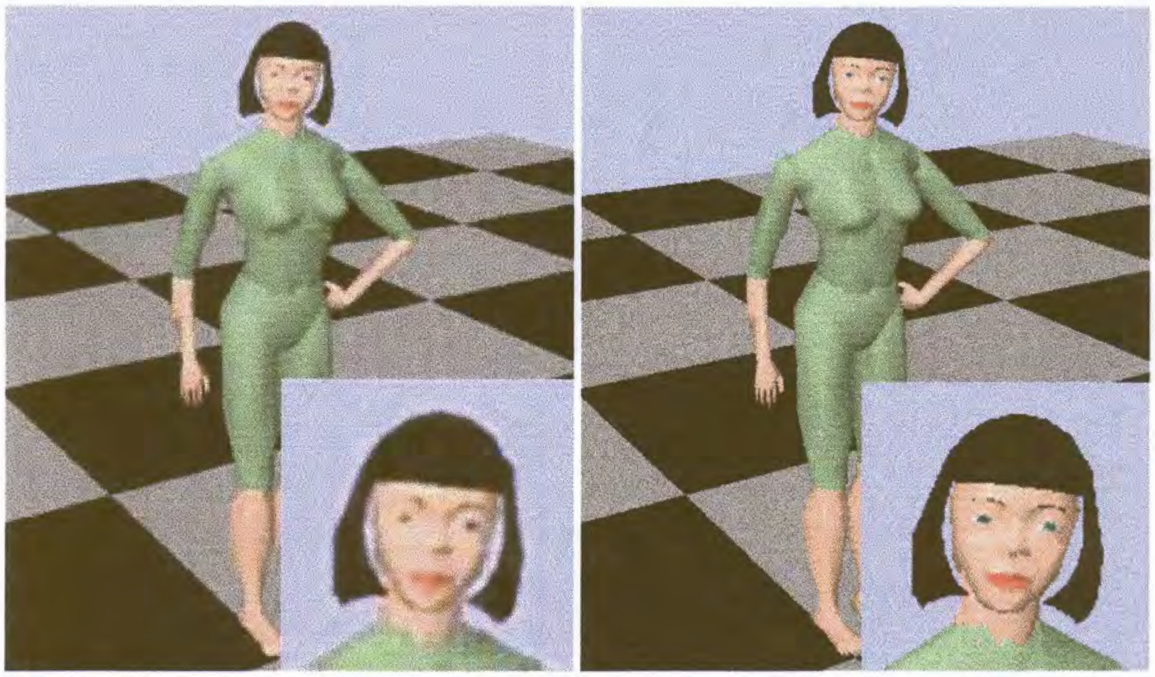
Chapter 1 Introduction

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.

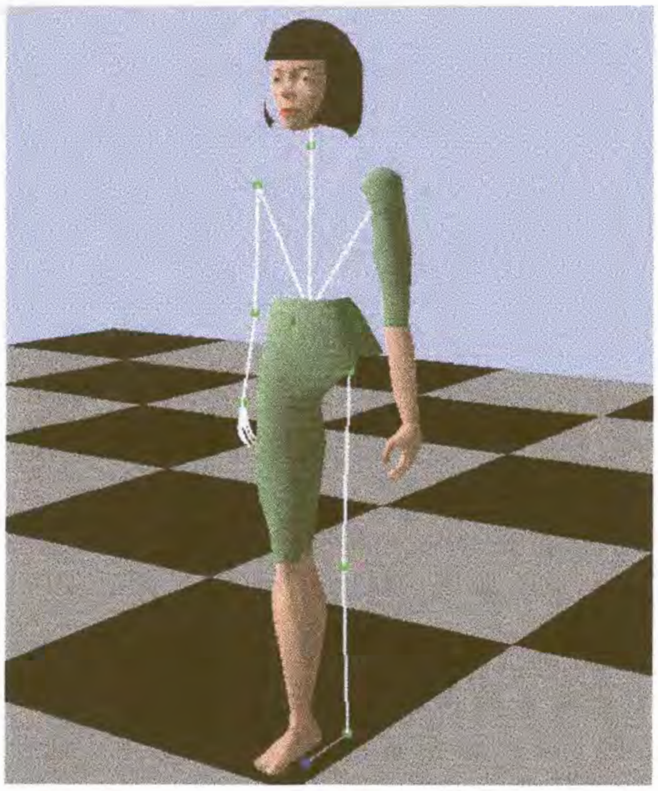


Figure 2-1: Hierarchical skeleton with partial solid surfaces.

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 freeform 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.

