{aligned} \tag{4-3}$$

where **S** is the current sensor matrix from equation (4-1), **A** is the rotational part of the calibration matrix and **B** is the translational part of the calibration matrix. **M** is therefore an identity matrix when the sensor aligns exactly with the calibration position/orientation, and indicates the offset from the calibration otherwise. If the calibration posture differs from

the "zero" posture as given in figure 2-7, the calibrated sensor matrix needs to be transformed to the proper position and orientation. From figure 4-6 it is clear that this is the case for the two wrist sensors. The hip segment sensor and head segment sensor calibration position and orientation align with the zero posture, and no transformation is necessary.

The data glove sensors are inherently *absolute* or *sourceless* devices compared to the electromagnetic sensors, which are *relative* or *sourced* devices (relative to the transmitter). The glove sensor output needs to be scaled and offset to a known value, usually to the joint limits of the fingers. The automatic, normalized calibration of the finger sensors as presented in equation (4-2) provides a reasonably accurate method to provide a known flexure output over the full range of motion.

4.4 Sensor data converter

Consider the simplified human model of figure 4-7 (which is reproduced here from chapter 2 for convenience), and the minimal configuration discussed above. There are seven body sections that could be addressed separately. These are the root, hip, spine and head section, the two arm sections, the two leg sections and the hand and finger sections. The left and right leg, arm and hand sections are similar, and only the calculations for the right hand side will be discussed. In the following sections a new set of variables ϕ_k are defined, instead of the standard joint and DOF notation $\theta_{i,j}$, in order to avoid confusion. It is a simple matter to map the resulting ϕ_k's to the proper DOFs.

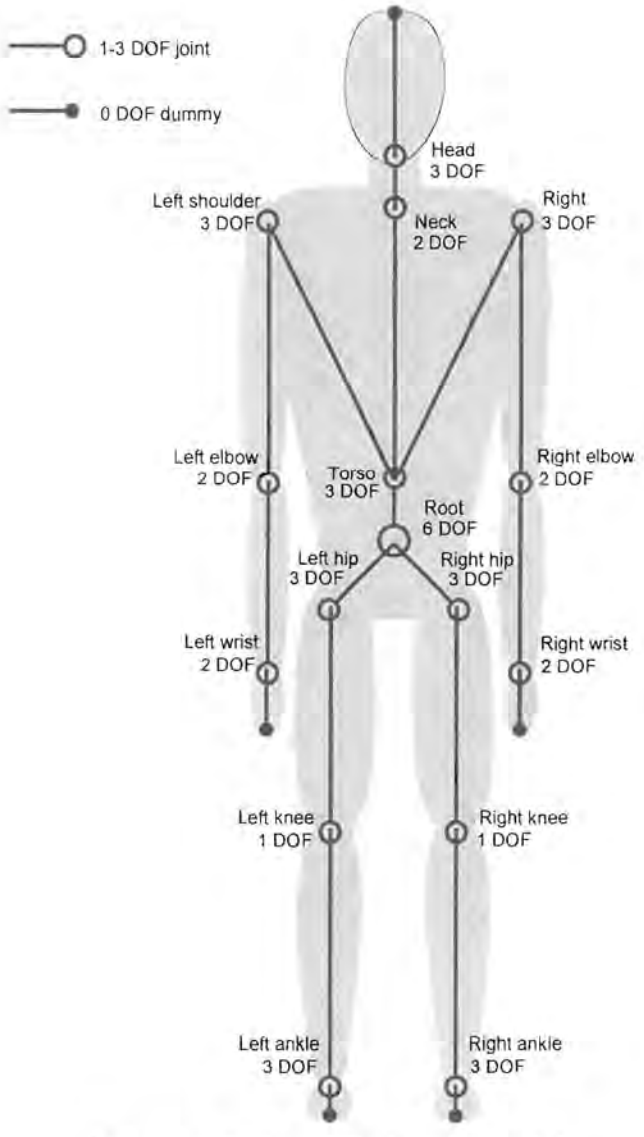


Figure 4-7: Simplified body model.

4.4.1 Spine section

Figure 4-8 depicts the skeleton for the hip to head section, with the fixed world space origin located at the root. To simplify the calculations, it is assumed that the offset from the root to the torso base, the offset from the torso base to the neck base, and the offset from the neck base to the head base can be approximated with vectors consisting only of a y component. It is also assumed that the root and torso can be combined as a single joint, as well as the neck and head. The angles calculated for these combined joints could later be split to give movement that is more natural.

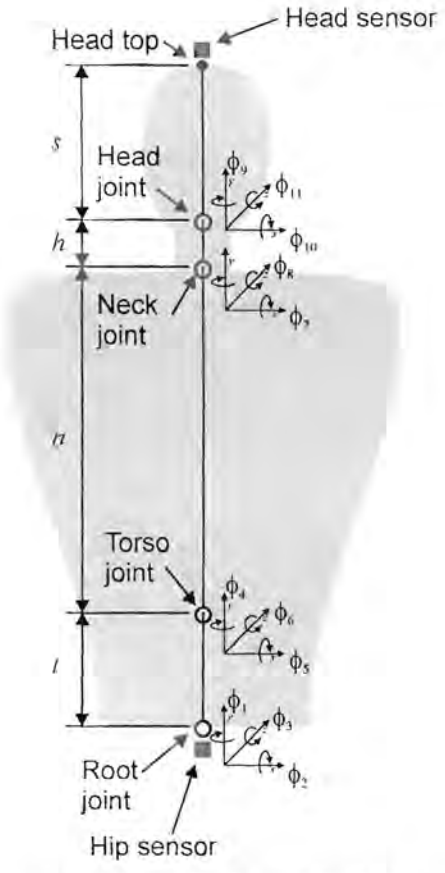


Figure 4-8: Simplified model of the spine.

Given these approximations, we find that

$$\phi_4 = \phi_5 = \phi_6 = \phi_7 = \phi_8 = 0\,. \tag{4-4}$$

The combined hierarchical transformation from the root to the head can be found by defining $\mathbf{A}_i$ as the coordinate transformation matrix from frame $\{i\text{-}1\}$ to frame $\{i\}$ as a function of the joint variable ϕ_i:

$$\begin{aligned}
\mathbf{A}_{11} &= \mathbf{R}_z(\phi_{11}),\\
\mathbf{A}_{10} &= \mathbf{R}_x(\phi_{10}),\\
\mathbf{A}_9 &= \mathbf{R}_y(\phi_9),\\
\mathbf{A}_3 &= \mathbf{T}(0, t+n+h, 0)\mathbf{R}_z(\phi_3),\\
\mathbf{A}_2 &= \mathbf{R}_x(\phi_2),\\
\mathbf{A}_1 &= \mathbf{R}_y(\phi_1).
\end{aligned} \tag{4-5}$$

Given the head transformation matrix in world space $\mathbf{A}_h$, the inverse kinematics problem is to find the angles ϕ_1, ϕ_2, ϕ_3, ϕ_9, ϕ_{10}, ϕ_{11} that satisfy the equation

$$\mathbf{A}_h = \mathbf{A}_{11}\mathbf{A}_{10}\mathbf{A}_9\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1. \tag{4-6}$$

Denote the current calibrated hip sensor matrix by

$$\mathbf{S}_1 = \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix}. \tag{4-7}$$

For the simplified spine model, we only use the position and y-axis rotation information of the hip sensor. By symbolically calculating $\mathbf{A}_1$ and equating the elements to $\mathbf{S}_1$, we find that

$$\phi_1 = \tan^{-1}\left(\frac{s_{31}}{s_{33}}\right). \tag{4-8}$$

We know that $\mathbf{A}_{11}\mathbf{A}_{10}\mathbf{A}_9$ has no effect on the position of the head joint in world space, therefore

$$\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\mathbf{A}_h = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1. \tag{4-9}$$

To obtain an expression for $\mathbf{A}_h$, we need to transform the calibrated head sensor matrix to the world space calibration position, and then project back to the base of the head. By denoting the current calibrated head sensor matrix with $\mathbf{S}_2$, we get

$$\mathbf{A}_h = \mathbf{T}(0,-s,0)\mathbf{S}_2\mathbf{T}(0,t+n+h+s,0). \tag{4-10}$$

Since we have already calculated ϕ_1, we can postmultiply by $\mathbf{A}^{-1}$ to get

$$\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\mathbf{A}_3\mathbf{A}_2 = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\mathbf{A}_h\mathbf{A}_1^{-1} = \begin{bmatrix}a_{41} & a_{42} & a_{43} & a_{44}\end{bmatrix}. \tag{4-11}$$

By equating the left and right hand matrix translation components we find that

$$\begin{aligned}\phi_2 &= \tan^{-1}\left(\frac{a_{43}}{a_{42}}\right),\\ \phi_3 &= \sin^{-1}\left(\frac{-a_{41}}{t+n+h}\right).\end{aligned} \tag{4-12}$$

Equation (4-4) can now be written as

$$\mathbf{A}_{11}\mathbf{A}_{10}\mathbf{A}_9\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1(\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1)^{-1} = \mathbf{A}_h(\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1)^{-1} = \begin{bmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\\ a_{31} & a_{32} & a_{33} & a_{34}\\ a_{41} & a_{42} & a_{43} & a_{44}\end{bmatrix}. \tag{4-13}$$

By calculating the left hand side symbolically and equating to the right we get

$$\theta_9 = \tan^{-1}\left(\frac{a_{31}}{a_{33}}\right),$$
$$\theta_{11} = \tan^{-1}\left(\frac{a_{12}}{a_{22}}\right), \tag{4-14}$$
$$\theta_{10} = \tan^{-1}\left(\frac{-\cos(\theta_{11})a_{32}}{a_{22}}\right).$$

Once ϕ_1, ϕ_2, ϕ_3, ϕ_9, ϕ_{10} and ϕ_{11} are known, there are numerous ways to assign the torso and neck joint angles. One possibility is simply

$$\begin{aligned}
\phi_1' &= \frac{\phi_1}{2}, \\
\phi_2' &= \frac{\phi_2}{2}, \\
\phi_3' &= \frac{\phi_3}{2}, \\
\phi_4 &= \phi_1', \\
\phi_5 &= \phi_2', \\
\phi_6 &= \phi_3', \\
\phi_7 &= \frac{\phi_{10}}{2}, \\
\phi_8 &= \frac{\phi_{11}}{2}, \\
\phi_{10}' &= \phi_7, \\
\phi_{11}' &= \phi_8.
\end{aligned} \tag{4-15}$$

Where $\phi_1', \phi_2', \phi_3', \phi_{10}'$ and ϕ_{11}' are the new root and head angles. Because the length n is in general much larger than either t or h, the slight error in the final posture that this approach introduces, is negligible.

4.4.2 Arm Section

Figure 4-9 shows the skeleton for the shoulder-to-wrist section, with the fixed world space origin located at the shoulder. The arm is modeled as a crude five degree of freedom mechanism with a spherical joint at the shoulder and a hinge and twist joint at the elbow [61]. The inherent singularities of this approach were discussed in chapter 3. Although we lose a degree of freedom considering that the wrist sensor is a 6 DOF device, this configuration ensures a reasonably stable and unique solution. To simplify the calculations, we assume that we can approximate the offset from the shoulder to the elbow and the offset from the elbow to the wrist with vectors consisting only of a y component.

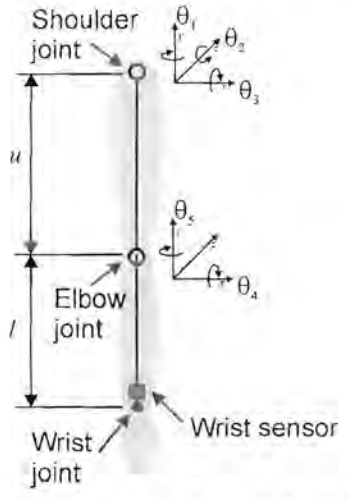


Figure 4-9: Simplified model of the arm.

The combined hierarchical transformation from the shoulder to the wrist can be found by defining $\mathbf{A}_i$ as the coordinate transformation matrix from frame $\{i\text{-}1\}$ to frame $\{i\}$ as a function of the joint variable ϕ_i:

$$\begin{aligned}
\mathbf{A}_5 &= \mathbf{T}(0,-l,0)\mathbf{R}_y(\phi_5), \\
\mathbf{A}_4 &= \mathbf{R}_x(\phi_4), \\
\mathbf{A}_3 &= \mathbf{T}(0,-u,0)\mathbf{R}_y(\phi_3), \\
\mathbf{A}_2 &= \mathbf{R}_z(\phi_2), \\
\mathbf{A}_1 &= \mathbf{R}_y(\phi_1).
\end{aligned} \tag{4-16}$$

Given the wrist transformation matrix in world space $\mathbf{A}_w$, the inverse kinematics problem is to find the angles $\phi_1, \ldots, \phi_5$ that satisfies the equation

$$\mathbf{A}_w = \mathbf{A}_5\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1. \tag{4-17}$$

The wrist matrix $\mathbf{A}_w$ is given by

$$\mathbf{A}_w = \mathbf{S}\mathbf{A}_s^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}. \tag{4-18}$$

where $\mathbf{A}_s$ is a matrix that defines the shoulder origin in world space and $\mathbf{S}$ is the current calibrated wrist sensor matrix. The results from the previous section define the value of $\mathbf{A}_s$. Let $\mathbf{w} = [a_{41}\ a_{42}\ a_{43}]$ be the wrist position. From figure 4-9 it can be seen that the elbow hinge angle ϕ_4 is given by the cosine rule

$$\phi_4 = \pi \pm \cos^{-1}\left(\frac{u^2 + l^2 - \|\mathbf{w}\|^2}{2ul}\right). \tag{4-19}$$

Only one solution of ϕ_4 is physically realizable due to joint limits. The next step is to calculate the elbow position $\mathbf{e} = [e_x\ e_y\ e_z]$, which is simply given by

$$[\mathbf{e}\ \ 1] = [0\ \ -l\ \ 0\ \ 1]\mathbf{A}_w. \tag{4-20}$$

From equation (4-16), it is clear that the elbow position is just a function of the shoulder angles, i.e.

$$[\mathbf{e}\ \ 1] = [0\ \ 0\ \ 0\ \ 1]\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1. \tag{4-21}$$

By expanding the right hand side symbolically and equating to the left, we find that

$$\phi_1 = \tan^{-1}\left(\frac{e_z}{e_x}\right),$$
$$\phi_2 = \begin{cases} \tan^{-1}\left(\dfrac{e_x}{\cos\phi_1 e_y}\right) & \cos\phi_1 \neq 0 \\ \tan^{-1}\left(\dfrac{e_z}{\sin\phi_1 e_y}\right) & \sin\phi_1 \neq 0 \end{cases} \tag{4-22}$$

Alternatively, ϕ_2 is given by the dot product and cosine rule

$$\phi_2 = \cos^{-1}\left(\frac{[0 \quad -u \quad 0]\cdot\mathbf{e}}{u\|\mathbf{e}\|}\right). \tag{4-23}$$

From figure 4-9 it can be seen that the lower arm twist angle ϕ_5 has no effect on the wrist position, as the axis of rotation aligns with [0 –*l* 0]. The value for ϕ_3 can therefore be found by solving the equation

$$[\mathbf{w} \quad 1] = [0 \quad 0 \quad 0 \quad 1]\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1. \tag{4-24}$$

By multiplying both sides with $(\mathbf{A}_2\mathbf{A}_1)^{-1}$ and using the proper components of the vector equation, we get

$$\phi_3 = \tan^{-1}\left(\frac{-a_{41}c_1c_2 + a_{42}s_2 - a_{43}s_1c_2}{-a_{41}s_1 + a_{43}c_1}\right), \tag{4-25}$$

where $c_1 = \cos\phi_1$, $c_2 = \cos\phi_2$, $s_1 = \sin\phi_1$ and $s_2 = \sin\phi_2$. The angle ϕ_5 can now be found by solving the full equation (4-16). Rearranging gives

$$\mathbf{A}_w(\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1)^{-1} = \mathbf{A}_5\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1(\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1)^{-1}. \tag{4-26}$$

In this case symbolic multiplication becomes a bit tedious. By denoting the elements of the left hand matrix with b_{ij} where $\{i = 1...4, j = 1...4\}$, we have

$$\mathbf{A}_w(\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1)^{-1} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \mathbf{A}_5, \tag{4-27}$$

and one possible solution for ϕ_5 is given by

$$\phi_5 = \tan^{-1}\left(\frac{b_{13}}{b_{11}}\right). \tag{4-28}$$

When ϕ_1, ..., ϕ_5 are applied to the hierarchy in figure 4-9, the position of the wrist will often differ from the actual position as found in $\mathbf{A}_w$, due to sensor misalignment and noise. Such an error is shown in figure 4-10, where $\mathbf{w}$ represents the actual wrist position as given by the wrist sensor and $\mathbf{w}'$ the position given by traversing the hierarchy and current angles. It is sometimes desirable to have the wrist position exactly right at the cost of errors in the joint angles. From the use of equation (4-19) we know that $\|\mathbf{w}\| = \|\mathbf{w}'\|$, and the only way to alter the computed wrist position is to adjust the shoulder angles ϕ_1, ϕ_2, and ϕ_3. The axis of rotation can be found by the cross product

$$\mathbf{u} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \times \frac{\mathbf{w}'}{\|\mathbf{w}'\|}, \tag{4-29}$$

while the amount of rotation around $\mathbf{u}$ is given by the dot product

$$\varphi = \cos^{-1}\left(\frac{\mathbf{w} \cdot \mathbf{w}'}{\|\mathbf{w}\|\|\mathbf{w}'\|}\right). \tag{4-30}$$

Denote the correction matrix by **M**, where

$$\mathbf{M} = \mathbf{R}(\mathbf{u}, \varphi). \tag{4-31}$$

We now have that $\mathbf{A}_w' = \mathbf{A}_5\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1$ and $\mathbf{A}_w = \mathbf{A}_5\mathbf{A}_4\mathbf{A}_3'\mathbf{A}_2'\mathbf{A}_1'$ where $\mathbf{A}_3', \mathbf{A}_2'$ and $\mathbf{A}_1'$ correspond to the corrected angles ϕ_3', ϕ_2' and ϕ_1' respectively. The corrected angles can be found by solving the equation

$$\mathbf{A}_5\mathbf{A}_4\mathbf{A}_3'\mathbf{A}_2'\mathbf{A}_1' = \mathbf{A}_5\mathbf{A}_4\mathbf{A}_3\mathbf{A}_2\mathbf{A}_1\mathbf{M}, \tag{4-32}$$

which is in a similar format as the original equation (4-16).

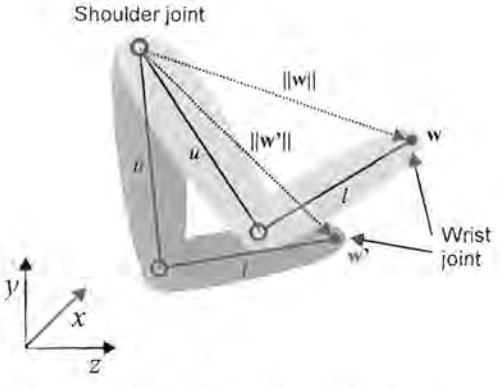


Figure 4-10: Wrist position errors.

4.4.3 Leg section

The left and right leg degrees of freedom are hopelessly underspecified by the four useable degrees of freedom available from the hip sensor (i.e. the hip joint position and the root *y*-angle). The following solution was developed for simplicity and is but a single possibility out of numerous options. Figure 4-11a shows a schematic of the right leg with the ankle joint rooted at the world origin. Using this configuration, it is implied that the tracked

human cannot move his or her feet. Figure 4-11b shows a clearer view in the *xy* and *yz* planes.

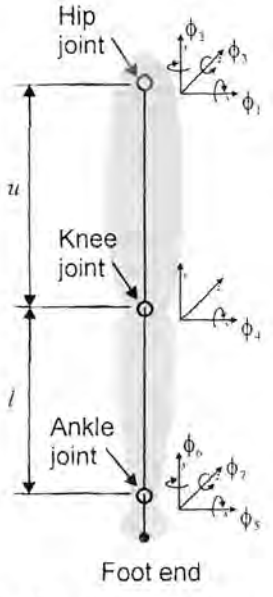


Figure 4-11a: Simplified leg model.

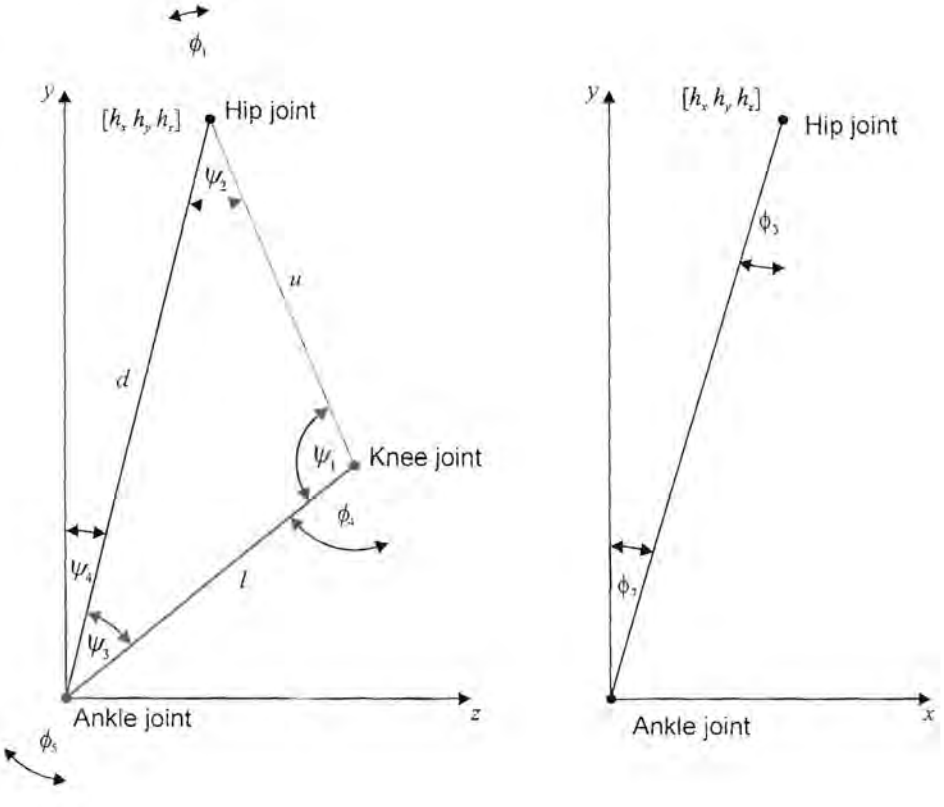


Figure 4-11b: Leg model in the *yz* and *xy* planes.

The internal angles of the triangle formed by the ankle, knee and hip joints are given by

$$\begin{aligned}\psi_1 &= \cos^{-1}\left(\frac{u^2+l^2-d^2}{2ul}\right),\\ \psi_2 &= \cos^{-1}\left(\frac{d^2+u^2-l^2}{2du}\right),\\ \psi_3 &= 180-\psi_1-\psi_2.\end{aligned} \tag{4-33}$$

It should be noted that if $d > u + l$, there is a violation in the allowable workspace for the hip position and equation (4-32) is invalid. The best we can do in this case is to make $\psi_1 = 180$ and $\psi_1 = 0$. The angle formed by the vector $\mathbf{h} = [h_x\ h_y\ h_z]$ from the ankle to the hip is given by

$$\psi_4 = \tan^{-1}\left(\frac{h_z}{h_y}\right). \tag{4-34}$$

From these values the following leg angles can be computed directly:

$$\begin{aligned}\phi_1 &= \psi_2 - \psi_4,\\ \phi_4 &= 180-\psi_1,\\ \phi_5 &= \psi_3 + \phi_4.\end{aligned} \tag{4-35}$$

From figure 4-11b, the rotation of the hip and ankle joints around the z-axis can be found by inspection:

$$\begin{aligned}\phi_3 &= \tan^{-1}\left(\frac{h_x}{h_y}\right),\\ \phi_7 &= -\phi_3.\end{aligned} \tag{4-36}$$

The hip twist angle ϕ_2 is left at zero. To prevent the feet from rotating with the hips, we assign ϕ_6 as

$$\phi_6 = -\phi_y, \tag{4-37}$$

where ϕ_y is the y-rotation of the hips as calculated with equation (4-8).

4.4.4 Hand section

The fingers of both hands can be represented as simple one and two DOF segments as shown in figure 2-4b. The output from the glove sensors are normalized real values in [0...1], and can be scaled and mapped directly to the finger angles. The glove sensors only return the average curvature of the fingers in one dimension. The best we can do in this case is to assign rotations around the z-axis for all the joints of each finger, except the thumb, for which the average flexure lies roughly in a plane perpendicular to the vector [1 0 0.5]. The abduction angles cannot be estimated. By denoting the angles for the fingers as ϕ_{ij} where $\{i = 1...5, j = 1...4\}$, we get

$$\begin{aligned}
\phi_{11} &= \frac{\phi_{L11} + s_1(\phi_{U11} - \phi_{L11})}{(\phi_{U11} - \phi_{L11})}, \\
\phi_{12} &= \frac{\phi_{L12} + s_1(\phi_{U12} - \phi_{L12})}{(\phi_{U12} - \phi_{L12})}, \\
\phi_{ij} &= \frac{\phi_{Lij} + s_i(\phi_{Uij} - \phi_{Lij})}{(\phi_{Uij} - \phi_{Lij})} \quad i \neq 1, j \neq 2.
\end{aligned} \tag{4-38}$$

where the s_i's are the calibrated normalized sensor outputs and ϕ_{Lij} and ϕ_{Uij} are the lower and upper joint limits respectively. Joints not specified by equation (4-38) are left at zero.

4.4.5 Joint limits

There are a number of instances where the joint angles are undefined or poorly defined due to numerical instabilities, singularities, sensor misalignment and noise. Soft clipping (or referred to as "ease-to" in the animation community) is a non-linear method to compensate for hard joint limits. Although not strictly necessary, it can alleviate some of the jerkiness associated with the singularities and discontinuities associated with inverse kinematics. The variables or angles that are calculated often exhibit jerky behaviour near joint limits or holes in the workspace where the formulations that are used are invalid. This is worsened by the fact that the sensors are often misaligned and noisy. Preventing these angles (and other parameters, such as the arguments of the $\cos^{-1}$ function) from reaching certain values in a "soft" manner gives a much smoother effect. There are various ways to implement soft clipping. We have chosen a simple method where the input-output function is modified by a second order parabolic curve near the clipping edges. There is a one-to-one linear relationship where no clipping occurs. In the clipping region, there is a second order relationship, and beyond that it is constant:

$$y(x) = \begin{cases} x & x \le x_1 \\ ax^2 + bx + c & x_1 < x < x_2, \\ x_c & x \ge x_2 \end{cases} \tag{4-39}$$

where x_1 and x_2 are constant values just before and after the input clip point x_c respectively. The parameters for the parabolic curve are found by setting the derivative to 1 and 0 respectively at the crossover points, and are given by

$$\begin{aligned} d &= 4(x_c - x_1), \\ a &= \frac{-1}{d}, \\ b &= 1 + \frac{2x_1}{d}, \\ c &= \frac{-x_1^2}{d}. \end{aligned} \tag{4-40}$$

4.5 Summary

This chapter presented methods for capturing human motion in real-time using electromagnetic position and orientation sensors and optical data glove devices. Inverse kinematic algorithms were developed to estimate missing and incomplete data. Separate algorithms were used for the spine, arm, leg and hand sections. The result was a mapping from the limited degree of freedom sensor space to a high degree of freedom joint space suitable for compression algorithms.

Chapter 5 Data analysis

5.1 Introduction

Analysis of human motion can be interpreted on a number of levels. Low level analysis includes parameters such as limb position and orientation. High level analysis includes posture, gesture and expression analysis. Human motion has been studied for many years on both high and low level. For the purpose of this study it is convenient to represent the data as a discrete-time stochastic process, and the human motion is viewed as low level digital waveform representations. Mathematically tractable and precise engineering approaches can be used to analyze and characterize the waveforms. Specific attention will be given to the analysis of joint angles, since this is the primary source of information that will be compressed and coded.

There are fundamentally two approaches in determining the statistics of human motion. The first is to look at the *driving force* or *process* behind the motion and to analyze the motion from a purely analytical perspective. The other approach is to look at an *infinite* amount of stored motion data and to interpret it purely *numerically*. Both of these methods are fraught with difficulties. A useable mathematical model might not always exist for every human motion variable to be analyzed. Even with appropriate models there are still too many unknowns, which tend to undermine an analytical approach. On the other hand, it is impossible to store and process an infinite amount of data, which raises the question whether the sample used is representative of the population. This research approaches the analysis of the motion numerically, provided it is understood that the results are only applicable to the few *types* of motion discussed here.

5.2 Numerical analysis

It is convenient to assume that the human motion waveforms can be represented by an ergodic random process. Although this is a gross simplification, such a statistical point of view can yield useful results. The random process in question here is applicable only to *specific* types of motion (i.e. to the examples presented here) – we do not attempt to derive statistics for human motion in general. Such a task would be almost impossible. It is occasionally convenient to group a number of DOFs together to avoid tedious repetition and to clarify results. We assume that there are a number of such groups that are independent of each other, and that each has different characteristics. Clearly, foot movement does not depend on hand movement for normal human behaviour. Where appropriate, the characteristics of head movement, torso movement, arm movement, leg movement and finger movement will be jointly investigated. Table 5-1 shows this in more detail.

Table 5-1: Joint and segment grouping

Group	Reference number	Number of joints	Number of segments	Number of DOFs
Root and torso	0	2	2	7
Neck and head	1	2	2	5
Left arm	2	3	3	8
Left hand	3	14	14	19
Right arm	4	3	3	8
Right hand	5	14	14	19
Left leg	6	3	3	7
Right leg	7	3	3	7

Denote the jth DOF of the ith joint as a sequence $\{\theta_{i,j}(n)\}$ of a discrete-time random process. From table 2-1 and figure 2-4 it can be seen that $0 \leq i \leq 48$ and $1 \leq j \leq 6$. The same notation can be used for a group of DOFs, with the subscript i indicating the group number, and in this case we have from table 5-1 that $1 \leq i \leq 8$ and $1 \leq j \leq 19$. Refer to Appendix I for additional information. For the purposes of this research it is adequate to characterize the motion signal and its derivative in terms of its first order probability density, and in terms of its autocorrelation and power spectral density functions. These methods will be discussed in the following sections.

5.2.1 Examples

Three examples of human motion will be used for statistical analysis. The first example is obtained from general conversational movements, the second is obtained from fast dance movements and the third from hand gestures. The latter is used specifically for analysis of finger movement – general body activity often lacks detailed hand gestures. We assume that most common motion will fall between the extremes represented by these examples. Figure 5-1a shows a 10 second segment of conversational movement for the left arm, figure 5-1b a 10 second segment of dance movement for the left arm, and figure 5-1c a 10 second segment of finger movement. Figure 5-1d depicts an image of 1 second's worth of overlaid 3D rendered frames for the dance sequence (skeleton only). The length of the original sequences is 300 seconds each. For clarity these figures show but a fraction of the available DOFs and sequence lengths. The full motion sequences are available on request. The motion sequences were captured using the techniques described in chapter 3, at a sampling rate of 30 Hz. We therefore assume that the frequency content of the motion is less than 15 Hz to satisfy the Nyquist criterion. It will later be shown that this is indeed the case, except possibly for extremely fast motion such as found in sport activities.

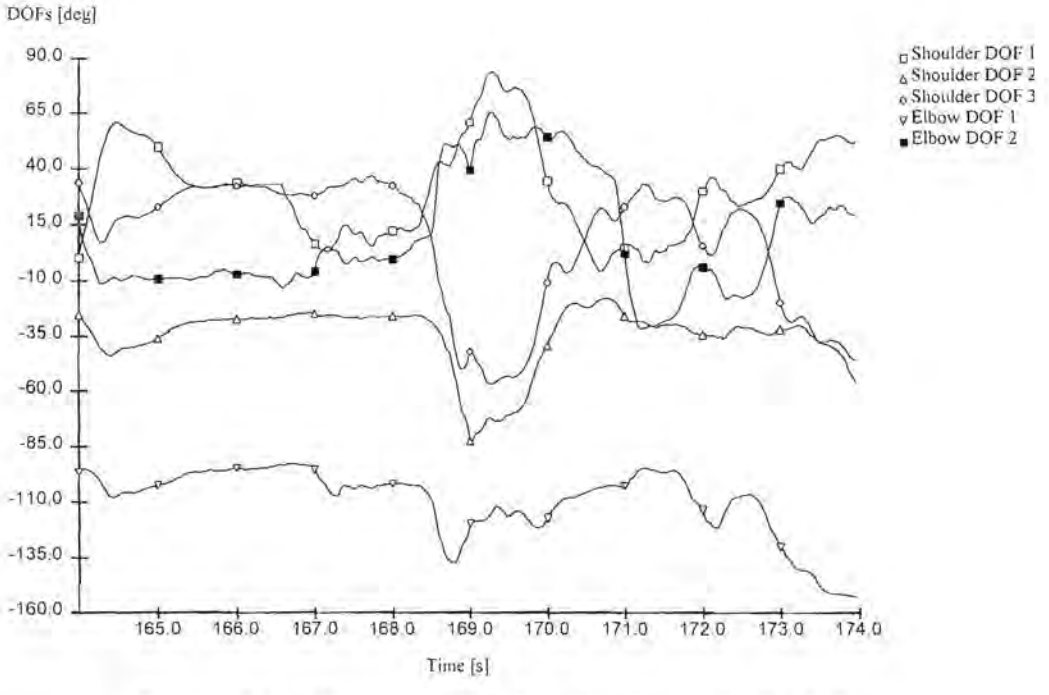


Figure 5-1a: Conversational motion example for left arm.

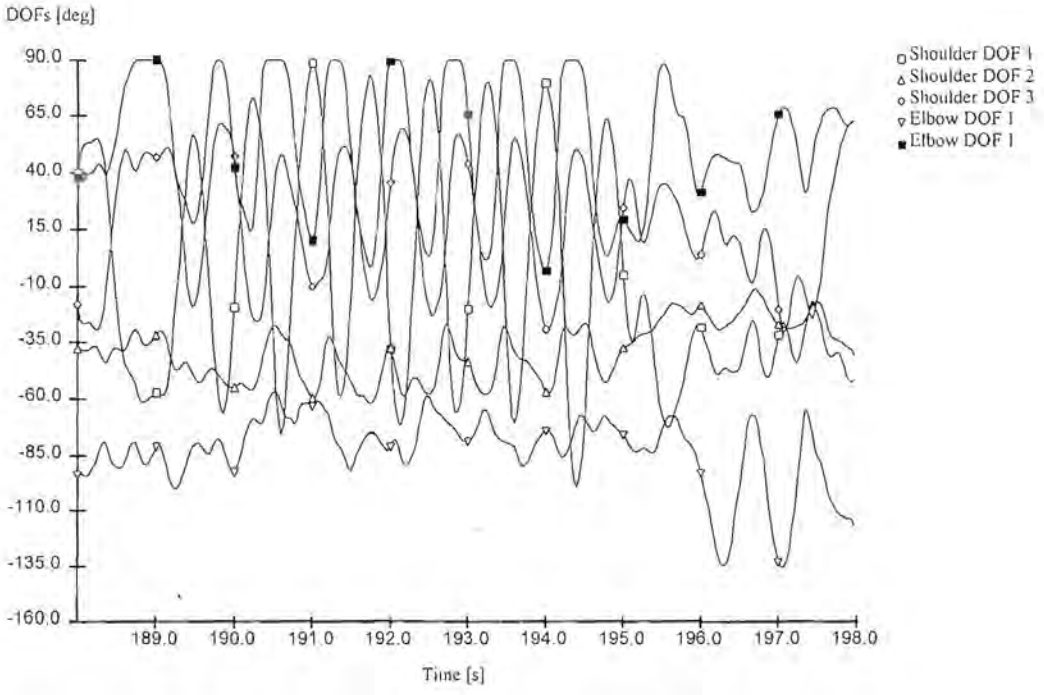


Figure 5-1b: Dance motion example for left arm.

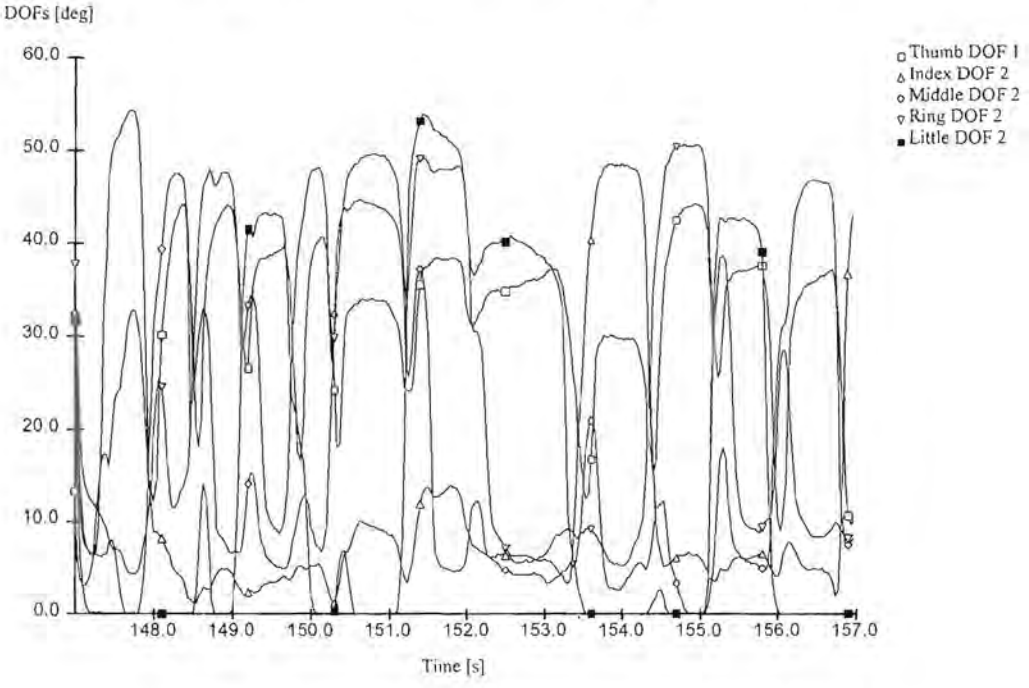


Figure 5-1c: Finger gesture example for left hand.

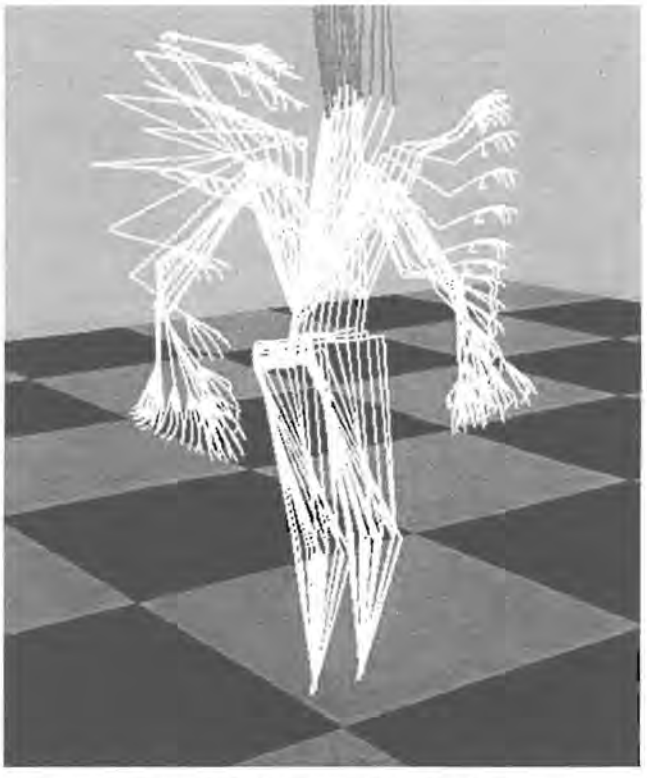


Figure 5-1d: Overlaid frames for the dance sequence.

5.2.2 Spatial content

Ranges

The range of each degree of freedom is mainly a function of the theoretical joint limits as discussed in chapter 2. However, table 5-2 summarizes the practical ranges as obtained from the motion sequences of figure 5-1, excluding the hands. Table 5-3 summarizes the characteristics of the finger movement for the gesture sequence. Also shown in the tables are the mean and standard deviation for each DOF.

Table 5-2: Summary of body DOF characteristics.

Description		Conversational sequence					Dance sequence				
Joint name	DOF	Min	Max	Range	Mean	Std. dev	Min	Max	Range	Mean	Std. dev.
Root	$\theta_{0,1}$	-71.4	59.7	131.2	-0.1	18.16	-63.6	51.5	115.1	-1.8	14.75
Root	$\theta_{0,2}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Root	$\theta_{0,3}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Root	$\theta_{0,4}$	-0.100	0.165	0.265	0.016	0.046	-0.163	0.247	0.410	0.034	0.088
Root	$\theta_{0,5}$	0.860	1.068	0.208	0.997	0.010	0.804	1.037	0.233	0.965	0.026

Root	$\theta_{0,6}$	-0.118	0.085	0.203	-0.011	0.028	-0.280	0.060	0.340	-0.060	0.049
Torso	$\theta_{1,1}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Torso	$\theta_{1,2}$	-16.3	31.9	48.2	6.5	6.05	-8.2	30.0	38.2	10.7	5.26
Torso	$\theta_{1,3}$	-29.6	20.4	50.0	-1.7	5.35	-26.8	28.9	55.7	1.1	6.61
Neck	$\theta_{2,1}$	-24.1	22.2	46.4	0.5	7.99	-29.1	22.8	51.9	-6.1	7.63
Neck	$\theta_{2,2}$	-15.0	15.0	30.0	0.7	4.97	-15.6	15.4	30.9	2.4	6.58
Head	$\theta_{3,1}$	-76.3	82.9	159.3	9.5	25.92	-88.5	89.2	177.7	-9.9	30.56
Head	$\theta_{3,2}$	-24.1	22.2	46.4	0.5	7.99	-31.4	22.8	54.2	-6.1	7.65
Head	$\theta_{3,3}$	-15.0	15.0	30.0	0.7	4.97	-15.6	15.4	30.9	2.4	6.58
Left shoulder	$\theta_{5,1}$	-82.8	131.8	214.5	29.5	35.66	-100.0	100.9	200.9	-7.5	30.10
Left shoulder	$\theta_{5,2}$	-110.3	-6.5	103.8	-27.0	11.65	-91.1	-3.9	87.2	-32.4	11.77
Left shoulder	$\theta_{5,3}$	-167.9	169.7	337.6	29.0	55.34	-95.4	85.0	180.4	8.8	30.63
Left elbow	$\theta_{6,1}$	-153.0	-4.0	149.0	-66.7	45.69	-148.8	-4.3	144.6	-86.0	22.42
Left elbow	$\theta_{6,2}$	-91.0	110.7	201.7	0.5	45.68	-38.7	90.4	129.0	45.8	22.83
Left wrist	$\theta_{7,1}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Left wrist	$\theta_{7,2}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Right shoulder	$\theta_{22,1}$	-113.6	104.8	218.4	-8.1	38.75	-103.5	104.5	208.0	1.8	29.79
Right shoulder	$\theta_{22,2}$	6.0	137.0	131.0	23.6	18.83	3.3	129.4	126.0	32.9	15.28
Right shoulder	$\theta_{22,3}$	-112.2	132.9	245.1	-7.5	39.55	-90.3	167.7	258.0	-4.7	28.92
Right elbow	$\theta_{23,1}$	-156.2	-4.0	152.1	-77.1	46.30	-161.4	-4.3	157.1	-74.4	29.41
Right elbow	$\theta_{23,2}$	-90.3	90.0	180.3	-15.3	31.93	-90.4	89.8	180.2	-24.4	33.60
Right wrist	$\theta_{24,1}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Right wrist	$\theta_{24,2}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Left hip	$\theta_{39,1}$	-32.6	7.2	39.8	-5.4	3.57	-42.6	2.7	45.4	-17.4	7.90
Left hip	$\theta_{39,2}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Left hip	$\theta_{39,3}$	-10.7	7.6	18.3	-0.9	3.18	-18.8	12.0	30.8	-2.2	5.95
Left knee	$\theta_{40,1}$	0.1	68.0	67.9	8.4	5.69	-0.8	80.4	81.2	25.1	13.92
Left ankle	$\theta_{41,1}$	-30.0	14.5	44.6	-2.9	4.15	-30.0	19.9	49.9	-7.7	8.27
Left ankle	$\theta_{41,2}$	-15.0	15.0	30.0	0.5	10.40	-15.1	15.8	30.9	1.3	9.80
Right hip	$\theta_{43,1}$	-32.4	4.8	37.2	-5.5	3.41	-41.9	4.0	45.9	-17.2	7.82
Right hip	$\theta_{43,2}$	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00
Right hip	$\theta_{43,3}$	-11.1	6.6	17.7	-1.3	3.22	-15.0	12.1	27.1	-2.5	5.88
Right knee	$\theta_{44,1}$	0.1	68.0	67.9	8.5	5.71	-0.8	80.5	81.3	25.4	13.91
Right ankle	$\theta_{45,1}$	-30.0	10.2	40.2	-3.0	3.84	-30.0	21.5	51.6	-8.1	8.06
Right ankle	$\theta_{45,2}$	-15.0	15.0	30.0	0.5	10.40	-15.1	15.8	30.9	1.3	9.80

Table 5-3: Summary of finger DOF characteristics.

Joint name	DOF	Min	Max	Range	Mean	Std. dev.
Left thumb 1	$\theta_{8,1}$	-0.0	30.0	30.1	11.7	4.73
Left thumb 2	$\theta_{9,1}$	-0.1	80.1	80.2	31.1	12.62
Left index 1	$\theta_{10,1}$	0.0	0.0	0.0	0.0	0.00
Left index 1	$\theta_{10,2}$	-0.1	74.9	75.0	27.4	16.74
Left index 2	$\theta_{11,1}$	-0.1	74.9	75.0	27.4	16.74
Left index 3	$\theta_{12,1}$	-0.1	60.0	60.0	21.9	13.39
Left middle 1	$\theta_{13,1}$	0.0	0.0	0.0	0.0	0.00
Left middle 1	$\theta_{13,2}$	-0.0	75.0	75.1	27.6	17.29
Left middle 2	$\theta_{14,1}$	-0.0	75.0	75.1	27.6	17.29
Left middle 3	$\theta_{15,1}$	-0.0	60.0	60.1	22.1	13.83
Left ring 1	$\theta_{16,1}$	0.0	0.0	0.0	0.0	0.00
Left ring 1	$\theta_{16,2}$	-0.1	75.1	75.2	24.5	16.08
Left ring 2	$\theta_{17,1}$	-0.1	75.1	75.2	24.5	16.08
Left ring 3	$\theta_{18,1}$	-0.0	60.1	60.1	19.6	12.86
Left little 1	$\theta_{19,1}$	0.0	0.0	0.0	0.0	0.00
Left little 1	$\theta_{19,2}$	-0.1	75.1	75.2	20.6	15.87
Left little 2	$\theta_{20,1}$	-0.1	75.1	75.2	20.6	15.87
Left little 3	$\theta_{21,1}$	-0.1	60.1	60.1	16.5	12.70
Right thumb 1	$\theta_{25,1}$	-0.0	30.1	30.1	11.7	6.44
Right thumb 2	$\theta_{26,1}$	-0.0	80.2	80.2	31.3	17.18
Right index 1	$\theta_{27,1}$	0.0	0.0	0.0	0.0	0.00
Right index 1	$\theta_{27,2}$	-0.0	75.1	75.1	21.7	20.26
Right index 2	$\theta_{28,1}$	-0.0	75.1	75.1	21.7	20.26
Right index 3	$\theta_{29,1}$	-0.0	60.1	60.1	17.4	16.21
Right middle 1	$\theta_{30,1}$	0.0	0.0	0.0	0.0	0.00
Right middle 1	$\theta_{30,2}$	-0.1	75.0	75.1	26.3	21.49
Right middle 2	$\theta_{31,1}$	-0.1	75.0	75.1	26.3	21.49
Right middle 3	$\theta_{32,1}$	-0.1	60.0	60.1	21.0	17.19
Right ring 1	$\theta_{33,1}$	0.0	0.0	0.0	0.0	0.00
Right ring 1	$\theta_{33,2}$	-0.1	75.1	75.1	29.6	20.58
Right ring 2	$\theta_{34,1}$	-0.1	75.1	75.1	29.6	20.58
Right ring 3	$\theta_{35,1}$	-0.1	60.0	60.1	23.7	16.46
Right little 1	$\theta_{36,1}$	0.0	0.0	0.0	0.0	0.00
Right little 1	$\theta_{36,2}$	-0.1	75.2	75.3	26.9	22.49
Right little 2	$\theta_{37,1}$	-0.1	75.2	75.3	26.9	22.49
Right little 3	$\theta_{38,1}$	-0.1	60.2	60.2	21.5	17.99

Resolution

The resolution of each DOF is a function of the resolution of the input device used to digitize the motion. As described in chapter 3, we use 6 DOF electromagnetic sensors for body tracking and fiber optical data gloves for finger flexure sensing. The electromagnetic sensors output a 16-bit value for a ±3 m and a ±180° range respectively [74], while the glove device outputs an 8-bit value for full flexure [75]. The output of both of these

devices is converted to a floating-point representation for internal use. The useable hardware resolution of the electromagnetic sensors is a function of the distance from the transmitter, as was indicated in figure 4-1. The useable glove sensor hardware resolution is constant under all circumstances. The total useable resolution in bits is a function of both the hardware and range resolution, denoted by N_h and N_r respectively. Using the noise power σ_p^2 from figure 4-1, we find that the maximum number of useable bits for the 3 position DOFs is given by

$$\begin{aligned} N_p &= 16-(N_h+N_r) \\ &= 16-\left(\log_2\left(\frac{65536.\sigma_p}{6}\right)+\log_2\left(\frac{6}{R_p}\right)\right), \end{aligned} \tag{5-1}$$

where R_p is the range as found in table 5-2. The bits for the 3 angular DOFs is given by

$$\begin{aligned} N_a &= 16-(N_h+N_r) \\ &= 16-\left(\log_2\left(\frac{65536.\sigma_r}{360}\right)+\log_2\left(\frac{360}{R_a}\right)\right), \end{aligned} \tag{5-2}$$

where R_a is the range as found in table 5-2, and σ_a^2 is the noise power from figure 4-1. Similarly, if we assume a full finger flexure of roughly 70°, from chapter 4 we find that the useable number of flexion bits is given by

$$N_f = 8-\log_2\left(\frac{256.\sigma_f}{70}\right). \tag{5-3}$$

Assuming that the non-linear inverse kinematics calculations do not adversely influence resolution and range, the useable number of bits for each DOF can be found using the above mentioned equations. Table 5-4 shows the bit quantities with respect to the test sequences discussed above. A maximum, typical and minimum resolution is presented. Note that the number of finger bits is constant, and is only shown once. Throughout the

rest of the text, the *typical* value will be used for comparison purposes. Table 5-4 shows only the DOFs that we can actually measure, as described in chapter 4. In all fairness, only these DOFs should be used to calculate the raw, uncompressed bit-rate requirement. From table 5-4, it can be seen that the reduced skeleton model typically requires 345 bits/frame for the body and 168 bits/frame for the hands. At a sampling rate of 30 Hz, the bit-rate requirement is 15390 bits/second.