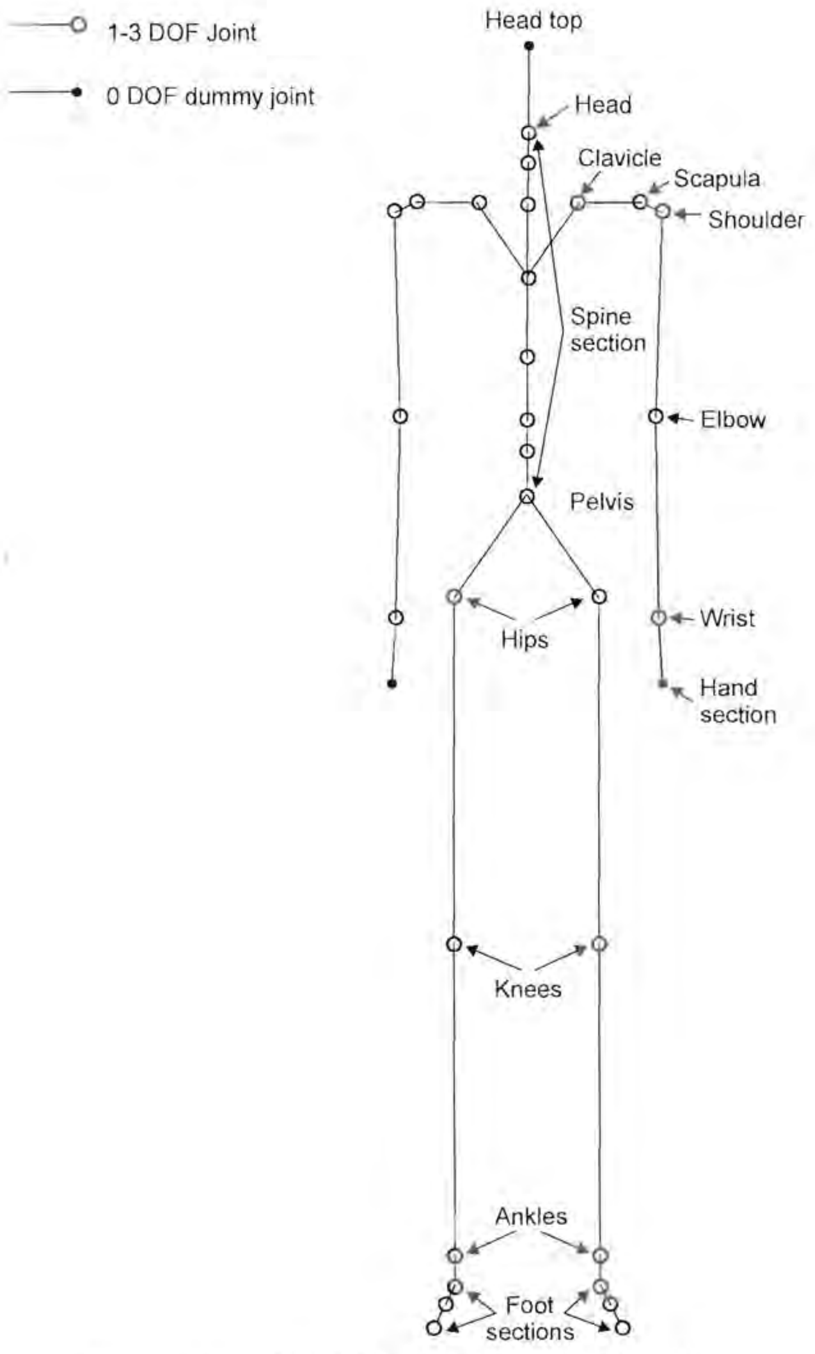


Figure 3-2a: MPEG-4 main body specification.

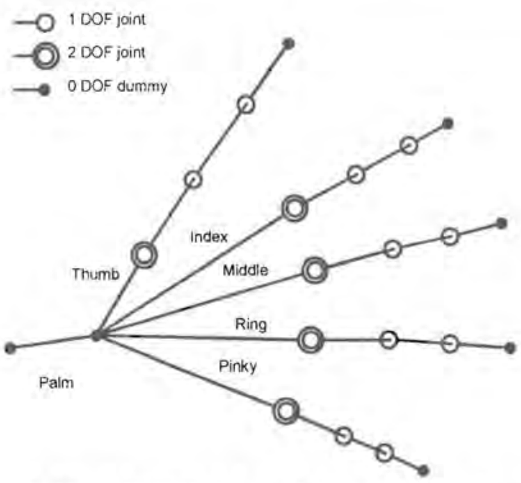


Figure 3-2b: MPEG-4 hand section specification.

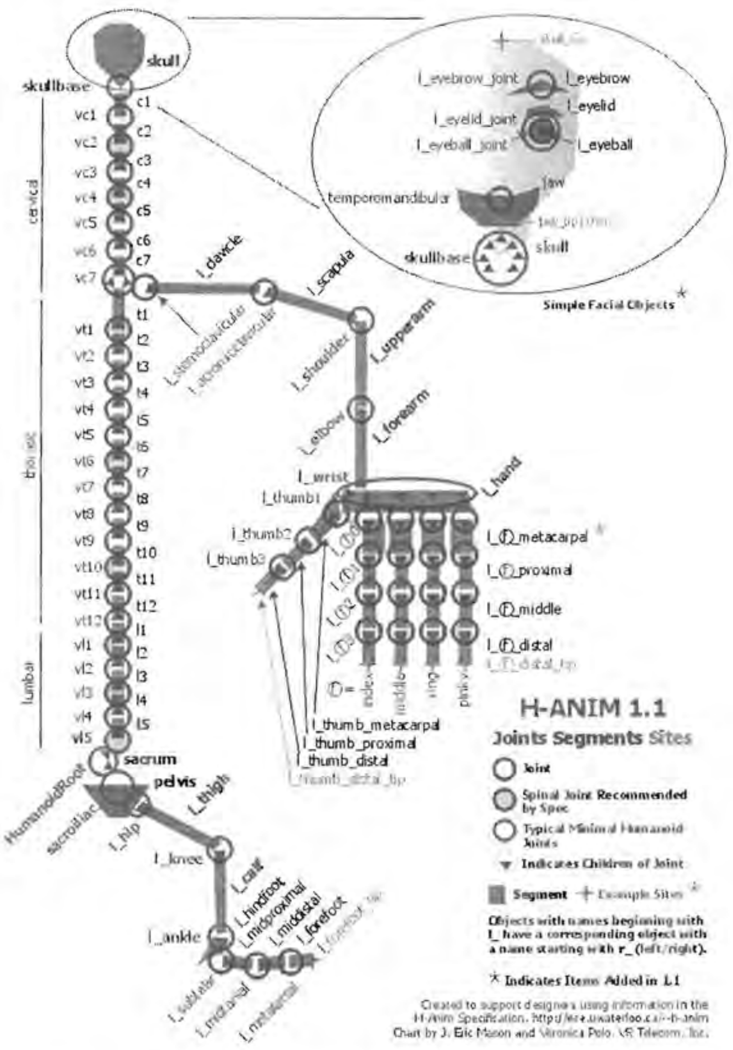


Figure 3-3: VRML 2.0 H-Anim specification.

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.

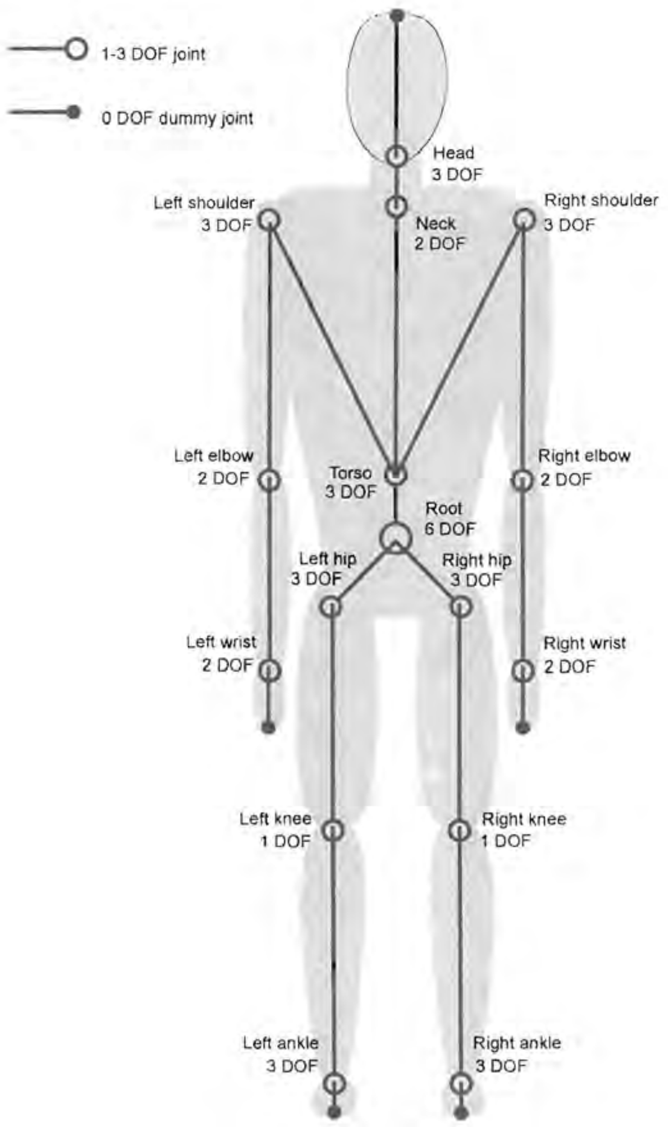


Figure 3-4a: Simplified human body model.