Table 5-4: Resolution in bits for each DOF

Joint name	DOF	Range	Maximum	Typical	Minimum
Root	$\theta_{0,1}$	131.2	14	12	9
Root	$\theta_{0,4}$	0.410	14	11	7
Root	$\theta_{0,5}$	0.264	13	11	7
Root	$\theta_{0,6}$	0.364	13	11	7
Torso	$\theta_{1,2}$	48.2	13	10	7
Torso	$\theta_{1,3}$	58.4	13	11	8
Neck	$\theta_{2,1}$	51.9	13	10	8
Neck	$\theta_{2,2}$	30.9	12	10	7
Head	$\theta_{3,1}$	177.7	15	12	9
Head	$\theta_{3,2}$	54.2	13	11	8
Head	$\theta_{3,3}$	30.9	12	10	7
Left shoulder	$\theta_{5,1}$	231.8	15	13	10
Left shoulder	$\theta_{5,2}$	106.4	14	12	9
Left shoulder	$\theta_{5,3}$	337.6	16	13	10
Left elbow	$\theta_{6,1}$	149.0	14	12	9
Left elbow	$\theta_{6,2}$	201.7	15	12	10
Right shoulder	$\theta_{22,1}$	218.4	15	13	10
Right shoulder	$\theta_{22,2}$	133.6	14	12	9
Right shoulder	$\theta_{22,3}$	279.9	15	13	10
Right elbow	$\theta_{23,1}$	157.3	15	12	9
Right elbow	$\theta_{23,2}$	180.4	15	12	9
Left hip	$\theta_{39,1}$	49.8	13	10	8
Left hip	$\theta_{39,3}$	30.8	12	10	7
Left knee	$\theta_{40,1}$	81.2	14	11	8
Left ankle	$\theta_{41,1}$	49.9	13	10	8
Left ankle	$\theta_{41,2}$	30.9	12	10	7
Right hip	$\theta_{43,1}$	46.7	13	10	7
Right hip	$\theta_{43,3}$	27.1	12	10	7
Right knee	$\theta_{44,1}$	81.3	14	11	8
Right ankle	$\theta_{45,1}$	51.6	13	10	8
Right ankle	$\theta_{45,2}$	30.9	12	10	7
Fingers	14 DOFs per hand	70	8	6	6

5.2.3 Temporal content and statistics

Average

The average of $\{\theta_{i,j}(n)\}$ is given by

$$\eta_{i,j} = E[\theta_{i,j}(n)] = \frac{1}{N}\sum_{n=1}^{N}\theta_{i,j}(n), \tag{5-4}$$

for a sequence of length N. For a stochastic process to be ergodic, we must be able to prove that the time averages are equal to the ensemble or probability averages. Although we do not prove it explicitly here, it is reasonable to assume that this is the case and that $\{\theta_{i,j}(n)\}$ is ergodic. The results from the following sections also give strong indications that this assumption is reasonable.

Variance

The variance of $\{\theta_{i,j}(n)\}$ is given by

$$\sigma_{i,j} = E\left[\left(\theta_{i,j}(n)-\eta_{i,j}\right)^2\right] = \frac{1}{N}\sum_{n=1}^{N}\left(\theta_{i,j}(n)-\eta_{i,j}\right)^2, \tag{5-5}$$

for a sequence of length N.

Autocorrelation

The autocorrelation function of $\{\theta_{i,j}(n)\}$ is given by

$$r_{i,j}(k,l) = E[\theta_{i,j}(k)\theta_{i,j}(l)]. \tag{5-6}$$

A stochastic process is wide sense stationary (WSS) if its mean is constant, i.e. $\eta_{i,j}$ is not a function of n, and its autocorrelation function depends only on the lag or time difference $m = k - l$. In this case, the autocorrelation function can be written as

$$r_{i,j}(m) = E[\theta_{i,j}(n)\theta_{i,j}(n+m)] = \frac{1}{N-m}\sum_{n=1}^{N-m}\theta_{i,j}(n)\theta_{i,j}(n+m), \qquad (5\text{-}7)$$

for a sequence of length N. We have evaluated the mean and autocorrelation functions of the example sequences for arbitrary DOFs and various time origins to test the validity of a WSS process.

Figure 5-3 shows the mean of the arbitrarily chosen head yaw angle $\theta_{3,0}$, the left shoulder elevation angle $\theta_{5,1}$, the index finger flexion $\theta_{27,1}$ and the middle finger flexion $\theta_{30,1}$ for the test sequences as a function of sample origin n. Although there is a small variation, the mean can comfortably be approximated by a constant. Figure 5-4a shows a number of overlaid autocorrelation functions of the head pitch angle $\theta_{3,1}$ for the conversational test sequence, evaluated at a number of arbitrary sample origins n. Figure 5-4b shows the same functions for the dance sequence. Similarly, figure 5-4c and 5-4d depict the autocorrelation functions for the left shoulder twist angle $\theta_{5,2}$. Figure 5-4e shows the overlaid autocorrelation functions for the right index finger flexion angles $\theta_{27,1}$. It is clear from the results that these functions depend little on the sample origin n, and are mainly a function of the lag m. The autocorrelation functions and mean of all the other DOFs exhibit similar behaviour, and it is therefore reasonable to assume that the stochastic process $\{\theta_{i,j}(n)\}$ is wide sense stationary. It should be noted that this is true *only* if all of $\{\theta_{i,j}(n)\}$ is within the *same type of motion*, and the concept of WSS cannot be extended to include human motion in general.

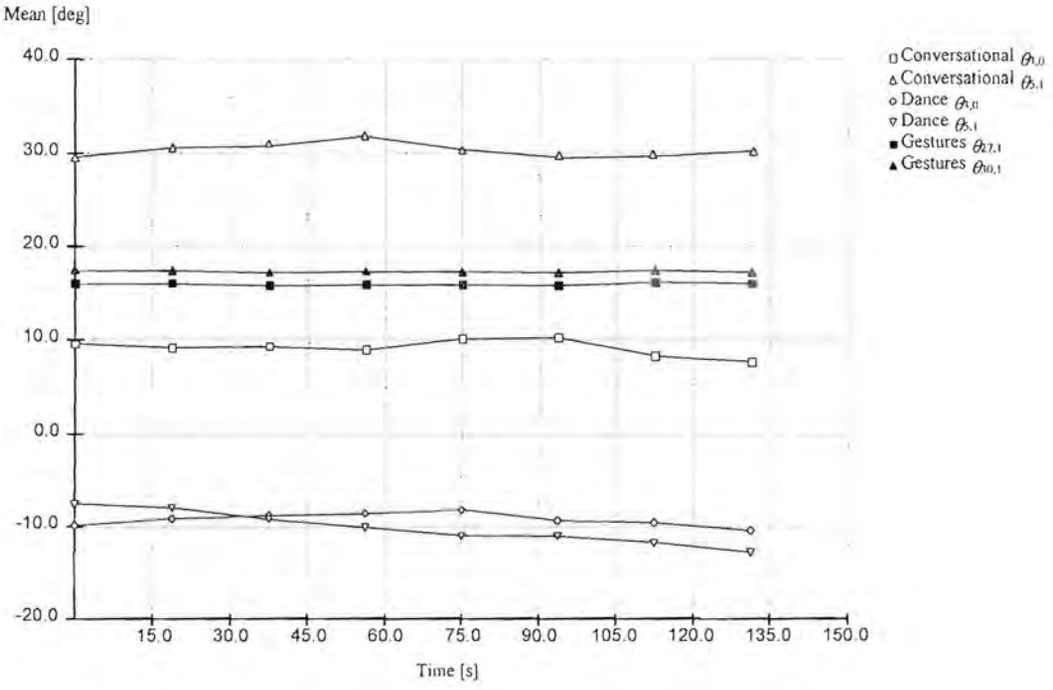


Figure 5-3: Mean as a function of time.

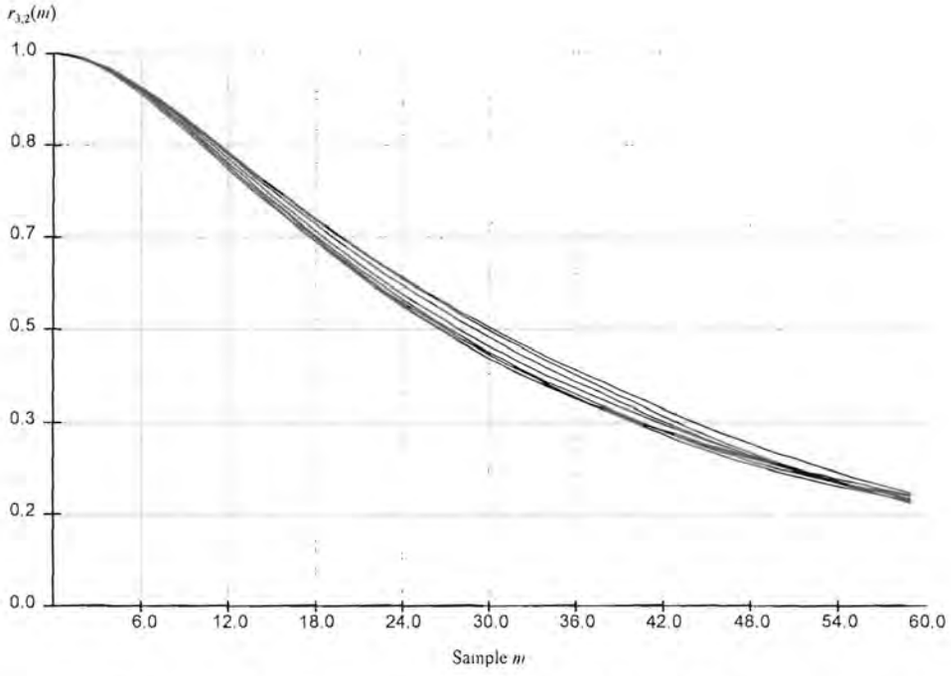


Figure 5-4a: Autocorrelation function of head pitch $\theta_{3,1}$ for conversational motion.

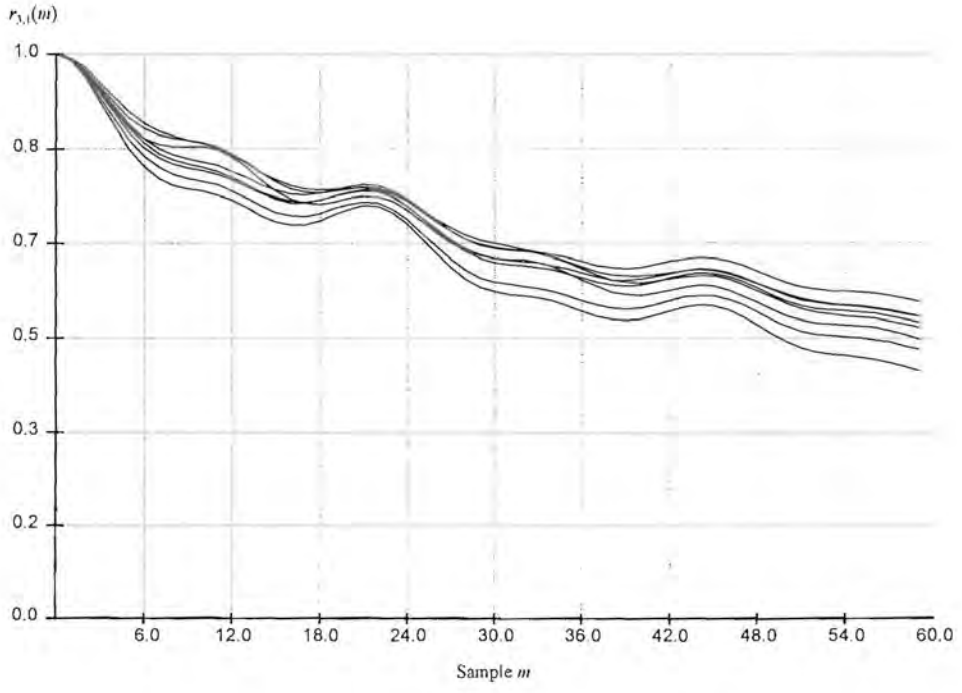


Figure 5-4b: Autocorrelation function of head pitch $\theta_{3,1}$ for dance motion.

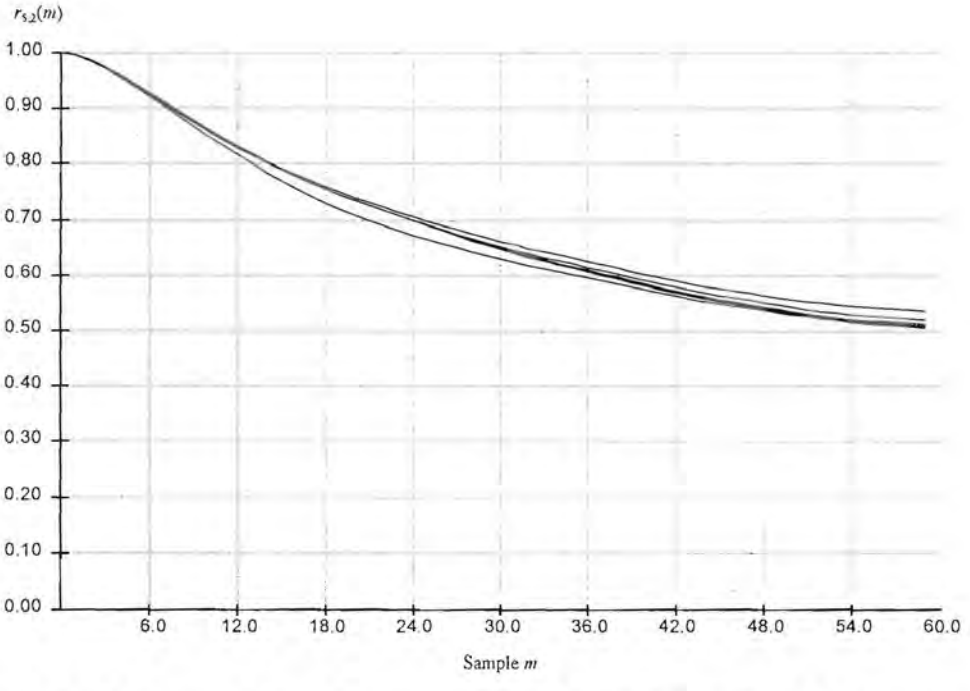


Figure 5-4c: Autocorrelation function of shoulder twist angle $\theta_{5,2}$ for conversational motion.

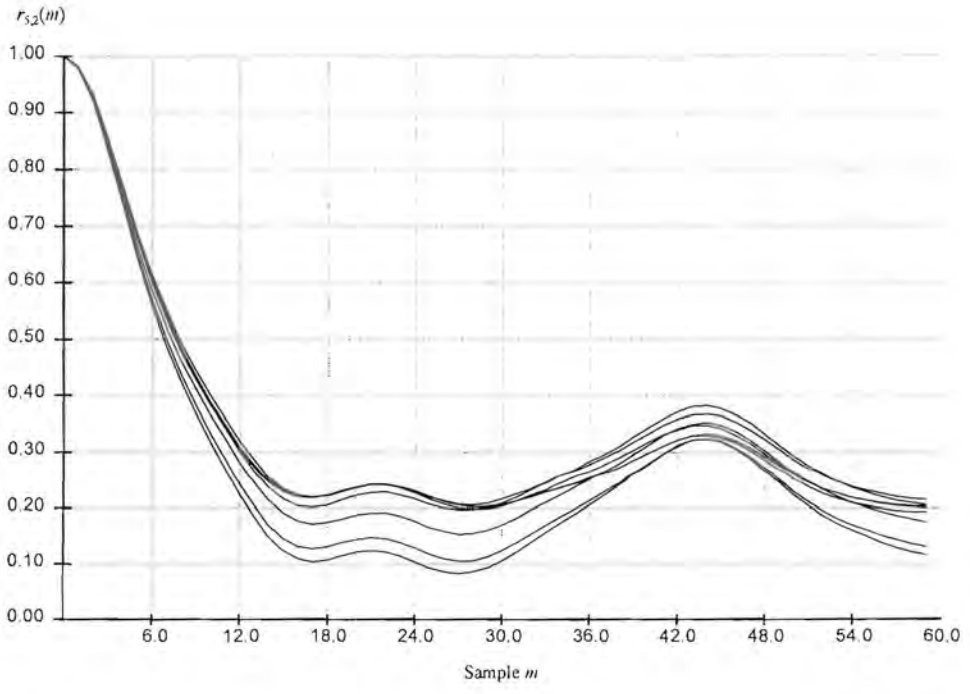


Figure 5-4d: Autocorrelation function of shoulder twist angle $\theta_{5,2}$ for conversational motion.

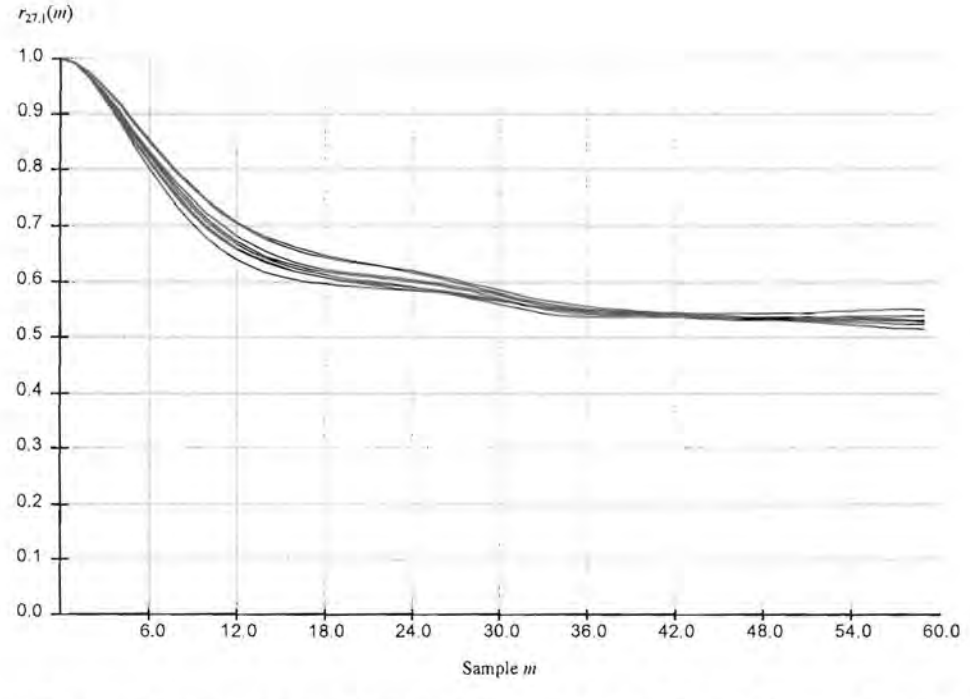


Figure 5-4e: Autocorrelation function of right index finger flexion angle $\theta_{27,1}$ for gesture motion.

It is often convenient to model a time signal as a Markov process, for which the correlation between samples is proportional to their time difference. The autocorrelation function of a discrete, second order, zero-mean Markov process can be written as

$$\phi(m) = \lambda e^{-\alpha^2 m^2}, \tag{5-8}$$

where λ and α are scaling constants. Figure 5-5a depicts the autocorrelation function for the left shoulder elevation angle $\theta_{5,1}$ for the conversational test sequence, with $\phi(m)$ where $\lambda = 1$ and $\alpha = 2.5\text{e-}4$. Figure 5-5b shows the same function for the dance sequence with $\alpha = 0.015$. It can be seen that there is a close match for $m < 10$, and the assumption that the motion can be modeled as a Markov process is reasonable. The other DOFs exhibit similar behaviour.

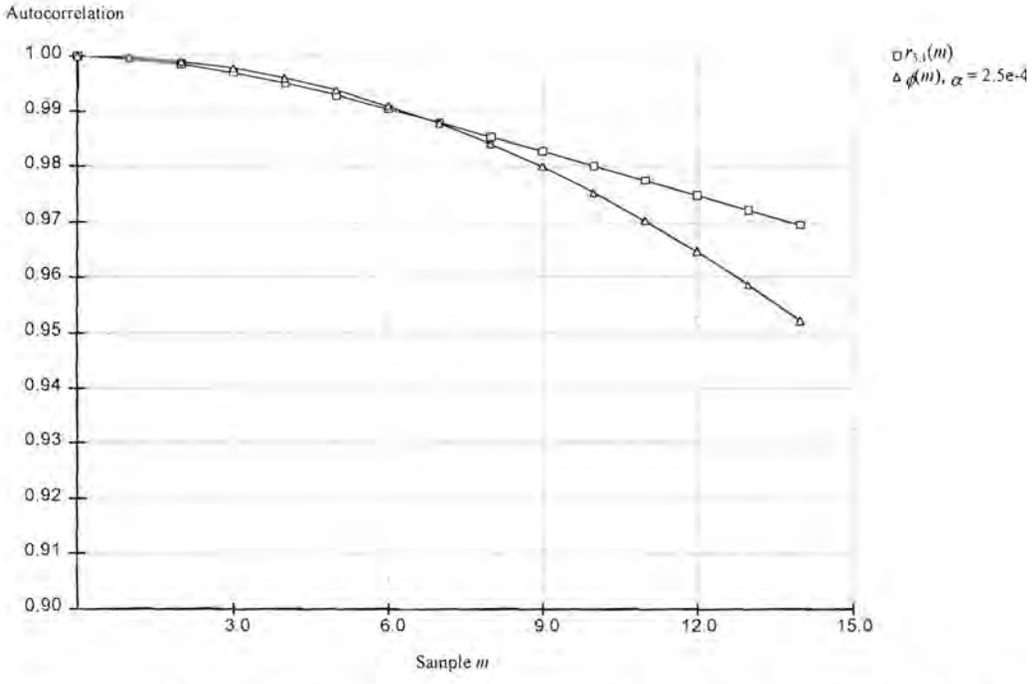


Figure 5-5a: Autocorrelation comparison with a Markov process for the conversational sequence.

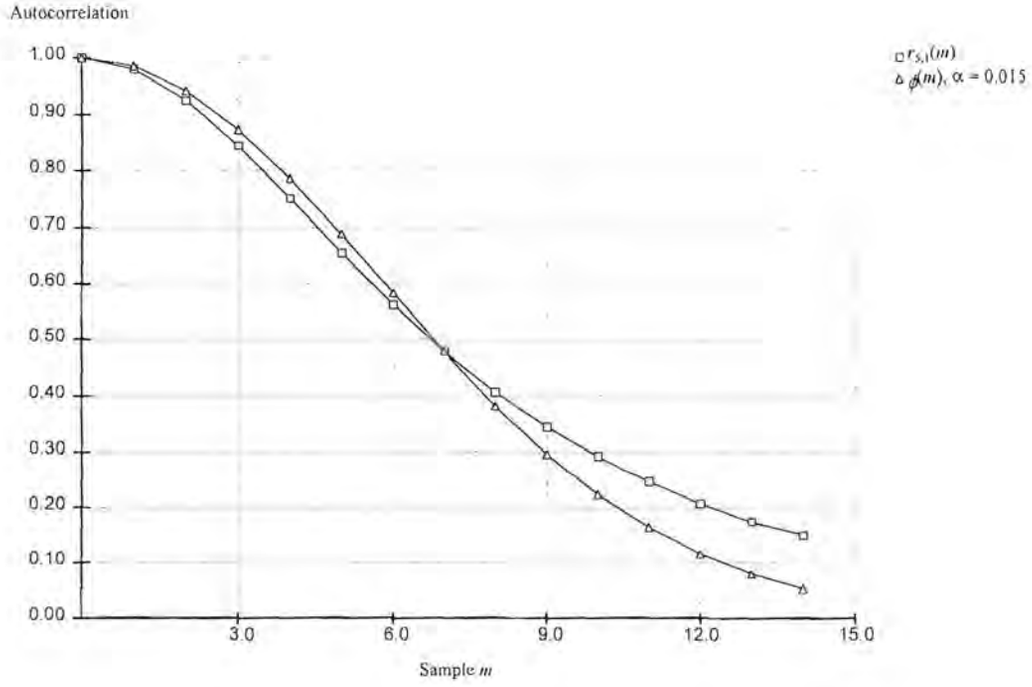


Figure 5-5b: Autocorrelation comparison with a Markov process for the dance sequence.

Probability density function (PDF)

From tables 5-1 and 5-2 it can be seen that most of the DOFs have vastly different ranges and characteristics, and that it would be necessary to obtain separate PDFs for each DOF. For clarity we only show examples of four such PDFs in figure 5-6a to 5-6d, which represent the arbitrary DOFs for the head, arm, leg and finger sections. It is clear that the graphs peak at the orientation favoured by the specific motion sequence, and are also an indication of the mean value. The shoulder angle shows two distinct peaks for the conversational motion. This is an indication that the person who performed the actions favoured two separate postures. The finger PDFs indicates either an open or closed gesture. Some of the body PDFs resembles a Gaussian-like distribution, except for those with strong peaks, in which case the sum of a number of distributions would be more appropriate. The finger PDFs resembles a one sided Rayleigh-like distribution for the open handed gesture and a Gaussian-like distribution for the other gestures.

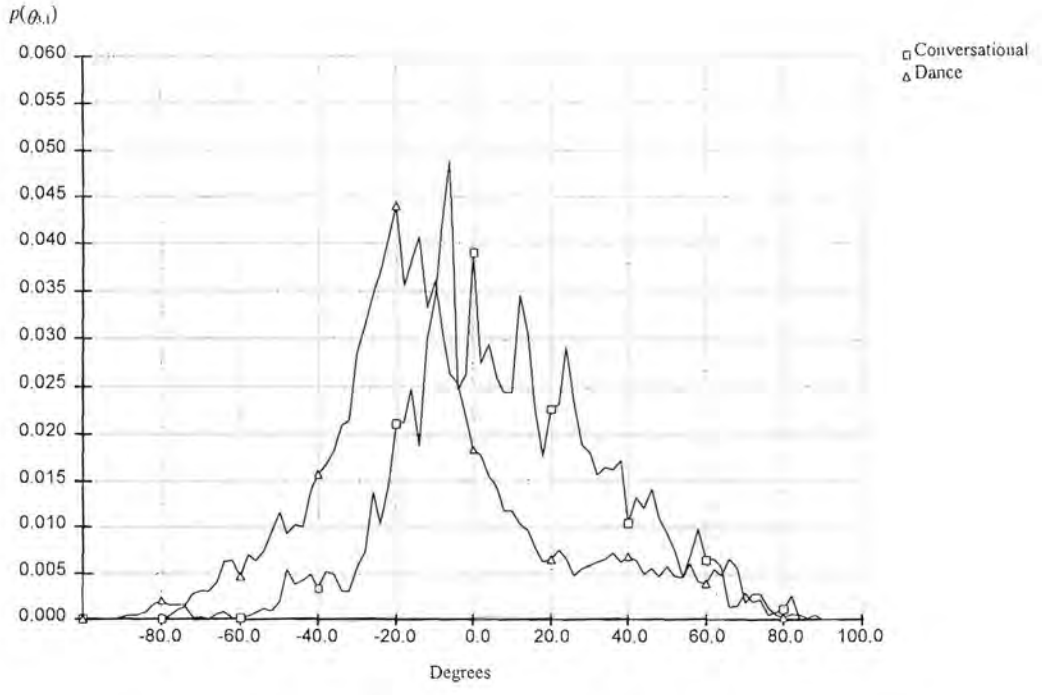


Figure 5-6a: PDF of the head angle $\theta_{3,1}$ for the conversational and dance motion.

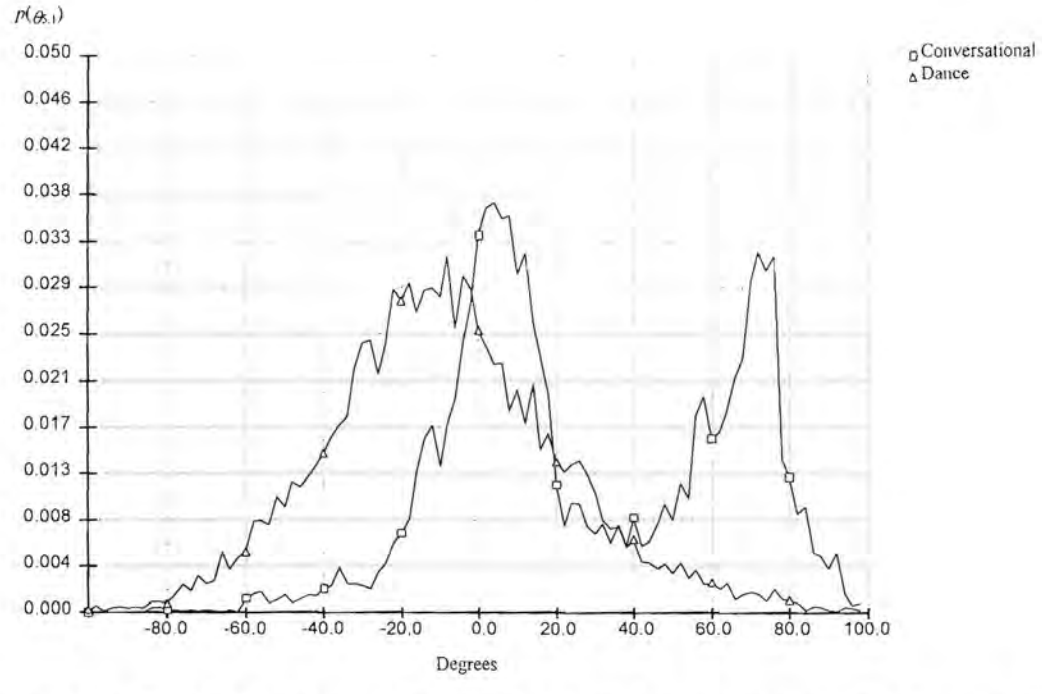


Figure 5-6b: PDF of the shoulder angle $\theta_{5,1}$ for the conversational and dance motion.

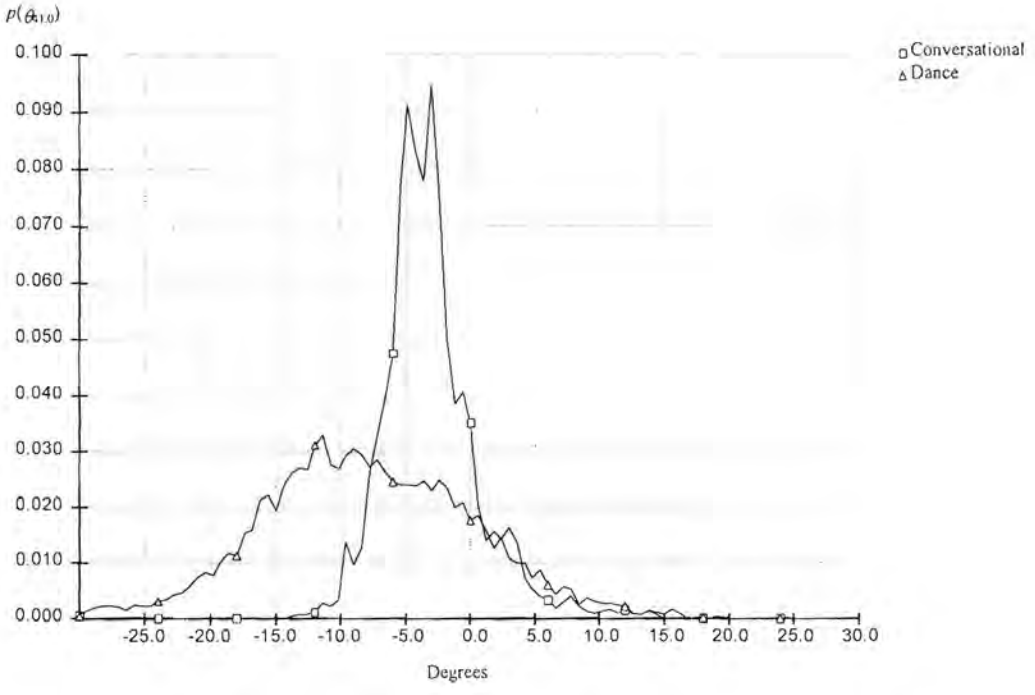


Figure 5-6c: PDF of the ankle angle $\theta_{41,0}$ for the conversational and dance motion.

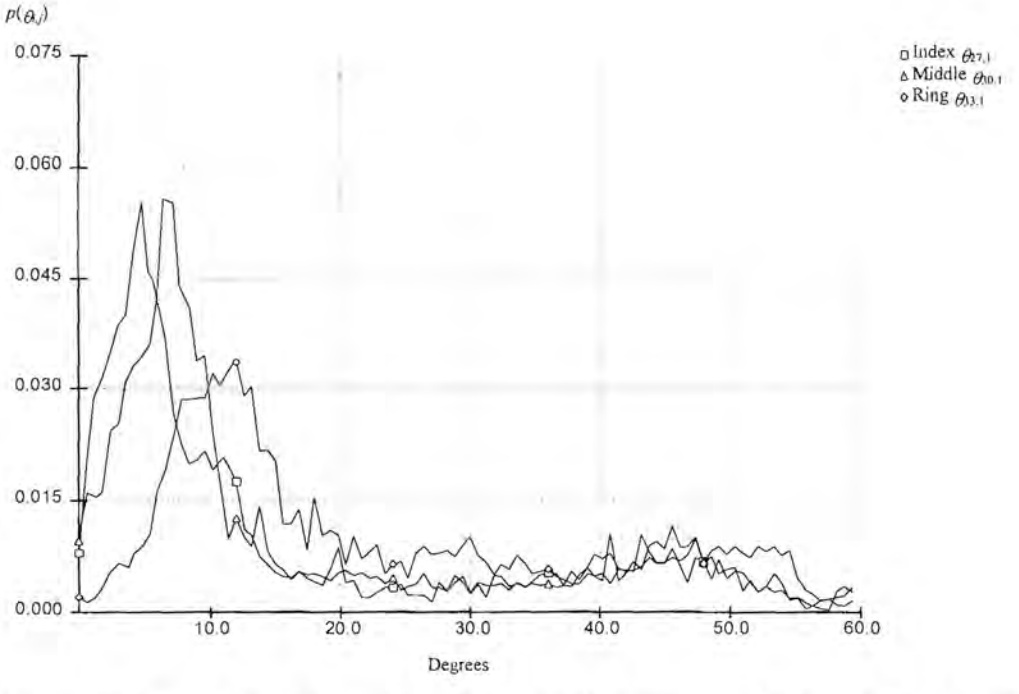


Figure 5-6d: PDF of the index, middle and ring finger flexion for the gesture sequence.

Derivative PDF

Linear predictive coders generally calculate a difference signal that is subsequently coded and transmitted. It is convenient to obtain a PDF for the first order difference $d_{i,j}(n) = \theta_{i,j}(n) - \theta_{i,j}(n\text{-}1)$ as an indication of the range and statistics of this difference signal. Under the assumption that there are a number of DOFs that exhibit similar difference behaviour, such as the arm DOFs or leg DOFs, it is possible to obtain a combined PDF by grouping these together. Figure 5-7 a-d depict the first difference PDFs for the head, arm, leg and finger groups. It can be seen that the first difference is a zero mean sequence, with considerably less variance than the DOF PDFs. Most of the difference PDFs resembles a Laplace-like distribution.

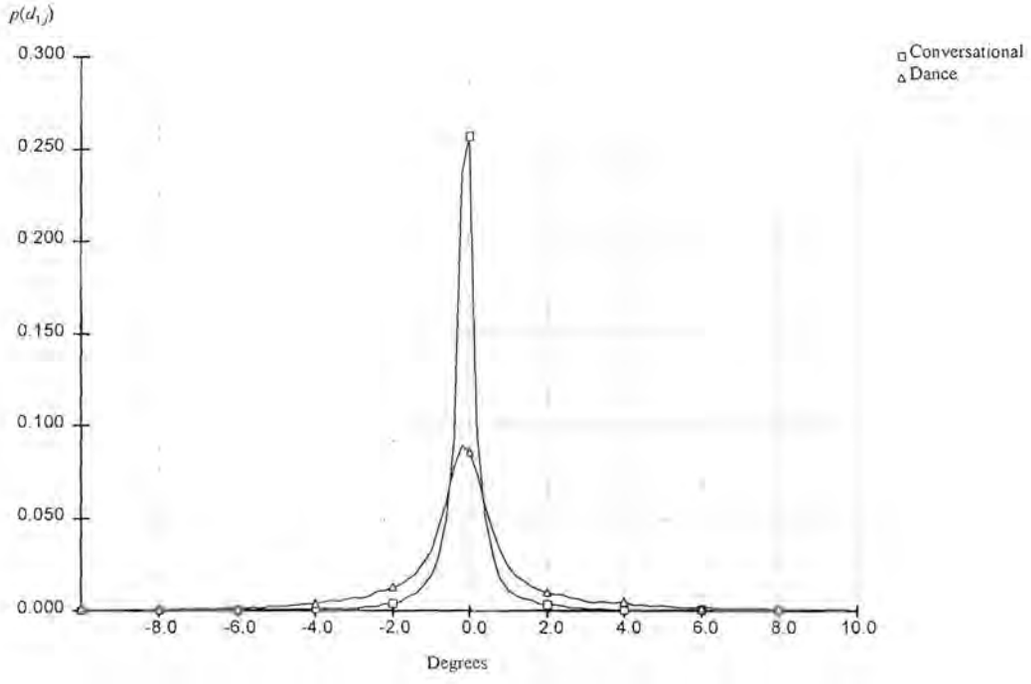


Figure 5-7a: PDF of the head joint group difference angle.

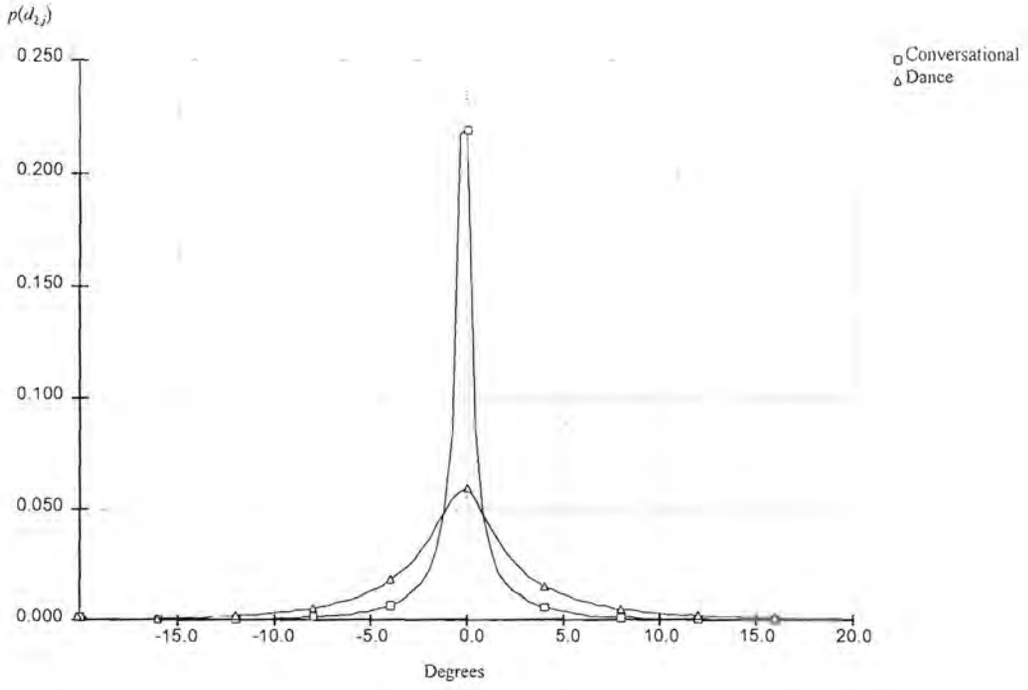


Figure 5-7b: PDF of the left arm joint group difference angle.

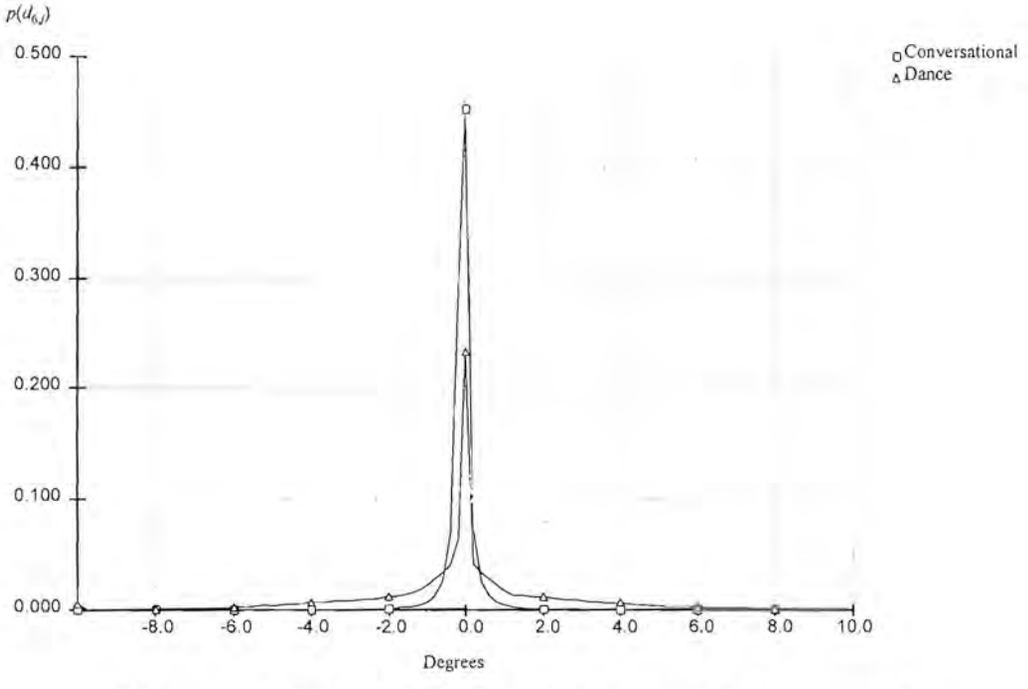


Figure 5-7c: PDF of the left leg joint group difference angle.

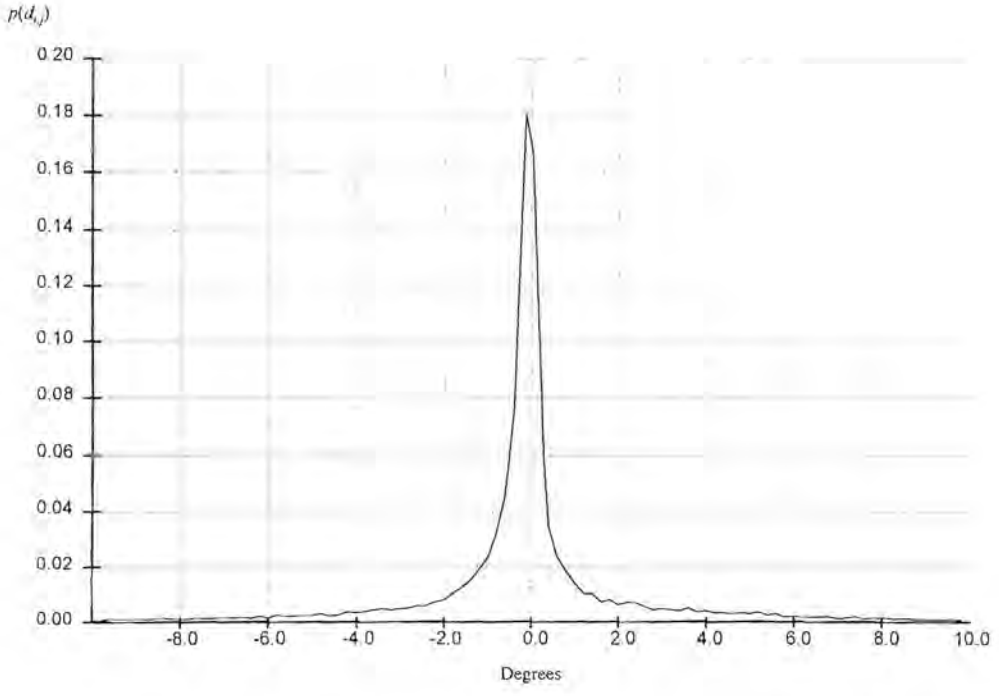


Figure 5-7d: PDF of the right hand finger joint group difference angle for the gesture sequence.

Joint cross dependence

There are a number of coding techniques that rely on the correlation or dependence between two or more joints. It is reasonable to assume that joints or DOFs from completely different body sections, such as the legs and arms, will have very little correlation. Therefore, only the correlation between the DOFs belonging to the body, head, arm, leg and finger sections will be investigated. One way of visualizing the relationship between DOFs is to plot them on a phase space diagram. Figure 5-8 shows such a plot of the left shoulder elevation angle $\theta_{5,1}$ against the other four arm DOFs for the dance sequence. There is a clear clustering behaviour, and it can be concluded that these DOFs are indeed dependant on $\theta_{5,1}$. It becomes tedious to plot every DOF against all the others in a group, and the graph quickly becomes cluttered, especially for long sequences. The rest of the body sections exhibit similar behaviour, and the results are not shown here.

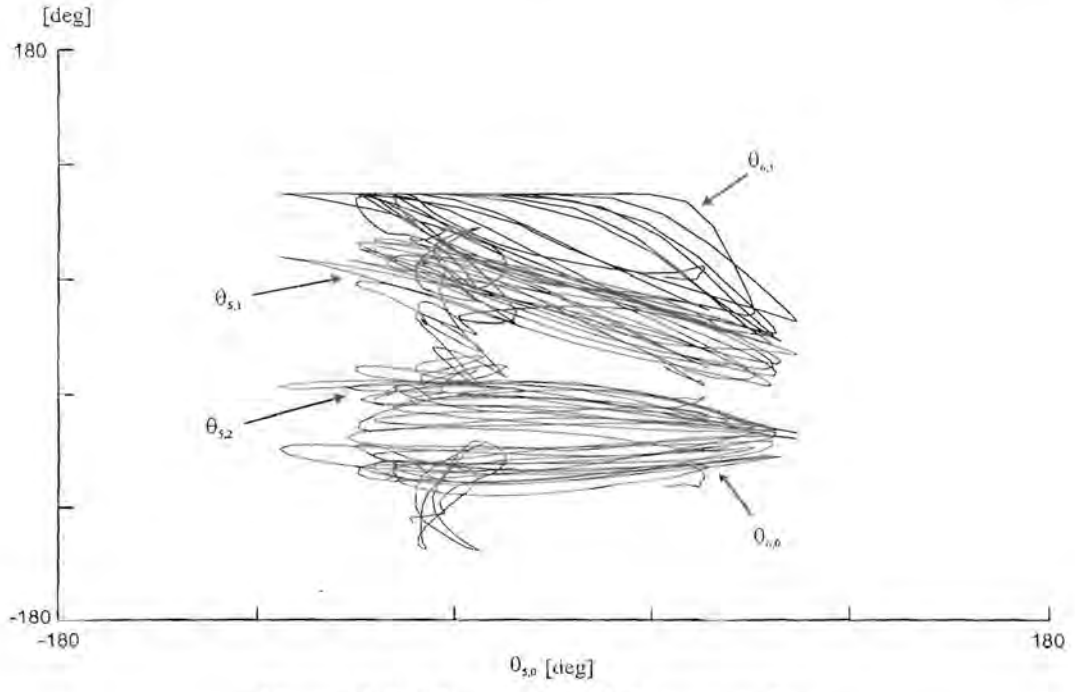


Figure 5-8: Phase plot of left arm angles.

A statistically more correct method is to calculate the joint PDF (where it is understood that joint does not mean physical human joint) for the DOFs in the relevant section. The visual results are kept to two-dimensional PDFs, as it is difficult to visualize more than that in a three-dimensional world. Figures 5-9 a-d show various PDFs of arbitrarily chosen body DOF pairs for the conversational test sequence, while figures 5-10 a-d similar PDFs for the dance sequence. The PDFs are shown as gray scale bitmaps, with a darker value indicating a higher occurrence. They all range from -180° to 180° on both axis. A single point or line on the PDF indicates that one or both of the DOFs is constant, while a large round cluster indicates not much of a cross correlation. However, it is clear from the images that the DOFs are indeed dependent on each other. Figures 5-11a and 5-11b show joint PDFs of arbitrarily chosen finger DOF pairs for the gesture sequence. The range is 0° to 60° on both axis. The finger DOFs are extremely correlated, and this fact will be used later to achieve higher compression ratios.

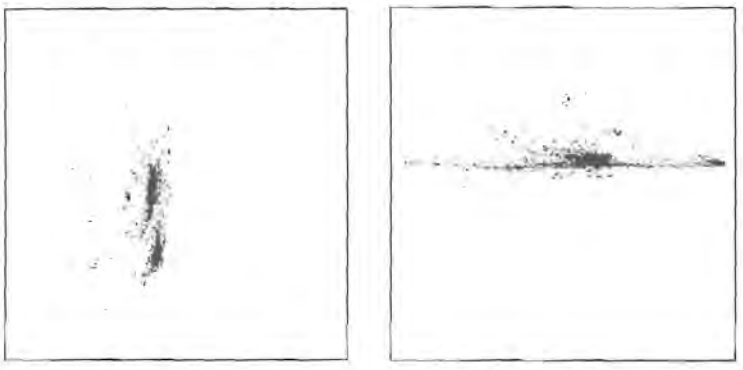


Figure 5-9: a) Joint PDF for $\theta_{5,0}$ and $\theta_{5,1}$ and b) Joint PDF for $\theta_{5,1}$ and $\theta_{5,2}$ for the conversational sequence.

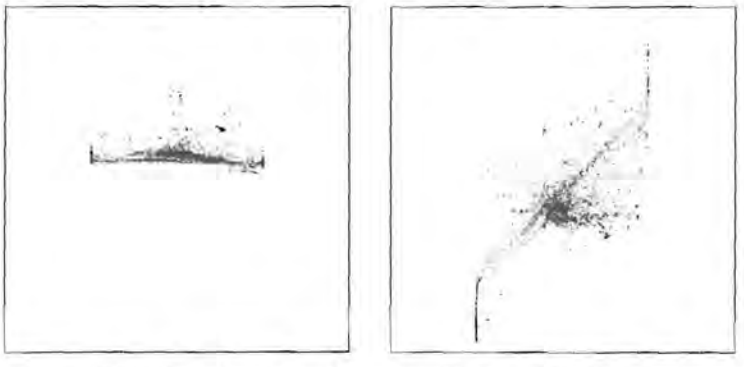


Figure 5-9: c) Joint PDF for $\theta_{5,1}$ and $\theta_{6,1}$ and d) Joint PDF for $\theta_{5,2}$ and $\theta_{6,1}$ for the conversational sequence.

Figure 5-10: a) Joint PDF for $\theta_{5,0}$ and $\theta_{5,1}$ and b) Joint PDF for $\theta_{5,1}$ and $\theta_{5,2}$ for the dance sequence.