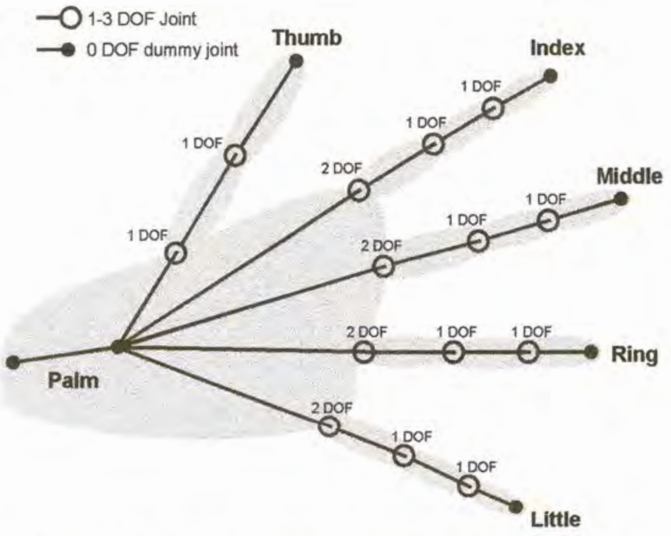
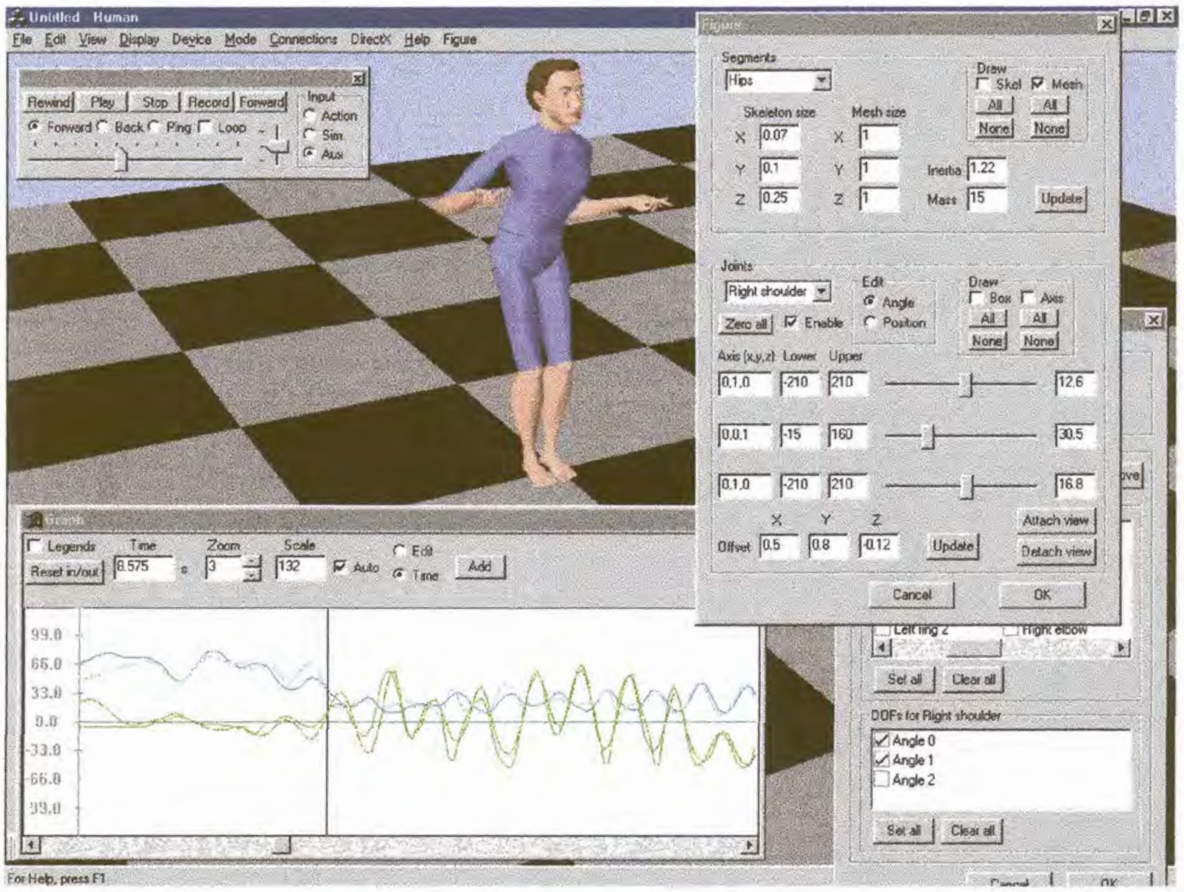


Figure 3-4b: Simplified human hand model.