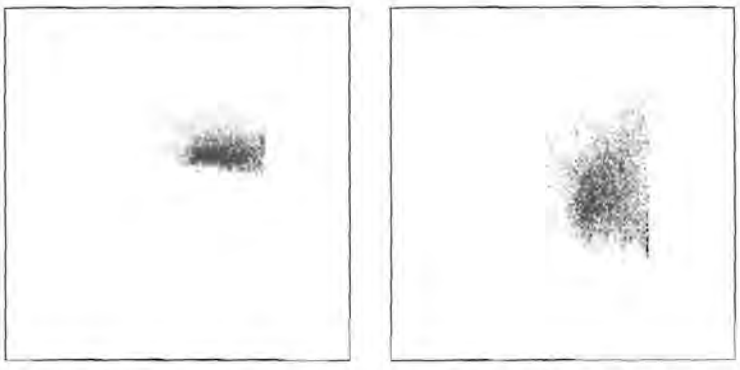


Figure 5-10: c) Joint PDF for $\theta_{5,1}$ and $\theta_{6,1}$ and d) Joint PDF for $\theta_{5,2}$ and $\theta_{6,1}$ for dance sequence.

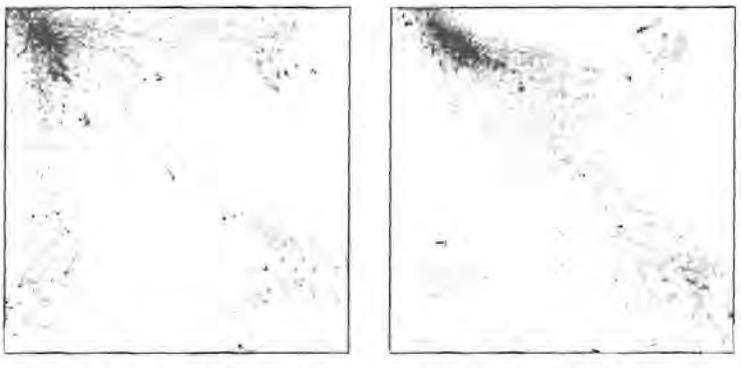


Figure 5-11: a) Joint PDF for $\theta_{27,1}$ and $\theta_{30,1}$ and b) Joint PDF for $\theta_{30,1}$ and $\theta_{33,1}$ for the gesture sequence.

5.2.4 Frequency content

A very important measure of motion information can be found by investigating the frequency content of the motion. It can be used to conclude the minimum sample rate required to capture, process or display human motion. The power spectrum or power spectral density (PSD) of a WSS process $\{\theta_{i,j}(n)\}$ is defined as the Fourier transform of its autocorrelation function. To avoid an impulse at the origin in the case of a process where the mean $\eta_{i,j}$ is non-zero, it is often more convenient to use the autocovariance of the process, which is given by

$$c_{i,j}(m) = E\left[\left(\theta_{i,j}(n) - \eta_{i,j}\right)\left(\theta_{i,j}(n+m) - \eta_{i,j}\right)\right] \tag{5-9}$$

The PSD is then defined by

$$P_{i,j}(\omega) = \sum_{n=-\infty}^{n=\infty} c_{i,j}(n) e^{-j\omega n} . \qquad (5\text{-}10)$$

For a finite length sequence, only an estimate of the PSD can be made, but the term will be used anyway. There are a number of efficient algorithms that can be used to evaluate equation (5-10). We use the Blackman-Tukey [47] method for general PSD calculations and a parametric model based approach for smooth spectra. In the latter case we assume that human motion spectra have broadband characteristics, and autoregressive (AR) parameters are obtained using the autocorrelation method [47].

It is convenient to assume that there are DOFs that exhibit similar frequency behaviour, and to group them together to obtain a combined PSD for the relevant body section. Similar to groupings done elsewhere, we calculate PSDs for the body, head, arm, leg and finger sections. Figures 5-12 a-d show long-term PSDs (i.e. the average PSD of the *whole* sequence) for both the conversational and dance sequences. Figure 5-12e shows the PSD of the fingers for the gesture motion sequence, together with the PSD for arm movement of the same sequence. If a suppression of 50 dB is taken as the cut-off threshold for perceptible motion, then it is clear that the average frequency content is limited to roughly 3 Hz and 6 Hz for the conversational and dance sequences respectively. The finger content is slightly more, which is to be expected since the inertial forces are the smallest on the fingers. However, these results do not imply that the short-term content will necessarily follow the same pattern. Figure 5-13 shows a short term PSD of arm movement at various time intervals for the conversational test sequence, and figure 5-14 a similar PSD for the dance sequence. It can be seen that the short term frequency content stays relatively constant with time. The higher mid-frequencies can clearly be seen across the time range for the dance sequence.

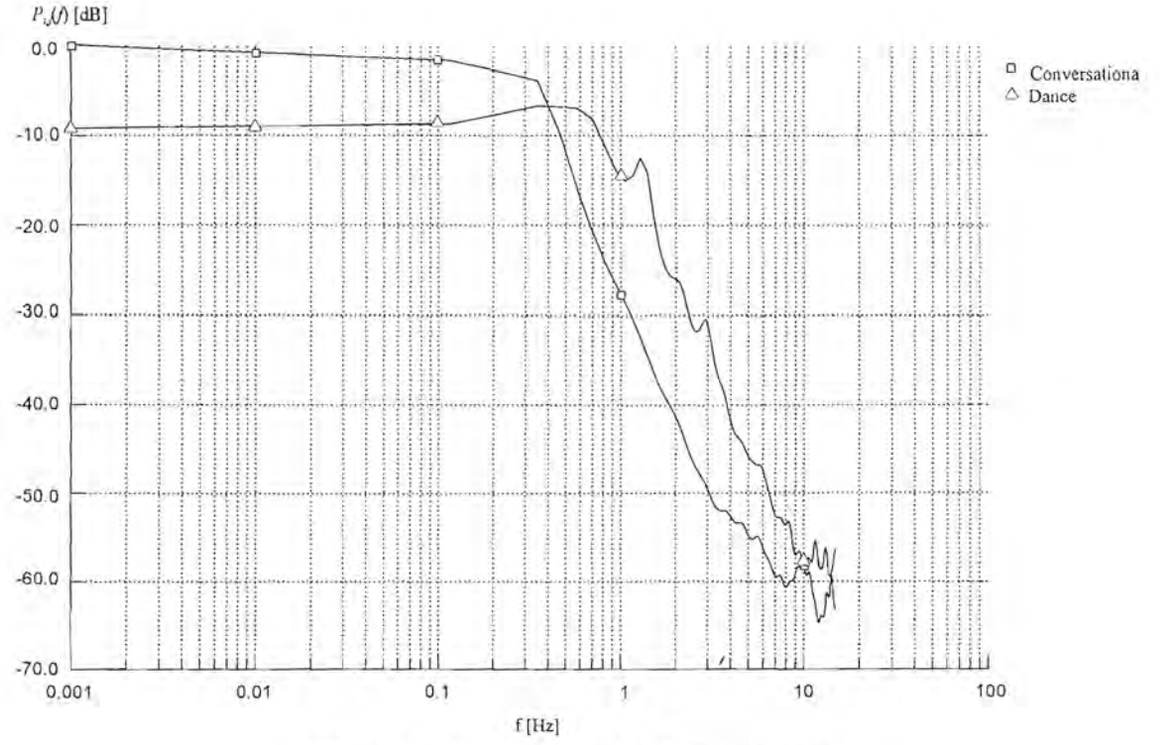


Figure 5-12a: PSD for body movement.

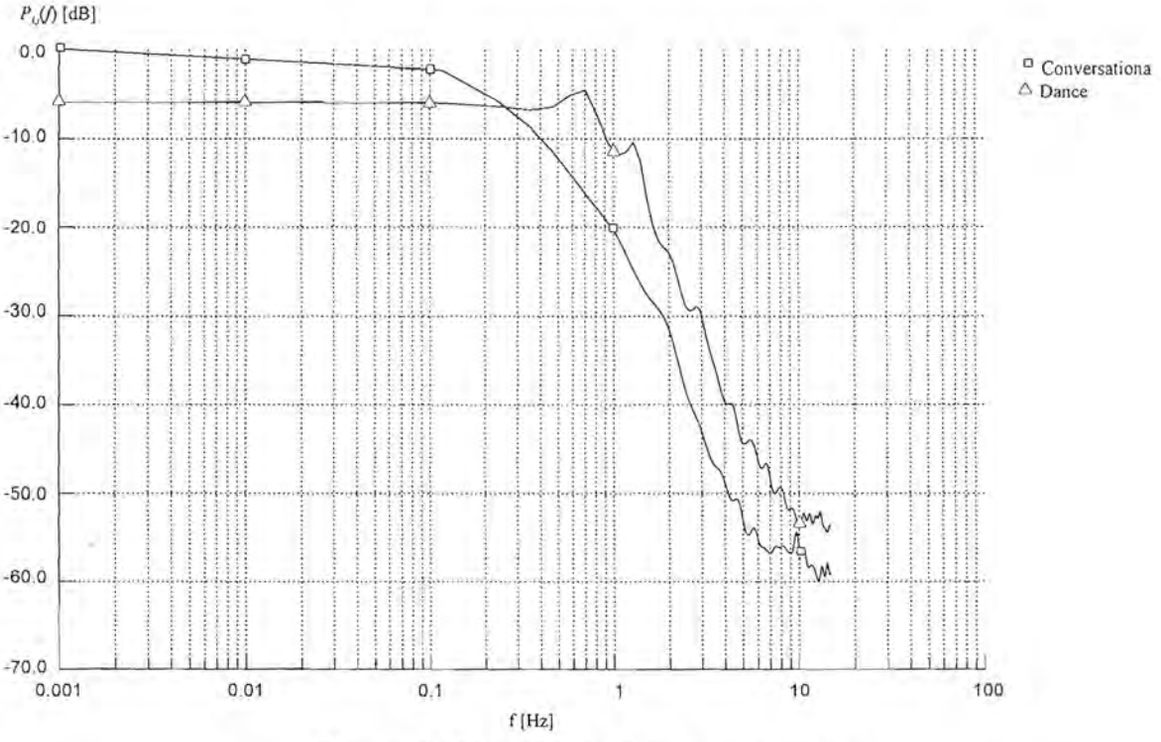


Figure 5-12b: PSD for head movement.

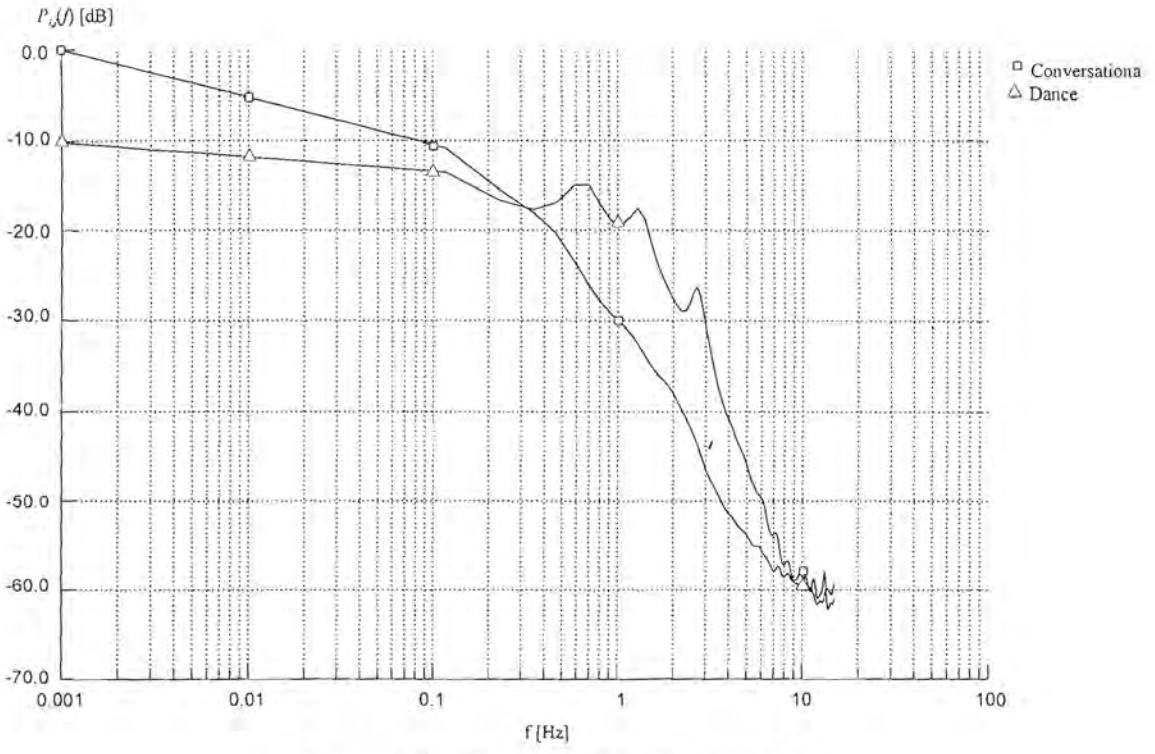


Figure 5-12c: PSD for arm movement.

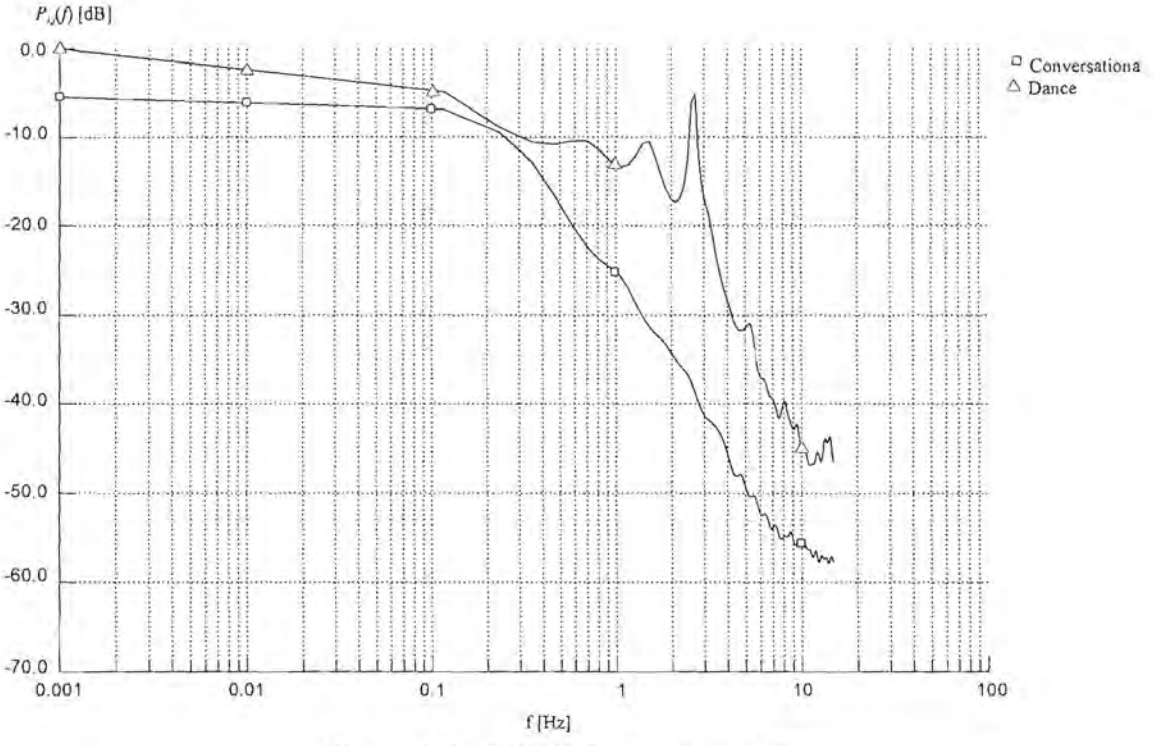


Figure 5-12d: PSD for leg movement.

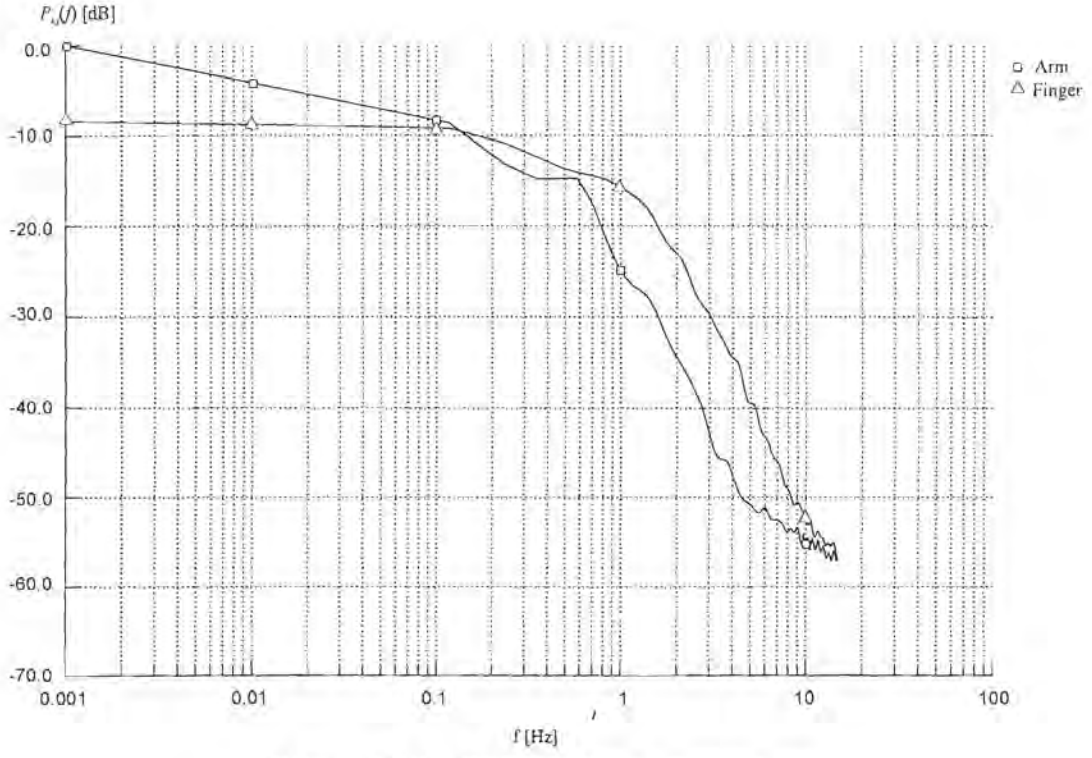


Figure 5-12e: PSD for finger movement.

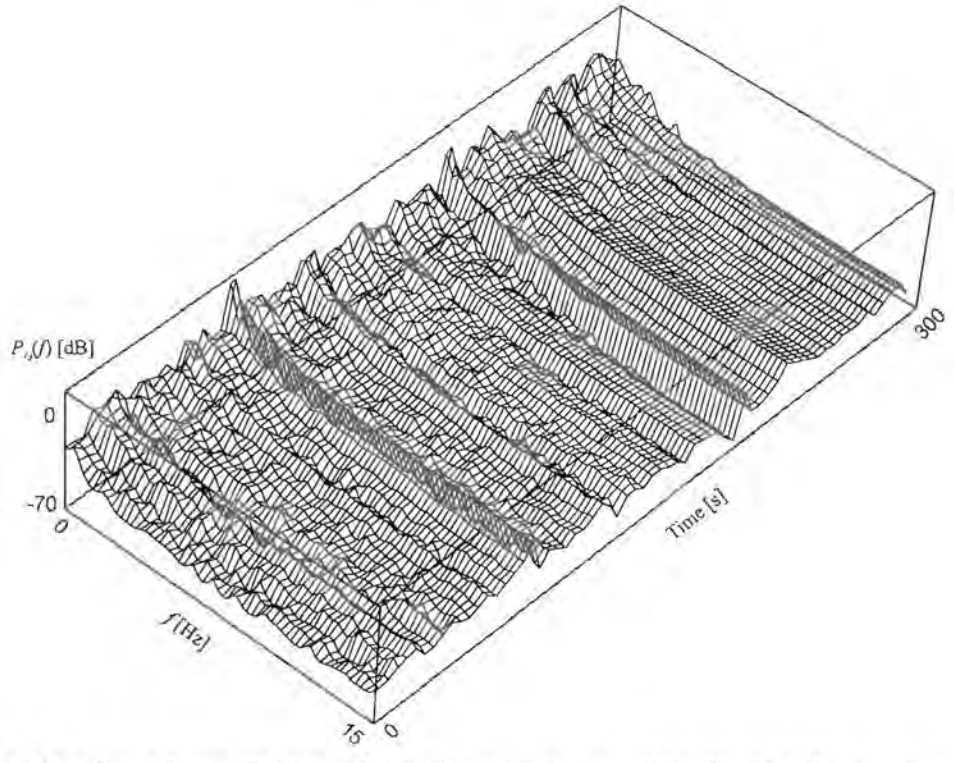


Figure 5-13: Short term PSD vs. time of the body position $\theta_{0,4}$ for the conversational sequence.

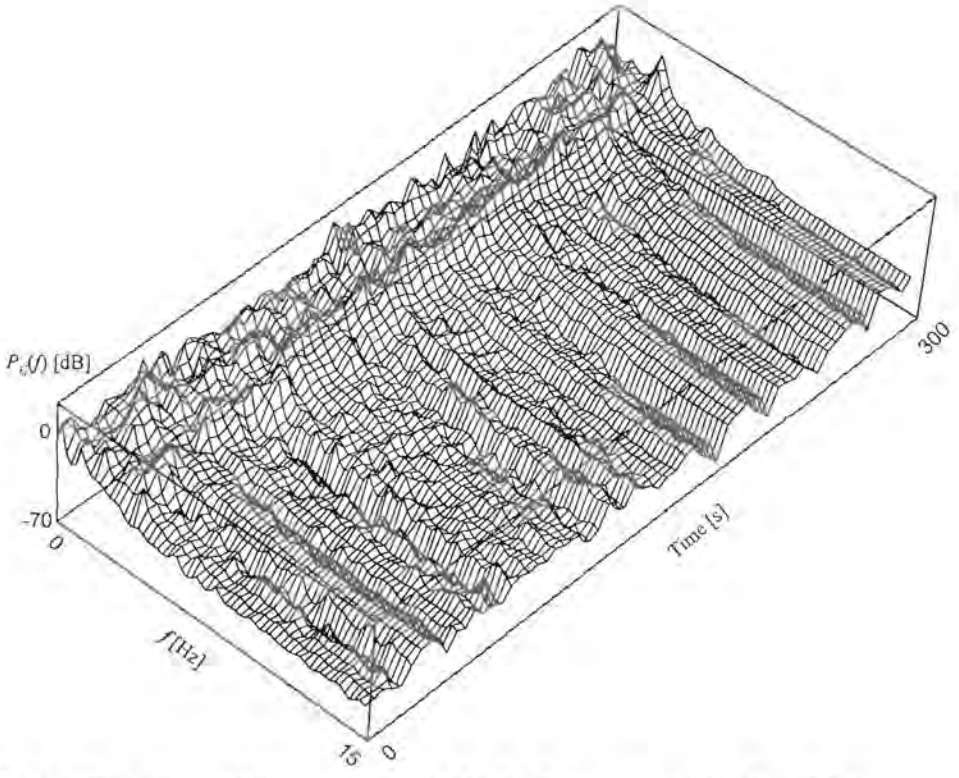


Figure 5-14: Short term PSD vs. time of the body position $\theta_{0,\theta}$ for the dance sequence.

5.3 Summary

This chapter presented a detailed statistical analysis of the human motion captured by the techniques described in chapter 4. The spatial content in terms of range and resolution was investigated, and it was found that these quantities rely on both the nature of the motion as well as the performance of the capturing hardware. Temporal content and statistics were investigated and it was found that it is reasonable to assume the motion data to be ergodic and wide sense stationary. Probability density studies revealed similarities in joint angle behaviour and indicated potential for predictive coding methods. Frequency content analysis indicated that human motion in general is rather band-limited, with the exception of a few peculiar movements. 40 dB cut-off was achieved at as low as 3 Hz for relaxed movement and the frequency content almost never exceeded 8 Hz, even for the dance motion.

Chapter 6 Error measurement

An error measure is a quantitative *or* qualitative indication of the amount of dissimilarity or distortion between two processes. Quantitative measures can be expressed mathematically and the result is some numerical value. Qualitative measures are a bit more difficult to pin down. They are usually given in some descriptive form, such as "it looks horrible". In the following sections, we will give an analysis of a number of error measurement techniques, and their applicability to human motion. Low level coding methods, such as waveform coding, require quantitative error measures that are mathematically tractable. High level methods, such as model based coding play havoc with strictly quantitative error measurements, since there is usually not a one-to-one relationship between the original and coded motion. The best that one can do for model based coding is to define some long-term measurement that will give an indication of the visual quality, or to develop subjective testing mechanisms.

6.1 Quantitative measures

6.1.1 MS error

One of the most common and well-known error measurements is the Mean Square (MS) error. Assume a sequence of values (or degrees of freedom) $\{\theta(n)\}$, and a processed sequence $\{\theta'(n)\}$, which is an approximation of $\{\theta(n)\}$. For clarity the subscript i,j is dropped, and it is understood that the sequence $\{\theta(n)\}$ can represent any DOF. The mean square error is given by

$$\mathrm{MSE} = \frac{1}{N}\sum_{n=1}^{N}\left(\theta(n)-\theta'(n)\right)^2, \tag{6-1}$$

for a sequence of length N. The square root of the MS error is sometimes more convenient, and is given by

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}. \tag{6-2}$$

Some useful variations on the MSE are the normalized MSE

$$\mathrm{NMSE} = \frac{\sum_{n=1}^{N}\left(\theta(n) - \theta'(n)\right)^2}{\sum_{n=1}^{N}\theta(n)^2} \tag{6-3}$$

and the peak MSE

$$\mathrm{PMSE} = \frac{\frac{1}{N}\sum_{n=1}^{N}\left(\theta(n) - \theta'(n)\right)^2}{R^2}, \tag{6-4}$$

where R is the range of $\{\theta(n)\}$. The mean square error is often described in logarithmic or decibel form as an equivalent signal-to-noise ratio (SNR)

$$\mathrm{NSNR} = -10\log_{10}(\mathrm{NMSE}), \tag{6-5}$$

or

$$\mathrm{PSNR} = -10\log_{10}(\mathrm{PMSE}). \tag{6-6}$$

Mean square error measurements are generally used as an evaluation tool after some process or operation has been completed, i.e. it is performed on a whole sequence of values.

6.1.2 Instantaneous error

Many of the compression algorithms require an error measurement that is applicable to the current frame or update. The best that can be done in this case is to use a distance metric as the error measurement. If $\theta(n)$ is a value at the nth sample, and $\theta'(n)$ is an approximation of $\theta(n)$, the distance is simply given by

$$d(n) = |\theta(n) - \theta'(n)|. \tag{6-7}$$

Sometimes it is desirable to use a metric that is mathematically more tractable (such as being easily differentiable), and we can use

$$d(n) = (\theta(n) - \theta'(n))^2. \tag{6-8}$$

6.1.3 Vector error

Although the distance errors specified in equation (6-7) and (6-8) are useful on their own, it is often convenient to group a number of dependent variables together as a vector and use their combined error (the reasons for doing so are explained in more detail in chapters 6 and 7). Mathematically it serves no purpose to group independent variables, as the uncorrelated result will be meaningless to the compression algorithm. Table 6-1 repeats the grouping scheme, together with the number of joints, segments and DOFs for each group. Refer to the human skeleton representation in figure 3-4 for details.

Table 6-1: Joint and segment grouping

Group	Reference number	Number of joints	Number of segments	Number of DOFs
Root and torso	0	2	2	7
Neck and head	1	2	2	5
Left arm	2	3	3	8
Left hand	3	14	14	19
Right arm	4	3	3	8
Right hand	5	14	14	19
Left leg	6	3	3	7
Right leg	7	3	3	7

Although we do not attempt to prove it here, it is reasonable to assume that the above combined joints or variables are correlated to some extent. The body specification of MPEG-4 uses a similar grouping scheme [39].

We denote the sequence of a group or vector of DOFs by $\{\theta_i(n)\}$, and the vector of the approximated DOFs by $\{\theta'_i(n)\}$, where $\{i = 0...7\}$. Individual components of the vector are denoted by $\{\theta_{i,j}(n)\}$ or $\{\theta'_{i,j}(n)\}$, where $\{i = 0...7, j = 0...K\text{-}1\}$ and K is the number of DOFs of the ith vector as given in table 6-1. Using this notation, we define the normalized weighted vector error of the ith group for the nth sample as

$$w_i(n) = \frac{1}{K}\sum_{j=0}^{K-1}\frac{a_{i,j}\left(\theta_{i,j}(n)-\theta'_{i,j}(n)\right)^2}{b_{i,j}^2}, \tag{6-9}$$

where K is the number of DOFs and $b_{i,j}$ is the range of the jth DOF. The quantity $a_{i,j}$ is a weighing coefficient that defines the contribution of the jth DOF to the error. If $a_{i,j}$ is in [0, 1], then $w_i(n)$ will be in [0, 1] with lower values indicating a good match. The values for $a_{i,j}$ and $b_{i,j}$ can also be defined in such a manner that the quantity $w_i(n)$ has meaningful units, such as [deg^2].

We are often interested in the *maximum* error for a group of joints, instead of a linear combination of errors. The maximum normalized weighted error for the nth sample of the ith group is given by

$$m_i(n) = MAX\left(\frac{a_{i,j}\left|\theta_{i,j}(n)-\theta'_{i,j}(n)\right|}{b_{i,j}}\right), \quad 0 \le j < K, \tag{6-10}$$

where K is the number of DOFs and $b_{i,j}$ is the range of the jth DOF. The quantity $a_{i,j}$ is a weighing coefficient that defines the contribution of the jth DOF to the error. If $a_{i,j}$ is in [0, 1], then $m_i(n)$ will be in [0, 1] with lower values indicating a good match. It should be

noted that equation (6-10) is not easily differentiable (compared to equation (6-9)), and is not very useful in error minimizing algorithms.

Equations (6-7) to (6-10) can also be used on a sequence of values similar to the definition of MS error and its variants. For example, we can write

$$W_i = \frac{1}{N}\sum_{n=1}^{N} w_i(n) \tag{6-11}$$

as the normalized weighted error on a whole sequence of length N. The maximum error can be redefined in a similar manner.

6.1.4 Joint and segment errors

Cases of special interest in human motion analysis are those of joint and segment position and/or orientation error, which are often geometrically more meaningful and intuitive than individual joint angle errors. By taking three-dimensional volume displacement into consideration, we get a bit closer to visual based comparisons between various body postures. We denote a sequence of joint positions by $\{\mathbf{u}_{i,j}(n)\}$, and that of the approximated joints by $\{\mathbf{u}'_{i,j}(n)\}$, where $\{i = 0...7, j = 0...K\text{-}1\}$. K is the number of joints for the ith group, and is given in table 6-1. The joint position error for the ith group of joints is given by

$$p_i(n) = \frac{1}{K}\sum_{j=0}^{K-1} \frac{a_{i,j}\left\|\mathbf{u}_{i,j}(n) - \mathbf{u}'_{i,j}(n)\right\|}{b_{i,j}}, \tag{6-12}$$

where $a_{i,j}$ and $b_{i,j}$ are weighing and normalizing coefficients similar to equation (6-9) and (6-10). It is often more meaningful to define the coefficients such that $p_i(n)$ has units of meters. The coefficients $a_{i,j}$ can also be defined as an impulse function to obtain the error for a single joint in the group.

When the axis of rotation is parallel to the rotated segment, the use of equation (6-12) on its own can sometimes result in complete failure to detect a rotation error. An example of this is the upper and lower arm twisting motion, both of which can result in a constant elbow or wrist joint position. To satisfy both position and rotation errors in a single generalized equation, we define additional DOFs for each group. The original and additional DOFs are grouped together in a *configuration* vector **c**. Configuration vectors describe both joint position and rotation. For example, the 14-dimensional configuration vector for the left arm group would be given by

$$\mathbf{c}_2 = [e_x\ e_y\ e_z\ w_x\ w_y\ w_z\ \theta_{2,0}\ \theta_{2,1}\ \theta_{2,2}\ \theta_{2,3}\ \theta_{2,4}\ \theta_{2,5}\ \theta_{2,6}\ \theta_{2,7}], \tag{6-13}$$

assuming that the shoulder is fixed at the world origin [0 0 0]. The additional DOFs are the elbow position, which is given by $[e_x\ e_y\ e_z]$, and the wrist position, which is given by $[w_x\ w_y\ w_z]$. A sequence of configuration vectors for the ith group is written as $\{\mathbf{c}_i(n)\}$, and individual components as $\{c_{i,j}(n)\}$. Using similar notation as in equation (6-9), the generalized error for the ith group can be written as

$$\varepsilon_i(n) = \frac{1}{K}\sum_{j=0}^{K-1}\frac{a_{i,j}\left(c_{i,j}(n) - c'_{i,j}(n)\right)^2}{b_{i,j}^2}, \tag{6-14}$$

where K is the number of elements in the configuration vector. Given proper coefficients, equation (6-14) is a useful error measure under many conditions. We obtained suitable values for $a_{i,j}$ and $b_{i,j}$ for equations (6-9), (6-10), (6-12) and (6-14) using heuristic methods and subjective testing.

6.2 Visual measures

Visual error measurement implies a method that will tell us whether the visual posture and motion of the human figure are acceptable, and if possible, to what extent. It should be noted that there is often a vast difference between a visual measure and a strictly mathematical measure. If the animation has natural and pleasing motion, it does not

necessarily mean it has the correct original position or orientation. Visual measurement techniques often rely on subjective tests by a panel of viewers. Many parameters of our compression techniques were obtained in this manner. However, it need not be done only subjectively. In fact, it would be desirable to define an objective visual measure that is mathematically tractable. When seeking such a solution, there is often no clear mathematical relationship between the original quantity and distorted quantity, and we are forced to look at the characteristics of these quantities separately.

6.2.1 Natural movement

One method of identifying visual artifacts is by evaluating the joint angles and their first and second derivatives for discontinuities or abnormally large values. Naturally, if both the original and coded values contain such anomalies not much can be said about the error. However, if the decoded motion exhibits values that are out of bounds compared to the original, it is reasonable to assume that something had gone wrong in the coding process. A more advanced method than simply identifying discontinuities is to compare the decoded human motion with dynamically simulated motion. One way of doing this is to calculate the metabolic energy spent in performing a motion, and to compare it to the original. It has been established that humans try to accomplish movement using the least amount of energy [44]. Abnormally large values indicate unnatural movement, and can be considered as an error in the coding process. Unfortunately, the methods described above rely primarily on the decoded sequence. We need at least some reference to the original sequence, otherwise the error between completely different original and decoded actions will be pronounced acceptable.

Discontinuities and unnatural movement aside, common errors on a waveform level are primarily due to *phase* and *amplitude* differences[1]. Phase errors are usually generated by coding delay and motion interpolation approximations. Amplitude errors are primarily generated by quantization in the spatial, temporal and frequency domains. We have found

[1] Not to be confused with the actual amplitude and phase functions of the original signal.

that phase errors are visually more tolerable than amplitude errors, especially high frequency amplitude errors. For example, spatial quantization generates high frequency discontinuities and jerkiness, and the differentiating characteristics of the human visual system causes such errors to be perceived as visually annoying. It is common practice to compensate for the (known) coding delay when calculating quantitative errors. The remaining phase error is therefore primarily a function of the compression method. These errors vary relatively slowly over time compared to quantization errors, which can occur at every sample. Phase errors in general result in fewer high frequency discontinuities and artifacts.

6.2.2 Visual MS error (VMSE)

The observed low and high frequency relationship between phase and amplitude errors led us to develop the visual mean square error, or VMSE. Figure 6-1 shows a conceptual diagram of the method. The difference between the original and coded signal (i.e. the error) is divided into a number of frequency bands, each is assigned a certain weight, and the results are combined again. By adjusting the coefficients α_i, the importance of various visual dissimilarities and artifacts that exist between the original and coded sequences can be set. Naturally, by setting all of the coefficients to unity, the VMSE measurement reduces to the normal MSE measurement. Similar to the MS error defined in equation (6-1), it is understood that by *signal* we mean any DOF, and that the VMSE of the total figure is given by the sum of the VMS errors of some or all of the DOFs. Mathematically, the VMSE can be written as

$$VMSE = \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{m=1}^{M}\alpha_m e_m(n)\right)^2, \qquad (6\text{-}15)$$

for a sequence of length N, with M frequency bands. The quantity $e_m(n)$ is the output of the mth bandpass filter. The filtering can be implemented in any number of convenient ways. Similar to equations (6-4) and (6-6), the peak visual mean square error is defined as

$$VPMSE = \frac{VMSE}{R^2}, \tag{6-16}$$

where R is the range of the original input sequence. The equivalent peak signal-to-noise ratio is defined as

$$VPSNR = -10\log_{10}(VPMSE). \tag{6-17}$$

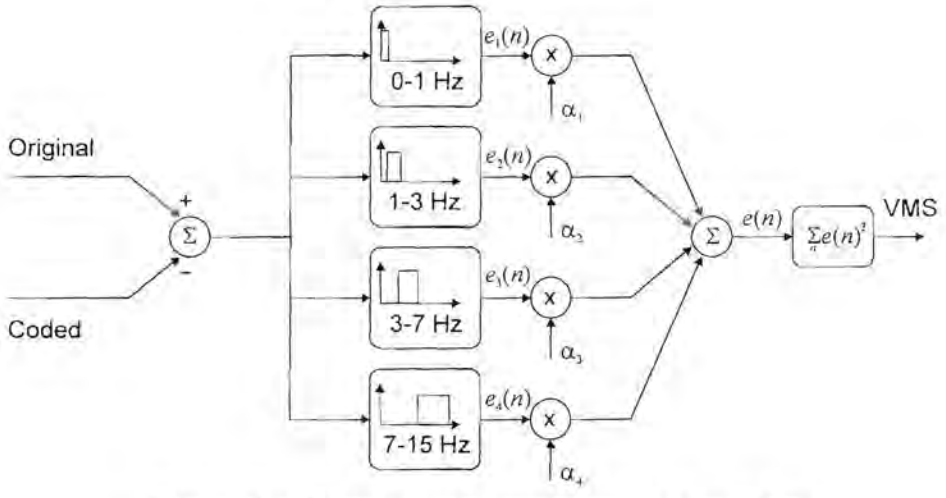


Figure 6-1: Visual mean square error algorithm.

Figure 6-2 shows a comparison between the normal MS measurement and the visual MS measurement (using the peak signal-to-noise ratio variation). The rate axis indicates a dimensionless quantity chosen for convenience. We simulate noisy amplitude errors by quantizing a signal to various levels, and phase errors by shifting a signal in time by various amounts. The amplitude errors are visually quite obvious, while the phase errors are indistinguishable without reference to the original sequence. Although this is an oversimplification of errors encountered from real compression methods, it gives an indication of what to expect from best and worst case scenarios. We use the filter banks as shown in figure 6-1, i.e. the error signal is divided into four consecutive frequency bands, with bandwidth increments by a power of two starting at one. The coefficients were heuristically chosen as $\alpha_i = \{0.25, 0.5, 1, 2.25\}$, i.e. low frequency and mean errors are subdued while high frequency errors are emphasized. In chapter 5 it was shown that the

original signal contains very little or no high frequency components. It is therefore in order to set α_4 to quite a high value, since errors in this band can originate only from the compression method. The coefficients defined above clearly forms a high-pass filter and equation (6-15) could have been implemented as such. However, we have found it more intuitive to work with a number of discrete frequency bands, each representing a certain type of visual artifact. For example, the lowest frequency band contains the general gist of the motion, while the middle frequency bands add *emotion* to the movement. High frequency bands contain jerky behaviour, which is often a result from quantization errors.

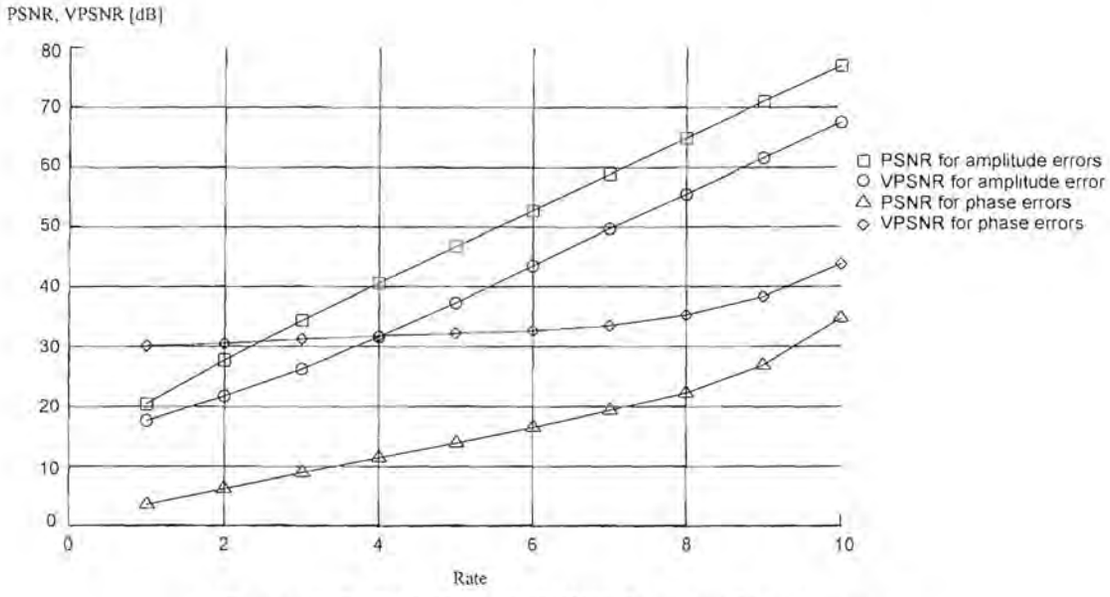


Figure 6-2: PSNR vs. VPSNR for simulated errors.

It is clear from figure 6-2 that the VMSE measure consistently indicates a lower SNR compared to the MSE measure for high frequency amplitude errors. Severe quantization results in long constant values with occasional high frequency jumps to adjacent levels. In this case, it can be seen that the VMSE starts to favour the low frequency errors introduced by these constant values. As is to be expected, at high quantization levels the error diminishes (i.e. the coding becomes lossless), and the two measures converge (not shown). In the case of phase errors, the MSE measure starts failing even for moderate errors. In this case the VMSE in figure 6-2 shows a clear advantage, which is consistent with the visual appearance of the errors.

6.3 Summary

This chapter presented a number of error measurement techniques. A distinction was made between purely quantitative methods such as the naive mean square error (MSE) measure, and qualitatively motivated methods such as the newly proposed *visual* mean square error (VMSE). Quantitative methods such as the MSE and its variants are easy to implement, are mathematically tractable and are suitable for direct implementation in a wide variety of compression algorithms. However, these methods clearly failed to distinguish acceptable error artifacts from annoying visual errors such as severe quantization noise. In order to accommodate visual errors the VMSE was introduced, which is similar in concept to the noise-shaping error measures used in speech coding.

Chapter 7 Waveform coding

7.1 Introduction

Waveform coding implies algorithms and methods that focus on single variables, such as body position and joint angles. No knowledge of the actual action that the figure is performing (such as walking, waving etc.) is assumed, and the exact source of the motion is also not under consideration, only that it is valid human motion. Whether the motion is captured in real-time, or generated by synthetic animation techniques, is of no concern. It is assumed that all of the body parameters can be decomposed into single DOF values that are independent of each other. An exception to this is the spatial vector quantization method presented at the end of the chapter, where it is assumed that there is a correlation between the variables.

In general a distinction can be made between coding (or compression) in the temporal domain and coding in the spatial domain. These two domains can be seen as orthogonal[1] to each other, and it is often advantageous to *combine* methods from each domain to get maximum compression. Temporal coding techniques take advantage of the temporal correlation of a single variable, while spatial techniques take advantage of spatial correlation between several variables.

Another distinction that can be made is the concept of *uniform* vs. *non-uniform* sampling. Traditionally we have become accustomed to sampling, frame or simulation updates that occur at well specified, regular intervals. However, there are many random processes in nature that need not be discretized in such a way, of which human motion is probably one. As has been reported by [45] for head orientations, human movement remains relatively

[1] Orthogonal is used not in a strictly mathematical sense.

static except for occasional bursty moments. The speed and acceleration of the movement are non-zero only during these moments of erratic actions. It is therefore natural to use a lower sample rate during slow movements and to increase the rate proportionally to faster movements. The solution of exactly *how* to do this is not very obvious. One such example is the dead reckoning algorithm, which is discussed a little later in this chapter. In the next chapter on model based coding, the use of non-uniform sampling will be discussed more extensively.

7.1.1 Compression

Compression is defined as the procedure that takes a stream of input samples $\{\theta(n)\}$, and transforms them to a finite string of codes or messages $\{c(n)\}$ that is a compressed version of the input stream. Decompression is the procedure that takes the string $\{c(n)\}$ and converts it to an equivalent output stream $\{\hat{\theta}(n)\}$. If the output stream is an exact or very similar duplicate of the input stream, the compression scheme is lossless. Lossy compression schemes introduce a controlled amount of distortion in the output stream in exchange for better compression. All of the compression methods discussed in this chapter are of a lossy nature (except statistical coding, but it is never used on its own).

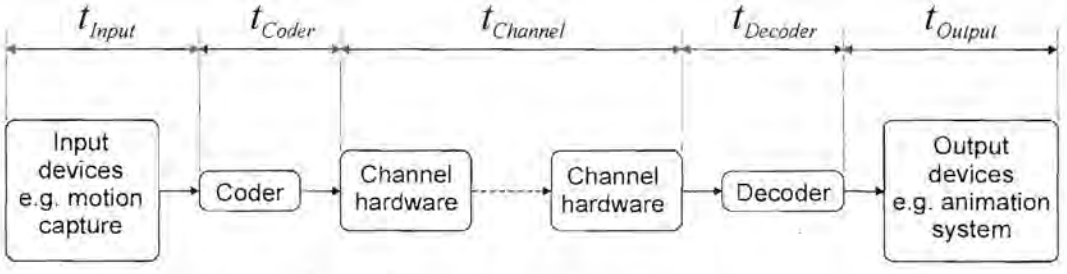


Figure 7-1: Coding/decoding delay.

7.1.2 Delay

Figure 7-1 shows a generic layout of a human motion coding/decoding system. The total coding delay is the lapsed time from the execution of an action until reconstruction at the receiving end, and is given by

$$t_T = t_{Input} + t_{Coder} + t_{Channel} + t_{Decoder} + t_{Output}.$$

Of these, we are not concerned with t_{Input}, $t_{Channel}$ and t_{Output}, which are the delays for the input devices, communications channel and output devices respectively. The time t_{Coder} is the time the coding or compression step takes from input to output, and similarly is $t_{Decoder}$ the time the decoding or decompression step takes. The delay $t_{Channel}$ is often quite severe, but there is usually not much that can be done about that. The values of t_{Coder} and $t_{Decoder}$ should be kept as low as possible – every little bit helps. Unfortunately, we are committed to a discrete sampling system at the input stage. The more coder delay we allow, the more samples we have to work with and the better information estimation we can get. When comparing compression results with the original motion, it is often convenient to compensate for the coding delay in order to use convenient error measures such as the MSE. However, it is still important to properly define the effects and tolerability of delay, especially when different systems are compared. Some systems can be used for off-line storage purposes, while others are more suitable for real-time interactive applications. In this thesis, we are more interested in the latter, hence more attention will be given to such systems.

The rest of this chapter discusses various methods for human motion compression. We start with the definition of quantization and statistical coding. Both of these methods are not really used on their own, but are "building blocks" for other compression algorithms. We then look at the class of predictive and adaptive predictive coders. This is followed by a DCT coding method as an example of a frequency domain algorithm. Vector quantization is difficult to classify as a waveform coding technique, since it can be used temporally or spatially or both, but is presented at the end of this chapter anyway. Typical results are presented with each method in the form of a representative DOF for each test sequence, as well as an overall rate-distortion graph. The following DOFs were arbitrarily chosen:

- The head angle $\theta_{3,0}$ for the conversational sequence,
- The left upper arm elevation angle $\theta_{5,0}$ for the wave sequence,

- The left elbow hinge angle $\theta_{6,0}$ for the dance sequence,
- The right hand index finger flex angle $\theta_{27,1}$ for the gesture sequence.

In each case, the rate-distortion graph shows the average PSNR (defined in chapter 6) for the whole body, except for the gesture sequence, which shows *only* the results for the right hand. The effective bit-rate is therefore considerably lower. The PSNR is used instead of the VPSNR measure for waveform coding techniques in order to compare the results with related work done by others [36]. In the following chapter on model based coding, it will be seen that the PSNR fails to give a good error measure and the VPSNR will be used instead. As a general rule-of-thumb, we have found that a PSNR of roughly 20–30 dB is visually acceptable. A PSNR of less than 10 dB is considered completely unacceptable, and a PSNR of more than 40 dB is considered almost lossless. In chapter 5, the undistorted bit-rate requirement for the whole body was found to be roughly 15000 bits/second. A compression method is regarded as useful when it can reduce the information by at least a factor two while still maintaining an acceptable error level. Any method that exceeds 8000 bits/second is therefore regarded as not worth the effort. The undistorted bit-rate for the right hand alone is roughly 2500 bits/second, and rates of less than 1250 bits/second are regarded as useful.

The results shown in this chapter are but a very small subset of the complete human due to space limitations. Representing a DOF graphically as a time varying signal is also not very intuitive, but that is the best that can be done on paper. On occasion a rendered sequence of the human figure is shown, but the results are best viewed using the video clips provided on CD-ROM with this document. Appendix III describes the contents of the accompanying CD-ROM, as well as the parameters used for each coding algorithm.

7.2 Quantization

Quantization is the mapping of a variable θ to an approximated variable $\hat{\theta}$, $\hat{\theta} = Q(\theta)$, where Q is some sort of quantization function. It can be described as the process of comparing a real value $\theta \in \mathbf{R}$ to a set of decision levels d_i and a set of reconstruction

levels r_i, where i is a *finite* integer. The problem entails the specification of a set of decision levels and reconstruction levels such that if

$$d_i \leq \theta < d_{i+1}, \tag{7-1}$$

the input variable is quantized to the reconstruction value r_i. The decision and reconstruction levels are chosen to minimize some error measure between θ and $\hat{\theta}$. An example of a mathematically tractable measure is the mean-square error, and is often used. If θ is seen as a random variable, then for N quantization levels the mean-square error is

$$\varepsilon = E\{(\theta - \hat{\theta})^2\} = \sum_{i=0}^{N-1} \int_{d_i}^{d_{i+1}} (\theta - r_i)^2 p(\theta) d\theta, \tag{7-2}$$

where $p(\theta)$ is the probability density function (PDF) of θ. It can be shown that the optimum placement of r_i can be found by minimizing ε with respect to r_i, and is given by

$$r_i = \frac{d_i + d_{i+1}}{2}, \tag{7-3}$$

which is the midpoint between each pair of decision levels. Finding the optimum choice of decision levels d_i involves the minimization of ε with respect to d_i. This is rather involved and requires knowledge of the probability density function $p(\theta)$. Max [48] developed a solution for optimum levels of a Gaussian distribution, and it can be extended to include uniform, Laplacian and Rayleigh densities. Calculation of the probability density function of human motion is virtually impossible due to the wide variation in human physiology and human motion. We will look at the more general case of uniform quantization and non-uniform quantization, and the minimum parameters that define each.