Untitled - Human
File Edit View Display Device Mode Connections DirectX Help Figure
Rewind Play Stop Record Forward
Forward Back Ping Loop
Input Action Sim Aux
Graph
Legends Time Zoom Scale
Reset in/out Auto Edit Time Add
99.0 66.0 33.0 0.0 -33.0 -66.0 -99.0
Figure
Attach view Detach view Update Cancel OK
Set all Clear all
For Help, press F1

Segments: Hips
Draw: [ ] Skel [✓ Mesh]
Skeleton size: X 0.07, Y 0.1, Z 0.25
Mesh size: X 1, Y 1, Z 1
Inertia 1.22
Mass 15

Joints: Right shoulder
[✓ Enable]
Edit: [✓ Angle] Position
Draw: [ ] Box [ ] Axis

Axis (x,y,z)	Lower	Upper	
0,1,0	-210	210	12.6
0,0,1	-15	160	30.5
0,1,0	-210	210	16.8

Offset: X 0.5, Y 0.8, Z -0.12

Time 8.575 s, Zoom 3, Scale 132, [✓ Auto], [✓ Time]

Left ring 2, Right elbow

DOFs for Right shoulder
[✓ Angle 0]
[✓ Angle 1]
[ ] Angle 2

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

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

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 $\theta_{0,0}$ $\theta_{0,1}$ $\theta_{0,2}$ $\theta_{0,3}$ $\theta_{0,4}$ $\theta_{0,5}$	-180 -40 -30 $-\infty$ $-\infty$ $-\infty$	180 40 30 $+\infty$ $+\infty$ $+\infty$	Figure *x* angle Figure *y* angle Figure *z* angle Figure *x* position Figure *y* position Figure *z* position
Torso : Root		3 $\theta_{1,0}$ $\theta_{1,1}$ $\theta_{1,2}$	-30 -40 -30	30 40 30	Torso twist Torso bend Torso flexion
Neck : Torso		2 $\theta_{2,0}$ $\theta_{2,1}$	-30 -15	30 15	Neck bend Neck flexion
Head : Neck		3 $\theta_{3,0}$ $\theta_{3,1}$ $\theta_{3,2}$	-90 -50 -15	90 50 15	Head yaw Head pitch Head roll
Left shoulder : Torso		3 $\theta_{5,0}$ $\theta_{5,1}$ $\theta_{5,2}$	-180 -160 -180	180 0 180	Upper arm flexion Upper arm abduction Upper arm twist
Left elbow : Left shoulder		2 $\theta_{6,0}$ $\theta_{6,1}$	-180 -90	0 90	Elbow hinge Lower arm twist
Left wrist : Left elbow		2 $\theta_{7,0}$ $\theta_{7,1}$	-60 -30	70 30	Wrist yaw Wrist pitch
Left thumb 1 : Left wrist		2 $\theta_{8,0}$ $\theta_{8,1}$	0 -20	60 60	Thumb flexion Thumb abduction
Left thumb 2 : Left thumb 1		1 $\theta_{9,0}$	0	60	Thumb segment 2 flexion
Left index 1 : Left wrist		2 $\theta_{10,0}$ $\theta_{10,1}$	-5 0	5 60	Index finger segment 1 abduction 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 :		1			
Left index 2		$\theta_{12,0}$	0	45	Index finger segment 3 flexion
Left middle 1 :		2			
Left wrist		$\theta_{13,0}$	-5	5	Middle finger segment 1 abduction
		$\theta_{13,1}$	0	60	Middle finger segment 1 flexion
Left middle 2 :		1			
Left middle 1		$\theta_{14,0}$	0	75	Middle finger segment 2 flexion
Left middle 3 :		1			
Left middle 2		$\theta_{15,0}$	0	45	Middle finger segment 3 flexion
Left ring 1 :		2			
Left wrist		$\theta_{16,0}$	-5	5	Ring finger segment 1 abduction
		$\theta_{16,1}$	0	60	Ring finger segment 1 flexion
Left ring 2 :		1			
Left ring 1		$\theta_{17,0}$	0	75	Ring finger segment 2 flexion
Left ring 3 :		1			
Left ring 2		$\theta_{18,0}$	0	45	Ring finger segment 3 flexion
Left little 1 :		2			
Left wrist		$\theta_{19,0}$	-5	5	Little finger segment 1 abduction
		$\theta_{19,1}$	0	60	Little finger segment 1 flexion
Left little 2 :		1			
Left little 1		$\theta_{20,0}$	0	75	Little finger segment 2 flexion
Left little 3 :		1			
Left little 2		$\theta_{21,0}$	0	45	Little finger segment 3 flexion
Right shoulder :		3			
Torso		$\theta_{22,0}$	-180	180	Upper arm flexion
		$\theta_{22,1}$	-160	0	Upper arm abduction
		$\theta_{22,2}$	-180	180	Upper arm twist
Right elbow :		2			
Right shoulder		$\theta_{23,0}$	-180	0	Elbow hinge
		$\theta_{23,1}$	-90	90	Lower arm twist
Right wrist :		2			
Right elbow		$\theta_{24,0}$	-60	70	Wrist yaw
		$\theta_{24,1}$	-30	30	Wrist pitch
Right thumb 1 :		2			
Right wrist		$\theta_{25,0}$	0	60	Thumb flexion
		$\theta_{25,1}$	-20	60	Thumb abduction
Right thumb 2 :		1			
Right thumb 1		$\theta_{26,0}$	0	60	Thumb segment 2 flexion
Right index 1 :		2			
Right wrist		$\theta_{27,0}$	-5	5	Index finger segment 1 abduction
		$\theta_{27,1}$	0	60	Index finger segment 1 flexion
Right index 2 :		1			
Right index 1		$\theta_{28,0}$	0	75	Index finger segment 2 flexion
Right index 3 :		1			
Right index 2		$\theta_{29,0}$	0	45	Index finger segment 3 flexion
Right middle 1 :		2			
Right wrist		$\theta_{30,0}$	-5	5	Middle finger segment 1 abduction
		$\theta_{30,1}$	0	60	Middle finger segment 1 flexion
Right middle 2 :		1			
Right middle 1		$\theta_{31,0}$	0	75	Middle finger segment 2 flexion
Right middle 3 :		1			
Right middle 2		$\theta_{32,0}$	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.