7.2.1 Uniform quantization

Uniform quantization is applicable to variables with a uniform PDF. This is mostly the case if quantization is to be directly applied to joint angles (see chapter 4). The parameters that define a uniform quantizer are the lower and upper limits, denoted by θ_L and θ_U respectively, and the number of quantization steps N. This means that the input variable θ must be restricted to $\theta_L \leq \theta \leq \theta_U$. For practical purposes N should also be restricted to a power of two, since we do not want to deal with split bits in an output bit stream. In this case N will be an even number, and the quantizer can be designed to be symmetric or asymmetric in the case of a bipolar system. If quantization is to be applied directly to joint angles, the joint limits define the lower and upper limits as well as the symmetry. It is a good idea to have separate quantizers for variables with radically different limits. If some other quantity is to be quantized, the defining parameters should be known or calculated. The generalized equation for decision level i is given by

$$d_i = \theta_L + \frac{i(\theta_U - \theta_L)}{N}, \tag{7-4}$$

and the reconstruction is given by equation (7-3). The term

$$\frac{\theta_U - \theta_L}{N}$$

in equation (7-4) is the *step size* of the quantizer and is often denoted by Δ.

7.2.2 Non-uniform quantization

Non-uniform quantization will be applied to variables with non-uniform PDFs. The spacing of decision levels is narrow in large amplitude regions of the PDF and widens in low amplitude portions of the PDF. Other than that not much can be said about the exact mathematical expression for the decision levels. There are a number of non-linear functions that can be used to generate an appropriate quantizer. Popular examples are the

A-law and μ-law quantizers used in speech coding. We have found that the bipolar μ-law quantizer gives good results, and can easily be adjusted to match a variety of non-uniform PDFs, such as the Laplace density that is encountered in predictive or difference coding. The basic parameters to specify such a non-linear quantizer are the value of μ (a "measure" of the non-linearity), the maximum bipolar limits θ_{MAX} and the number of steps N. The generalized equation for the positive half (i.e. $\theta > 0$) ith decision level is given by

$$d_i = \frac{\theta_{MAX} \log\left(1 + \mu \frac{i}{(N/2)}\right)}{\log(1+\mu)}, \tag{7-5}$$

and the reconstruction is given by equation (7-3). The negative half is a mirror of the positive half.

Finding the reconstruction level r_i given an input θ is trivial in the case of a uniform quantizer, and involves the conversion of θ to the integer space of i using simple multiply and add operations. The same cannot be said for a non-uniform quantizer, and some search algorithm has to be implemented. We use a recursive binary method, where the input level θ is compared with the midpoint of two decision levels, and a choice between the left or right branch is made.

7.2.3 Quantization noise

A useful mathematical concept is that of quantization noise, i.e. a measure that indicates the amount of distortion introduced by the quantizer. By "noise" we mean *visual* noise, and not the more traditional term of audible noise. Severe quantization noise is much more offensive in the visual sense compared to audible noise, and can render some compression algorithms completely useless. When analyzing quantization noise, it is useful to represent the quantized samples as

$$\hat{\theta}(n) = \theta(n) + e(n), \tag{7-6}$$

where $e(n)$ is the quantization noise or error. To study the effects of quantization noise, and in order to solve certain mathematical equations, it is convenient to assume a simple statistical model for the quantization noise:

- The quantization noise is a stationary white noise process, i.e.

$$E[e(n)e(n+m)] = \sigma_e^2, \quad m = 0$$
$$= 0, \quad otherwise$$

- The quantization noise is uncorrelated with the input signal, i.e.

$$E[\theta(n)e(n+m)] = 0, \quad \forall m$$

- The quantization error distribution is uniform over each quantizer interval Δ.

Although these assumptions are unrealistic for some types of human motion, our experiments have shown that it is reasonable for a step size Δ that is small enough.

Quantization can be seen as a compression technique, since the quantized output usually occupies fewer bits than the original signal for a given error in representation. Figures 7-2a to 7-2d show the results for direct quantization of the representative joint angles discussed previously. Shown are the original, 8-level quantized, 64-level quantized and the error signal of the 64-level quantization. Using less than 64 levels (or 6 bits) usually results in severe visual artifacts, except for the interesting case shown in figure 7-2d, where the open/close gesture movements can be quantized quite well with very few levels. In any case, direct quantization of DOF values results in an effective compression ratio below 2:1, and such a naive method is not recommendable as a compression mechanism. Figure 7-2e depicts a number of consecutive 3D wireframe images from the dance sequence. The original is overlaid with a 16-level quantized sequence (shown in red), and the frame-to-fame difference can clearly be seen.

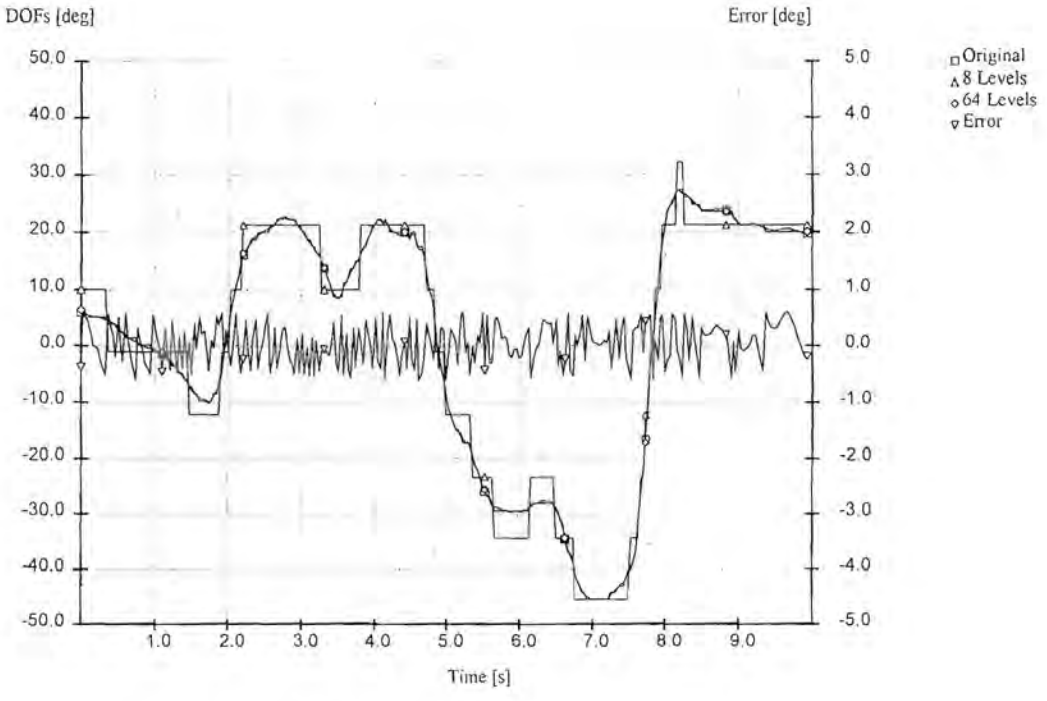


Figure 7-2a: Quantization of $\theta_{3,0}$ with 8 and 64 levels for the conversational sequence.

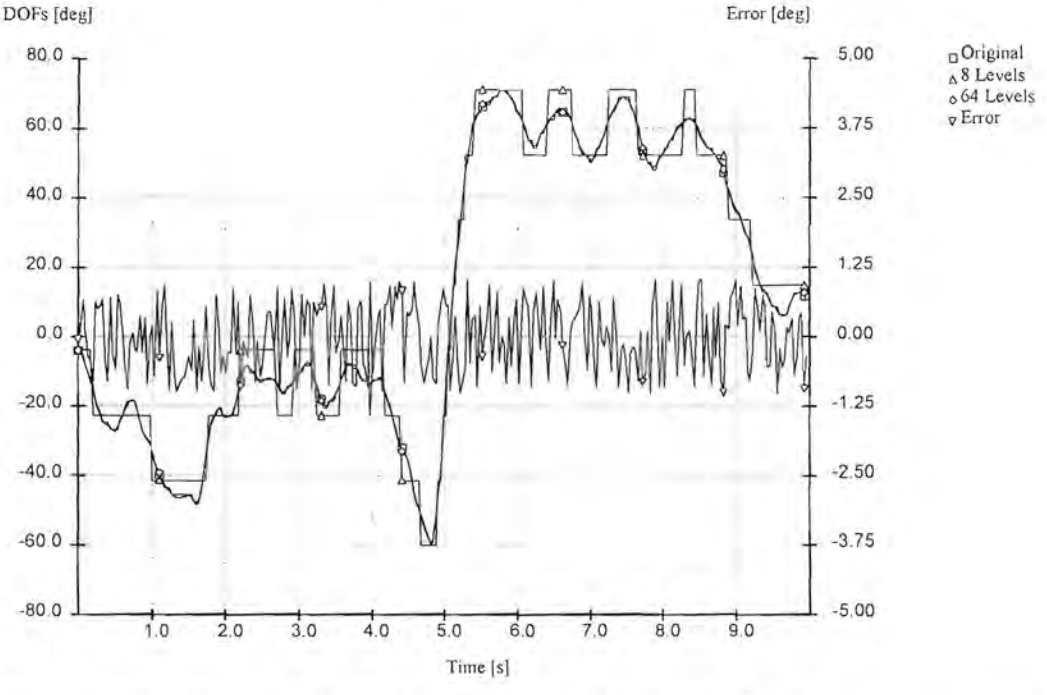


Figure 7-2b: Quantization of $\theta_{5,0}$ with 8 and 64 levels for the wave sequence.

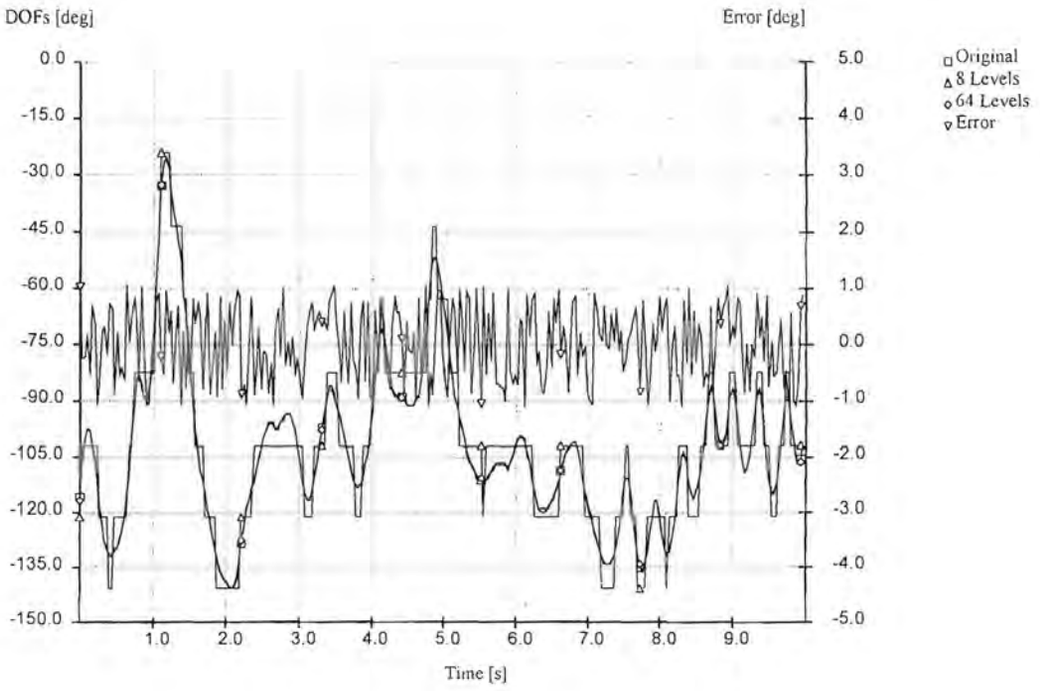


Figure 7-2c: Quantization of $\theta_{6,0}$ with 8 and 64 levels for the dance sequence.

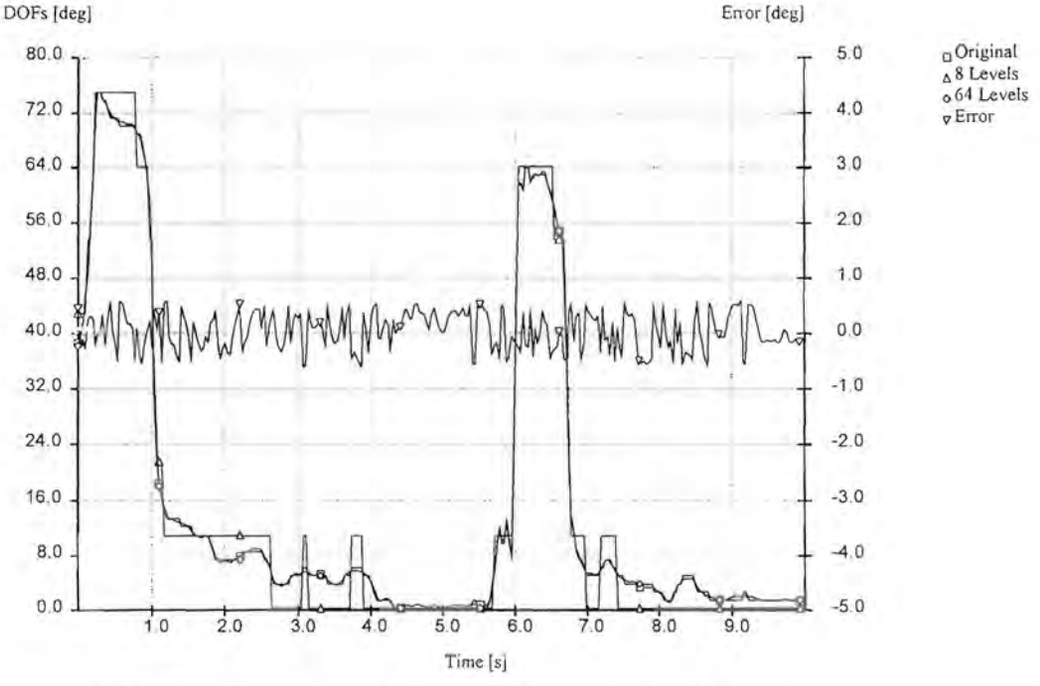


Figure 7-2d: Quantization of $\theta_{27,1}$ with 8 and 64 levels for the gesture sequence.

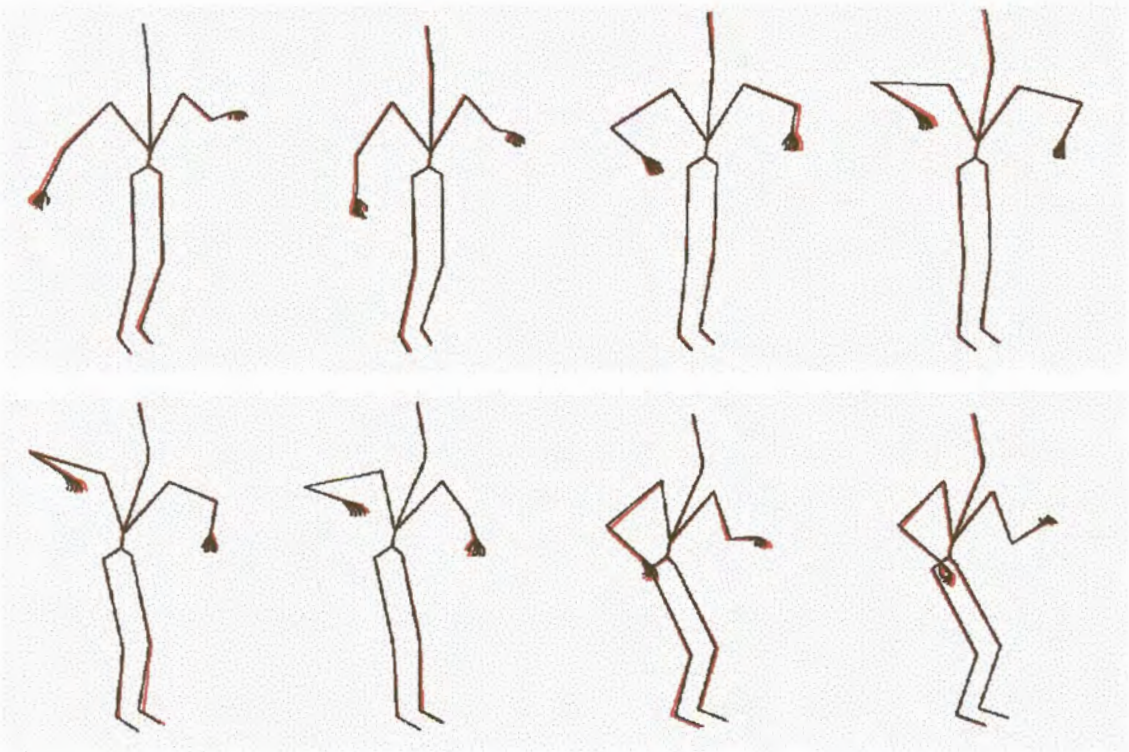


Figure 7-2e: Wireframe images from the dance sequence with 16-level quantization.

Figure 7-3 shows the PSNR against bit-rate. Note that the effective bit-rate and error for the gesture sequence is for the right hand only. The other body DOFs contained almost no information, and was not compressed.

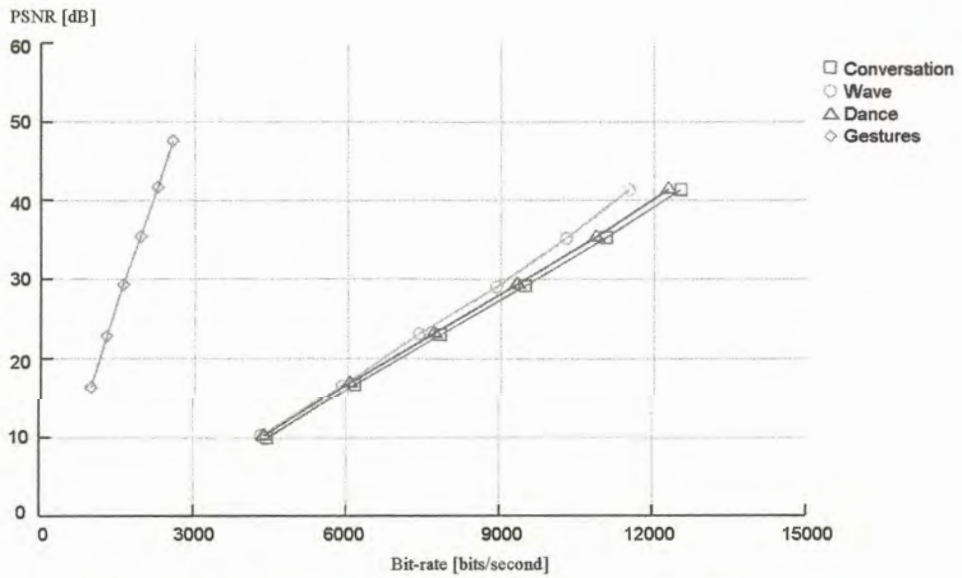


Figure 7-3: Distortion vs. bit-rate for direct quantization.

7.3 Adaptive quantization

One is often confronted with the dilemma of choosing the correct quantization step size Δ. On the one hand it should be large enough to accommodate the maximum peak-to-peak range of the input signal. On the other hand it should be small enough to minimize quantization noise. One way of alleviating this problem is to use non-linear quantization, while the other is to adapt the quantizer to some property of the input signal.

The basic idea of adaptive quantization is to let the quantizer levels and ranges vary to match the variance of the input signal, or alternatively to adjust the gain of the input signal inversely with the variance of the input signal. There are two methods of doing this. A *feed-forward* scheme estimates the matching function from the input itself. A feedback scheme estimates the matching function from the output of the quantizer (or even the whole coding system). Feed-forward systems require us to transmit the quantizer settings as well (albeit only every nth update), while the feedback system can use the received messages to derive the quantizer settings.

For simplicity, we have chosen a simple feedback algorithm where the step size $\Delta(n)$ of a uniform quantizer is modified at update n by a function of the form

$$\Delta(n) = \beta\Delta(n-1), \tag{7-7}$$

where β is a step size multiplier and is a function of the previous code $c(n-1)$. In practice, we use a table containing values of β for each code word. These values have been obtained in a heuristic fashion to accommodate a large variety of input signals. Direct adaptive quantization of DOF variables is not recommended because the joint angles are generally non-zero-mean quantities, and do not exhibit symmetric behaviour. It is difficult under such circumstances to establish a proper adaptation table for β, and the performance of the adaptive scheme approaches that of the standard quantization method discussed in the

previous section. Adaptive quantization will be used extensively in the more advanced compression techniques discussed below.

7.4 Statistical coding

Although statistical coding (sometimes referred to as entropy coding) is not a lossy compression technique in itself, it is usually inserted at the end of a lossy compression pipeline to ensure that the stream of codes have optimal statistics. As a starting point for statistical coding development, it is necessary to model, estimate or measure the probabilities of occurrence for each value to be encoded. In our case, we use statistical coding after some other coding technique, and this measurement will be done in "message space" rather than in joint angle space. Most often a quantizer is superseded by a statistical coder. In this case, suppose that the probability of a quantized value $\hat{\theta}$ (or message) to be equal to the nth reconstruction level, is given by

$$P(n) = P\{\hat{\theta} = r_n\}. \tag{7-8}$$

In the coding process, a code word of $b(n)$ bits is assigned to each quantization level, resulting in an average code length of

$$L = \sum_{n=0}^{N-1} P(n)b(n), \tag{7-9}$$

where N is the length of the code book. There are a number of techniques that can be used to produce a codebook [71,72,73]. These include arithmetic coding, Shannon-Fano coding and Huffman coding, of which the latter is the most efficient in terms of length. In this coding process [71], the two messages with the lowest probability are combined in a tree structure and their probabilities summed at the junction. The probability is then combined again in the same manner with the next lowest probability until the tree converges to a single junction. The branches of the tree are then assigned arbitrarily bit values of one or

zero. A code is formed by traversing the tree back to the message node in question and recording the path designation.

It is possible to calculate a fixed codebook beforehand, or to adaptively build the codes as the transmission progresses. In the latter case, we start with a codebook of equal length codes. For every message sent the probabilities of the codes are updated and the codebook is calculated according to the method discussed above. For a fixed codebook, the probabilities can be calculated using an appropriate test data set.

Most of the compression algorithms in the remainder of this chapter use a statistical coder as a "black box" between the coder and decoder sections. The statistical coder can never increase the average code length, and can have no adverse effects on the coding process if used correctly. However, a properly implemented compression algorithm should not rely on the use of a statistical coder to achieve high compression ratios. In fact, more than a 20-30 percent decrease in average code length is an indication that the code words from the output of the compression algorithm have a non-uniform distribution. This implies that the compression algorithm is probably poorly designed, and that further gain can be achieved with a better implementation.

7.5 Predictive coding

In chapter 5, it was shown that there is considerable correlation between adjacent samples. On average, joint angles do not change rapidly from sample to sample, therefore the difference between adjacent samples should have a lower variance than the original signal itself. Figure 7-4 depicts the general layout of a *predictive coder*. The dotted lines indicate an adaptive section, and can be ignored for now. The input to the quantizer is a difference signal

$$d(n) = \theta(n) - \tilde{\theta}(n), \tag{7-10}$$

where $\tilde{\theta}(n)$ is a *predicted* version of the input signal $\theta(n)$. If the prediction is good, the variance of $d(n)$ will be smaller than that of $\theta(n)$, and the quantizer could be adjusted to give a smaller quantization error for a fixed number of levels. Figure 7-4 also shows the layout of a corresponding decoder. The output is given by

$$\hat{\theta}'(n) = \tilde{\theta}'(n) + \hat{d}'(n), \tag{7-11}$$

where $\tilde{\theta}'(n)$ is the output of a similar predictor as in the coder. Clearly if $c'(n) = c(n)$, then $\hat{\theta}'(n) = \hat{\theta}(n)$, and the only difference between the input and output is the quantization error incurred in $d(n)$.

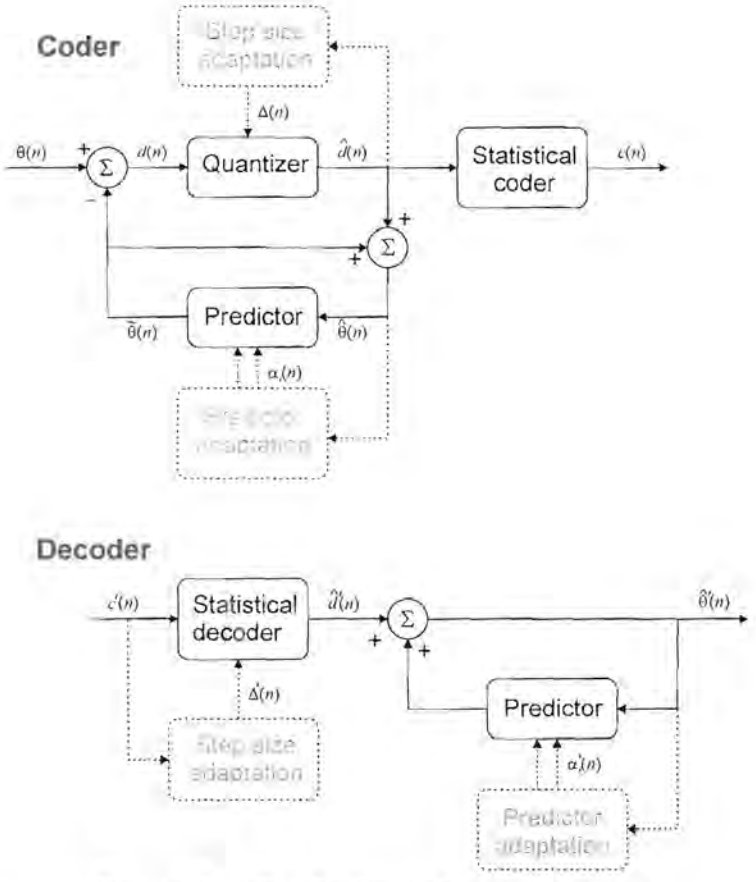


Figure 7-4: Predictive coder and decoder.

The predictor could have a variety of forms. A mathematically tractable and widely used form is a linear predictor [72,73], i.e. the output is a linear combination of past input values. The general form of the predictor can be written as

$$\tilde{\theta}(n) = \sum_{k=1}^{p} \alpha_k \hat{\theta}(n-k). \tag{7-12}$$

Since we would like to minimize the variance of $d(n)$, as denoted by σ_d^2, it would be appropriate to differentiate σ_d^2 with respect to each coefficient α_i:

$$\frac{\partial \sigma_d^2}{\partial \alpha_i} = 0, \quad 1 \le i \le p. \tag{7-13}$$

Finding an exact solution for equation (7-13) is quite involved and requires extensive knowledge about the input signal. In [73] a number of approximations are discussed. It has been found that not much is gained with high order predictors, and that it is best to keep $p < 4$. For comparison purposes, we use a first order predictor in this section, and a higher order adaptive predictor in the next section. In the case of $p = 1$, it can be shown [73] that

$$\alpha_1 = \frac{R_\theta(1)}{R_\theta(0)}, \tag{7-14}$$

where R_θ is the autocorrelation function of θ.

Figures 7-5a to 7-5d show the results for predictive coding of the representative joint angles. Indicated are the original, 8-level quantized, 64-level quantized and the error signal of the 8-level quantization. Even with as little as 4 quantization levels (not shown), the coder still provides acceptable results. It can be seen that in most cases the 8-level error signal is similar to that of 64-level direct quantization, which is a saving of almost 3 bits. The use of 64-level (or 6 bit) quantization results in motion that is almost indiscernible from the original.

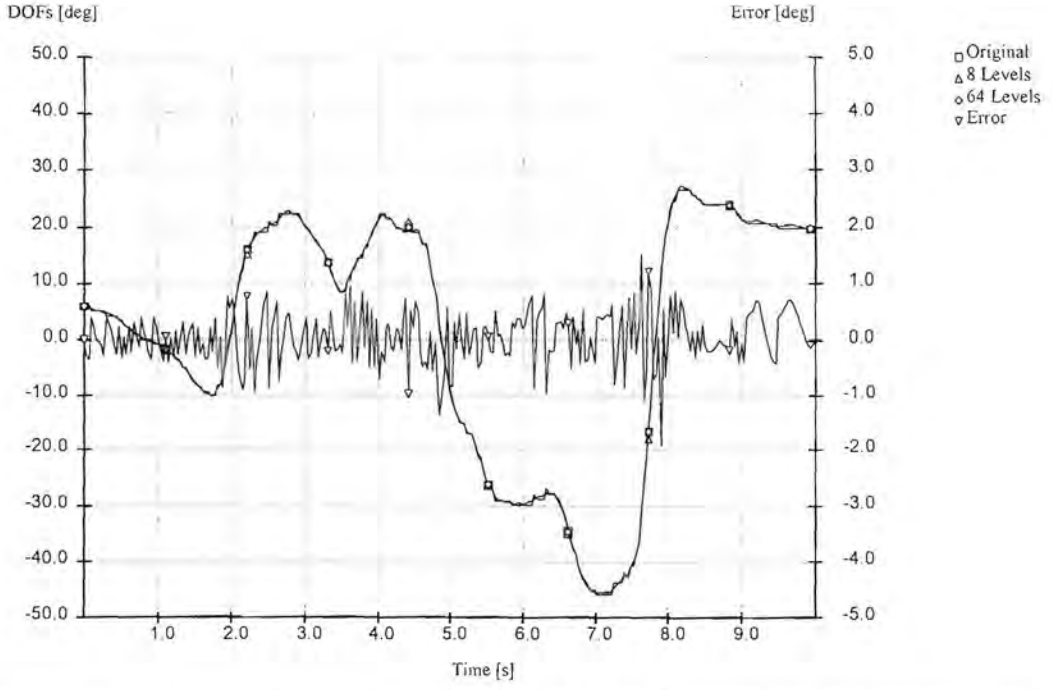


Figure 7-5a: Predictive coding of $\theta_{3,0}$ with 8 and 64 levels for the conversational sequence.

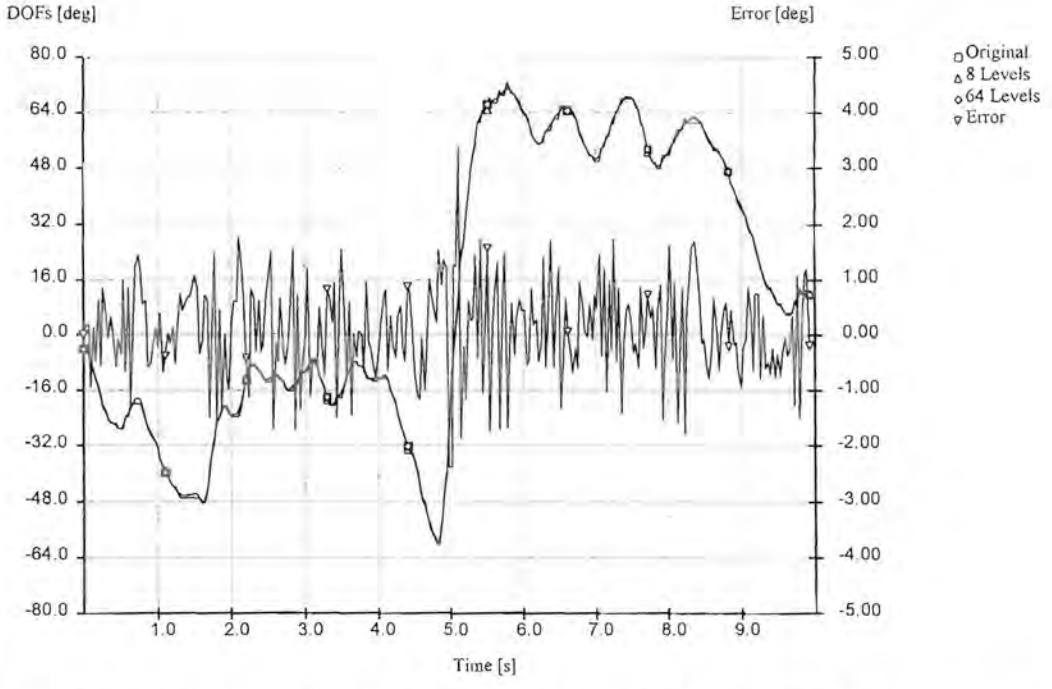


Figure 7-5b: Predictive coding of $\theta_{5,0}$ with 8 and 64 levels for the wave sequence.

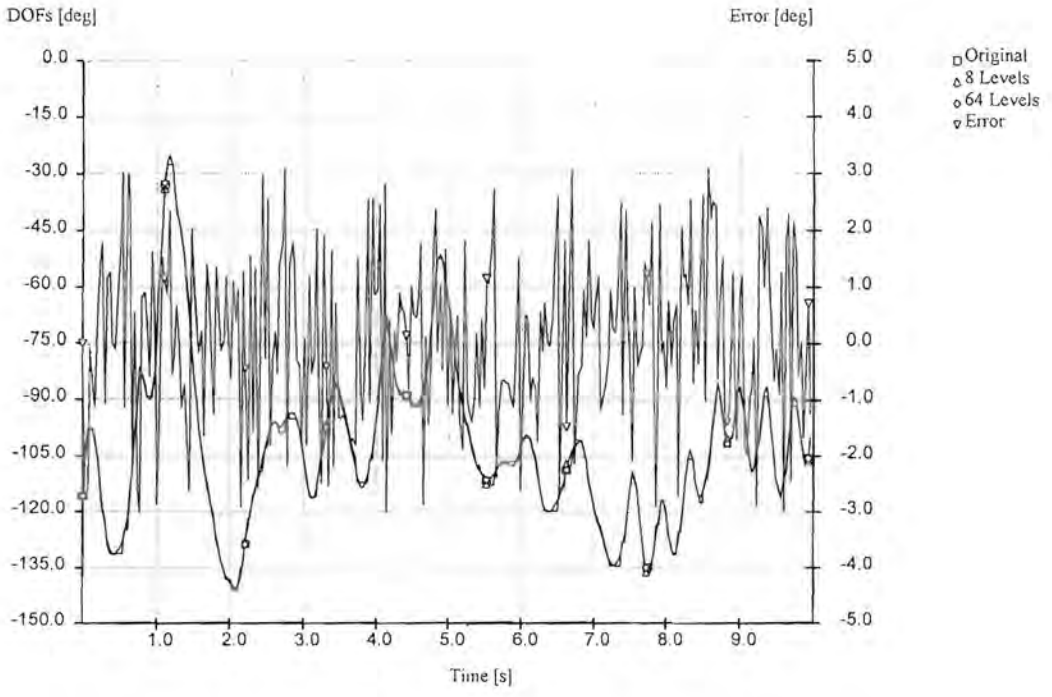


Figure 7-5c: Predictive coding of $\theta_{6,0}$ with 8 and 64 levels for the dance sequence.

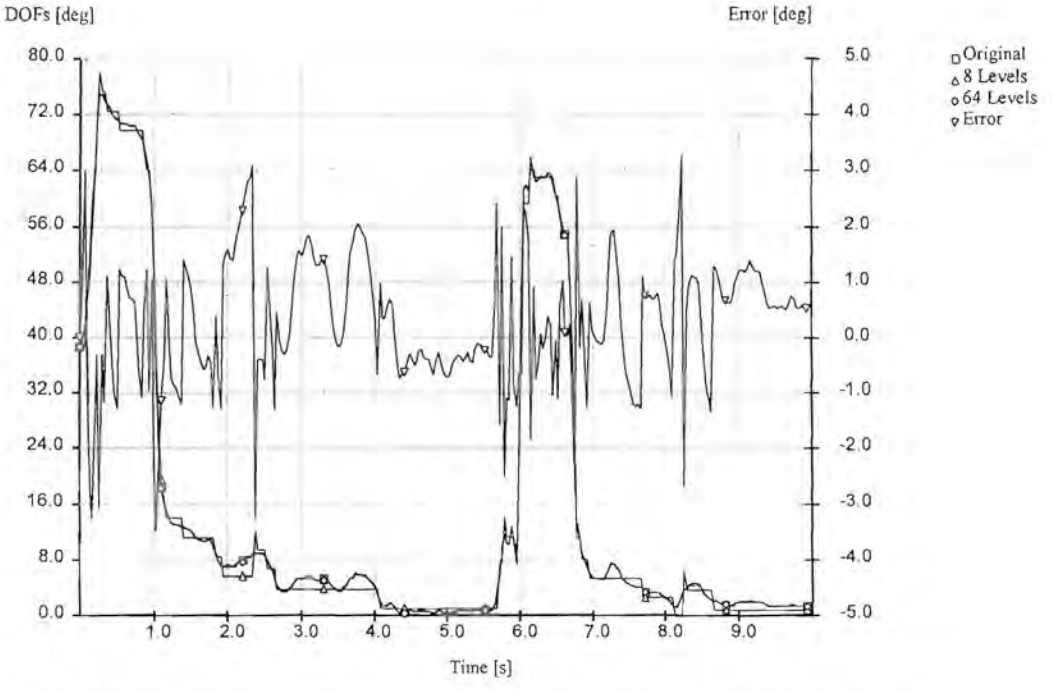


Figure 7-5d: Predictive coding of $\theta_{27,1}$ with 8 and 64 levels for the gesture sequence.

Figure 7-6 shows the PSNR against bit-rate for predictive coding. Note that the effective bit-rate and error for the gesture sequence are for the right hand only. The saving over

direct quantization can clearly be seen (chapter 9 contains a comparison between various coding methods for the same sequence). The graphs are also more spread out relative to each other, which indicates that the coding method is sensitive to temporal variation, which direct quantization is not. It can be seen that predictive coding, for all practical purposes, becomes lossless for more than 257-level quantization. However, the practical range for this method lies between 3000 and 6000 bits/second.

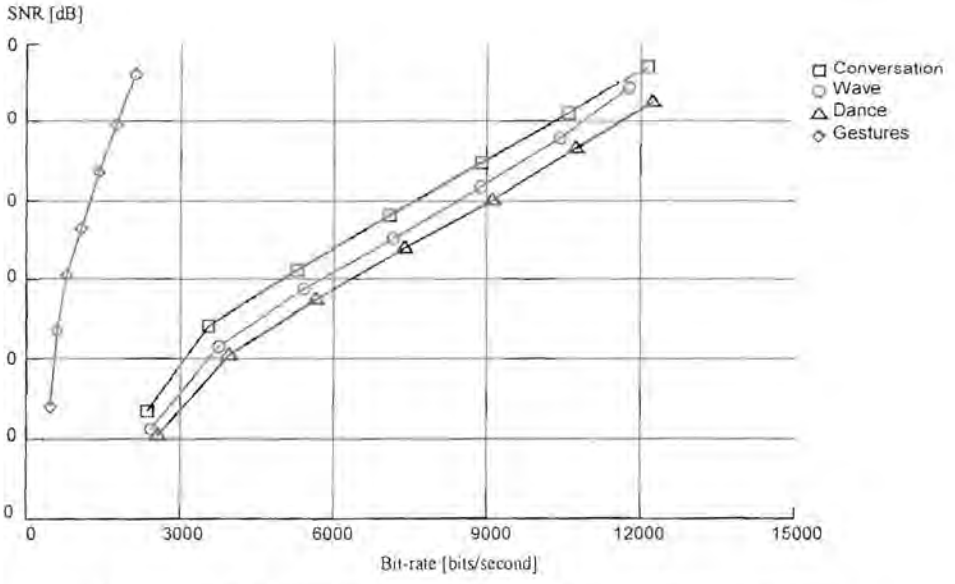


Figure 7-6: Distortion vs. bit-rate for predictive coding.

7.6 Adaptive predictive coding

The structure of an adaptive predictive coder is very similar to that of a predictive coder, and it shares the same layout as shown in figure 7-4. However, in the case of an adaptive predictive coder it is important to note that either the quantizer, or the predictor, or both are adapted to give an improved output $\hat{\theta}(n)$. Note that the scheme shown in figure 7-4 is a feedback system, i.e. the adaptation parameters are calculated from the decoded signal. The concept of adaptive quantization was already covered in section 7.3. It is natural to consider adapting both the quantizer and predictor to match the temporal variations in the human motion signal. Adaptive prediction implies that the prediction coefficients $\{\alpha_i\}$ are

now functions of the current sample or update, i.e. they should be written as $\{\alpha_l(n)\}$. The predictor function of equation (7-12) now becomes

$$\tilde{\theta}(n) = \sum_{k=1}^{p} \alpha_k(n)\hat{\theta}(n-k). \tag{7-15}$$

Similar to speech signals, the prediction coefficients could be chosen to minimize the average mean-squared prediction error over short time intervals. Again there are numerous approaches and methods to solve this problem, such as the autocorrelation and covariance methods [73]. By evaluating equation (7-13) for short motion segments and omitting the effects of quantization noise, it can be shown that the autocorrelation method provides a set of equations that can be written in matrix form

$$\mathbf{A}\boldsymbol{\alpha} = \mathbf{B}, \tag{7-16}$$

where $\mathbf{A}$ is a $p \times p$ matrix of autocorrelation values

$$\mathbf{A} = \begin{bmatrix} R_\theta(0) & R_\theta(1) & R_\theta(2) & \dots & R_\theta(p-1) \\ R_\theta(1) & R_\theta(0) & R_\theta(0) & \dots & R_\theta(p-2) \\ R_\theta(2) & R_\theta(0) & R_\theta(0) & \dots & R_\theta(p-3) \\ \dots & \dots & \dots & \dots & \dots \\ R_\theta(p-1) & R_\theta(p-2) & R_\theta(p-3) & \dots & R_\theta(0) \end{bmatrix}, \tag{7-17}$$

and $\mathbf{B}$ is a $p \times 1$ vector of autocorrelation values

$$\mathbf{B} = \begin{bmatrix} R_\theta(1) \\ R_\theta(2) \\ R_\theta(3) \\ \dots \\ R_\theta(p) \end{bmatrix}. \tag{7-18}$$

The matrix **A** has a Toeplitz structure, and the solution $\alpha = \mathbf{A}^{-1}\mathbf{B}$ can be computed using a variety of numerical methods. In equations (7-17) and (7-18), the term $R_\theta(n)$ refers to the *short-term* autocorrelation function of the sequence $\{\theta(n)\}$, which can be calculated from a windowed segment of motion. The choice of window and window length depends on the characteristics of the DOF in question and the type of motion. We use window lengths of between 0.5 seconds (15 samples at 30 Hz) and 4 seconds (120 samples at 30 Hz).

We have found that not much is gained for values of p greater than 2, and the following results were obtained using a second order adaptive predictor, together with an adaptive quantizer. Figures 7-7a to 7-7d show the results for adaptive predictive coding of the representative joint angles. Depicted are the original, 8-level quantized, 64-level quantized and the error signal of the 8-level quantization. It can be seen that the error for an 8-level quantizer is worse than that of simple first order non-adaptive prediction, a fact that is also evident on the rate-distortion graph of figure 7-8. The sharp increase in error at very low bit-rates is due to the omission of quantization noise effects in the calculation of the adaptation coefficients (the difference signal is severely quantized to 4 levels at these low rates). At low rates the system also exhibits oscillatory quantization behaviour, which can clearly be seen in figure 7-7d. For higher bit-rates, the adaptive coding scheme clearly outperforms the non-adaptive techniques.

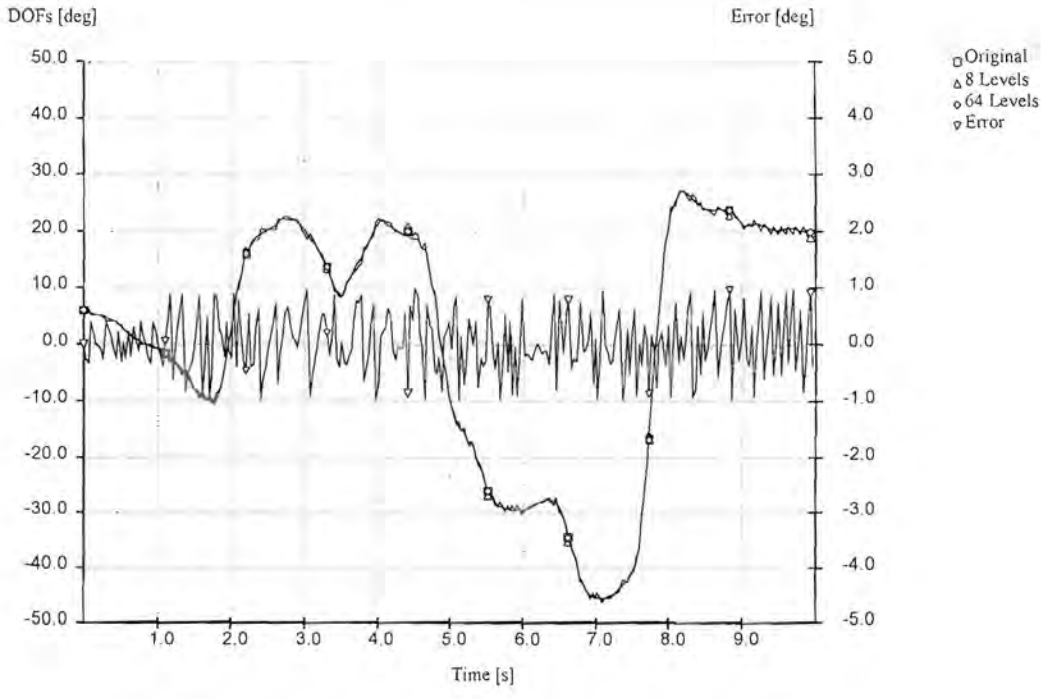


Figure 7-7a: Adaptive predictive coding of $\theta_{3,0}$ with 8 and 64 levels for the conversational sequence.

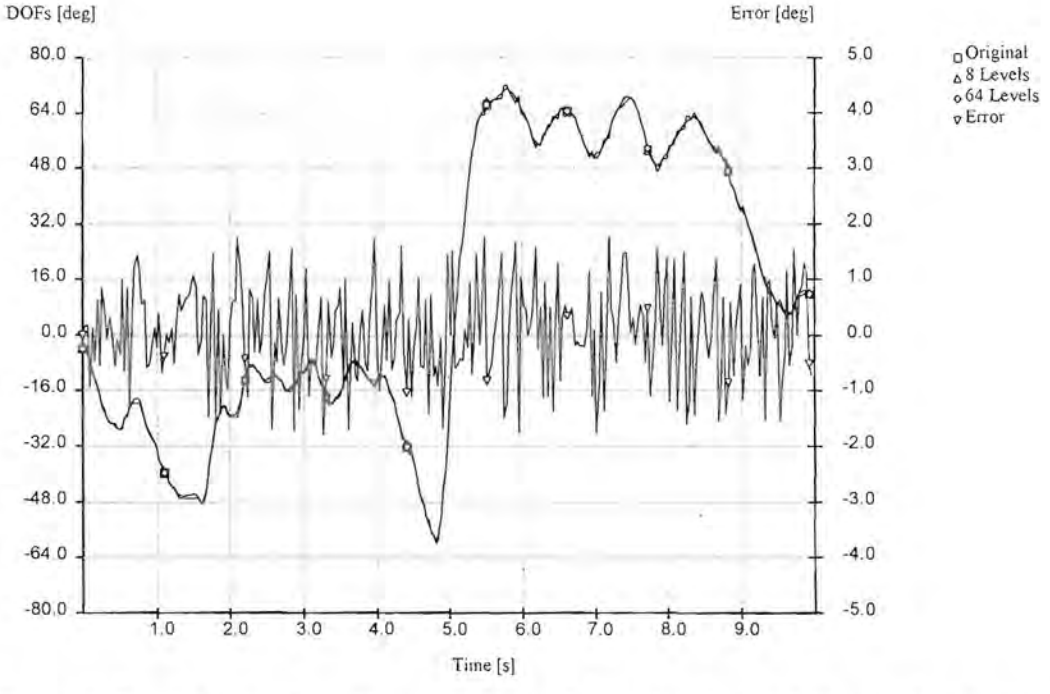


Figure 7-7b: Adaptive predictive coding of $\theta_{5,0}$ with 8 and 64 levels for the wave sequence.

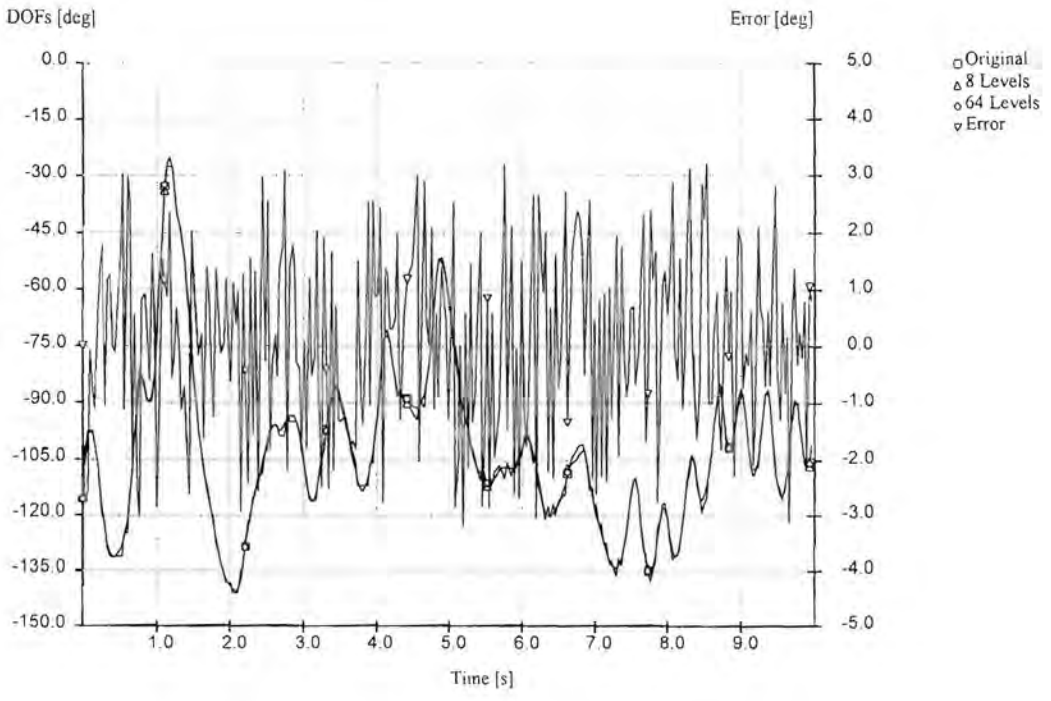


Figure 7-7c: Adaptive predictive coding of $\theta_{6,0}$ with 8 and 64 levels for the dance sequence.