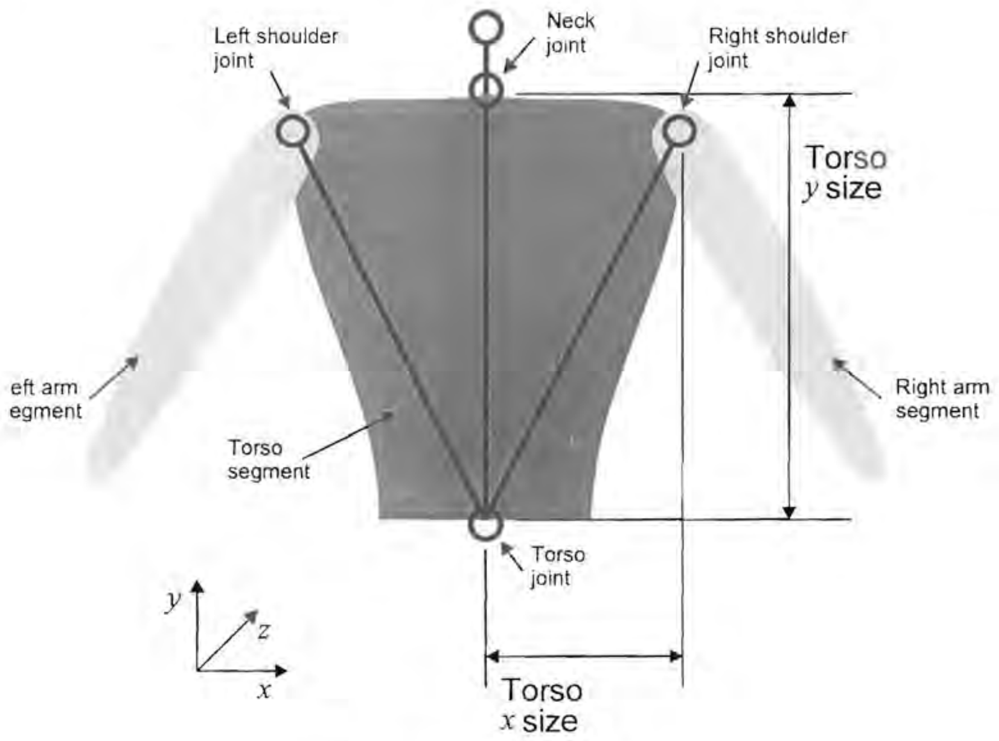


Figure 3-6: Segment sizes.

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.

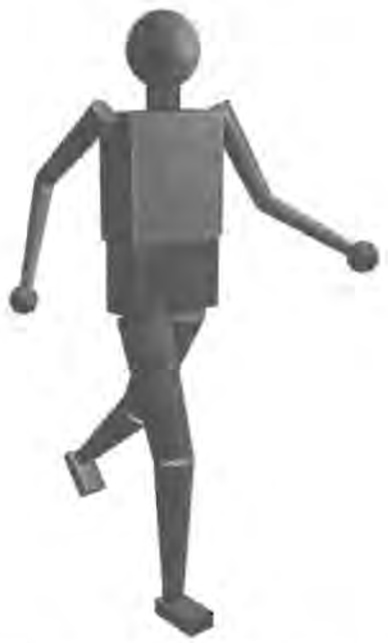


Figure 3-7: Approximated segment shapes, excluding fingers.

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.

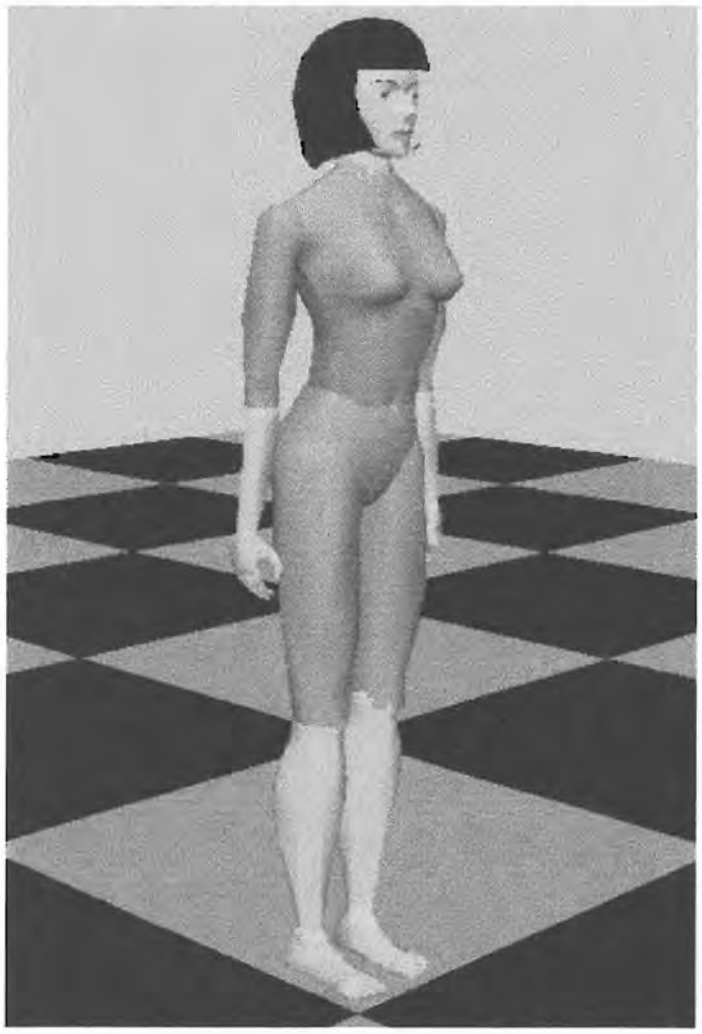


Figure 3-8a: Female model.

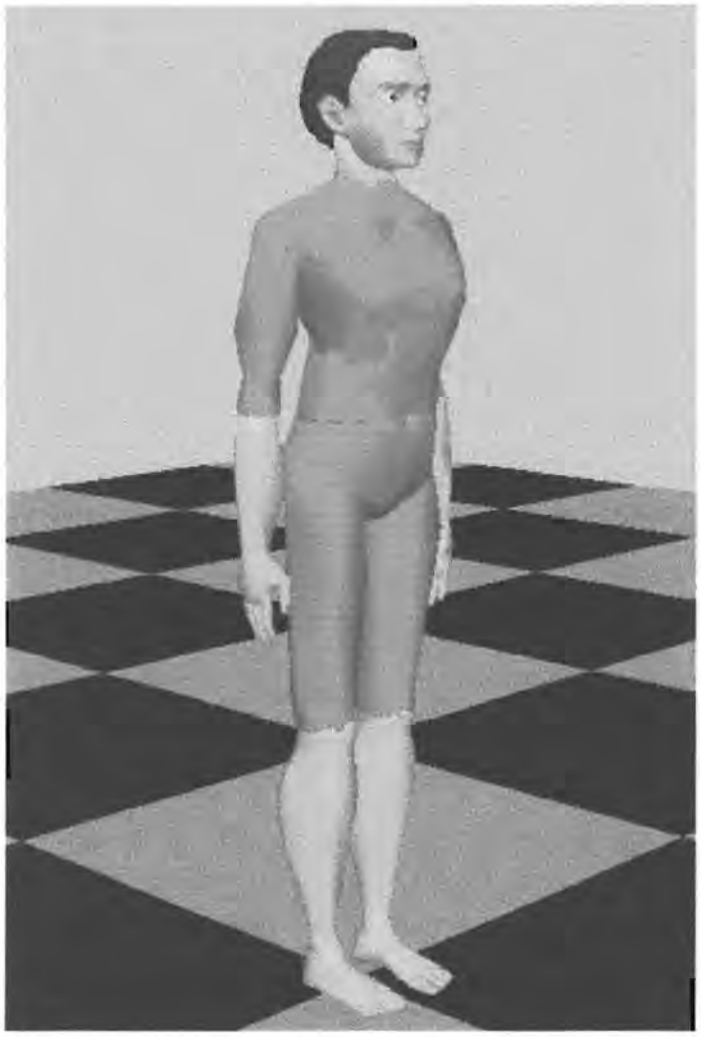


Figure 3-8b: Male model.