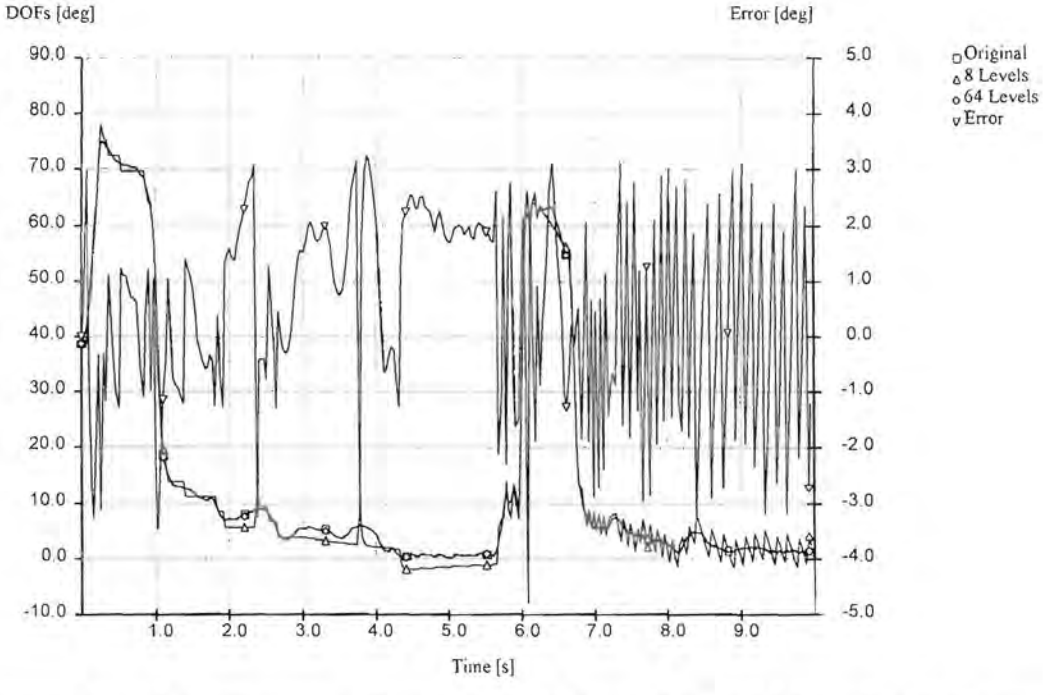


Figure 7-7d: Adaptive predictive coding of $\theta_{3,0}$ with 8 and 64 levels for the gesture sequence.

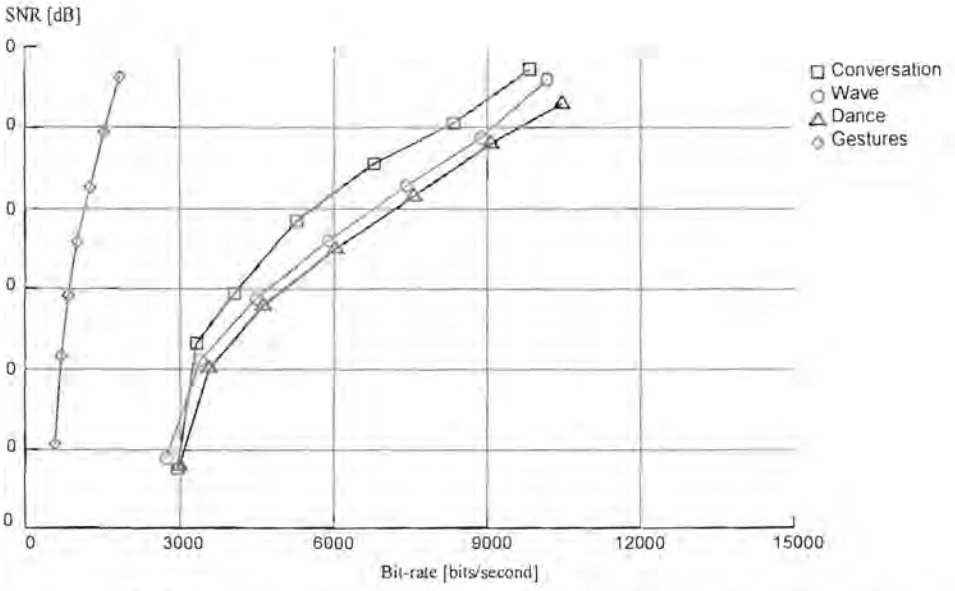


Figure 7-8: Distortion vs. bit-rate for adaptive predictive coding.

7.7 Dead reckoning

Dead reckoning (DR) refers to a traditional method of calculating the position of an object given its velocity and/or acceleration. This concept can be used to decrease the number of update messages that need to be sent in a networked virtual environment, since the position of the object can be extrapolated. Examples of such usage can be found in the Distributed Interactive Simulation (DIS) protocol [49] and the NPSNET system [30,31].

More formally, assume that there is a known function $\theta(t)$ of time, and that the first and second derivatives $\dot{\theta}(t)$ and $\ddot{\theta}(t)$ are known, or can be calculated. For convenience, let $\omega(t) = \dot{\theta}(t)$ and $\alpha(t) = \dot{\omega}(t) = \ddot{\theta}(t)$. If $\alpha(t)$ is constant over a time period $0 \le t < T$ with a value of α, we can write

$$\begin{aligned} \omega(t) &= \omega_0 + \alpha t, \\ \theta(t) &= \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2, \quad t < T \end{aligned} \tag{7-19}$$

where $\theta_0 = \theta(0)$ and $\omega_0 = \omega(0)$. Similarly, if $\omega(t)$ is constant over a period $0 \le t < T$ with a value of ω, we can write

$$\theta(t) = \theta_0 + \omega t, \quad t < T \tag{7-20}$$

where $\theta_0 = \theta(0)$. Clearly, with θ_0, ω_0, ω and/or α known, the function $\theta(t)$ can be evaluated over the period $0 \le t < T$. The dead reckoning algorithm uses this concept in the following manner:

> Two copies of the human model are maintained. One is the "real" or local human model, as obtained from the input devices or animation system. The other is a so-called *ghost* model, which is updated by the likes of equations (7-19) and (7-20). Denote a single DOF in the real model as the function $\theta(t)$ [2] and the corresponding DOF in the ghost model as $\theta'(t')$. Similarly we define the functions $\omega'(t')$ and $\alpha'(t')$. We also define some error measure $\varepsilon = e(\theta(t), \theta'(t'))$ between the local and ghost model. For simplicity, we will look only at the case of constant $\omega(t)$ (equation (7-20)) in the following discussion. It can at any time be extended to use equation (7-19) as well. At time t, we set $\theta'_0 = \theta(t)$ and obtain a value for the constant ω. The real model is locally updated by the input devices, and the ghost model is updated using equation (7-20). After some time the approximation of constant $\alpha(t)$ or $\omega(t)$ will not hold anymore, and the value of ε will exceed some threshold. At this point, we reset the dead reckoning time t' to zero, again set $\theta'_0 = \theta(t)$ and obtain a new value for ω. By adjusting the error threshold for ε, we can control the resetting frequency of the DR process.

[2] In the case of joint angles, $\theta(t)$ conveniently becomes the angle, $\omega(t)$ becomes the angular velocity and $\alpha(t)$ becomes the angular acceleration.

This concept is shown in figure 7-9. In terms of a networked environment, we only need to transmit messages containing θ_0' and ω when the error threshold is exceeded. The above reasoning assumes that the relative time reference of the local and remote process is exact or at least very close. A similar DR process is used at every receiver for each participant in the virtual world. A problem that is immediately evident from figure 7-9 is that we cannot simply set $\theta_0' = \theta(t)$ during a reset, as it will result in a discontinuity in $\theta'(t')$. There are various methods to compensate for this, but they will not be discussed here. For us, the important point is that it should not influence the frequency at which new updates are sent, nor should it influence the amount of information that needs to be sent.

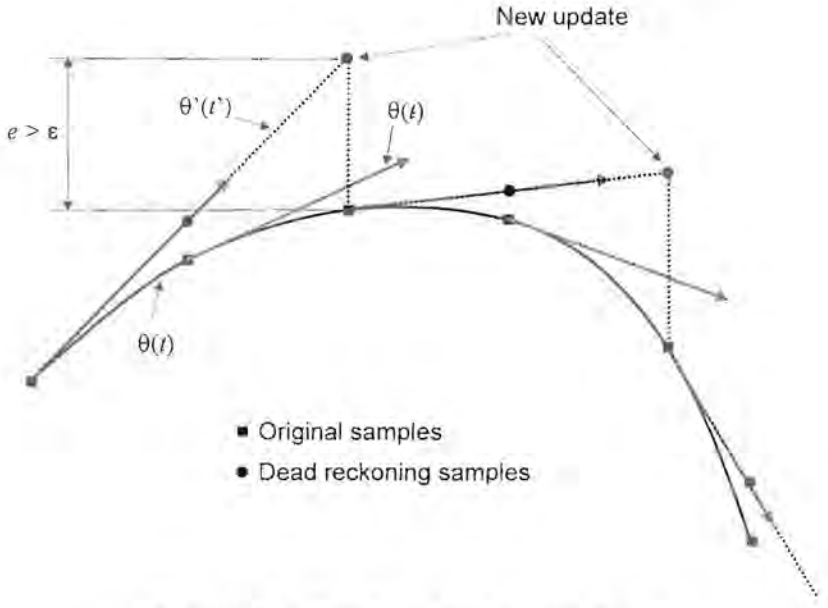


Figure 7-9: Dead reckoning operation.

A generalized schematic for a DR coder and decoder is shown in figure 7-10. This includes the use of a quantizer, and possibly a statistical coder. Note that the DR update parameters are taken *after* the quantizer to account for quantization noise. It is not clear from other DR implementations [35] whether quantization and/or statistical coding are used at all. In our opinion, at least quantization is crucial for further reduction, as this is a basic component of most compression systems. In addition, we do not transmit the absolute position for each new update, but rather the *difference* between the desired and predicted position. The difference can be quantized with fewer levels and results in a further rate reduction. There

is the need for an accurate estimation of the speed and acceleration, if it cannot be measured directly. The use of a Kalman filter has been proposed by [35] for noisy data. We have found that our motion data is relatively noise free, and that it is sufficient to estimate the quantities $\dot{\theta}(t)$ and $\ddot{\theta}(t)$ by fitting at least a second order polynomial to $\theta(t)$, using a simple least-squares solution with a 7 sample window. This method introduces a 3 sample delay, and should be kept in mind when different systems are compared to each other.

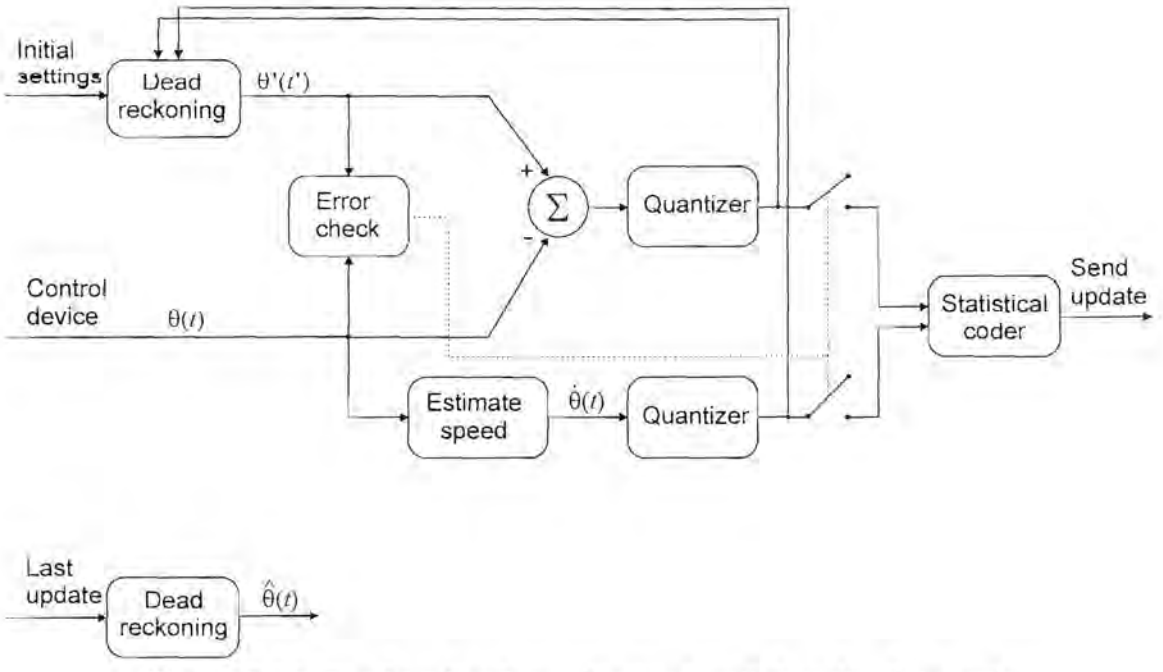


Figure 7-10: Dead reckoning coder (top) and decoder (bottom).

The error measurement can be done in a variety of ways, each of which will result in different performances of the algorithm. A number of possibilities are discussed in chapter 6. The most obvious method for independent single DOF variables is to take the distance $\varepsilon = |\theta(t) - \theta'(t')|$ as the error. However, some DOFs contribute less to visual errors, and it is often advantageous to group a number of DOFs together and to use a weighed mutual error measurement, such as joint distance.

It is possible to use a higher level DR algorithm that analyzes actions and/or motion control methods to make an update decision. Such algorithms will fall under the more

general case of non-uniform sampling methods rather than dead reckoning, and is beyond the scope of this chapter.

Figures 7-11a to 7-11d show the results for the dead reckoning compression method on the representative joint angles. All the DOFs were processed separately using a simple absolute difference as the error measurement. Figure 7-12 shows PSNR against bit-rate for the various test sequences. It can be seen that the dead reckoning algorithm does not perform very well, except possibly for the gesture sequence. In fact, it is evident from the dance sequence that the resulting bit-rate can even *exceed* the original raw bit-rate. This can happen when twice the amount of original information is sent (i.e. both angle and angular speed) too frequently. A check can be done by calculating a histogram of the frequency of updates for the whole sequence. Figure 7-13 shows such an average histogram that covers the whole of the rate-distortion range in figure 7-12. It is clear that the conversational sequence requires updates for *every* new sample for almost 70% of the time. The gesture sequence performs better, and requires new updates only 40% of the time. Still, such frequent updates result in high bit-rates, considering that the value, as well as the derivative, of a DOF is sent.

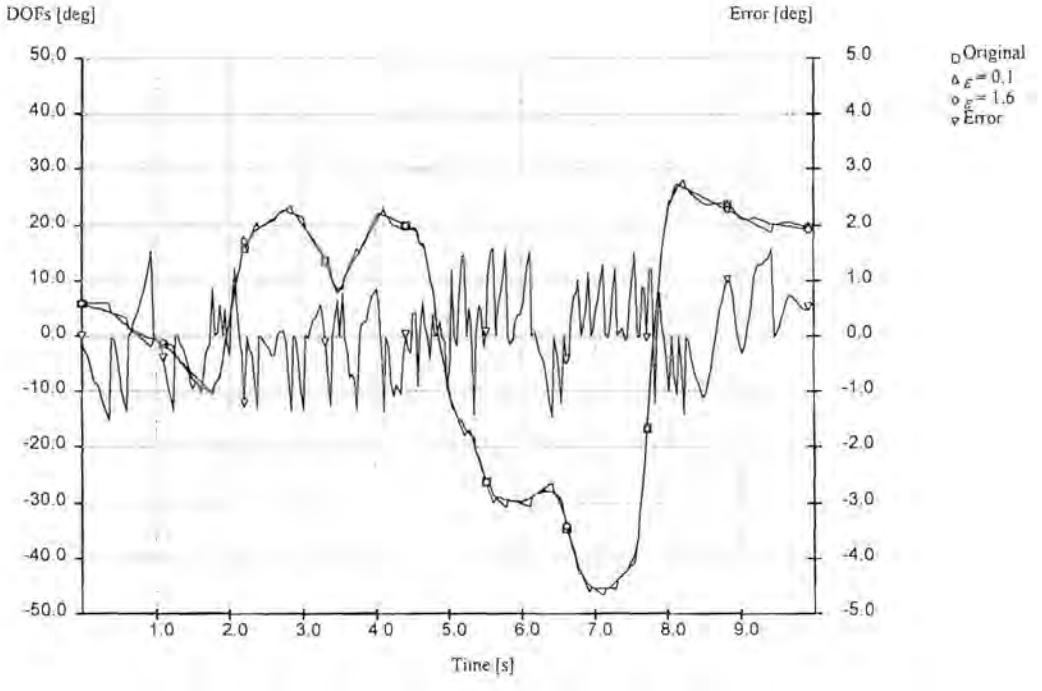


Figure 7-11a: Dead reckoning of $\theta_{3,0}$ for the conversational sequence.

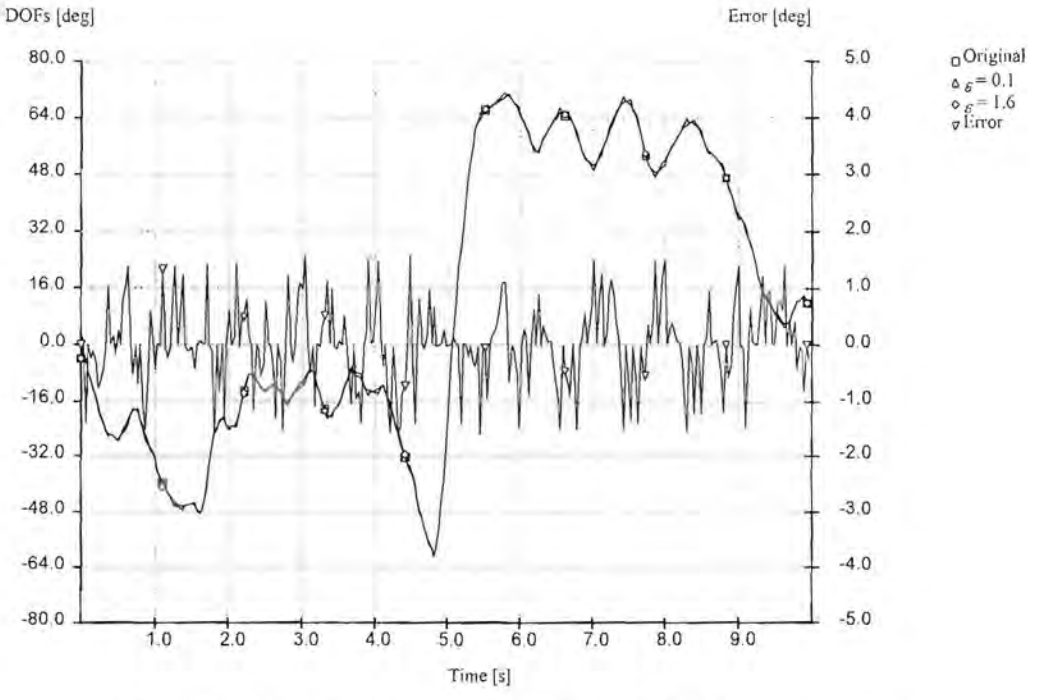


Figure 7-11b: Dead reckoning of $\theta_{5,0}$ for the wave sequence.

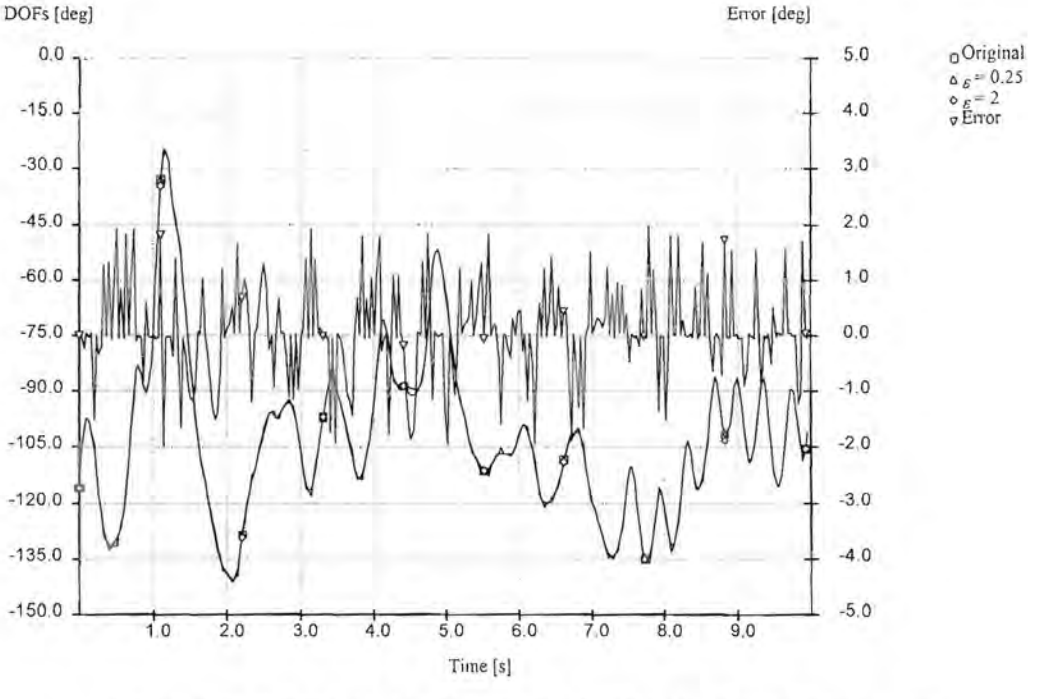


Figure 7-11c: Dead reckoning of $\theta_{6,0}$ for the dance sequence.

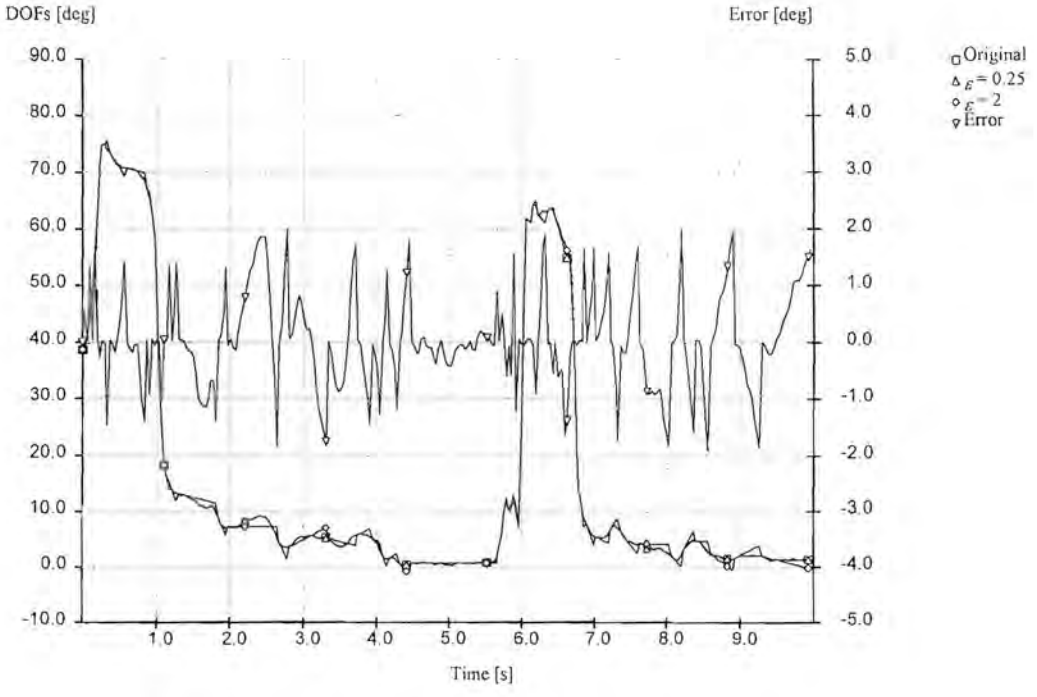


Figure 7-11d: Dead reckoning of $\theta_{27,1}$ for the gesture sequence.

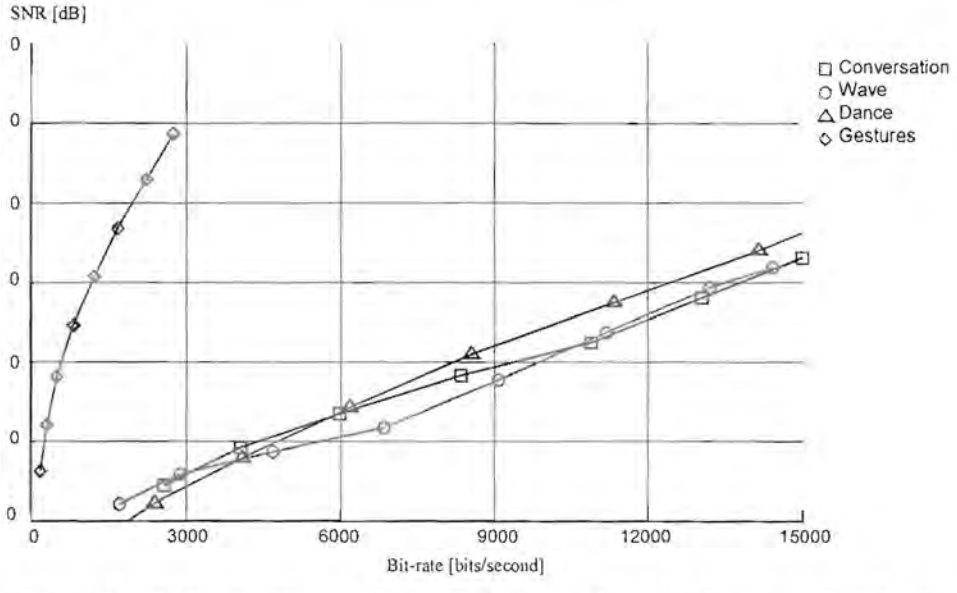


Figure 7-12: Distortion vs. bit-rate for the dead reckoning algorithm.

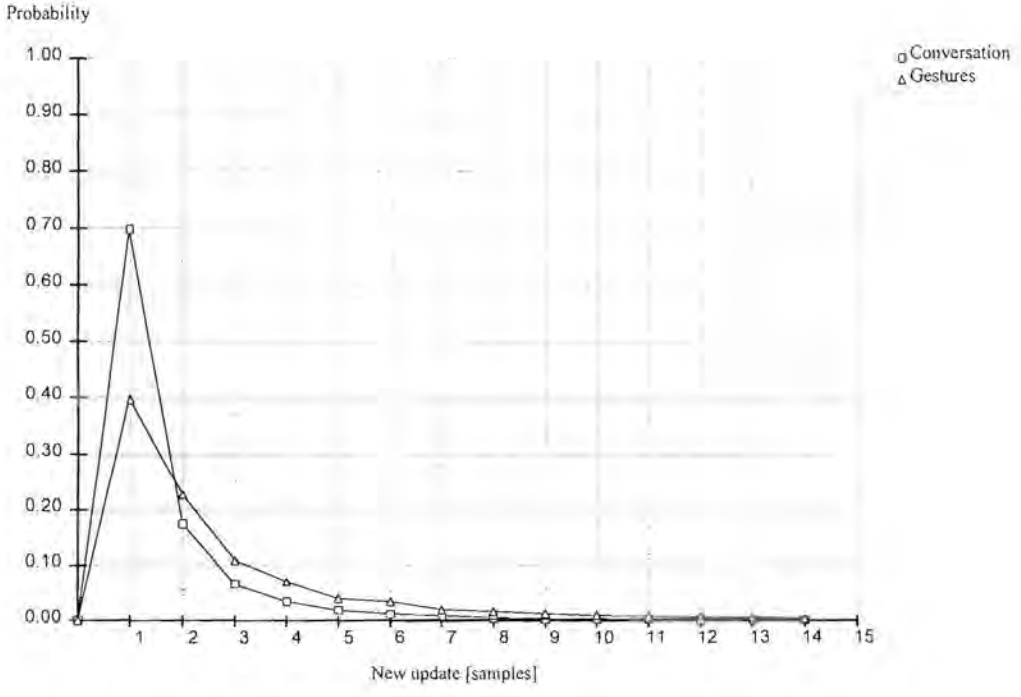


Figure 7-13: Probability of updates.

The dead reckoning algorithm performs reasonably well for slow motion with infrequent bursts of fast motion, and indicates that non-uniform sampling is well suited to such movement. Motion with continuous fast and "busy" sections cannot be predicted accurately with a first order curve. The dance sequence fails completely at low bit-rates. Second order prediction requires an additional acceleration parameter, and we have found that the bit-rate performance even worse. A higher level dead reckoning technique, such as one where the prediction is based on a full dynamic simulation of human motion, might perform better, but is not investigated here.

7.8 Frequency based coding

Frequency or transform based coding methods usually imply a transformation from the time to frequency domain, from which a normal coding route is then taken. Many signals have a more suitable representation in the frequency domain for coding than the time domain representation. The reason for this is that the inherent sample-to-sample correlation that exists in most natural signals tends to cluster the energy in the frequency domain in a relatively small number of transform samples. To achieve bandwidth reduction, those

frequency components or samples with low magnitude could be grossly quantized or even discarded, without introducing serious degradation. Unfortunately, human motion is characterized by large temporal and frequency variations. Natural human motion consists of slow movements with long time windows and small frequency windows, with occasional fast movements with narrow time windows and large frequency windows. It would therefore be convenient to describe the signal in both the time *and* frequency domain. This can be accomplished by a number of so-called time-frequency methods, such as short-term Fourier spectrum manipulation and wavelet decomposition. Wavelet decomposition is particularly attractive, since it helps observing rapidly changing functions by using shorter time windows, and low frequency components by using longer time windows (this is different from the Fourier transform, where the bases are characterized by an infinite time window).

7.8.1 Discrete cosine transform (DCT)

The Karhuhen-Loeve Transform (KLT) (sometimes referred to as the eigenvector transform) is a technique for transforming a signal into a set of uncorrelated representational coefficients. However, it is well known that signals with Markovian properties can be decorrelated with faster and simpler approaches such as the Discrete Cosine Transform (DCT), while approaching the efficiency of the KLT process. In chapter 4, we have seen that human motion indeed exhibits such temporal causal relations, or Markov properties, due to the inherent inertial forces at work.

The DCT transforms a real sequence $\{\theta(n)\}$ of length N into an array $\{\Theta(n)\}$ of length N frequency coefficients or components. The value of $\Theta(0)$ is often referred to as the DC component, and represents the average or mean of the sequence $\{\theta(n)\}$. The rest of coefficients are referred to as the AC components, and contains increasingly higher frequency information about $\{\theta(n)\}$. Human motion data have relatively low frequencies, and the higher frequency components of $\{\Theta(n)\}$ are often very small and can be discarded. This could dramatically reduce the amount of data that is presented to the channel. The forward DCT is defined by the formula

$$\Theta(k) = C(k)\sum_{n=0}^{N-1}\theta(n)\cos\left(\frac{(2n+1)\pi k}{2N}\right), \tag{7-21}$$

and the inverse DCT is defined by

$$\theta(n) = \sum_{k=0}^{N-1}C(k)\Theta(k)\cos\left(\frac{(2n+1)\pi k}{2N}\right). \tag{7-22}$$

The quantity $C(k)$ is a scaling coefficient and can be implemented in various ways. We use

$$C(k) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 0 \\ \frac{\sqrt{2}}{\sqrt{N}}, & k > 0 \end{cases}. \tag{7-23}$$

Refer to [50] for more details on the DCT.

Figure 7-14 shows a block diagram of a DCT based coder and decoder. Disregard the adaptive section for a moment. The input sequence is segmented in groups of length N, where N is a power of two, to be able to use a fast algorithm for the DCT. The resulting DC and AC coefficients are quantized and statistically coded. Again, there is a choice between using inter-frame prediction for the coefficients, or using straight quantization. This section of the coder is very similar to the predictive and adaptive predictive coders discussed earlier, and will not be explained in detail. The DCT coder is a bit more complicated than the simple predictive coder in the sense that there are more parameters to be optimized, especially regarding the quantization of various frequency components. We use empirically determined step sizes and ranges for the different AC and DC frequency components. The DCT based decoder, also shown in figure 7-14, simply performs the inverse operations of the coder, including an Inverse Discrete Cosine Transform (IDCT).

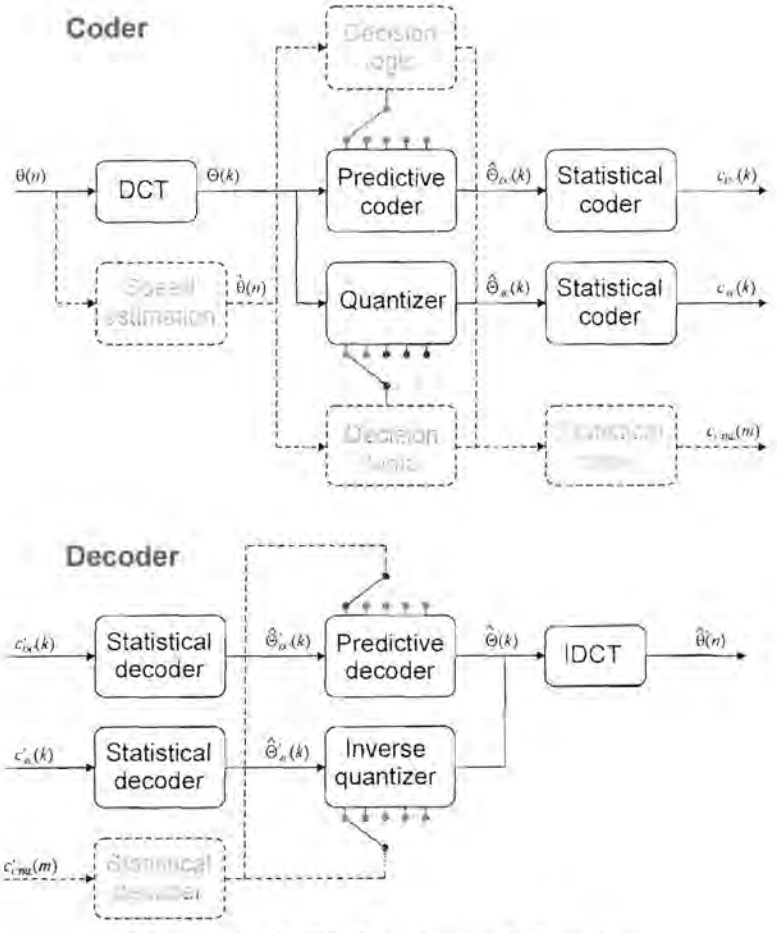


Figure 7-14: DCT coder and decoder.

Similar to techniques used in video compression such as the MPEG format, the operation of the DCT coder can be adapted to short-term motion characteristics. We have found that the visual artifacts generated by quantization errors are less noticeable during fast movement. The quantization can therefore be coarsened in such situations, resulting in an increased compression ratio. The dotted lines in figure 7-14 indicate an adaptive solution that has been developed for human motion. An estimate of the first derivative (i.e. the speed) is obtained from the input sequence. This quantity is then passed through a threshold decision algorithm, which in turn chooses from a set of quantizers and predictive coders. This is a feed-forward approach, and the decision information must be transmitted along with the coded frequency components. It should be noted that the use of adaptive coding based on an subjective quality observation would adversely affect the use of the MS error measurement, although there is a decrease in bit-rate. The results and PSNR measurements presented at the end of the section were obtained using the adaptive method.

The DCT operation could obtain a more efficient frequency estimate of the input sequence for larger N, but we also have to realize that we introduce a delay of N samples by doing so. Furthermore there are more non-zero AC coefficients to be coded. On the other hand, for small N there are more DC components to be coded, each of which uses more bits than the corresponding AC components. We use a relatively small segment length of $N = 16$, which was determined empirically. Even so, at an input sample rate of 30 Hz, this represents a coding delay of 0.5 seconds. This delay sets the DCT method completely apart from the other techniques presented in this chapter. If real-time interaction is of cardinal importance, high delay techniques such as transform coding cannot be used. However, since the exact delay is known, it can be compensated for when comparison studies with other algorithms are done.

Figures 7-15a to 7-15d show the results for the DCT compression method on the representative joint angles. Depicted are the original signal, the decoded signal with two different adaptive threshold decision schemes, as well as the error signal of the second adaptive method. The threshold and quality parameters are given in Appendix III. It is clear that the DCT method outperforms predictive coding by at least a factor two. The errors introduced by the DCT algorithm are also visually much more pleasing, except for block effects due to the finite length window used, which results in a periodic jerk. We compensate for this by smoothing the first and last samples of two adjacent blocks (not shown in figure 7-15). The MS error increases slightly in doing so, but the visual results are much more pleasing, as can be seen clearly in the video clips.

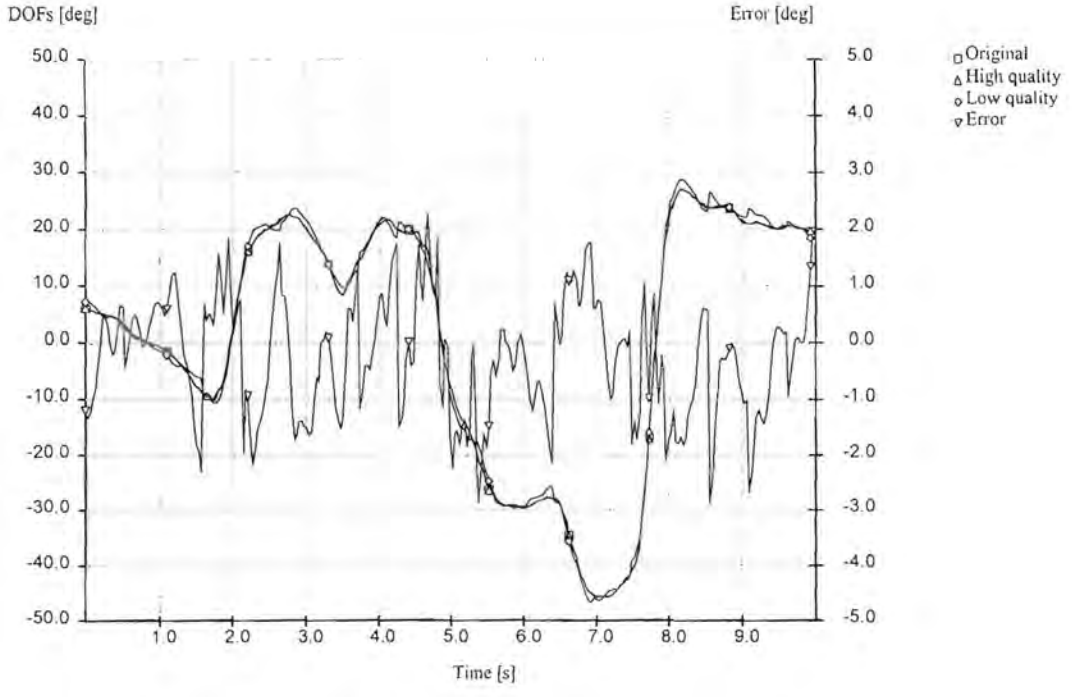


Figure 7-15a: DCT coding of $\theta_{3,0}$ for the conversational sequence.

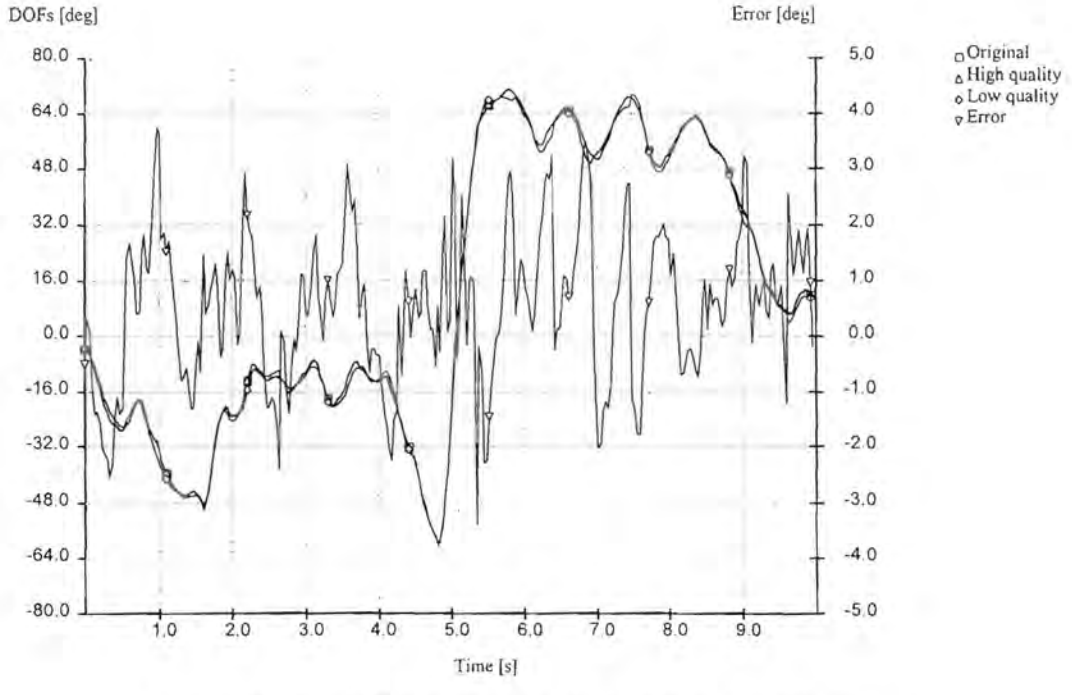


Figure 7-15b: DCT coding of $\theta_{5,0}$ for the wave sequence.

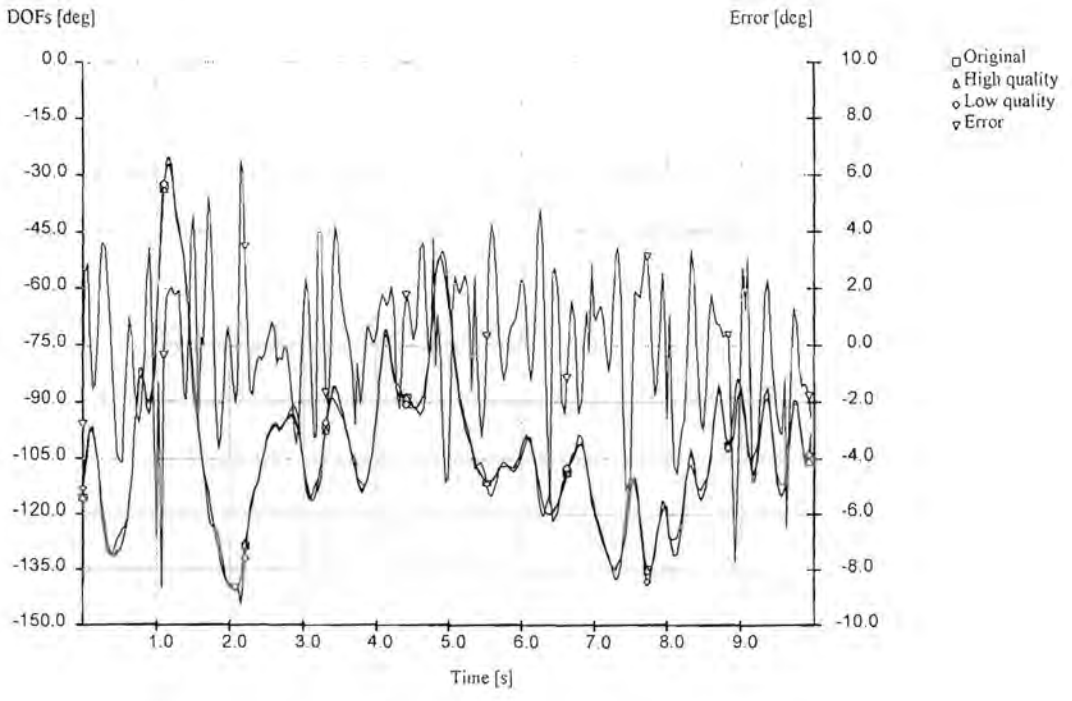


Figure 7-15c: DCT coding of $\theta_{6,0}$ for the dance sequence.

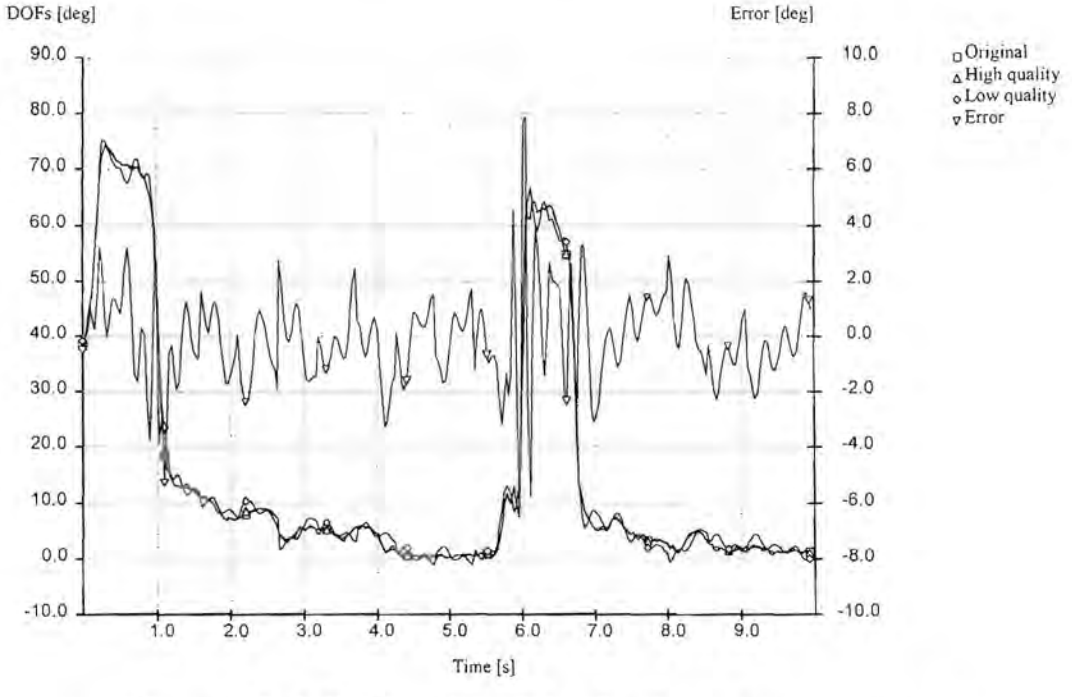


Figure 7-15d: DCT coding of $\theta_{27,1}$ for the gesture sequence.

Figure 7-16 shows the PSNR against bit-rate for the DCT coding method. There is a clear advantage compared to the other methods presented thus far.

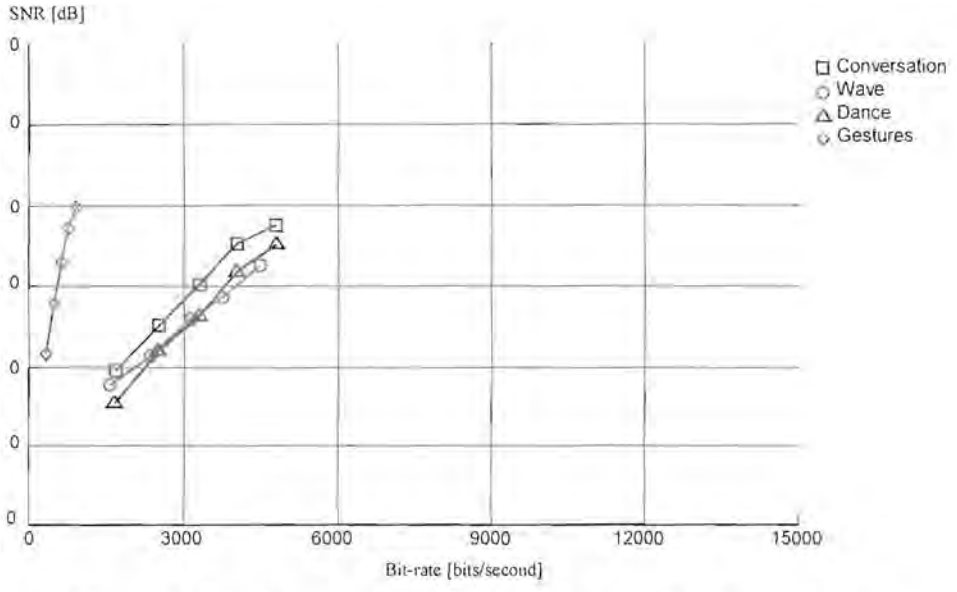


Figure 7-16: Distortion vs. bit-rate for the DCT algorithm.

7.8.2 Time-frequency methods

We briefly investigate the use of wavelet decomposition and subband coding as human motion compression methods. Both of these methods offer a more intuitive approach towards quantization strategies. For example, the structure of a subband coder closely resembles the definition of the VMSE discussed in the previous chapter. A natural bit allocation and quantization strategy could therefore be applied to favour the formulation of the VMSE in order to obtain visually pleasing decoded motion. This concept is very similar to the noise masking characteristics of the human ear. Speech coders (such as subband coders) often make use of these characteristics to enhance the perceived audio quality. In a similar manner wavelet decomposition allows us to choose and manipulate various properties of the motion signal to achieve a better visual effect.

Figure 7-17 shows a generalized subband coder/decoder and figure 7-18 a basic wavelet approach. The subband coder divides the input signal into a number of frequency bands, each of which is separately coded with a predictive coder. The performance of the predictive coders can be adjusted to better suit visual fidelity, even if it results in a drop in PSNR. We use a frequency division scheme similar to figure 6-1. The wavelet approach

decomposes the input signal into a number of decomposition levels. We use a full decomposition, i.e. there are $\log_2(N)$ levels for an input length N that is a power of two. The M largest decomposition values are retained by means of a threshold system, where $M < N$. These values are separately coded with predictive coders that can be adjusted for visual fidelity.

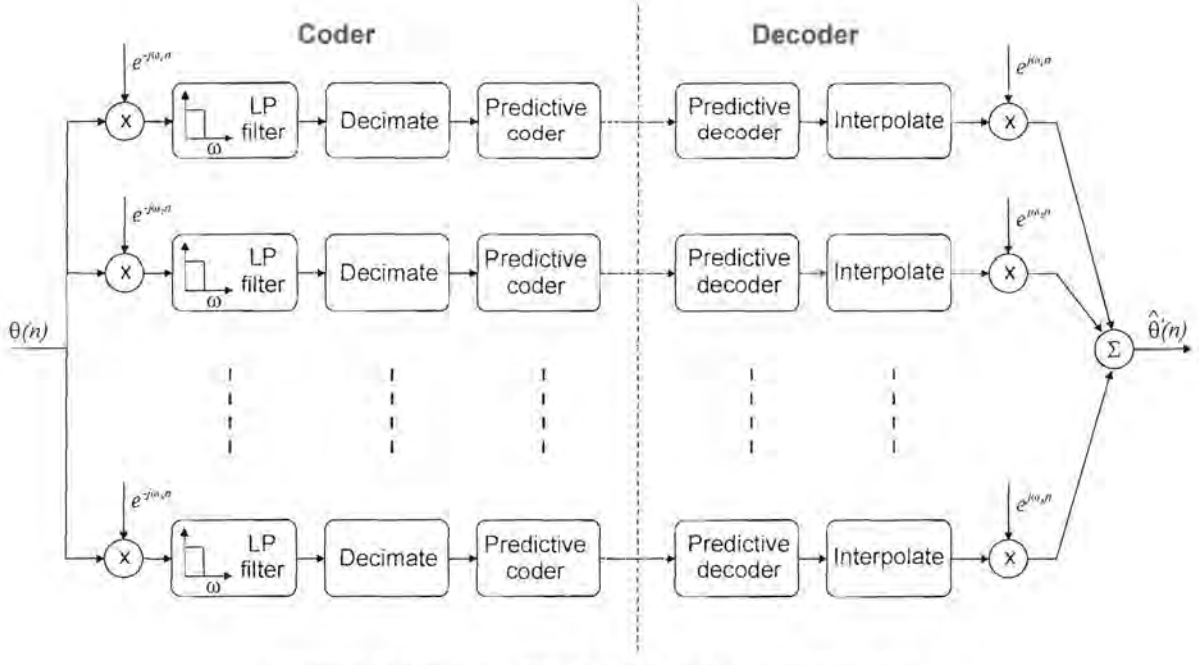


Figure 7-17: Simplified subband coder/decoder.

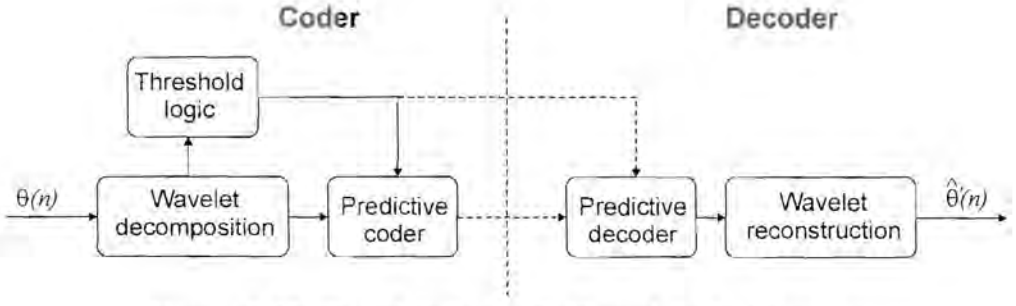


Figure 7-18: Simplified wavelet based coder/decoder.

In theory, the time-frequency methods appear very appealing and intuitive. However, due to the very low sampling rate there are few samples to work with. Similar to the DCT method, we cannot afford a coding delay of much more than 0.5 seconds or 16 samples. This results in practical issues such as filter length constraints and windowing problems.

The actual implementation also suffers from annoying artifacts caused by block edge effects that completely mask the visual gain obtained by the selective coding process. Due to these edge effects we have found the resulting performance to be roughly the same as the DCT method, and visually below that of the adaptive DCT method for similar bit-rates.

7.9 Vector quantization methods

Vector quantization (VQ) involves the grouping of a number of variables together to form a vector. This group can then be quantized, as opposed to the single variable quantization discussed above. Variables or joints that are grouped together must be correlated in some fashion, otherwise nothing is gained and the results will be no different from single variable quantization. Variables can either be grouped spatially or temporally, or both.

VQ can be formulated as follows [51]: Assume that Θ is a k-dimensional vector consisting of real-valued random variables. Vector quantization is defined as the mapping of Θ onto another k-dimensional vector Θ' such that we can write

$$\Theta' = Q(\Theta). \tag{7-24}$$

Θ' takes one of a *finite* set of values $\{\theta_i\}, 1 \le i \le N$. The set $\{\theta_i\}$ is referred to as the codebook, the individual entries are the code vectors and the size N of the codebook is referred to as the number of levels. To design the codebook, we divide the k-dimensional space of Θ into N regions $\{\mathbf{C}_i\}, 1 \le i \le N$, with a vector θ_i associated with each region $\mathbf{C}_i$. The quantizer then assigns the code vector θ_i if Θ is in $\mathbf{C}_i$.

If we denoted some error measure between Θ and Θ' as $e(\Theta,\Theta')$ the overall average error is given by

$$\varepsilon = \lim_{M\to\infty} \frac{1}{M} \sum_{n=1}^{M} e(\Theta(n),\Theta'(n)). \tag{7-25}$$

The quantizer is optimal in terms of the error measure if the overall error is minimized over all N levels. Two conditions are necessary for this. The first is that the quantizer must be realized by using a nearest neighbour selection rule

$$Q(\boldsymbol{\Theta}) = \theta_i \quad \text{iff} \quad e(\boldsymbol{\Theta}, \theta_i) \leq e(\boldsymbol{\Theta}, \theta_j), \quad i \neq j, \quad 1 \leq j \leq N. \tag{7-26}$$

The second condition is that the code vector θ_i must be chosen to minimize the average error in region $\mathbf{C}_i$. Such a vector θ_i is called the centroid of region $\mathbf{C}_i$, and again depends on the error measure being used.

The establishment of an optimal codebook that fulfills the above requirements generally requires and exhaustive search method and is computationally extremely expensive. An alternative sub-optimal method is the Linde-Buzo-Gray (LBG) algorithm [72], which repeatedly splits a region into two smaller regions and assign a codebook entry to each, until a desired size is reached. The algorithm consists of the following steps:

- Create an initial region that contains the entire training set. The initial codebook therefore has one entry corresponding to the centroid of the entire set.
- Split the region into two using a well defined procedure. The codebook now has twice the amount of entries.
- Repeat the splitting process until the codebook reaches its desired size.

The splitting procedure has a big impact on the optimality of the codebook. Ideally each region should be split or divided by a hyperplane that is normal to the direction of maximum distortion. This ensures that the maximum distortions of the two new regions will be less than that of the parent region. However, calculating the maximum distortion as the codebook size increases becomes computationally expensive, and often a simpler scheme is used. We simply use Euclidean distances (L_2-norms) as error measurement and a "diameter splitting" technique, similar to the alternative LBG algorithm [72].

7.9.1 Spatial quantization

Spatial vector quantization can be done on a group of spatially correlated DOFs. In chapter 5 it was intuitively shown that there is correlation between joints and segments that form part of a limb, such as the arms, legs, torso etc. However, establishing a vector quantization codebook with sufficient entries to accommodate a large number of possible movements requires huge training sets and computational power. Such codebooks are also restricted to a very specific type of motion, and is difficult to generalize. Spatial vector quantization does not perform very well for arbitrary body movement where the spatial interrelationships between DOFs are not clearly specified (i.e. correlation is not necessarily measured accurately by the L_2 norm). Figures 7-19a to 7-19b show the results for the conversational and gesture sequences respectively, each with a codebook length of 32 and 256 entries. Also shown is the error for the 256 entry coding. The reconstruction of the other sequences is similar or even worse, and is not shown. Figure 7-20 depicts the PSNR for all the sequences. Note the change of scale compared to previous rate-distortion results. Although quite high compression ratios are achievable (10:1 or more), the error is clearly unacceptable. The video clips also confirm this.

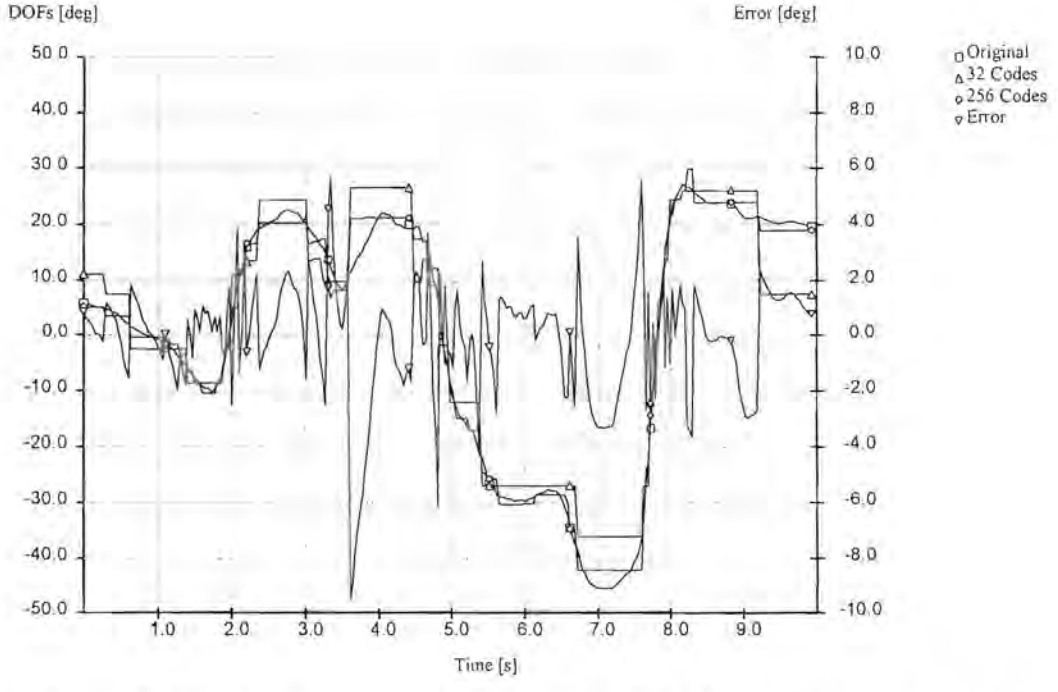


Figure 7-19a: Spatial vector quantization of $\theta_{3,0}$ for the conversational sequence.

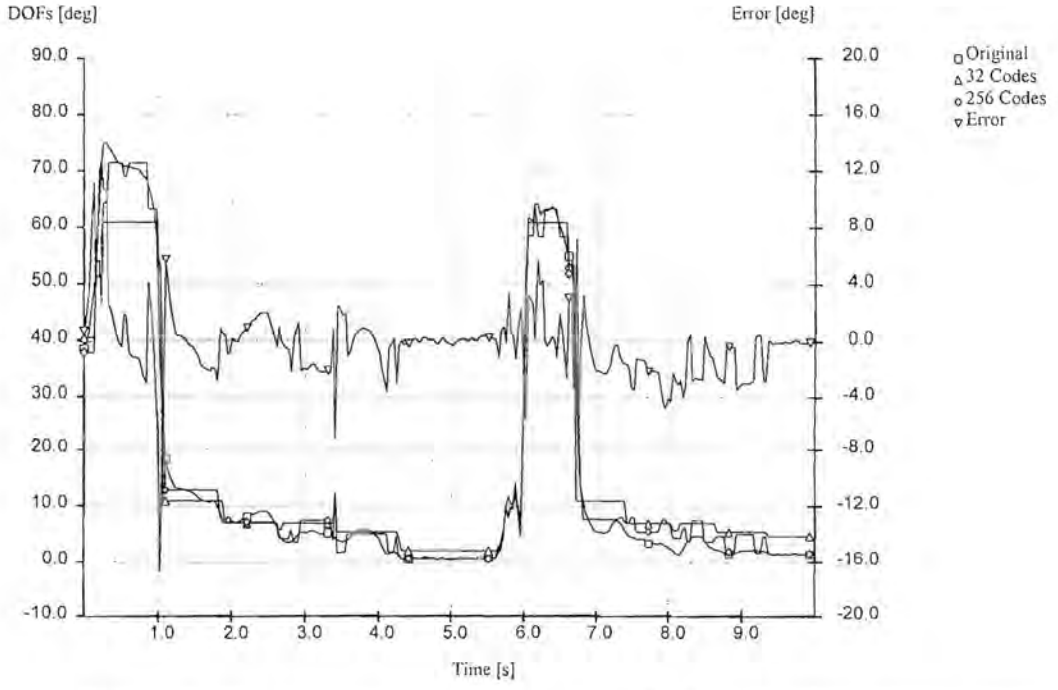


Figure 7-19b: Spatial vector quantization of $\theta_{27,1}$ for the gesture sequence.

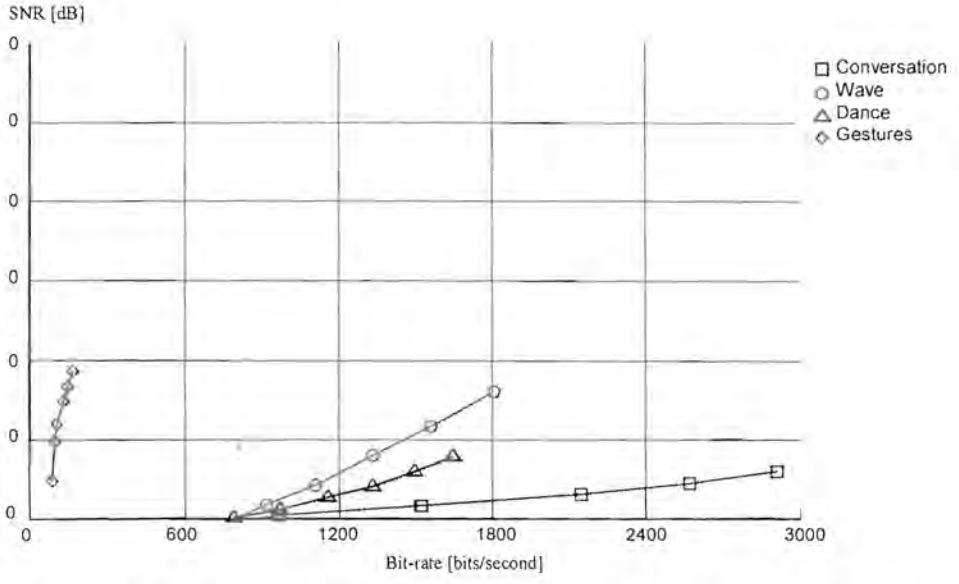


Figure 7-20: Distortion vs. bit-rate for spatial vector quantization.

One example of where spatial vector quantization does perform well, is hand gestures. There are a limited number of universally recognized hand gestures that are frequently used by people. It can almost be seen in the context of a universal sign language. These

gestures are usually a combination of open and closed fingers. From this viewpoint, the whole hand can intuitively be quantized with 32 codes (5 bits – one for each finger). Building such a codebook using the techniques described earlier requires a representative gesture motion sequence that contains examples of all possible gestures. Our gesture sequence contains a fair number of gestures (the small example segment in figure 5-1c shows a counting sequence). It should be noted that such severe vector quantization approaches a gesture recognition system, which is more appropriately described by model based coding (chapter 8). Although the PSNR shown in figure 7-20 indicates an improvement over other body parts, the MS error measurement is not appropriate for spatial vector quantization, especially if combined with some smoothing technique. For example, even at extremely low bit-rates (and hence high MS errors), the information contained in the counting sequence is still quite evident in figure 7-19, and even more so in the video clips.

7.9.2 Temporal quantization

We have already established in previous chapters that there is a high temporal correlation between adjacent DOF samples. A number of consecutive samples may be grouped together to form a vector, which can then be quantized as described in the previous section. Although this technique introduces a delay that is proportional to the dimension of the vector, quite high compression ratios can be achieved with an acceptable MS error. Due to the temporal correlation, a vector dimension of more than one will always yield better results than straight quantization (one-dimensional) of the DOF in question. Figures 7-21a to 7-21d show the results for the representative DOFs of all the sequences, as well as the reconstruction error for 256 codes. Figure 7-22 depicts the PSNR against bit-rate. In all cases a temporal vector dimension of 6 samples (180 ms) were used. Note the change of scale compared to previous rate-distortion graphs. A clear improvement can be seen compared to the results for direct quantization shown in figures 7-2 and 7-3.

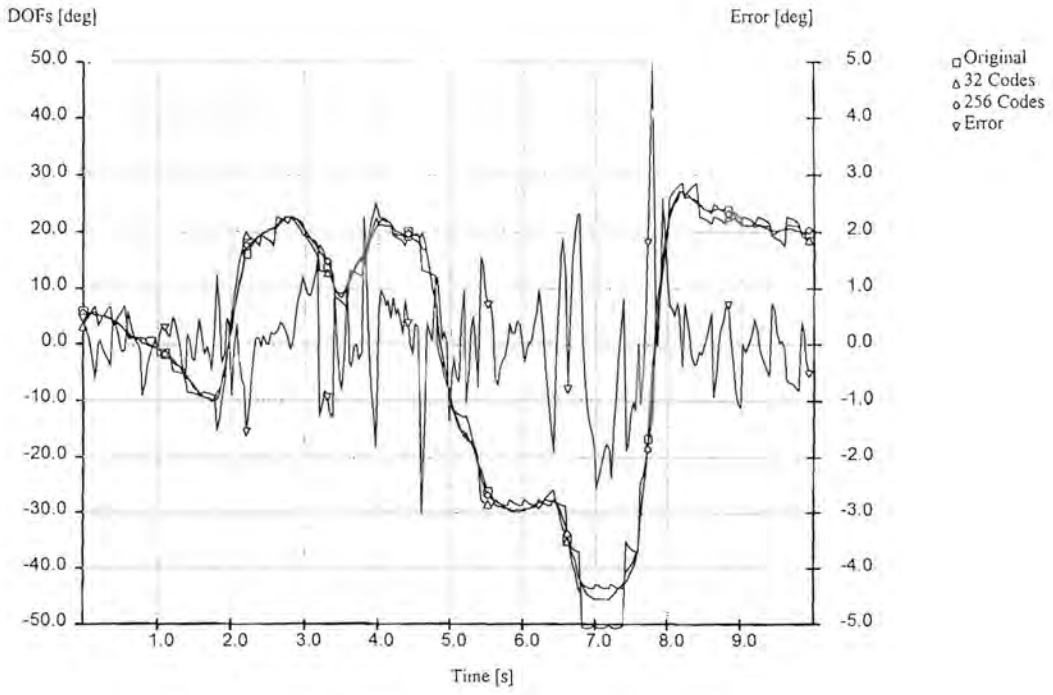


Figure 7-21a: Temporal vector quantization of $\theta_{3,0}$ for the conversational sequence.

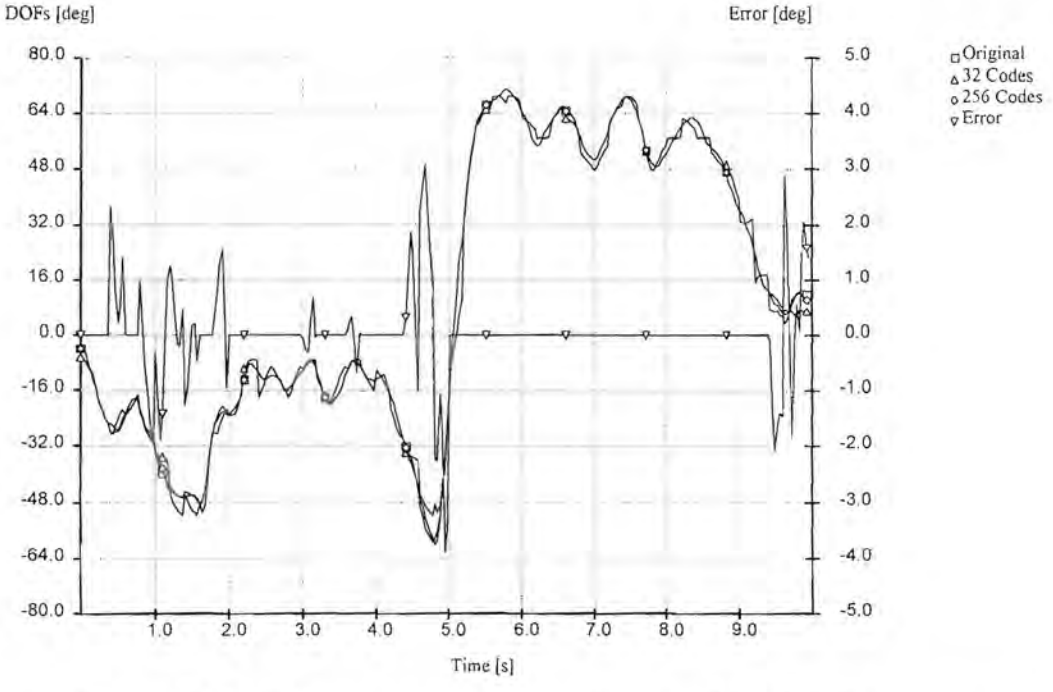


Figure 7-21b: Temporal vector quantization of $\theta_{5,0}$ for the wave sequence.

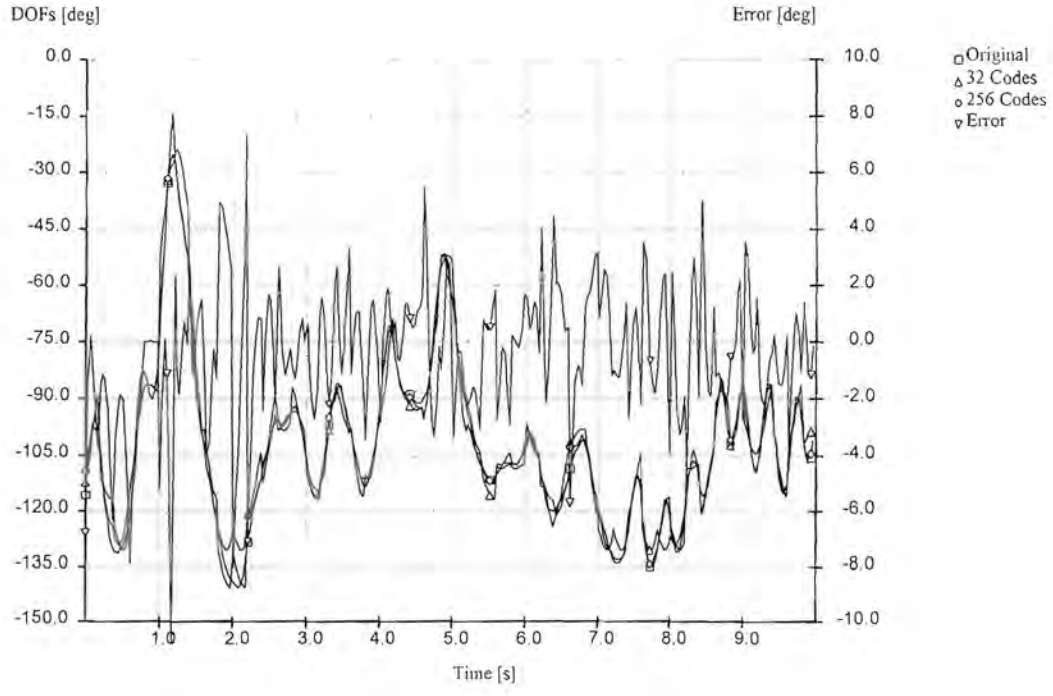


Figure 7-21c: Temporal vector quantization of $\theta_{6,0}$ for the dance sequence.

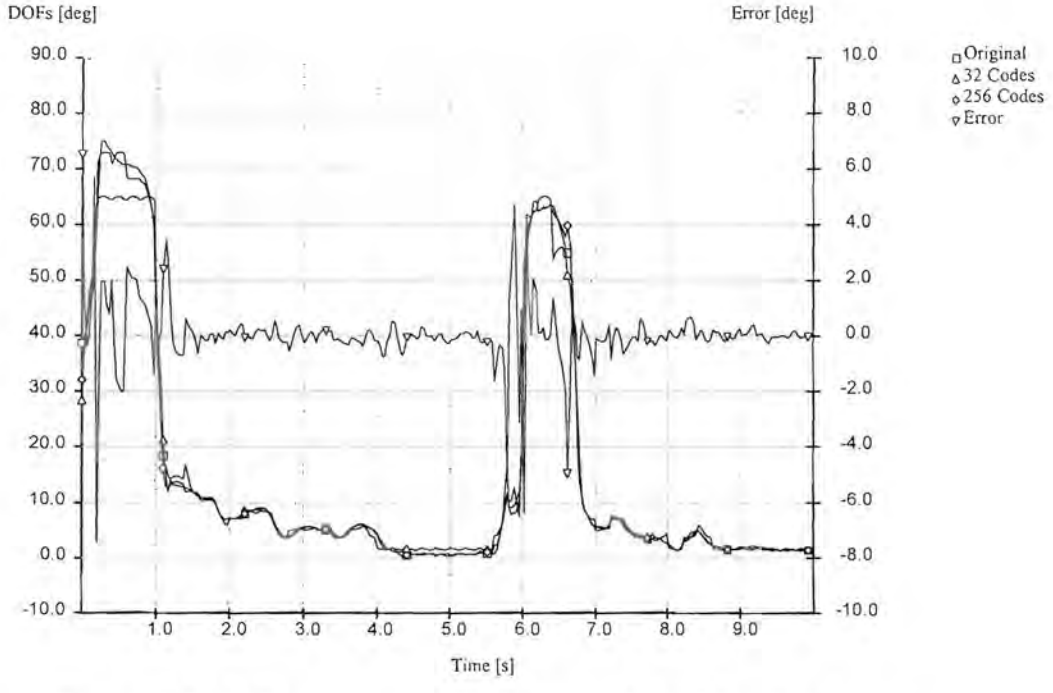


Figure 7-21d: Temporal vector quantization of $\theta_{27,1}$ for the gesture sequence.

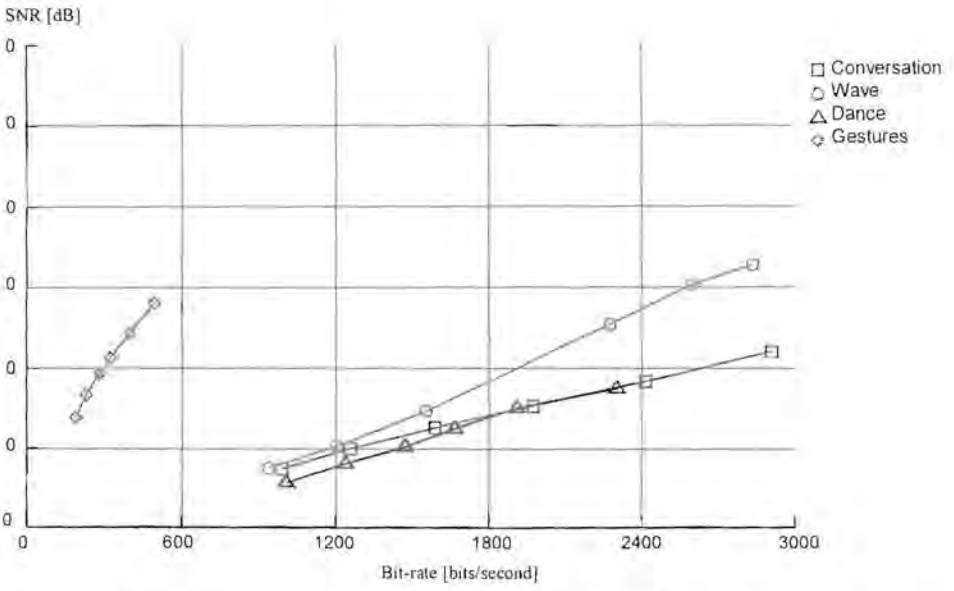


Figure 7-22: Distortion vs. bit-rate for temporal vector quantization.

7.10 Summary

This chapter presented a number of waveform compression methods and results. The use of direct joint angle quantization has been proposed by others [5,29], but at reasonable compression ratios the annoying quantization artifacts prohibits the use of such a naive method. Quantization is left as a functional step in higher complexity methods. The same reasoning is applied to adaptive quantization and statistical coding processes. Predictive coding and especially adaptive predictive coding were quite successfully applied as human motion compression methods. Compression ratios in the order of 5:1 can easily be achieved with these low complexity, low delay methods. Although dead reckoning has been successfully applied to synthetic objects in military applications as a bandwidth reduction technique [30], the results for human motion was not encouraging. Transform coding methods showed great potential, and compression ratios in excess of 10:1 can be expected. However, due to the inherent low sampling rate of human motion, these techniques are plagued by coding delay and block artifacts. The use of vector quantization methods was also investigated, but the requirement of a huge representative training sequence prohibits general use.

Chapter 8 Model based coding

8.1 Introduction

Under the general heading of virtual humans, *model* can either be understood as geometric modeling or motion modeling. Geometric modeling refers to the *synthetic* model or appearance of the computer rendered virtual human figure. This type of modeling, as opposed to traditional pixel-based approaches, already introduces a significant bandwidth reduction. Two-dimensional pixel arrays require vast amounts of storage and transmission bandwidth, whereas a synthetically rendered image requires only the definition of meshes, materials and textures. Once the definition process has been completed, no further information is required to reproduce a high quality image at the receiving end. However, to reproduce a computer rendered image that exhibits credible motion, we need to transmit motion parameters as well. The transmission of motion parameters is a continuous process as opposed to geometric modeling, which is usually completed at the beginning of a session. If we accept the compression that is inherent to geometric modeling as a given, further reduction can be found by examining the motion modeling. Model based coding for virtual humans can therefore be described as any technique whereby parameters that define and control a mathematical model are used to predict and/or reproduce existing motion in a controlled environment.

Similar to many other coding methods, purely temporal methods and purely spatial methods can be regarded as the two extremes in model based coding. These two methods are orthogonal to each other, where "orthogonal" is not used in the strict mathematical sense. The differences between these methods relate to the way the model parameters are obtained and used. The temporal approach is to obtain time-varying *dynamic* actions or motions using a model, and to reproduce the motion using the same or a similar model. The spatial approach is to obtain a set of *static* key postures, and to use a model to

reproduce the in-between motion. Of these two approaches, the temporal method has more potential, but it is also more difficult to implement for a wide range of arbitrary motion. The spatial approach is insensitive to most motion variations, but the compression ratio tends to saturate as the static postures become uncorrelated. The spatial approach is related to the temporal approach in that it *invokes* a model based animation upon changes in posture. Naturally, this is accompanied by a coding delay, which is undesirable, but at least controllable. On the other hand, motions recognized and reproduced using the temporal method cannot so easily be defined and controlled. The ideal solution lies somewhere in between these two extremes.

We define *postures*, *gestures* and *actions* as the essential elements of human motion or behaviour. Postures refer to static, spatial poses that the human figure can assume, and are described independently of temporal information. Gestures are based on temporal motions and behaviours, such as speed and acceleration. Some gestures require postural information, such as start or end conditions, but in general we try to keep the two definitions separate. Actions are a combination of both posture and gesture information. A sequence of one or more actions defines a complete motion. This terminology will be used in the remainder of this chapter.

In its most basic form, a model based coding/decoding scheme consists of three stages:

- A detection stage that recognizes postures, gestures, actions or any combination thereof,
- A reduction stage that reduces the information that is required to reproduce these elements according to a well defined error measurement, and
- A reproduction stage that animates the human figure.

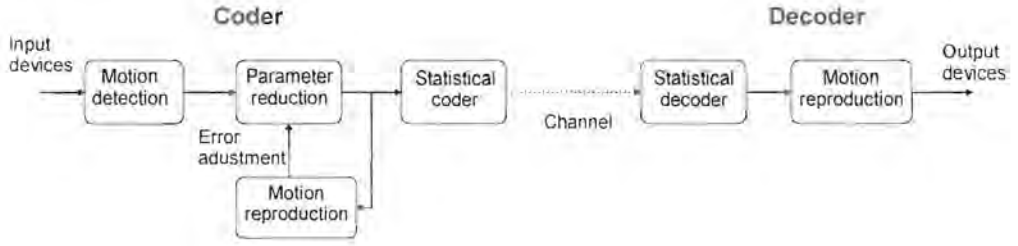


Figure 8-1: Generalized model based coder/decoder.

Figure 8-1 shows a diagram of a generalized model based coder. The reproduction stage is essentially part of the receiver, but also controls the error measurement of the reduction stage. Motion detection in itself is a particularly intricate subject, and can be classified in varying levels of complexity. A fair amount of previous (and current) research in human motion has focused on motion and action recognition [52-57,34]. For most of these systems an essential goal is *machine understanding*. Experimental motion recognition to date usually consists of a small alphabet of possible actions and an elaborate detection and classification algorithm that attempts to overcome problems such as gesture variation and performer independence. Methods such as Hidden Markov Models (HMM) [52] and Phase Space Constraints [54] have been implemented with a fair amount of success. These are high level methods, and applied to motion compression can result in tremendous reduction, but they lack generality.

Figure 8-2 depicts our view of the complexity of motion detection versus the number of parameters involved. At the low end of complexity, the number of parameters is high, such as the individual DOFs of the figure. For high complexity motion detection methods, there are very few parameters involved, in fact, it is conceivable that the whole motion or even the situation can be described by a single parameter. Figure 8-2 also shows four discrete levels on which motion detection can operate. The attribute level operates on individual DOFs, and each is processed independently from any other quantity. The posture level involves a group of DOFs that together form a static posture. On the gesture level, temporal information such as speed and acceleration are added. Finally, a situation level detection scheme attempts to classify whole sequences of motion, and tries to put the meaning of the action into context.

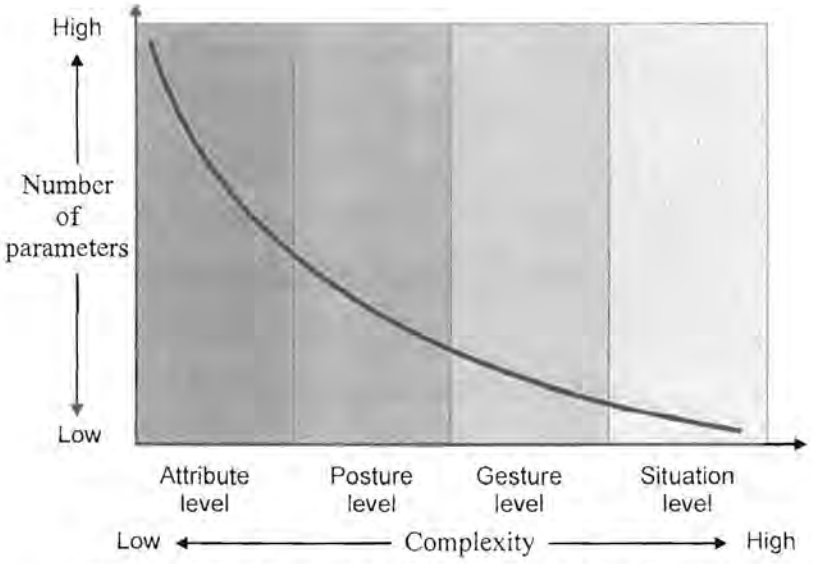


Figure 8-2: Motion detection complexity levels.

Fortunately, in our case machine understanding is not a requirement. As long as well defined and well behaved motion parameters can be detected, the actual *context* or *meaning* of those parameters is of no concern. In the model based coder of Figure 8-1, the motion detection stage plays a very important role. Proper motion detection will determine both the effectiveness and method of the reduction and animation stages. Attribute level parameters require vastly different animation techniques compared to situation level parameters. In this study, we limit ourselves to the first three levels of motion parameters. Attribute level parameters were discussed mainly in chapter 7 on waveform coding techniques (we rather use *parameter* instead of the term *detection*). In this chapter a posture level detection algorithm and a gesture level detection algorithm is investigated.

A final note: Although it is possible to achieve phenomenal compression ratios with model based coding methods, there are two severe penalties. Firstly, the inherent approach of parameter extraction requires a number of samples to be present. This introduces a variable and unpredictable coding delay, which could be as high as two seconds. Secondly, due to the highly synthetic nature of the decompressed motion, it is virtually impossible to compare the resulting motion on an objective scale to the original motion. Although the PSNR used in the waveform coding section still indicates a trend, this quantity frequently

falls below 0 dB, which of course indicates more "noise" than "signal". The VPSNR is used in this chapter to better effect. The coefficients that define the VPMSE can be adjusted to provide an error measure in any range, but we would like to keep it compatible with the normal MSE. The coefficients used throughout this chapter are $\alpha_i = \{0.25, 0.5, 1, 2.25\}$ (refer to chapter 5). Using these values, the VPSNR typically lies between 0 and 20 dB for the model based methods. A VPSNR above 20 dB can be considered as very good quality, while values below 0 dB are considered unacceptable. Although useful, the VPSNR still cannot distinguish between natural, credible motion and completely artificial motion. These penalties should be taken into consideration when various waveform and model based methods are compared in terms of real-time interaction, quality and bit-rate.

8.2 Human motion segmentation

Motion detection and extraction of suitable parameters form the basis of a model based compression algorithm. Once appropriate parameters are available, they can be processed, compressed and transmitted. At the receiving end, the parameters are decompressed and processed into a suitable form for animation purposes. It is the nature of these parameters that primarily determine the way in which processing, compression and animation will be achieved. If human motion is seen as a temporal sequence of events it is evident that we need to "sample" the process at certain points in time to obtain appropriate parameters. The decision of exactly where and how to sample the motion should be made with the objective to obtain the maximum amount of information with the minimum number of parameters. In the chapter on waveform coding, sampling was not really an issue since each DOF was seen as an independent time signal sampled at regular interval by an input device. Clearly, there is quite a lot of variation in human motion, and uniform sampling is not necessarily the optimal approach. The term *segment* is used to describe the portion of motion between two non-uniform samples. In the following discussion, two different segmentation methods are developed, a posture based approach and a gesture based approach.

8.2.1 Posture level segmentation

The posture based segmentation algorithm is a relatively low-complexity method, since no temporal information is directly used. In this approach, the motion of a group of DOFs is sampled at constant distances, where *distance* is defined as a scalar quantity that is obtained by integrating and combining individual DOFs. The scalar distance at sample n is defined as

$$s_i(n) = \sum_{m=1}^{n} \sqrt{\sum_{j=0}^{K-1} \left(\theta_{i,j}(m) - \theta_{i,j}(m-1)\right)^2}, \tag{8-1}$$

where K is the number of DOFs for the ith group (see table 5-1). Clearly, equation (8-1) is a monotonically increasing function, and by sampling s_i at constant distances d apart, a unique sample value n can be obtained on the time axis. These time values are the locations at which the original motion is non-uniformly sampled, and segments are represented as the portion between such samples. This concept is illustrated in Figure 8-3. Since the rate of the input signal is known, and the exact location of each non-uniform sample is known, the length of each segment can be calculated if desired. Care should be taken not to use values of d that are too large, as the resulting aliasing effects would make the information unrecoverable.

Figure 8-3 also shows an example of the segmentation scheme for the right arm group. The wave sequence with $d = 40$ was used in this example. For clarity, only the upper arm twist angle $\theta_{22,2}$ is shown. Also shown are the monotonic function s_i and the resulting segments. It can be seen that the group motion is segmented frequently during moments of high activity, and less so during low activity.

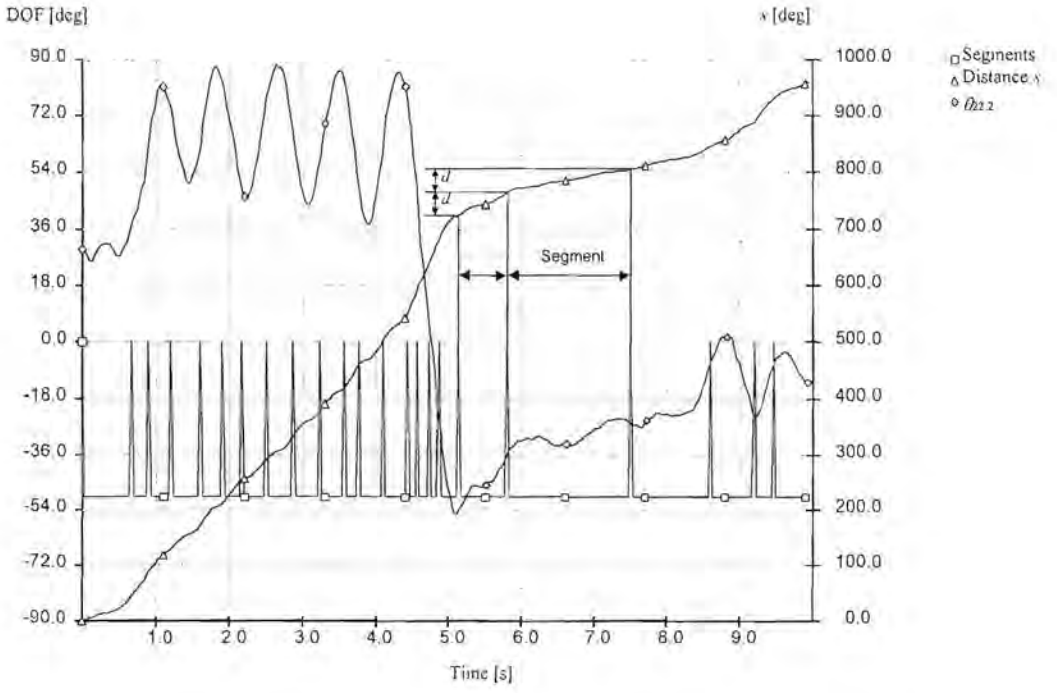


Figure 8-3: Posture based segmentation example.

8.2.2 Gesture level segmentation

Our gesture segmentation scheme follows a completely different approach than the posture based method. In fact, posture information is completely discarded and the temporal information contained in the first and second derivatives (speed and acceleration) of a group of DOFs is used. In this temporal context, a *gesture* is defined as the motion of a group of joints or DOFs between two successive stationary (or almost stationary) moments. A segment is the portion between the local minimums of the speed, provided that it is not larger than a certain threshold. The location of these minimum values are used to non-uniformly sample the motion. In order to segment the gestures of a group of DOFs as a whole, and still retain the convenience of mathematically tractable equations, we need to define some combination of individual DOF derivative information. A scalar speed quantity ω_i is defined as the length of a K-dimensional speed vector, whose elements are the first derivatives of the K DOFs of the ith group. It can be written as

$$\omega_i(n) = \sqrt{\sum_{j=0}^{K-1} \dot{\theta}_{i,j}(n)^2}, \tag{8-2}$$

which is a positive quantity. Clearly, there is a relationship between equations (8-2) and (8-1), i.e. $\omega_i(n) = \dot{s}_i(n)$. However, we use a more stable estimation of the speed as explained below. The scalar acceleration is simply given by

$$\alpha_i(n) = \dot{\omega}_i(n), \tag{8-3}$$

which is a bipolar quantity. Note that the acceleration is *not* defined as the length of the original DOF acceleration vector. There are a number of methods for estimating the derivative of a sequence of discrete time samples. We fit a third order polynomial through a set of points using a least squares solution, and then analytically calculate the derivative at the desired sample. It is reasonable to assume that a small enough window will not have an adversely averaging effect. We have found that a window length of seven samples gives acceptable results.

Practical implementation of the segmentation algorithm is a bit more involved than the posture based method. We propose an adaptive speed threshold algorithm to identify possible windows of interest. The zero crossings of the acceleration quantity within the window are then used to find the exact sampling locations. Figure 8-4a shows a blown up region of a typical signal. The threshold signal is a low-pass filtered version of the speed, and is given by

$$\zeta_i(n) = \beta \sum_{k=0}^{M-1} \omega_i(n-k)h(k). \tag{8-4}$$

The impulse response $h(k)$ can be any suitable form of a low-pass filter. The choice of filter length M is also not critical, and can be adjusted to suit the characteristics of the segmented motion. The quantity β is a scaling factor that determines the aggressiveness of

the threshold process, and is a convenient single parameter with which to adjust the performance of the algorithm.

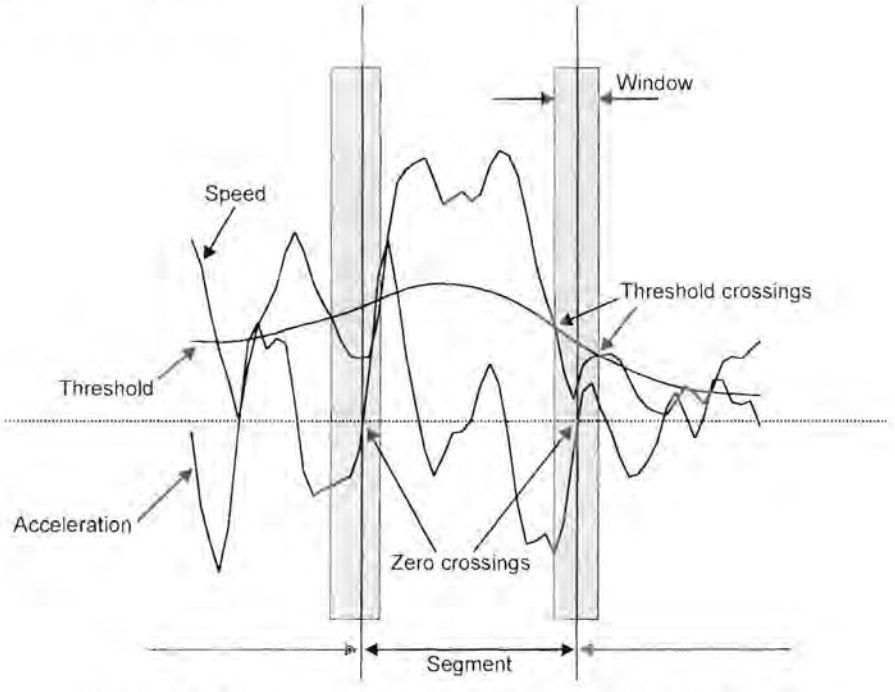


Figure 8-4a: Gesture based segmentation algorithm.

The algorithm is implemented recursively as follows: A possible window of interest is identified between an above/below threshold crossing and a below/above threshold crossing for the speed (see Figure 8-4a). The acceleration is then evaluated within the window for all the zero crossings. A minimum number of samples between each non-uniform sample are specified, and any values in this range are discarded. A maximum number of samples are also specified. When any new segment detects an excessive length, the value of β is adjusted to a slightly higher value and the algorithm is recursed again for the offending region. Some types of motion may result in an endless loop, and it is prudent to put an upper limit on the value of β, after which the recursion immediately unwinds. Excessively long segments can be left as they are, or subdivided by other means. It should be noted that for such long segments the motion probably had very little information content anyway, and the non-uniform sampling should be acceptable as it is.

Figure 8-4b shows an example of the segmentation scheme for the right arm group, and can be compared with Figure 8-3. Also indicated are the speed, threshold and acceleration

quantities. It is clear that the segmentation follows a completely different pattern compared to the posture based approach. There are fewer samples, and they are more uniformly spaced. The well defined sampling of $\theta_{22,2}$ can clearly be seen (i.e. where the derivative is almost zero), compared to the more random behaviour of the posture based approach in Figure 8-3.

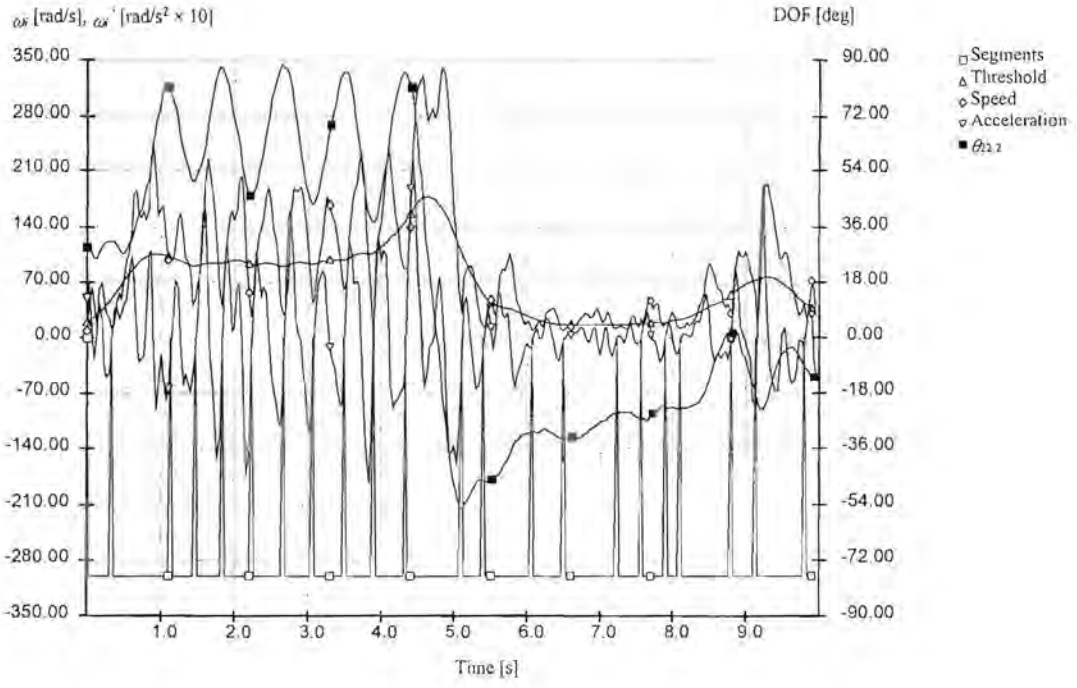


Figure 8-4b: Gesture based segmentation example.

8.3 Segment animation

As will be discussed later, data reduction can be achieved by transmitting only the non-uniform samples and possibly information concerning the lag between samples. This implies that the "missing" motion contained in the segments must be estimated. Segment animation can be done in a number of ways, of which three will be discussed briefly, namely end point interpolation, dynamic simulation and motion capture segments.

8.3.1 Interpolation

The most basic form of animation is the interpolation of motion between two static start and end postures. A variety of interpolation functions can be used depending on the boundary conditions. Smooth motion animation is a strong requirement. This condition implies that the interpolation function should at least have G^0 and G^1 continuity at the endpoints, i.e. the end/start positions of two consecutive segments should match exactly, and the first derivative should be equal on both sides. We have found that Hermite cubic polynomial curve segments [12] between two non-uniform samples provide satisfactory results. Such a function can be written in parameterized form as

$$w(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \tag{8-5}$$

where t is a parameter in [0, 1]. The lag between the actual start and end points should be normalized to the parameter t for equation (8-5) to be meaningful. It can be shown that the coefficients a_i are given by

$$\begin{aligned} a_0 &= \theta(m-1), \\ a_1 &= \Delta(m-1), \\ a_3 &= 3\big(\theta(m)-\theta(m-1)\big)-2\Delta(m-1)-\Delta(m), \\ a_4 &= 2\big(\theta(m-1)-\theta(m)\big)+\Delta(m-1)+\Delta(m). \end{aligned} \tag{8-6}$$

at each non-uniform sample m. Note that the subscript i,j has been dropped from the DOFs $\theta(m)$ for clarity. The quantity $\Delta(m)$ is the slope (first derivative or speed of the DOF) of the signal at sample m. $\Delta(m)$ is not directly available, and must be estimated since it is undesirable to transmit this value along with the posture information for *every* DOF in the group. We use an estimate of the form

$$\Delta(m) = s\big(\theta(m)-\theta(m-1)\big)+(1-s)\big(\theta(m+1)-\theta(m)\big), \tag{8-7}$$

where

$$s = \frac{r(m+1)}{r(m)+r(m+1)} \tag{8-8}$$

is a scaling factor in [0, 1], and $r(m)$ is the lag between non-uniform sample m and m-1. By using a sample at m+1, even further coding delay is incurred since we have to wait for the sample to arrive. However, the slope estimate of equation (8-7) is much more accurate than a first order approach, and especially at zero-derivative turning points the overshoot is much less.

It will be seen later that the gesture based segmentation method provides the implicit and very useful information that the derivatives of the interpolated DOF are zero (or almost zero) at the endpoints. After the work done by Perlin [18,19], we have found that a raised cosine interpolation function

$$w(t) = \frac{1+\cos(\pi t)}{2}, \tag{8-9}$$

where t is a parameter in [0, 1] gives very natural results, and fulfills the derivative condition at the endpoints. Interpolation using this function is straightforward, and is given by

$$\theta(t) = w(t)\theta(m-1) + \big(1-w(t)\big)\theta(m), \tag{8-10}$$

where it is understood that the parameter t has been normalized between the non-uniform samples at m-1 and m.

8.3.2 Dynamic simulation

The use of dynamic simulation of human motion has been put to very effective use recently in the field of computer animation. By making use of efficient algorithms, the basic equations of motion can be solved for complex hierarchical structures in real-time.

The use of dynamic simulation has been discussed in chapters 2 and 3, and Appendix II gives a summary of an algorithm that can be used.

Unfortunately, proper dynamic simulation requires well defined initial conditions and parameters to ensure a stable solution of the differential equations. The rather poor estimate of the speed (or slope) using the posture based approach causes numerical instabilities and overshoot when using dynamic simulation as a segment animation method. On the other hand, the closely spaced and fixed segments of the gesture based approach diminish the benefits of full dynamic simulation. Due to the adaptive algorithm, segments are seldom overly long, and dynamic simulation typically gives similar results as simple interpolation. The fact that the speed is not always exactly zero at the segment endpoints also destroys the finer detail and subtlety that could have been obtained with dynamic simulation.

8.3.3 Motion capture segments

A third approach to segment animation is to fill the segment with actual motion captured data. Given a substantial list of properly processed motion capture segments, a search can be performed against a given error criteria, and the best match fitted to the segment. When the error requirement cannot be fulfilled, the segment can simply be interpolated using one of the previous methods. The actual generation of a suitable table of motion entries is quite a daunting task, and requires extensive training sequences. The concept of such a table of motion is described in more detail in section 8.4, where the actual implementation of a model based coder is discussed.

Segmented motion requires a method of combining a sequence of actions to form smooth, continuous motion. Similar to the interpolation scheme discussed above, the end and start sections of two motion captured segments can be blended together using an interpolation function. Figure 8-5 shows an example of blending two DOFs $\theta_{i,j}$ and $\theta'_{i,j}$ from completely different motion segments. In practical compression methods, such severe differences will

usually not be tolerated. The interpolation function of equation (8-9) has been found to give good results and is used for blending motion segments.

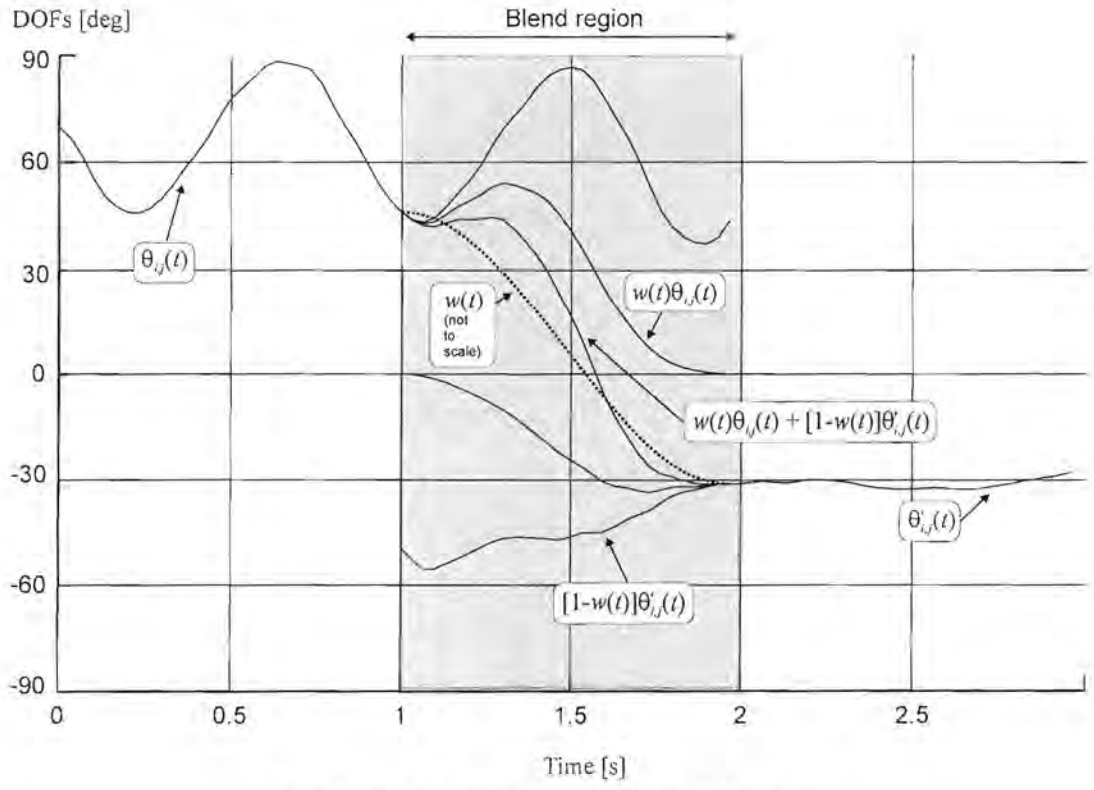


Figure 8-5: Example of motion blending.

8.4 A model based coder/decoder

Once a proper segmentation and animation algorithm is in place, a model based coder can be constructed by realizing that it is simply needed to transmit the non-uniform samples, as well as some information conveying the location of the samples. As with waveform coding techniques, it is crucial to quantize the actual values of the samples to a well-defined set of symbols in a controlled fashion. Although the values of the individual DOFs in a group can be quantized separately, more gain is achieved by making use of the correlated nature of the DOFs. It is therefore more appropriate to use some form of vector quantization technique. As discussed in the previous section, a further quantity that will be transmitted is a code describing the motion segment that will be fitted between the current samples. We define the term *posture table* for the vector quantization table containing the group

DOF values, and the term *gesture table* for the table containing the list of motion segment candidates.

8.4.1 Posture table

A posture table can be constructed in a similar fashion to the spatial vector quantization method presented in the previous chapter. However, in this case we have found that the *position* of joints in 3D space gives better results than the joint angles directly. A vector sequence $\{\mathbf{p}_i(n)\}$, consisting of all the x, y and z Cartesian joint position components of group i, is constructed using a *training* motion sequence. The individual components of $\mathbf{p}_i$ are denoted by $p_{i,j}$, $0 \leq j < K$, where K is the dimension of the composite position vector. We use a greedy $O(N^2)$ search algorithm to obtain the M vectors that are spaced a distance d or greater apart. This is an alternative approach to the clustering LBG algorithm presented in chapter 7. Similar to the error measure of equation (6-13), all the entries in the sequence which satisfies

$$\sum_{j=0}^{K-1} \frac{a_{i,j}\left(p_{i,j}(n)-p_{i,j}(m)\right)^2}{b_{i,j}} > d^2, \quad 1 \leq n \leq N, n < m \leq N, \tag{8-11}$$

for a sequence length of N are retained. For simplicity, we set the matrices a_{ij} and b_{ij} to unity and take the square root to obtain units of meters. By adjusting the value of d, the number of codebook entries M can be adjusted. The joint angles corresponding to a vector position $\{\mathbf{p}_i(n)\}$ are also stored in the resulting table of entries, so that a position vector can directly be mapped to a set of angles for animation purposes. The complete figure contains 8 groups (see table 5-1 for details), and there are therefore 8 separate vector quantization tables. The posture table for group i is denoted by $\mathbf{P}_i$, and contains M codebook entries $\{\mathbf{p}_m\}_i$, $0 \leq m < M$.

8.4.2 Gesture table

The gesture table is constructed similarly to the posture table, except that we now consider small sequences of motion, rather than a single sample or static posture of motion. We propose the following method for the construction of a gesture table: A large training sequence is segmented using the gesture based algorithm described above. Although in theory it is possible to use posture based segmentation, the resulting segments will be poorly defined and not of much use. After segmentation, the gesture table contains *all* the motion from the training sequence. Segments that are similar are discarded according to some error or distance measure. We use a simple weighted MS similarity measure of the form

$$\varepsilon = \sum_{n=1}^{N} w(n) \sum_{j=0}^{K-1} \left(\theta_{i,j}(n) - \theta'_{i,j}(n)\right)^2, \tag{8-12}$$

where N is the length of the segment and K is the number of DOFs in the ith group. The weighting function $w(n)$ is given by

$$w(n) = \begin{cases} b + \dfrac{(a-b)}{n}, & 0 < n \le \dfrac{N}{2}, \; a \ge b \\ b + \dfrac{(a-b)}{N-n+1}, & \dfrac{N}{2} < n \le N, \; a \ge b \end{cases}. \tag{8-13}$$

For $b = a$, equation (8-13) is constant over the segment. For $b < a$ it can be seen that less importance is given to the interior of the segment. The segment start and end values are important to ensure a good match at the boundaries. A greedy $O(N^2)$ search algorithm is used to discard all the segments that are closer than some error distance. In order to accommodate segments of varying length, we propose that each segment is warped or normalized in time to a predetermined length. Small segments are therefore stretched in time, and long segments shrunk. The normalized length is a function of the maximum segment length that is allowed by the segmentation algorithm. We have found that a value of one second (or 30 samples at an input sampling rate of 30 Hz) is appropriate. Similar to

the notation used for the posture table, the gesture table for group i is denoted by $\mathbf{G}_i$, and contains M normalized codebook sequences $\{\mathbf{g}_m(n)\}_i$, $0 \leq m < M$. The quantity $\mathbf{g}_m$ is a vector for which its elements are the DOFs of the group in question.

8.4.3 Posture based compression

We define posture based compression as posture based segmentation followed by vector quantization of the samples using a posture table. After segmentation, a group position vector $\mathbf{p}'_i(m)$ is generated at each non-uniform sample m for the sequence to be coded (the coding sequence is similar, but not identical, to the training sequence). The position vector for the mth sample, $\mathbf{p}'_i(m)$, is matched against the posture table $\mathbf{P}_i$, and the closest entry is selected (we simply use an L_2-norm). The result is a string of codes that is passed to a statistical coder to obtain a symbol stream of optimal length. Decompression reverses the process by decoding the symbols and obtaining the corresponding posture from the posture table. The method described by equations (8-5) to (8-8) is used to interpolate the motion between consecutive non-uniform samples.

The algorithm has a penalty in the form of variable coding delay, which can be quite severe in some cases. For practical reasons, it is convenient to compensate for the (known) delay when generating comparison results such as rate distortion graphs. Figure 8-6a shows the results for two different posture segmentation distances d. The example shows the right upper arm twist angle $\theta_{22,2}$ for the wave sequence. Clearly, if d is too large severe aliasing occurs and the motion cannot be reconstructed. For a smaller value of d, the motion is non-uniformly sampled at sufficiently small intervals to reconstruct the motion, albeit in a very synthetic fashion. Figure 8-6b depicts a number of 3D wireframe images from the dance sequence at $d = 20$, which corresponds to roughly 450 bits/second. Although there is a substantial difference, the relationship between the original and coded motion is still clear. These results are best viewed from the video clips provided on CD-ROM.

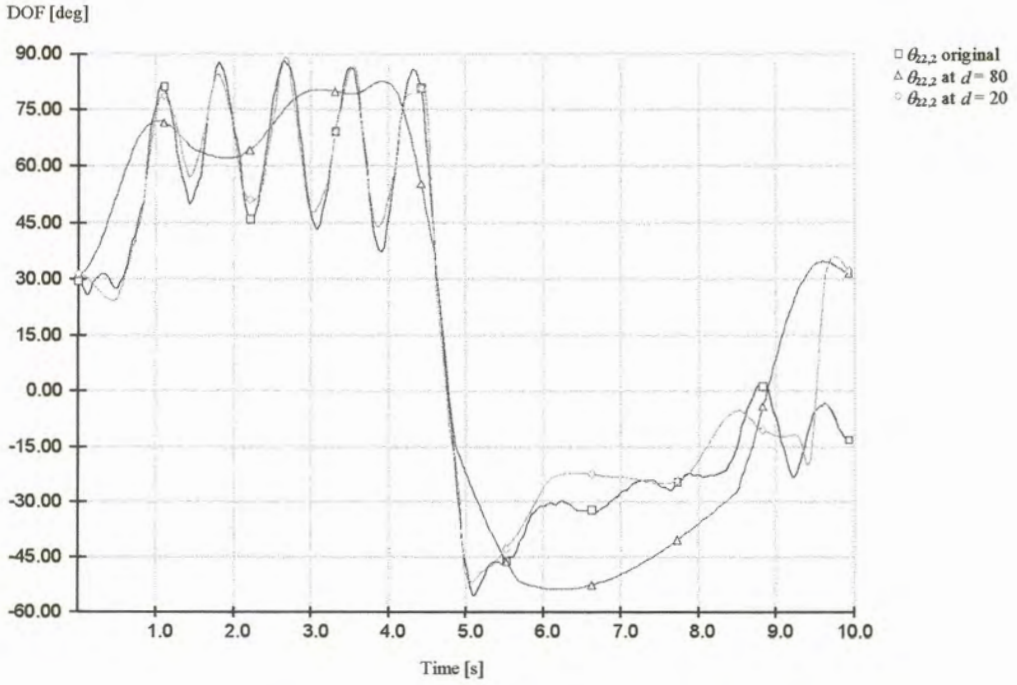


Figure 8-6a: Posture based coding of $\theta_{22,2}$ for the wave sequence.

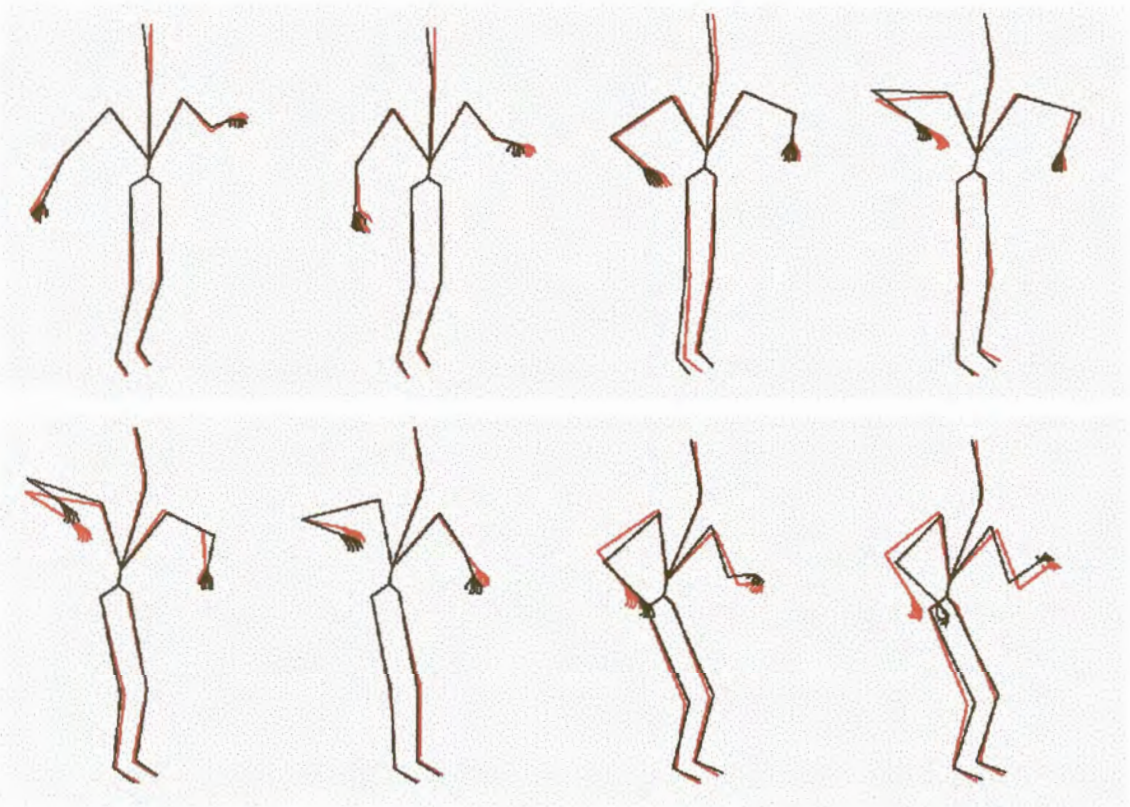


Figure 8-6b: Wireframe images from the dance sequence with $d = 20$.

Figure 8-7 shows the VPSNR against bit-rate for the various test sequences. The gesture example contains only compression results for the right hand, hence the lower bit-rate. Note the extreme change of bit-rate scale compared to the waveform coding examples. It can be seen that model based coding outperforms the previous methods by almost two orders of magnitude. In addition, a much larger variation in bit-rates is observed between the test sequences compared to the waveform coding techniques. The large variation indicates the high sensitivity of model based coding to the type of motion that is being coded. The highly active dance sequence requires many more bits than the passive conversational sequence. It is also clear from Figure 8-7 that not much is visually gained at higher bit-rates. This is primarily due to the sensitivity of the motion interpolation process to quantization errors. As the samples become more closely spaced, the vector quantization effects of the posture table cause significant and frequent errors in the slope estimation. A poor slope estimate in turn introduces overshoot in the interpolation function, and the resulting high-frequency errors are highlighted immediately by the formulation of the VPMSE.

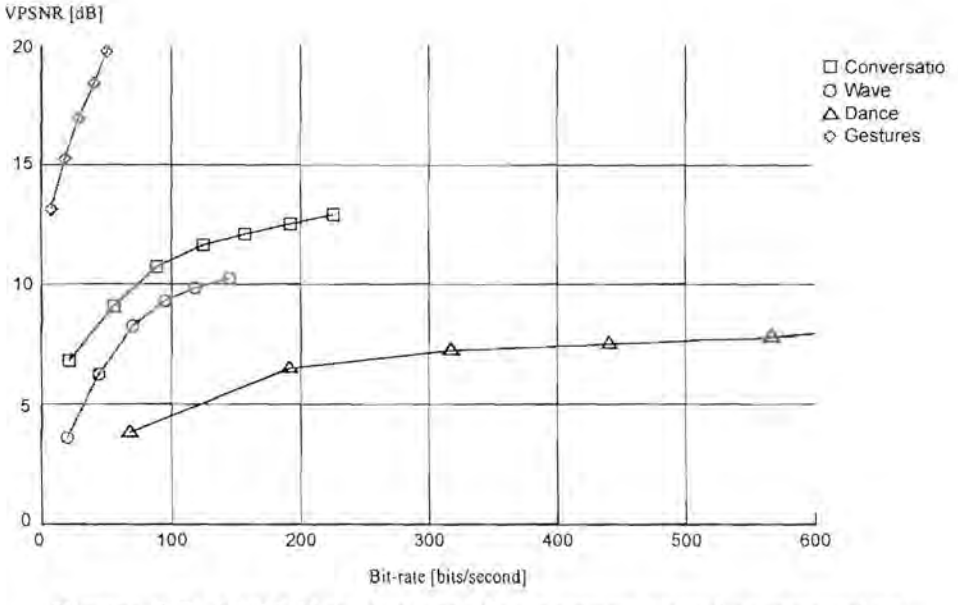


Figure 8-7: VPSNR vs. bit-rate for posture based compression.

8.4.4 Gesture based compression

We define gesture based compression in a similar fashion as posture based compression, except that gesture based segmentation is used, followed by gesture quantization (using a gesture table) and possibly also posture quantization.

After segmentation, a group gesture sequence $\{\mathbf{g}'(n)\}$ is extracted from each segment for the sequence to be coded. The sequence $\{\mathbf{g}'(n)\}$ is normalized to the same length as the entries in the gesture table $\mathbf{G}_i$, and the closest entry is selected using equation (8-12). The resulting string of codes is passed to a statistical coder to obtain a symbol stream of optimal length. Decompression reverses the process by decoding the symbols and obtaining the corresponding motion segment from the gesture table. The method shown in Figure 8-5 and equation (8-9) is used to interpolate the motion between consecutive segments.

Although the gesture table will always return the entry that is closest given the similarity measure, the error is often still too large. The resulting blended motion is usually completely inappropriate in the context. Note that except in cases of severe misalignment, the motion is almost never unnatural, just inappropriate. The concept of inappropriateness is different from the continuous unnatural synthetic look of pure posture interpolation. However, if a certain error threshold e is exceeded, we have found that it is visually still more acceptable to dismiss the motion segment completely and to revert to posture interpolation. In practice, extra bits are required to convey such changes, but do not significantly change the performance of the algorithm.

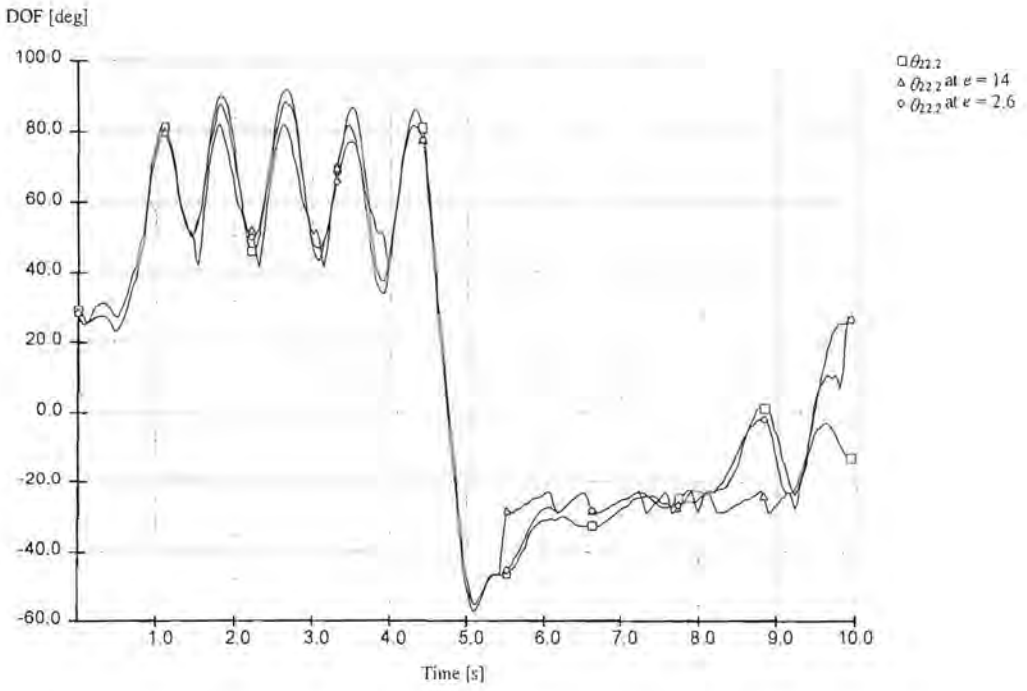


Figure 8-8a: Gesture based coding of $\theta_{22,2}$ for the wave sequence.

The gesture based algorithm also suffers from a variable coding delay, although not as severe as the posture based method due to the restrictions placed on the segmentation process. The known coding delay has again been compensated for in the following results. Figure 8-8a shows the results for two different similarity thresholds. The example shows the right upper arm twist angle $\theta_{22,2}$ for the wave sequence. For large values of the gesture/posture switch threshold e, the inappropriate choice of motion segments can be seen, especially during low activity. For smaller values of e, there is a clear improvement, albeit at the expense of a higher bit-rate. As is to be expected, Figure 8-8a shows more motion detail than Figure 8-6a, which is an inherent feature of using actual motion to fill in missing segments. Figure 8-8b depicts a number of 3D wireframe images from the dance sequence at $e = 5$, which corresponds to roughly 220 bits/second. The motion is coded at a lower rate, and follows the actual motion more closely compared to the posture based approach of figure 8-6b. These results are best viewed from the video clips provided on CD-ROM.

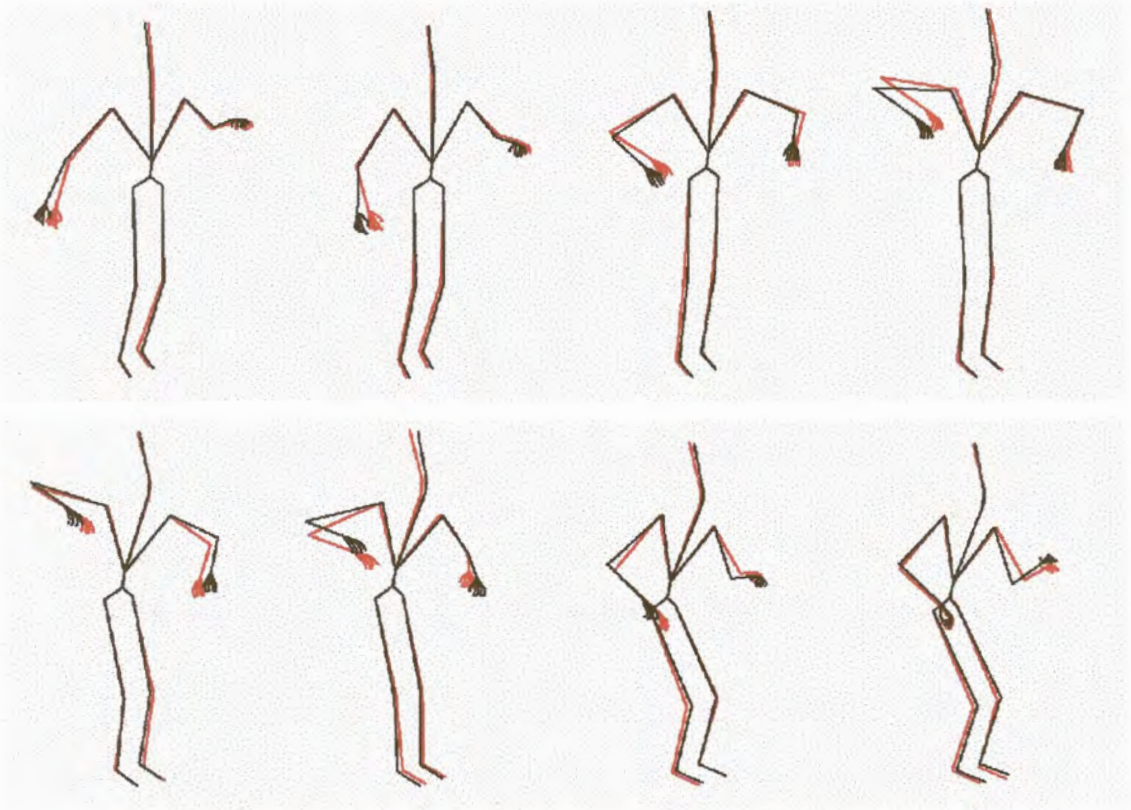


Figure 8-8b: Wireframe images from the dance sequence with $e = 5$.

Figure 8-9 shows the VPSNR against bit-rate for the various test sequences. Unfortunately, the gesture based segmentation is inherently fixed, and the spacing of non-uniform samples cannot be so conveniently adjusted as in the posture based segmentation scheme. The gesture based algorithm is also more complex than the posture based algorithm, and it is difficult to define a set of parameters that can be easily changed to produce a wide range of rate-distortion values. In any case, even the limited range in Figure 8-9 shows that the gesture based algorithm offers the same remarkable compression ratios as the posture based method. It exhibits the same sensitivity to variation in the type of motion, but there is a clear improvement in sensitivity to quantization noise. This is primarily due to the fact that there is no slope estimation requirement, and the use of the inherently more stable interpolation function of equation (8-9) compared to equation (8-5).

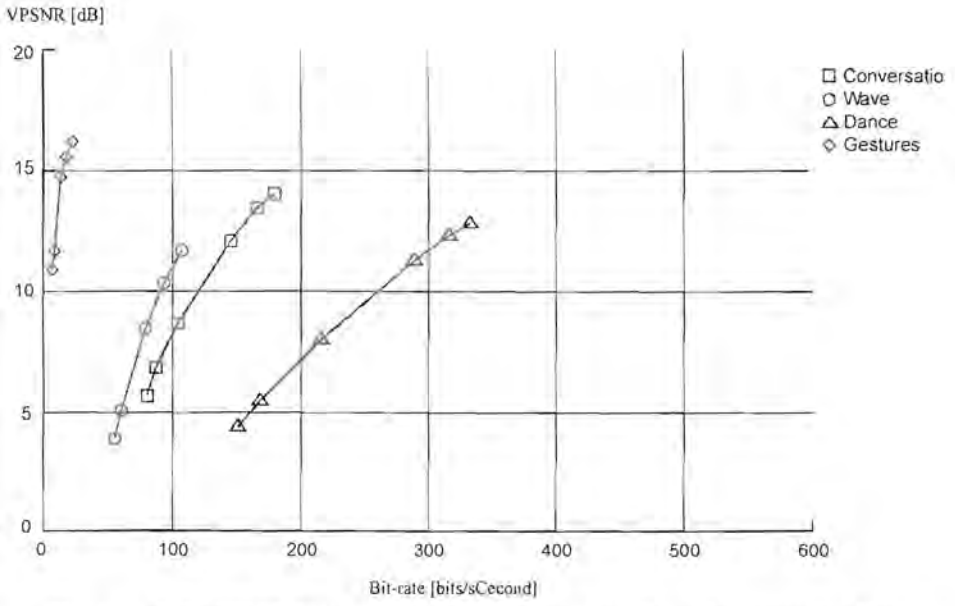


Figure 8-9: VPSNR vs. bit-rate for gesture based compression.

8.5 Hybrid coding

Model based coding methods, and especially motion interpolation between non-uniform samples, result in very high compression ratios. The price that we pay is that the result often does not resemble the original motion at all, and the animation has an unnatural and synthetic look to it. One way to overcome this would be to sacrifice a drop in compression ratio and to combine some of the waveform techniques from the previous chapter with model based methods. Obviously there are numerous ways to accomplish this combination. For example, one can simply use waveform coding techniques on groups where high accuracy is desired and model based techniques on the remaining groups. Visual artifacts in the root and torso group propagate through the whole figure, and high accuracy waveform coding on this group would be beneficial. Another example would be to continuously switch and blend on a temporal level between waveform coding and model based coding according to some rule. If the concept of virtual humans is extended to a whole virtual environment inhabited by other virtual entities and objects, interaction and collisions might force the use of higher resolution coding to resolve ambiguities.

There are endless possibilities and combinations of model based and waveform coding methods, most of which are quite trivial to implement, once the compression basis has been set. Rather than stating the obvious, we develop a more obscure combination of gesture based segmentation with transform coding of segments. It has been seen in chapter 7 that the DCT based method provides very efficient compression, but is plagued by block start/end mismatch effects. In model based coding, segment mismatch effects have been eliminated quite successfully through interpolation and blending techniques. Furthermore, it has been shown that the raised cosine interpolation function of equation (8-9) gives very natural looking results when combined with the gesture based segmentation method. This is primarily due to the fact that the first derivative (or speed) of the coded DOF is very small at the sampling points compared to the whole speed range. This raises the question of how well Fourier techniques can be used to represent motion, and what role each frequency component plays. Refer to [58] and [59] for an interesting discussion on such techniques.

The methods of DCT coding and gesture based segmentation have already been discussed, and will not be repeated here. In order to combine the methods, the DOFs of a group are segmented on a gesture level using the scalar speed method of equations (8-2) and (8-3) to obtain a segment of N samples. The segment is transformed using the DCT, which results in N frequency components. Note that N might not be a power of two and therefore an efficient DCT algorithm could not be used. The first three frequency components are retained, and each is quantized to a predetermined number of levels. The resulting codes are sent to a statistical coder, from which the optimal symbol stream can be transmitted. The decoder simply performs the inverse operations, and the contents of the segment can be obtained. By performing these steps, it is immediately clear that the more animation friendly non-uniform sampling method reduces the visual artifacts of the DCT block effects considerably, and it is possible to use fewer frequency components with coarser quantization. Since the segmentation cannot always be perfect, there are occasional jerks in the motion. We simply use the blending concept shown in figure 8-5 to blend the first few samples of the decoded segment with the last sample of the previous segment. The result is smooth motion throughout. The same variable coding delay issues are present here, but

since the waveform based DCT methods also have a (fixed) coding delay, not too much is sacrificed.

Figure 8-10 shows the results of the hybrid technique for the right upper arm twist angle $\theta_{22,2}$ for the wave sequence. Two possible transform coefficient bit allocations are indicated, one for low quality low bit-rate and one for high quality (the exact quantization scheme is detailed in Appendix III). It is quite clear that the results are not comparable with higher accuracy DCT methods, but the resulting bit-rate is substantially lower. We therefore still have to use the VPSNR measure as discussed below. The visual results are similar or better than the equivalent pure model based approaches. The benefits of such a hybrid method is that it is much more robust and general than some of the model based techniques. There is no requirement for extensive motion specific training sequences, and the hybrid coder is able to resolve some of the finer detail that is missing in more synthetic approaches.

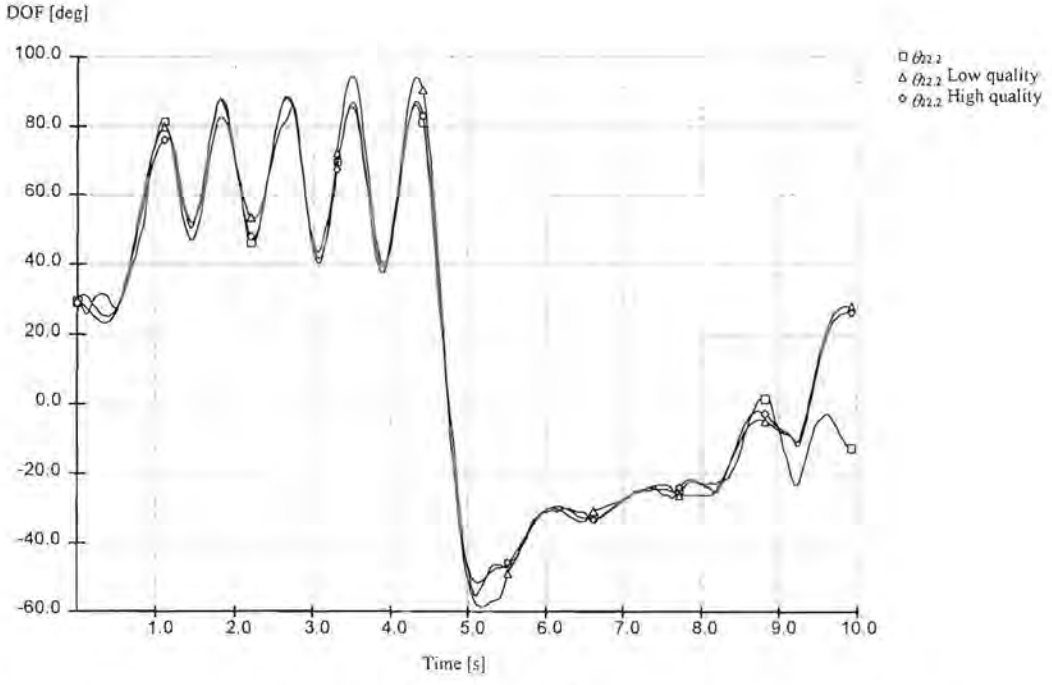


Figure 8-10: Hybrid coding of $\theta_{22,2}$ for the wave sequence.

Figure 8-11 shows the VPSNR against bit-rate for the various test sequences. Although the algorithm has its benefits, the information that needs to be sent is substantially more. Additionally, the information is on a per-DOF basis, and not in vectorized format. The expected drop in compression ration can clearly be seen in figure 8-11, compared to figures 8-7 and 8-9. The performance of the algorithm lies roughly halfway between equivalent model based and waveform based methods.

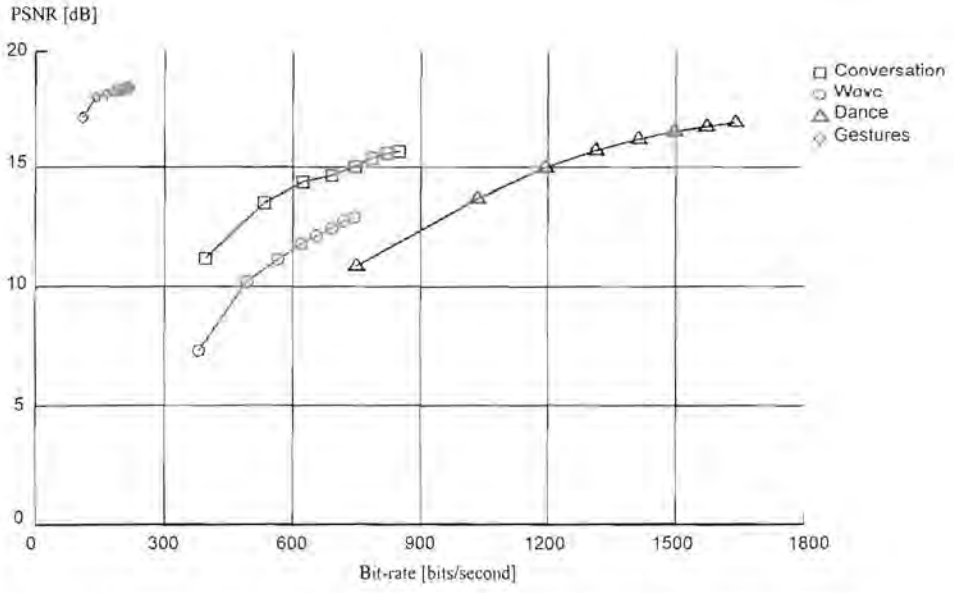


Figure 8-11: VPSNR vs. bit-rate for hybrid compression.

8.6 Summary

This chapter presented two model based compression methods as well as a hybrid model/waveform compression technique. The concept of non-uniform sampling or segmentation was introduced. Two segmentation methods were developed, namely posture or spatial based segmentation and gesture or temporal based segmentation. A combination of these segmentation methods with posture and gesture quantization tables led to the development of highly effective compression algorithms. Although the resulting motion animation from strictly model based compression techniques is very synthetic, there is no doubt about the underlying conveyance of information. A more robust hybrid compression

technique was also developed by combining transform based waveform coding with model based segmentation.

Chapter 9 Comparison and discussion

The previous chapters presented the compression results by type of *coding* process. A comparison on the performance of a specific technique could therefore be evaluated for the various test sequences. Once a certain type of motion has been established it is also desirable to compare the performance of various coding methods against each other. This chapter presents the compression results by type of *motion*. Unfortunately the waveform and model based approaches cannot be directly compared, as the MS error measurement is used for the first, and the VMS error for the latter. However, a lot of insight can still be gained by comparing various methods, both objectively and subjectively. Naturally, subjective comparison can only be done properly by evaluating the video clips provided on CD-ROM.

9.1 Waveform coding comparison

Figure 9-1a to 9-1d show the PSNR against bit-rate for the various waveform coding algorithms. Although there is some overlap, a clear distinction can be observed between the performances of the various methods. In general, the dead-reckoning and straight quantization algorithms do not perform very well. The adaptive predictive method is a good choice for a low complexity, low delay method that yields high signal-to-noise ratios at moderate bit-rates. If higher complexity and coding delay is tolerable, the DCT based method clearly outperforms all the other waveform coding techniques. The temporal vector quantization method shows interesting behaviour, but the underlying lack of generality of the method makes it undesirable. Spatial vector quantization performs well at very low bit-rates, but the almost non-existent signal-to-noise ratio renders it useless as a naive waveform coding method. However, note the interesting behaviour for the gesture sequence in figure 9-1d. This can be explained by the high spatial correlation that exists between the finger joints in general gesturing motion.

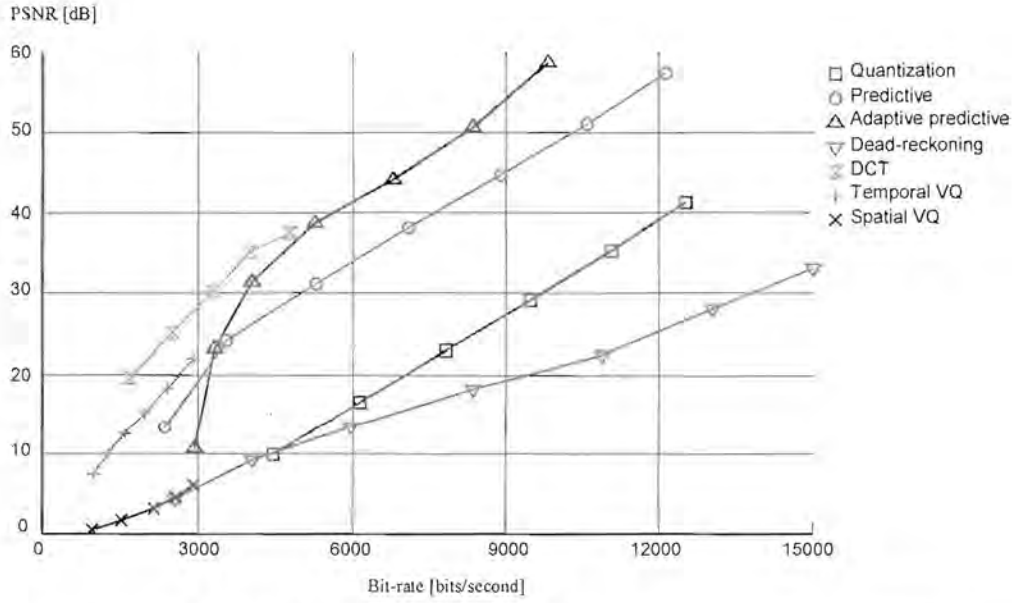


Figure 9-1a: PSNR vs. bit-rate for the conversational sequence.

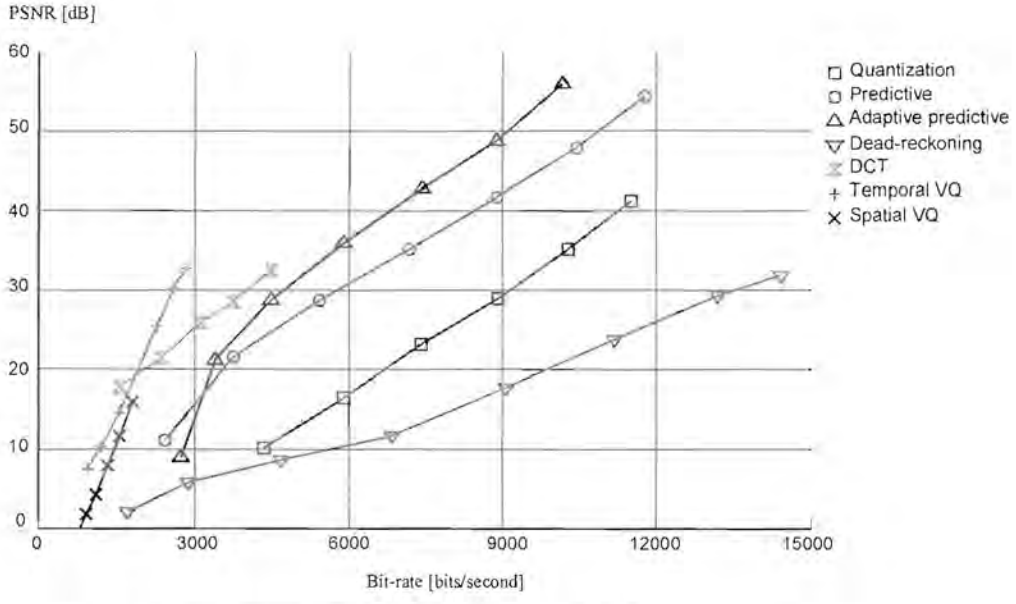


Figure 9-1b: PSNR vs. bit-rate for the wave sequence.

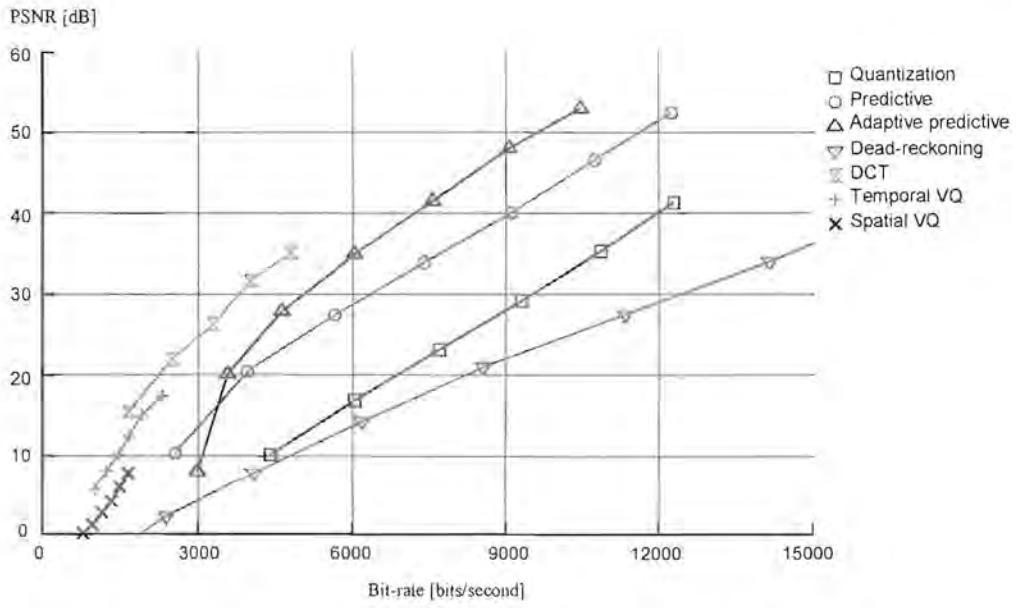


Figure 9-1c: PSNR vs. bit-rate for the dance sequence.

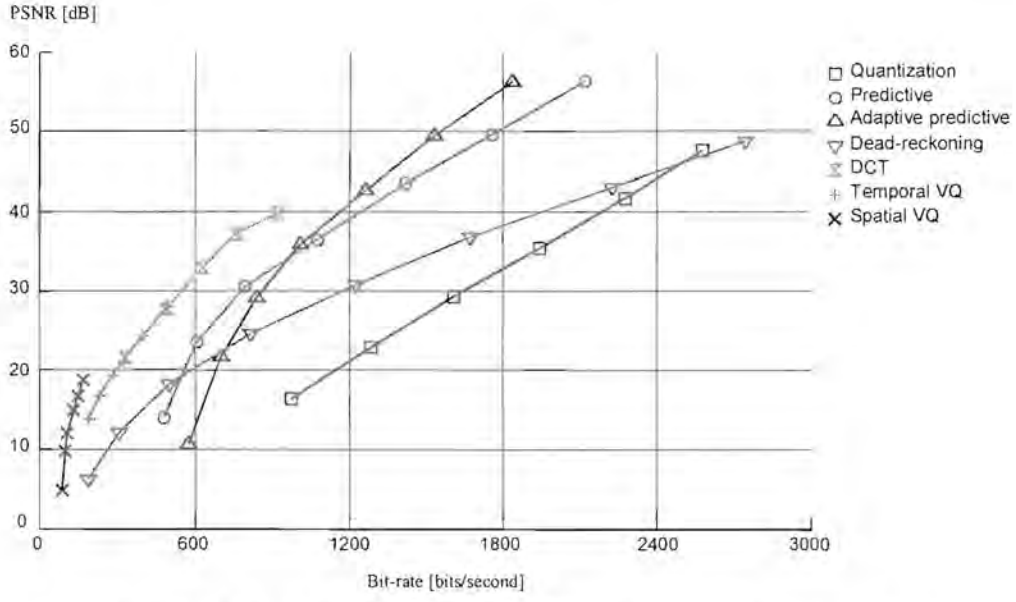


Figure 9-1d: PSNR vs. bit-rate for the gesture sequence.

At a glance, it can be seen that the waveform coding methods are surprisingly insensitive to type of motion. This is explained by the fact that most of the methods are based on uniform sampling techniques, and that the amount of "information" stays relatively constant from sample to sample. An exception of course is dead-reckoning, and it is indeed clear that this technique exhibits the most variation.

9.2 Model based comparison

Figures 9-2a to 9-2d show the VPSNR against bit-rate for the model based and hybrid coding algorithms. For all the test sequences there is a substantial increase in bit-rate for the hybrid method. This is to be expected, since the algorithm is more generalized and requires more information to be sent. The posture based and gesture based methods occupy more or less the same working range, and they intersect at a point within this range. However, at the extremes there are radical differences. The gesture based approach is in general more behaved, but it cannot cover the bit-rate range of the posture based approach. This is due to the fact that the non-uniform sampling process of the gesture based segmentation method is inherently rigid, and cannot be changed without sudden and serious degradation in the sampling process. The sampling process of the posture based segmentation can easily be changed by simply adjusting the sampling distance, as explained in the previous chapter. The result is that the posture based approach can achieve extraordinary high compression ratios, but the resulting motion is completely synthetic and unnatural. At low compression ratios, the posture based method suffers severely from quantization errors, and little is gained by using the coder in this range. The gesture based methods are much less sensitive to quantization errors. This is evident in the hybrid technique, where there is not much drop in signal-to-noise ratio for low bit-rates (i.e. coarse quantization).

All three of the model based compression algorithms presented in the previous chapter are quite useful. The posture based method can be used at extremely low bit-rates, provided that synthetic motion is acceptable. The gesture based method also gives excellent compression ratios, and although the motion might not always closely resemble the original, it appears smooth and natural. The compression of the hybrid method is not as substantial, but there is a gain in generality and robustness. These two aspects are a valid concern, especially compared to the gesture based segmentation, which on occasion can fall apart completely for peculiar motion behaviour.

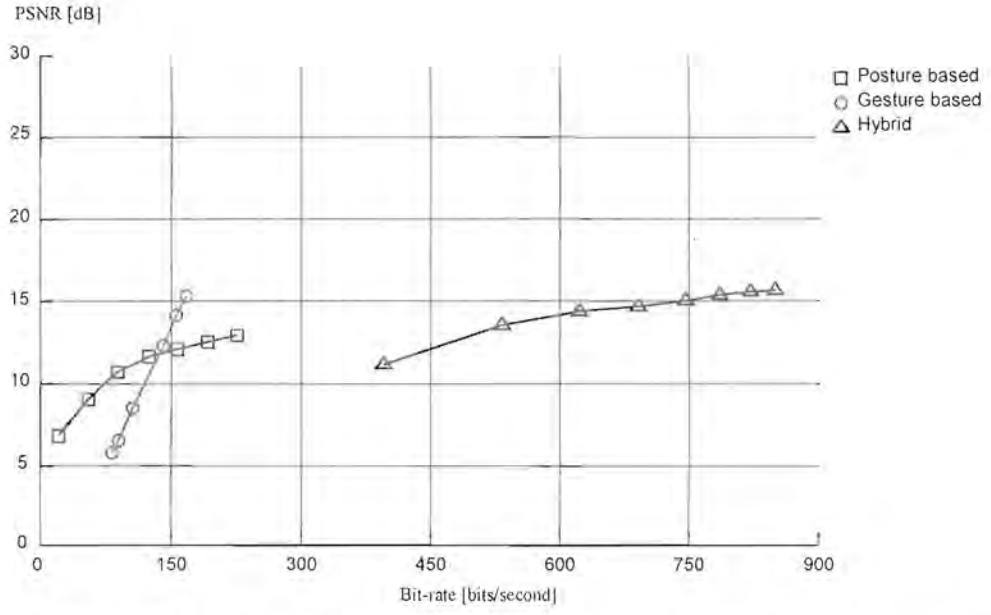


Figure 9-2a: VPSNR vs. bit-rate for the conversational sequence.

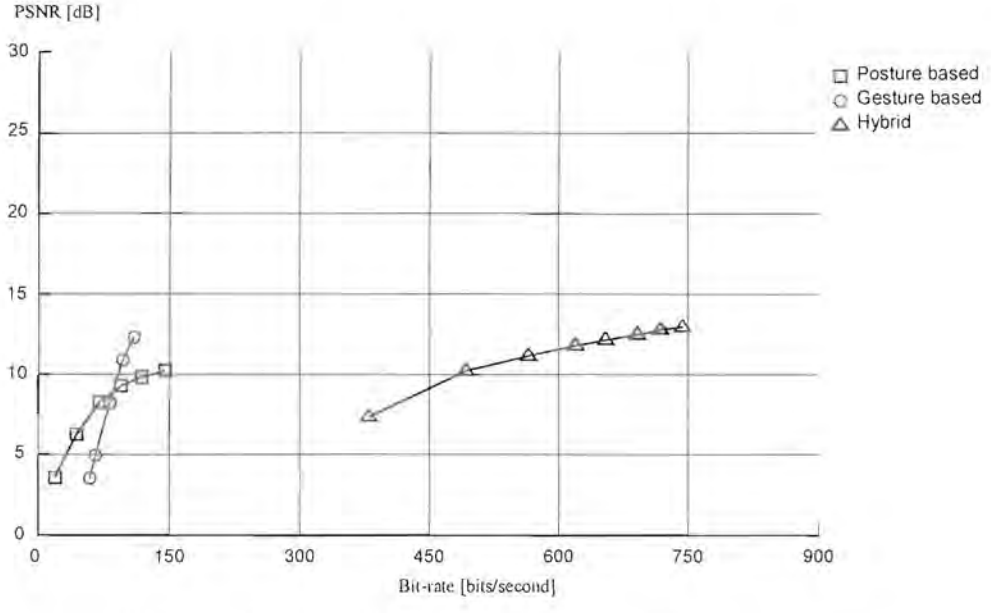


Figure 9-2b: VPSNR vs. bit-rate for the wave sequence.

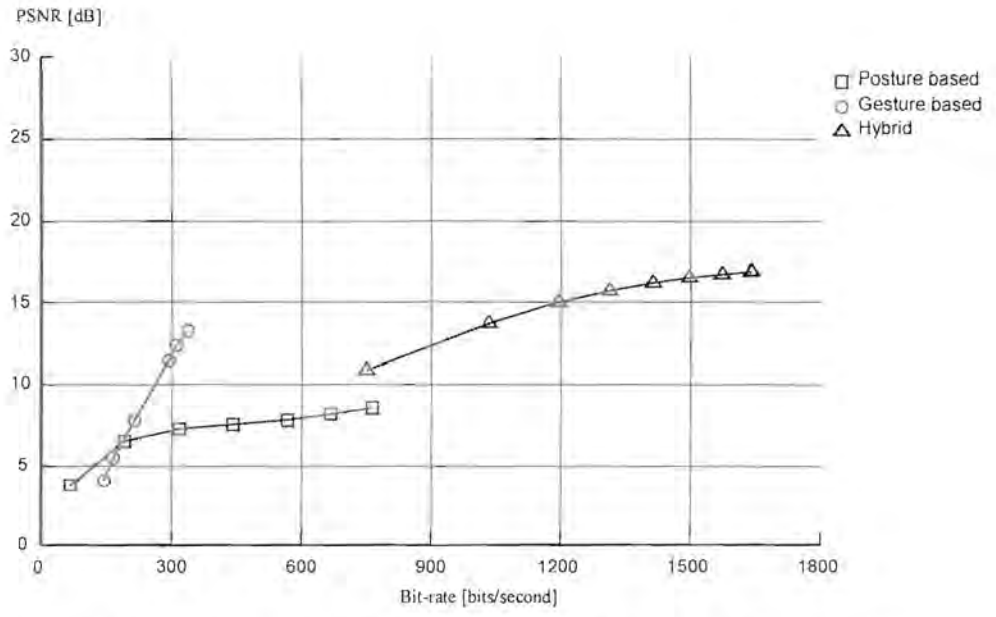


Figure 9-2c: VPSNR vs. bit-rate for the dance sequence.

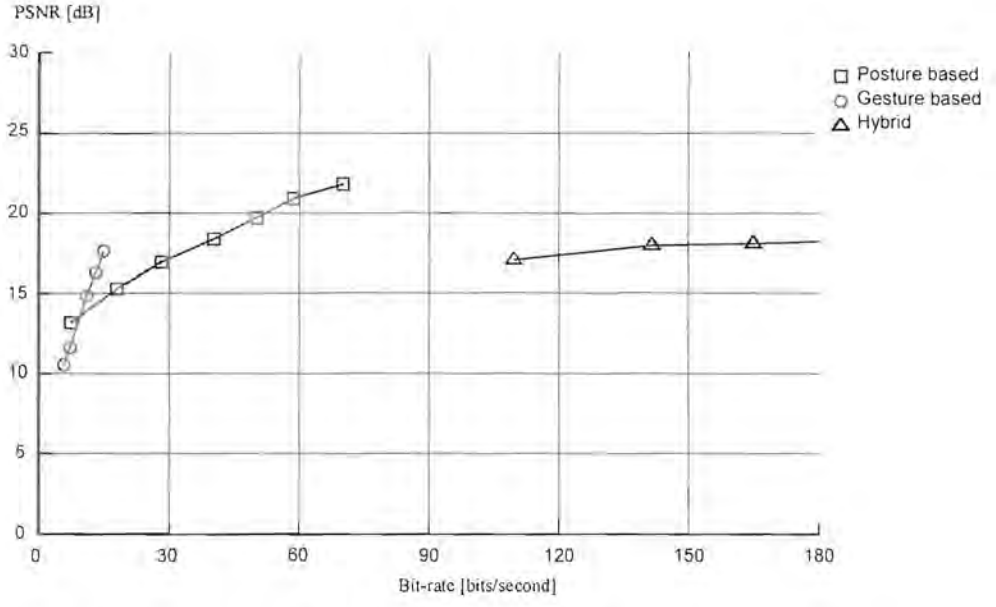


Figure 9-2d: VPSNR vs. bit-rate for the gesture sequence.

Clearly there is much more variation in the bit-rate spread between the various types of motion compared to the waveform coding methods. This is explained by the fact that the non-uniform sampling process or segmentation is highly dependent on the motion characteristics. For both the posture and gesture based methods there are frequent sampling during high activity and much less during low activity. This can clearly be seen for the

dance sequence in figure 9-2c, which occupies more than twice the bit-rate range of the preceding graphs. It is possible to adjust the waveform coding algorithms to produce constant rate bit streams, but the model based methods are strictly variable rate techniques, and suggest more complex network protocols and hardware.

9.3 Information entropy

The question "what is the entropy of the signal?" is invariably asked. In 1948, Shannon [60] published the famous paper, "A Mathematical Theory of Information". The foundation of Shannon's work rests on the concept of *entropy*. Briefly, the average information of a source is specified by its entropy, which is defined as

$$H(\theta) = -\sum_{i=1}^{N} P(\theta_i) \log_2 P(\theta_i),$$

where θ is some vector representing motion (in the context of this thesis), such as a temporal sequence or a spatial posture. There are N possible states of θ, denoted by all the θ_i's, each with an *a priori* probability of occurrence $P(\theta_i)$. The entropy definition above conveniently uses a base-two logarithm, and therefore specifies a binary measurement in bits. According to Shannon's "noiseless coding theorem", it is theoretically possible to code a source of entropy $H(\theta)$ bits without distortion using $H(\theta)+\varepsilon$ bits, where ε is an infinitesimally small positive quantity. It is understood that *distortion* is used in the context of information, and it is not necessarily the same as motion or animation distortion.

Unfortunately it is virtually impossible to measure the information of human motion contained in $P(\theta_i)$ and N. This is a problem even in the well defined field of speech analysis, a research area in which posture, gesture and situation level motion information analysis lags far behind. The reason for this difficulty lies primarily in the inability to properly measure error or distortion, and its relation to information loss. However, the following reasoning gives an approximation for the entropy of hand gesture motion.

Consider a set of possible information symbols for a human hand to be made up of open and closed finger positions. We use the average flexure for each finger, which can be represented by five bits of information, i.e. open (binary 1) or closed (binary 0). From informal experiments we have found that a reasonably dexterous person can perform approximately three to four recognizable symbols per second. The entropy is therefore roughly 3*5 = 15 to 4*5 = 20 bits/second. The counting test sequence example that was depicted in figure 5-1c is slightly less frantic, and about 1.5 symbols (or gestures) are produced every second. Finger counting is universally understood, and it is reasonable to assume that no information is lost using only five bits. We therefore conclude that the entropy of the gesture sequence is roughly 1.5*5 = 7.5 bits/second. If this entropy value is compared to figure 9-1d, it is clear that the waveform coding techniques still have a long way to go before the entropy limit is reached. On the other hand, figure 9-2d paints a completely different picture. The gesture based method rapidly approaches the entropy limit, and even so the VPSNR is still quite respectable. Figure 9-3 shows symbols "one" to "four" (starting at the index finger) extracted from the video clip. There is no question about information loss, even if the animated motion is a little synthetic.

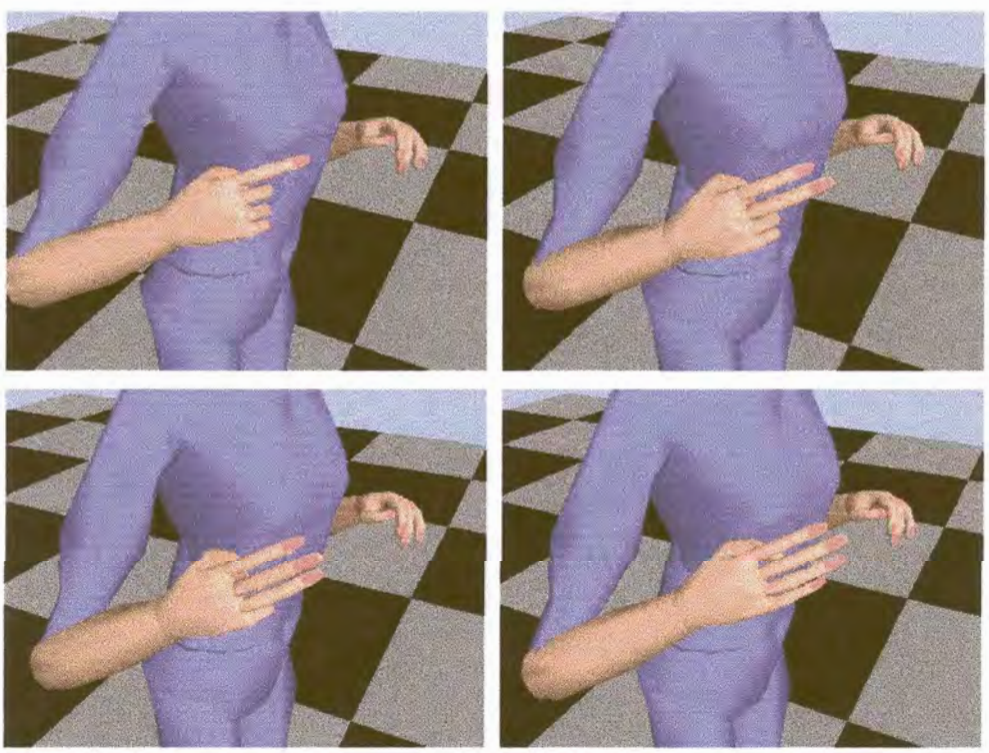


Figure 9-3: Symbols "one" to "four" for the gesture sequence.

Chapter 10 Conclusion

Future telecommunication applications will require the combination, interaction and transmission of natural audio and video streams with synthetic computer graphics models. Of these applications, those that employ aspects of virtual environments populated by virtual humans will play a key role. From the current state of applications it is clear that the insertion of highly detailed and articulated virtual humans into networked systems could place a burden on already taxed network resources. Research on networking virtual humans is still in its infancy. It would still be some time before the state of synthetic audio/visual object compression is on par with equivalent natural audio and video compression methods. Currently the research on facial parameter estimation, compression and transmission is well underway and its implementation in MPEG-4 well defined. The same cannot be said for full body motion methods.

Towards this end we have investigated and developed compression methods applicable to full body motion of virtual humans. A background study that puts the current state of virtual human research into context was presented. A virtual human model was introduced in terms of physical modeling, hierarchical modeling, surface or visual modeling and dynamic modeling. Human motion capture methods that employ inverse kinematic techniques to obtain high degree of freedom joint data, were developed. A detailed motion data analysis was presented, and it was shown that a general statistical model for human motion could be established for specific types of motion that included the properties of ergodicity and being wide sense stationary. The concept of a new visual coding error measurement that exhibits better performance than traditional quantitative methods was introduced. Waveform compression methods were used with good results as general, low complexity coding techniques. The bit-rate reduction that was obtained is comparable with results from research on facial motion done by others. Arguably these techniques are not

new, but the successful implementation on a "new" class of motion data is encouraging. Model based compression techniques based on a novel posture and gesture segmentation scheme were developed, and the results exhibited remarkable compression ratios with little loss in information. These techniques show the extreme importance of non-uniform sampling and segmentation methods as a preprocessing step for model based motion compression.

This study has clearly indicated the potential of human motion compression for communication purposes. Without compression, it would barely be possible to insert even two virtual humans into a networked virtual environment if current low-cost network technology is utilized. Using waveform compression techniques this number could be increased by a factor five. Model based compression techniques could increase it even further by a factor fifty or more, provided the loss of naturalness and subtlety is acceptable. This study contributes directly towards the current research of the SNHC group within the MPEG-4 standardization process, and brings the state of full body coding research closer to that of facial coding.

Future research

It is possible to adjust and refine the waveform coding methods to achieve better performance, but it is doubtful whether it will have dramatic effects on bit-rate requirements. Still, subjective testing by a panel of observers is required to optimize some of the coding parameters such as quantizer bit allocations and adaption settings. On the other hand, it is clear that model based methods show enormous potential for further research. Basic methods for segmentation and consequent segment animation were presented. Recent animation editing and transition methods based on combinations of motion capture, kinematic constraints and dynamic simulation can substantially improve the synthetic "feel" of these segment animation techniques. The use of higher level motion detection algorithms could also improve the non-uniform sampling methods and lower the bit-rate even more. However, care should be taken not to accidentally exceed the entropy limit (supposing that it can be measured). In human speech coding, intelligibility is clearly defined, even without the original speech. In contrast, it is possible to produce "coded"

human motion that does not resemble the original motion, yet it appears credible and natural. Lastly, further research can also benefit knowledge of the effects and implications of the variable coding delay and variable bit-rate characteristics of model based coding on real-time virtual environments.

References

[1] M. Slater and S. Wilbur, "A Framework for Immersive Virtual Environments (FIVE): Speculations on the Role of Presence in Virtual Environments", *Presence: Teleoperators and Virtual Environments*, 6(6) pp. 603-616, MIT Press, 1997.

[2] M. Usoh and M. Slater, "An exploration of immersive virtual environments", *Proc. IEEE Conf. VRAIS*, pp. 90-96, 1996.

[3] N.I. Badler, "Virtual humans for animation, ergonomics and simulation", *IEEE Workshop on Non-Rigid and Articulated Motion*, June 1997.

[4] M. Cavazza, R. Earnshaw, N. Magnenat-Thalmann and D. Thalmann, "Motion control of virtual humans", *IEEE Computer Graphics and Applications*, pp. 24-31, September/October 1998.

[5] T.K. Capin, H. Noser, I.S. Pandzic and N. Magnenat-Thalmann, "Virtual human representation and communication in VLNet", *IEEE Computer Graphics and Applications*, pp. 42-53, March/April 1997

[6] P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel and D. Thalmann, "Real-time animation of realistic virtual humans", *IEEE Computer Graphics and Applications*, pp. 42-55, September/October 1998.

[7] N.I. Badler, C.B. Phillips and B.L. Webber, *Simulating Humans: Computer Graphics, Animation and Control*, Oxford University Press, 1993.

[8] N.I. Badler, R. Bindiganavale, J. Bourne and J. Allbeck, "Real-time virtual humans", *International Conference on Digital Media Futures*, April 1999.

[9] C.A. Balafoutis and R.V. Patel, *Dynamic Analysis of Robot Manipulators: a Cartesian Tensor Approach*, Kluwer Academic Press, 1991.

[10] B. Paden, "Kinematics and Control of Robot Manipulators", *PhD. Thesis*, University of California, Berkeley, 1986.

[11] P.M. Isaacs and M.F. Cohen, "Controlling dynamic simulation with kinematic constraints", *Computer Graphics*, Proceedings of SIGGRAPH 1986.

[12] J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, 1990.

[13] P. Hanrahan and D. Sturman, "Interactive animation of parametric models", *The Visual Computer*, (1) pp. 260-266, 1985.

[14] WW. Armstrong and M.W. Green, "The dynamics of articulated rigid bodies for the purposes of animation", *Proceedings of Graphics Interface*, 1985.

[15] A. Witkin and M. Kass, "Spacetime constraints", *Computer Graphics*, pp. 159-168, Proceedings of SIGGRAPH 1988.

[16] C. Rose, B. Guenter, B. Bodenheimer and M.F. Cohen, "Efficient generation of motion transitions using spacetime constraints", *Computer Graphics*, pp. 147-154, Proceedings of SIGGRAPH 1996.

[17] Z. Popvic and A. Witkin, "Physically based motion transformation", *Computer Graphics*, pp. 11-20, Proceedings of SIGGRAPH 1999.

[18] K. Perlin, "Real-time responsive animation with personality", *IEEE Trans. on Visualization and Computer Graphics*, pp. 5-15, 1995.

[19] K. Perlin and A. Goldberg, "Improv: A system for scripting interactive actors in virtual worlds", *ACM Computer Graphics Annual Conf.*, pp. 205-216, 1996.

[20] J.K. Hodgins, P.K. Sweeney and D.G. Lawrence, "Generating natural-looking motion for computer animation", *Proceedings of Graphics Interface*, pp. 265-272, 1992.

[21] J.K. Hodgins, "Animating human motion", *Scientific American*, 278(3), pp. 64-69, 1998.

[22] J.K. Hodgins, W. Wooten, D. Brogan and J. O'Brien, "Animating human athletics", *ACM Computer Graphics, Annual Conf. Series*, pp. 71-78, 1995.

[23] M.H. Raibert and J.K. Hodgins, "Animation of dynamic legged motion", *Computer Graphics*, pp. 349-358, Proceedings of SIGGRAPH 1991.

[24] F. Azuola, N.I. Badler, P.-H. Ho, I. Kakadiaris, D. Metaxas and B. Ting, "Building anthropometry-based virtual human models", *Proc. IMAGE VII Conf.*, 1994.

[25] T. Sederberg and S. Parry, "Free-form deformation of solid geometric models", *ACM Computer Graphics*, 20(4), pp. 151-160, 1986.

[26] P. Kalra, "Simulation of facial muscle actions based on rational free-form deformations", *Computer Graphics Forum*, 2(3), pp. 65-69, 1992.

[27] J. Shen and D. Thalmann, "Interactive shape design using metaballs and splines", *Proc. Implicit Surfaces*, pp. 187-196, 1995.

[28] L. Moccozet and N. Magnenat-Thalmann, "Dirichlet free-form deformations and their application to hand simulation", *Proc. Computer Animation*, pp. 93-102, 1997.

[29] T.K. Capin, M. Jovovic, J. Esmerado, A. Aubel and D. Thalmann, "Efficient network transmission of virtual human bodies", *Proc. Computer Animation '98*, pp.41-48, 1998.

[30] M.R. Macedonia, M.J. Zyda, D.R. Pratt, P.T. Barham and S. Zeswitz, "NPSNET: A network software architecture for large scale virtual environments", *Presence*, 3(4), Fall 1994.

[31] M.R. Macdonida and M.J. Zyda, "A taxonomy for networked virtual environments", *Proceedings of the 1995 Workshop on Networked Realities*, Boston, MA, October 1995.

[32] M.J. Zyda, D.R. Pratt, P.T. Barham and J.S. Falby, "NPSNET-Human: Inserting the human into the networked synthetic environment", *Proceedings of the 13th DIS Workshop*, Orlando, Florida, pp.103-106, September 1995.

[33] M.S. Waldrop, S.M. Pratt, D.R. Pratt and R.B. McGhee, "Real-time upper body articulation of humans in a networked interactive virtual environment", *Proceedings of the First ACM Workshop on Simulation and Interaction in Virtual Environments*, University of Iowa, pp. 210-214, July 1995.

[34] L. Emering, R. Boulic and D. Thalmann, "Interacting with virtual humans through body actions", *IEEE Computer Graphics and Applications*, pp. 8-11, January/February 1998.

[35] T.K. Capin, J. Esmerado and D. Thalmann, "A dead-reckoning technique for streaming virtual human animation", *IEEE Transactions on Circuits and Systems for Video Technology*, 9(3), pp. 411-414, April 1999.

[36] H. Tao, H.H. Chen, W. Wu and T.S. Huang, "Compression of MPEG-4 facial animation parameters for transmission of talking heads", *IEEE Transactions on Circuits and Systems for Video Technology*, 9(2), pp. 264-276, March 1999.

[37] L. Sirovich and R. Everson, "Management and analysis of large scientific datasets", *Int. J. Supercomput. Appl.*, 6(1), pp. 50-68, Spring 1992.

[38] P. Doenges P, T.K. Capin, F. Lavagetto, J. Osterman, I.S. Pandzic and E.D. Petajan, "MPEG-4: Adio/video & synthetic graphics/audio for mixed media", *Signal Process.: Image Comm.*, 9(4), pp. 433-464, May 1997.

[39] "Text for CD 14496-2 video", ISO/IEC JTC1/SC29/WG11 N1902, November 1997.

[40] P. Ekman and W.V. Friesen, *Facial Action Coding System*, Palo Alto, CA: Consulting Psychologists Press, 1977.

[41] V. Zatsiorsky, V. Seluyanov *et al.*, *Contemporary Problems in Biomechanics*, pp. 272-291, CRC Press, Massachusetts, 1990.

[42] P. de Leva, "Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters", *J. of Biomechanics*, 29(9), 1223-1230, 1996.

[43] D.J. Pearsall, J.G. Reid and R. Ross, "Inertial properties of the human trunk of males determined from magnetic resonance imaging", *Annals of Biomed. Eng.*, 22, pp. 692-706, 1994.

[44] R.G. Burdett, G.S. Skrinar and S.R. Simo, "Comparison of mechanical work and metabolic energy consumption during normal gait", *J. of Orthopedic Research 1*, 1, pp. 63-72, 1983.

[45] R. Azuma and G. Bishop, "A frequency-domain analysis of head motion prediction", *Proc. ACM SIGGRAPH*, 1995.

[46] J. Craig, *Introduction to Robotics*, Addison-Wesley, 1989.

[47] S.M. Kay, *Modern Spectral Estimation*, Prentice-Hall, 1988.

[48] J. Max, "Quantizing for minimum distortion", *IRE Trans. Inf. Theory*, IT-6, 1, pp. 7-12, March 1960.

[49] Institute of Electrical and Electronic Engineers, International Standard, ANSI\IEEE Standard 1278-1993, Standard for Information Technology, Protocols for Distributed Interactive Simulation, March 1993.

[50] K.R. Rao and P. Yip, *Discrete Cosine Transform. Algorithms, Advantages and Applications*, Academic Press, San Diego, 1990.

[51] R.M. Gray, "Vector Quantization", *IEEE ASSP Magazine*, 1(2), pp. 4-29, April 1984.

[52] A.D. Wilson and A.F. Bobick, "Recognition and interpretation of parametric gesture", *Proc. Int. Conf. on Computer Vision*, 1998.

[53] T. Starner and A. Pentland, "Visual Recognition of American Sign Language using Hidden Markov Models", *International Workshop on Automatic Face- and Gesture- Recognition*, pp. 189-194, Zurich 1995.

[54] L.W. Campbell and A.F. Bobick, "Recognition of human motion using phase space constraints", *Proc. Int. Conf. on Computer Vision*, 1995.

[55] A.F. Bobick, A.D. Wilson and J. Cassell, "Temporal classification of natural gesture and application to video coding", *Proc. Comp. Vis. and Pattern Rec.*, 1997.

[56] C. Vogler and D. Metaxas, "ASL recognition based on a coupling between HMMs and 3D motion analysis", *Proc. Of the ICCV'98*, pp.363-369, 1998.

[57] N. Yanghee and K. Wohn, "Recognition of space-time hand gestures using hidden markov models", *ACM Symposium on Virtual Reality Software and Technology*, pp. 51-58, July 1996.

[58] A. Bruderlin and L. Williams, "Motion signal processing", *Computer Graphics*, 30, pp. 97-115, Proceedings of SIGGRAPH 1996.

[59] M. Unuma, K. Anjyo and R. Takeuchi, "Fourier principles for emotion-based human figure animation", *Computer Graphics*, 29, pp. 91-96, Proceedings of SIGGRAPH 1995.

[60] C.E. Shannon, *The Mathematical Theory of Communication*, University of Illinois Press, 1949.

[61] D. Tolani and N.I. Badler, "Real-time inverse kinematics of the human arm", *Presence*, 5(4), pp. 393-401, 1996.

[62] C.B. Phillips, J. Zhao and N.I. Badler, "Interactive Real-time Articulated Figure Manipulation Using Kinematic Constraints", *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, 24(2), pp. 245-249, March 1990.

[63] R. Bindiganavale and N.I. Badler, "Motion abstraction and mapping with spatial constraints", *Workshop on Motion Capture Technology*, November 1998.

[64] N.I. Badler, R. Bindiganavale, J.P. Granieri, S. Wei and X. Zhao, "Posture interpolation with collision avoidance", *Computer Animation '94*, Geneva, Switzerland, pp. 13-20, 1994.

[65] E. Kokkevis, D. Metaxas and N.I. Badler, "User-controlled physics based animation for articulated figures", *Proc. Comp. Animation*, 1996.

[66] R. Boulic, P. Bécheiraz, L. Emering and D. Thalmann, D., "Integration of motion control techniques for virtual human and avatar real-time animation", *Proc. ACM Int. Symp. VRST'97*, pp. 111-118, 1997.

[67] A. Lamouret and M. van de Panne, "Motion synthesis by example", *7th Eurographics Workshop on Animation and Simulation*, 1996.

[68] H. van der Elst and J.J.D. van Schalkwyk, "Data compression in distributed virtual environments", *Proc. 4th Africon Conf.*, pp. 1115-1118, September 1996.

[69] H. van der Elst and J.J.D. van Schalkwyk, "Coding of sampled human motion for the IS-95 standard", *Proc. ISSSTA*, pp. 748-751, September 1998.

[70] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 1984.

[71] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[72] J.R. Deller, J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing Company.

[73] L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall.

[74] 3SPACE InsideTrak User's Manual, Polhemus Inc., Doc. No. OPM3792-001, December 1993.

[75] 5DT Data Glove User's Manual, 5DT <Fifth Dimension Technologies>, 1995.

A. Appendix I: Definitions and notation

Coordinate system

A left handed world coordinate system is used throughout this document. Using the left hand, pointing forward with the index finger and upward with the thumb, and bending the middle finger to the right, the x, y, and z axis are formed. This is shown graphically in figure A-1.

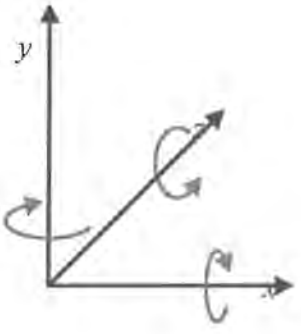


Figure A-1: Left handed coordinate system

The curl of the left hand fingers shows the direction of positive rotation with the thumb pointing in the direction of the desired axis. Figure A-1 also depicts the positive angle direction around each axis.

Vectors

A vector space consists of a set of elements, called *vectors*, together with the addition operator and the scalar multiplication operation. In this text we will use the vector space $\mathbf{R}^3$ or $\mathbf{R}^4$, the set of all ordered 3 or 4-tuples of real numbers. A vector is denoted by boldface letters, such as **u**, **v** or **w**. A vector in $\mathbf{R}^3$ consists of three elements, which is conveniently

denoted by x, y and z. Vectors in $\mathbf{R}^4$ contains one additional component. Vector components are written horizontally in row format as

$[x\ y\ z]$

which differs from some texts [65] which uses the vertical column format method

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ or } [x\ y\ z]^{\mathrm{T}},$$

The following vector operations are used in this text:

Multiplication by a scalar:	$\mathbf{v} = \mathbf{u}s$
Addition:	$\mathbf{w} = \mathbf{u}+\mathbf{v}$
Dot product:	$s = \mathbf{u}\cdot\mathbf{v}$
Cross product:	$\mathbf{w} = \mathbf{u}\times\mathbf{v}$
Vector length:	$s = \|\mathbf{u}\|$
Vector transpose:	$\mathbf{v} = \mathbf{u}^{\mathrm{T}}$

Matrices

A matrix is a rectangular array of real values. A matrix is denoted by a boldface capital letter such as **A**, **B** or **M**. Its elements are doubly indexed, with the first index indicating the row and the second index the column. A 4x4 matrix is therefore written as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

The following matrix operations are used in this text:

Matrix multiplication: $\mathbf{C} = \mathbf{AB}$

Matrix inverse: $\mathbf{B} = \mathbf{A}^{-1}$

Matrix transpose: $\mathbf{B} = \mathbf{A}^{T}$

Vector-matrix multiplication: $\mathbf{v} = \mathbf{uA}$

Matrix-vector multiplication: $\mathbf{v}^{T} = \mathbf{Au}^{T}$

The *identity matrix* is denoted by **I**. The *homogeneous transformation matrix* is a special 4x4 matrix that has the following form:

$$\mathbf{A} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & 0 \\ r_{yx} & r_{yy} & r_{yz} & 0 \\ r_{zx} & r_{zy} & r_{zz} & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix},$$

or more compactly

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{t} & 1 \end{bmatrix},$$

where $\mathbf{R}$ is a 3x3 rotation matrix, $\mathbf{t}$ is a translation vector in $\mathbf{R}^3$ and $\mathbf{0}$ is the zero vector [0 0 0]. The 1x3 row vectors of $\mathbf{R}$ are orthonormal and contains the x, y, and z axis of rotation respectively. If and only if we are using a transformation matrix, the inverse is conveniently given by

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{R}^{T} & \mathbf{0} \\ -\mathbf{tR}^{T} & 1 \end{bmatrix}.$$

The notation $\mathbf{R}(\mathbf{u}, \theta)$ or $\mathbf{R}(u_x, u_y, u_z, \theta)$ defines a rotation matrix around the axis $\mathbf{u}$ of θ degrees and is given by

$$\mathbf{R}(\mathbf{u},\theta)=\begin{bmatrix} u_x^2+\mathrm{c}(1-u_x^2) & u_xu_y(1-\mathrm{c})+u_z\mathrm{s} & u_xu_z(1-\mathrm{c})+u_y\mathrm{s} & 0 \\ u_xu_y(1-\mathrm{c})+u_z\mathrm{s} & u_y^2+\mathrm{c}(1-u_y^2) & u_yu_z(1-\mathrm{c})+u_x\mathrm{s} & 0 \\ u_xu_z(1-\mathrm{c})-u_y\mathrm{s} & u_yu_z(1-\mathrm{c})+u_x\mathrm{s} & u_z^2+\mathrm{c}(1-u_z^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where c = cos(θ) and s = sin(θ). If we rotate around one of the primary axis x, y or z, a convenient short hand notation is $\mathbf{R}(1, 0, 0, \theta) = \mathbf{R}_x(\theta)$, $\mathbf{R}(0, 1, 0, \theta) = \mathbf{R}_y(\theta)$ and $\mathbf{R}(0, 0, 1, \theta) = \mathbf{R}_z(\theta)$.

Similarly, the notation $\mathbf{T}(\mathbf{u})$ or $\mathbf{T}(u_x, u_y, u_z)$ defines a translation matrix and is given by

$$\mathbf{T}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ u_x & u_y & u_z & 1 \end{bmatrix}.$$

Variables and equations

General variables or degrees of freedom (DOFs) are represented by the symbol θ. A specific DOF is specified by the symbol $\theta_{i,j}$, where i is a zero-based integer representing either a joint or a group number, depending on the context. The zero-based integer j represents the DOF within the joint or group. The subscript i,j is sometimes omitted for clarity, in which case it can be assumed that θ represents *any* DOF. DOFs are functions of time, and can be written as $\theta(t)$. Since we are dealing with discrete time sampled quantities in practice, a *single* sample is written as $\theta(n)$. A *sequence* of samples is written as $\{\theta(n)\}$. For convenience, some expressions use a continuous time representation, while others use a discrete time representation.

Proofs

The following terminology is helpful:

- Proof by mutual consent

"We can show that..."

- Proof by vagueness

"It can be shown that..."

- Proof by intimidation

"It is understood that..."

B. Appendix II: Dynamic simulation

The purpose of dynamic simulation is to generate realistic animation using forces and torques calculated from physical laws. The process can be divided into two parts, namely forward dynamics and inverse dynamics. Forward dynamics simply use the forces and torques at each joint to *generate* motion. Inverse dynamics *calculate* the required forces and torques to generate the desired motion. These two processes are therefore related and dependent on each other. The forward and inverse dynamics problem has been extensively studied in the robotics field. See [69] for a good overview of these algorithms. Dynamic formulations range from non-recursive $O(N^4)$ algorithms to recursive $O(N)$ algorithms, where N is the number of DOFs. An example of an efficient recursive solution is the Armstrong-Green algorithm [10,69], which is based on the Newton-Euler formulations of motion. This algorithm is summarized as follows:

Scalars

m_i Mass of the ith segment

nd_i Number of DOFs at the ith joint

$\dot{\theta}_{i,j}$ First derivative of the jth DOF of the ith joint

$\ddot{\theta}_{i,j}$ Second derivative of the jth DOF of the ith joint

Vectors

ω^i Angular velocity of the *i*th segment

α^i Angular acceleration of the *i*th segment

$\mathbf{a}^i$ Linear acceleration of the *i*th segment at its pivot point

$\mathbf{a}_c^i$ Linear acceleration of the *i*th segment at its center of mass

$\mathbf{g}$ Acceleration due to gravity [0 –10 0] m/s^2

$\mathbf{g}_i$ Gravitation of the *i*th segment

$\mathbf{f}_i$ Force exerted on the *i*th segment by the (*i*-1)th segment

$\mathbf{n}_i$ Moment exerted on the *i*th segment by the (*i*-1)th segment

τ_i Generalized actuator torque at the *i*th joint

$\mathbf{p}_i$ Vector from the pivot point of the *i*th segment to the (*i*+1)th segment

$\mathbf{s}_i$ Vector from the pivot point of the *i*th segment to its center of mass

$\mathbf{z}_{i,j}$ Axis of rotation for the *j*th DOF of the *i*th joint

$\mathbf{F}_i$ Total force on the *i*th segment

$\mathbf{N}_i$ Total external torque on the *i*th segment

Matrices

$\mathbf{J}_i$ Inertia tensor of the *i*th segment

$\mathbf{A}_i^j$ Rotational matrix from the *j*th segment to the *i*th segment

Recursion initialization

$\omega^0 = 0$

$\alpha^0 = 0$

$\mathbf{f}_{n+1}$ = *Force at end-effector*

$\mathbf{n}_{n+1}$ = *Moment at end-effector*

Forward recursion

for $i = 1...n$ do

$$\boldsymbol{\omega}^i = \mathbf{A}_i^{i-1}\boldsymbol{\alpha}^{i-1} + \sum_{j=1}^{nd_i} \mathbf{z}_{i,j}\dot{\theta}_{i,j}$$

$$\boldsymbol{\alpha}^i = \mathbf{A}_i^{i-1}\boldsymbol{\alpha}^{i-1} + \sum_{j=1}^{nd_i} \mathbf{z}_{i,j}\ddot{\theta}_{i,j} + \mathbf{A}_i^{i-1}\boldsymbol{\omega}^{i-1} \times \sum_{j=1}^{nd_i} \mathbf{z}_{i,j}\dot{\theta}_{i,j}$$

$$\mathbf{a}^i = \mathbf{A}_i^{i-1}\left(\mathbf{a}^{i-1} + \boldsymbol{\alpha}^{i-1} \times \mathbf{p}_{i-1} + \boldsymbol{\omega}^{i-1} \times \left(\boldsymbol{\omega}^{i-1} \times \mathbf{p}_{i-1}\right)\right)$$

$$\mathbf{a}_c^i = \mathbf{a}^i + \boldsymbol{\alpha}^i \times \mathbf{s}_i + \boldsymbol{\omega}^i \times \left(\boldsymbol{\omega}^i \times \mathbf{s}_i\right)$$

$$\mathbf{g}_i = \mathbf{A}_i^0\left(m_i\mathbf{g}\right)$$

$$\mathbf{F}_i = m_i\mathbf{a}_c^i$$

$$\mathbf{N}_i = \mathbf{J}_i\boldsymbol{\alpha}^i + \boldsymbol{\omega}^i \times \mathbf{J}_i\boldsymbol{\omega}^i$$

Backward recursion

for $i = 1...n$ do

$$\mathbf{f}_i = \mathbf{F}_i + \mathbf{A}_i^{i+1}\mathbf{f}_{i+1} - \mathbf{g}_i$$

$$\mathbf{n}_i = \mathbf{N}_i + \mathbf{A}_i^{i+1}\mathbf{n}_{i+1} + \mathbf{s}_i \times \left(\mathbf{F}_i - \mathbf{g}_i\right) + \mathbf{p}_i \times \mathbf{A}_i^{i+1}\mathbf{f}_{i+1}$$

$$\tau_i = \mathbf{n}_i \cdot \mathbf{z}_i$$

C. Appendix III: CD-ROM contents

The visual results of this study are supplied on CD-ROM as video clips. The video clips all have a resolution of 368x272 pixels and are compressed with the MPEG-1 format. The CD-ROM disk is organized in a number of directories, each containing the results for a specific coding method. The following directories are provided:

Directory name	Description
`Original`	Original test sequences
`Quantization`	Straight joint quantization
`Predictive`	Predictive coding
`AdaptivePredictive`	Adaptive predictive coding
`DeadReckoning`	Dead reckoning coding
`Transform`	DCT based coding
`SpatialVQ`	Spatial vector quantization
`TemporalVQ`	Temporal vector quantization
`PostureBased`	Posture based coding
`GestureBased`	Gesture based coding
`Hybrid`	Waveform/model based hybrid coding

Four test sequences, namely the conversation, wave, dance and gesture sequences, are used for each coding method. Each directory contains the following files (excluding the directory `Original`):

File name	Description
`Readme.txt`	General information
`Conver.rd`	Rate-distortion information for the conversation sequence
`Conver?.mpg`	MPEG video clips for the conversation sequence
`Wave.rd`	Rate-distortion information for the wave sequence
`Wave?.mpg`	MPEG video clips for the wave sequence

`Dance.rd`	Rate-distortion information for the dance sequence
`Dance?.mpg`	MPEG video clips for the dance sequence
`Gestur.rd`	Rate-distortion information for the gesture sequence
`Gestur?.mpg`	MPEG video clips for the gesture sequence

The "?" wildcard is a single digit zero-based integer index that is an indication of the coding parameters that were used to obtain the sequence. The rate-distortion files contain a list of bit-rates and signal-to-noise ratios in the *same order* as the index. The units of the rate quantity are [bits/second] and the distortion quantity [dB]. The relevant coding parameters and rate-distortion results for each directory (compression method) are as follows:

`Quantization`

Coding parameters:

Index	Number of quantization levels
0	8
1	16
2	32
3	64
4	128
5	256

Rate-distortion results:

Sequence	Conversation		Wave		Dance		Gestures	
Index	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
0	4454.8	9.9	4338.0	10.2	4397.3	10.2	973.7	16.4
1	6135.6	16.7	5891.7	16.5	6038.4	16.9	1283.6	22.9
2	7812.9	23.0	7402.1	23.1	7692.0	23.2	1605.0	29.3
3	9476.0	29.1	8912.4	29.0	9319.5	29.3	1945.0	35.4
4	11062.1	35.2	10281.1	35.1	10872.1	35.3	2275.5	41.7
5	12526.4	41.3	11511.0	41.2	12288.1	41.3	2576.3	47.6

Predictive

Coding parameters:

Index	Number of difference quantization levels
0	4
1	8
2	16
3	32
4	64
5	128
6	256

Rate-distortion results:

Sequence Index	Conversation		Wave		Dance		Gestures	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
0	2356.5	13.4	2428.8	11.2	2555.6	10.4	479.3	14.0
1	3548.7	24.1	3750.9	21.6	3948.5	20.4	607.7	23.6
2	5285.1	31.1	5421.1	28.7	5648.1	27.4	792.9	30.6
3	7090.6	38.1	7168.0	35.1	7393.8	33.9	1074.8	36.5
4	8897.8	44.7	8886.5	41.7	9114.0	40.0	1416.5	43.5
5	10594.6	51.0	10438.5	47.8	10742.8	46.5	1756.8	49.6
6	12141.0	57.3	11774.9	54.3	12246.7	52.4	2120.8	56.3

AdaptivePredictive

Coding parameters:

Index	Number of difference quantization levels
0	4
1	8
2	16
3	32
4	64
5	128
6	256

Rate-distortion results:

Sequence Index	Conversation		Wave		Dance		Gestures	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
0	2939.8	10.7	2734.6	9.0	2984.4	8.0	574.0	10.7
1	3325.4	23.2	3395.8	21.2	3572.4	20.1	700.9	21.7
2	4048.2	31.4	4482.6	28.8	4623.4	27.9	838.4	29.2
3	5267.9	38.7	5891.8	36.0	6036.2	34.9	1008.0	35.9
4	6784.6	44.1	7421.6	42.8	7562.5	41.5	1262.7	42.7

5	8348.6	50.6	8882.1	48.8	9074.5	48.0	1531.2	49.4
6	9833.7	58.7	10167.9	55.9	10470.6	52.9	1836.6	56.3

DeadReckoning

Coding parameters:

Sequence Index	Conversation/Wave error threshold ε [deg]	Dance/Gesture error threshold ε [deg]
0	1.6	16
1	0.8	8
2	0.4	4
3	0.2	2
4	0.1	1
5	0.05	0.5
6	0.025	0.25
7	0.0125	0.125

Rate-distortion results:

Sequence Index	Conversation		Wave		Dance		Gestures	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
0	2567.3	4.4	1711.9	2.1	1164.3	-2.9	187.3	6.2
1	4049.3	9.2	2887.6	5.9	2392.6	2.3	306.7	12.1
2	5967.0	13.5	4678.5	8.7	4080.8	7.9	500.5	18.2
3	8344.5	18.3	6837.5	11.8	6175.2	14.3	815.2	24.6
4	10884.5	22.4	9071.5	17.8	8552.3	21.0	1223.3	30.8
5	13048.7	28.0	11184.5	23.7	11336.7	27.5	1671.2	36.8
6	14996.4	33.1	13196.9	29.3	14156.0	34.0	2227.2	43.0
7	16140.5	35.5	14420.9	31.9	16270.9	39.5	2743.7	48.7

Transform

Coding parameters:

Index	Quantizer bits	Normalized speed threshold
0	8, 7, 7, 6, 6, 5, 4, 4, 3, 3, 2, 2, 0, 0, 0, 0	0.9, 0.92, 0.94, 0.98, 1
1	8, 7, 6, 5, 5, 4, 4, 3, 3, 2, 2, 0, 0, 0, 0, 0	0.7, 0.75, 0.8, 0.85, 0.9
2	7, 6, 5, 4, 4, 3, 3, 3, 2, 2, 0, 0, 0, 0, 0, 0	0.3, 0.4, 0.5, 0.6, 0.8
3	6, 5, 4, 4, 3, 3, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0	0.1, 0.2, 0.3, 0.4, 0.6
4	5, 4, 3, 3, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0.05, 0.1, 0.2, 0.3, 0.4

Rate-distortion results:

Sequence Index	Conversation		Wave		Dance		Gestures	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
0	1669.5	20.0	1558.4	18.1	1653.9	15.7	331.4	22.0
1	2507.8	24.8	2353.7	21.1	2496.9	20.4	486.9	27.6
2	3295.5	28.3	3121.4	24.4	3293.6	22.5	629.4	31.1
3	4019.4	30.5	3742.2	25.8	4022.3	23.7	757.2	33.5
4	4784.7	31.1	4475.1	27.1	4797.9	24.1	920.1	34.5

SpatialVQ

Coding parameters:

Index	Number of VQ codebook entries
0	16
1	32
2	64
3	128
4	256
5	512

Rate-distortion results:

Sequence Index	Conversation		Wave		Dance		Gestures	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
0	747.3	-0.8	725.5	-1.1	789.1	0.2	86.8	4.9
1	959.6	0.5	915.9	1.8	969.0	1.2	98.4	9.8
2	1519.1	1.7	1106.5	4.3	1156.4	2.8	106.2	12.1
3	2145.1	3.2	1331.3	8.0	1331.2	4.2	130.9	14.9
4	2569.6	4.6	1558.1	11.7	1494.5	6.0	148.4	16.7
5	2910.3	6.1	1805.6	16.0	1645.9	7.9	167.5	18.7

TemporalVQ

Coding parameters:

Index	Number of VQ codebook entries
0	16
1	32
2	64
3	128
4	256
5	512

Rate-distortion results:

Sequence Index	Conversation		Wave		Dance		Gestures	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
0	988.4	7.5	937.6	7.6	1005.9	5.8	189.9	13.9
1	1254.7	10.0	1200.6	10.3	1236.2	8.1	229.5	16.7
2	1586.8	12.7	1551.8	14.7	1469.6	10.3	281.2	19.4
3	1972.5	15.3	2274.1	25.4	1666.1	12.5	321.9	21.2
4	2412.3	18.4	2593.3	30.3	1908.5	15.1	398.8	24.4
5	2906.4	22.0	2833.1	32.8	2301.2	17.5	494.2	28.1

PostureBased

Coding parameters:

Index	Sampling distance d
0	80
1	40
2	26.6667
3	20
4	16
5	13.3333
6	10

Rate-distortion results:

Sequence Index	Conversation		Wave		Dance		Gestures	
	Rate	VPSNR	Rate	VPSNR	Rate	VPSNR	Rate	VPSNR
0	20.9	6.8	19.5	3.6	67.7	3.8	7.4	13.1
1	55.2	9.1	43.7	6.2	191.2	6.5	17.9	15.2
2	88.4	10.7	69.8	8.2	317.2	7.3	28.1	16.9
3	124.5	11.6	94.7	9.3	440.5	7.5	40.2	18.4
4	156.7	12.1	118.1	9.8	566.8	7.8	50.1	19.7
5	192.1	12.5	144.8	10.2	666.5	8.2	58.6	20.9
6	225.5	12.9			762.8	8.5	70.0	21.8

GestureBased

Abbreviations:

nWin	Length of threshold filter
β	Initial value of β (before recursion)
nMin	Minimum number of samples per segment
nMax	Maximum number of samples per segment
D	Gesture table similarity discard distance
E	Gesture fit distance
b	Error weigthing constant *b* (*a* is always 1)
nNorm	Number of samples in normalized or warped segment
nBlend	Number of samples that can be used for blending

Parameters for coding of conversation sequence:

Param. Group	*nWin*	β	*nMin*	*nMax*	*D*	*E*	*b*	*nNorm*	*nBlend*
0	20	0.75	5	60	1	1	1	60	10
1	20	0.75	5	60	1	1	1	60	10
2	20	0.75	5	60	5	5	0.5	60	10
3	20	0.75	5	60	5	5	0.5	60	10
4	20	0.75	5	60	5	5	0.5	60	10
5	20	0.75	5	60	5	5	0.5	60	10
6	20	0.75	5	60	2	2	0.5	60	10
7	20	0.75	5	60	2	2	0.5	60	10

Parameters for coding of wave sequence:

Param. Group	*nWin*	β	*nMin*	*nMax*	*D*	*E*	*b*	*nNorm*	*nBlend*
0	20	0.75	5	60	1	1	1	60	10
1	20	0.75	5	60	1	1	1	60	10
2	20	0.75	5	60	10	10	0.5	60	10
3	20	0.75	5	60	5	5	0.5	60	10
4	20	0.75	5	60	10	10	0.5	60	10
5	20	0.75	5	60	5	5	0.5	60	10
6	20	0.75	5	60	2	2	0.5	60	10
7	20	0.75	5	60	2	2	0.5	60	10

Parameters for coding of dance sequence:

Param. Group	*nWin*	*β*	*nMin*	*nMax*	*D*	*E*	*b*	*nNorm*	*nBlend*
0	20	0.75	3	30	5	5	1	30	10
1	20	0.75	3	30	5	5	1	30	10
2	20	0.75	3	30	15	15	0.75	30	10
3	20	0.75	3	30	10	10	0.75	30	10
4	20	0.75	3	30	15	15	0.75	30	10
5	20	0.75	3	30	10	10	0.75	30	10
6	20	0.75	3	30	5	5	0.75	30	10
7	20	0.75	3	30	5	5	0.75	30	10

Parameters for coding of gesture sequence:

Param. Group	*nWin*	*β*	*Nmin*	*nMax*	*D*	*E*	*b*	*nNorm*	*nBlend*
0	20	0.75	5	60	1	1	1	60	10
1	20	0.75	5	60	1	1	1	60	10
2	20	0.75	5	60	5	5	0.5	60	10
3	20	0.75	5	60	10	10	0.25	60	10
4	20	0.75	5	60	5	5	0.5	60	10
5	20	0.75	5	60	10	10	0.25	60	10
6	20	0.75	5	60	2	2	0.5	60	10
7	20	0.75	5	60	2	2	0.5	60	10

Rate-distortion results:

Sequence	Conversation		Wave		Dance		Gestures	
Index	Rate	VPSNR	Rate	VPSNR	Rate	VPSNR	Rate	VPSNR
0	82.1	5.8	59.1	3.5	145.1	4.1	5.8	10.5
1	89.5	6.5	59.0	3.5	165.5	5.5	5.8	10.5
2	105.5	8.5	65.4	4.9	214.3	7.8	7.2	11.6
3	140.8	12.3	82.2	8.2	293.3	11.5	11.1	14.8
4	155.8	14.1	96.5	10.9	310.2	12.4	13.1	16.3
5	167.3	15.3	109.6	12.3	336.7	13.3	14.9	17.6

Hybrid

Coding parameters:

Index	Coeff. 0 [bits]	Coeff. 1 [bits]	Coeff. 2 [bits]
0	4	3	2
1	4	4	2
2	5	4	3
3	5	5	3
4	6	5	4
5	6	6	4
6	7	6	5
7	7	7	5

Rate-distortion results:

Sequence	Conversation		Wave		Dance		Gestures	
Index	Rate	VPSNR	Rate	VPSNR	Rate	VPSNR	Rate	VPSNR
0	395.4	11.2	380.0	7.3	749.7	10.8	109.6	17.1
1	532.5	13.5	492.6	10.2	1034.0	13.7	141.6	18.0
2	622.7	14.4	564.9	11.1	1194.5	15.0	165.0	18.1
3	690.7	14.7	619.6	11.8	1313.0	15.7	181.9	18.3
4	746.0	15.0	654.6	12.1	1413.5	16.2	194.3	18.3
5	785.1	15.4	691.3	12.5	1497.2	16.5	202.6	18.4
6	820.8	15.6	718.2	12.8	1573.2	16.7	211.5	18.4
7	849.5	15.7	743.9	12.9	1640.3	16.9	217.1	18.4