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.

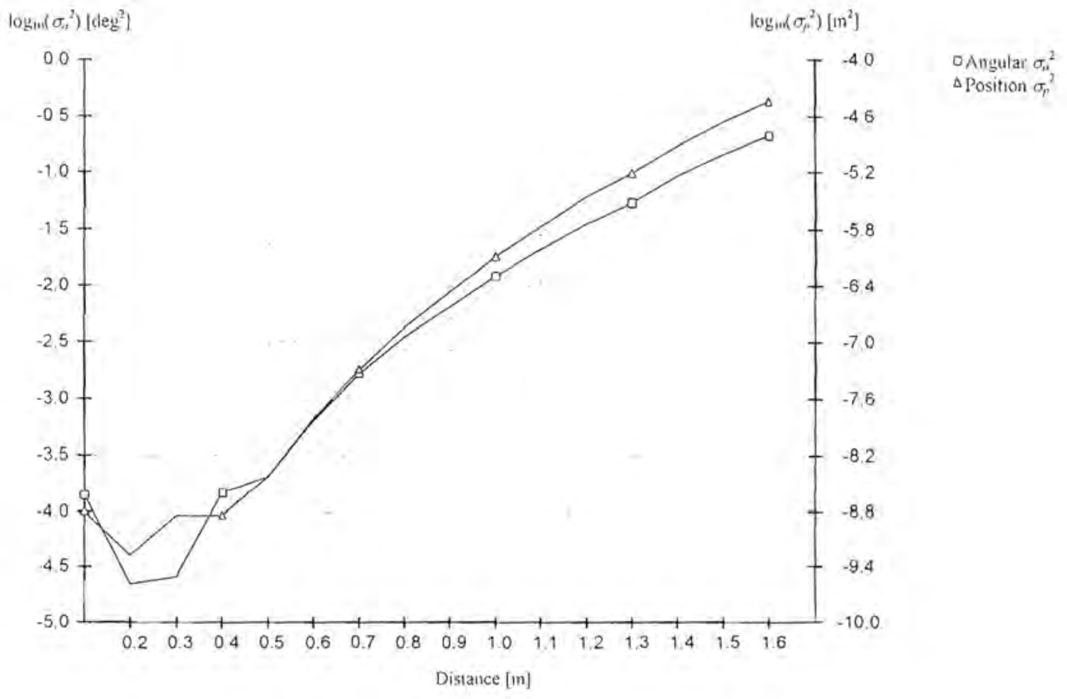


Figure 4-1: Noise power vs. distance from transmitter.

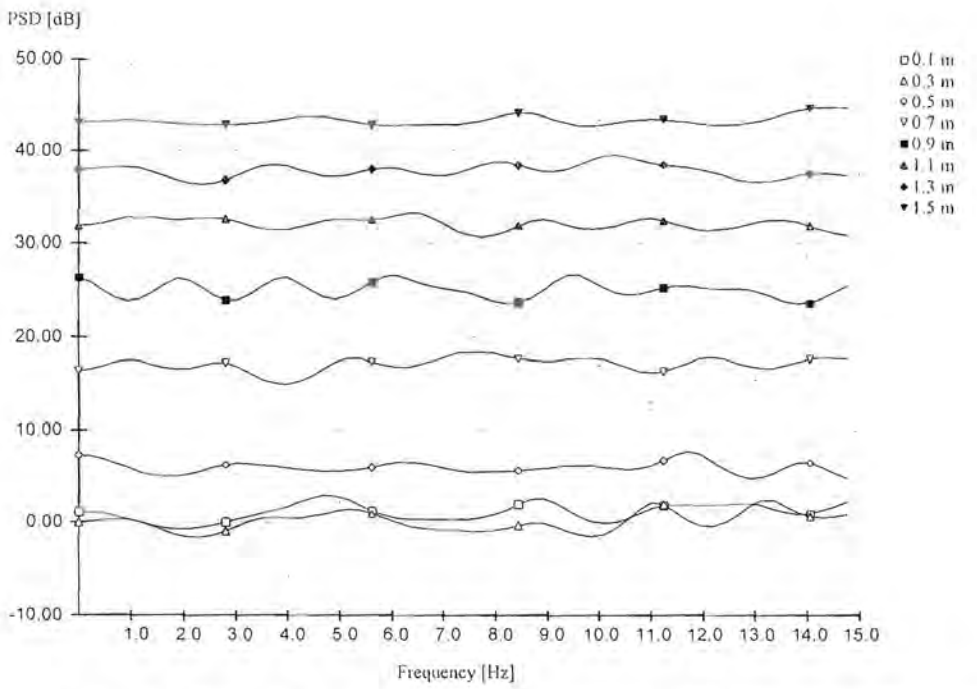


Figure 4-2a: Normalized PSD of sensor position noise with distance.

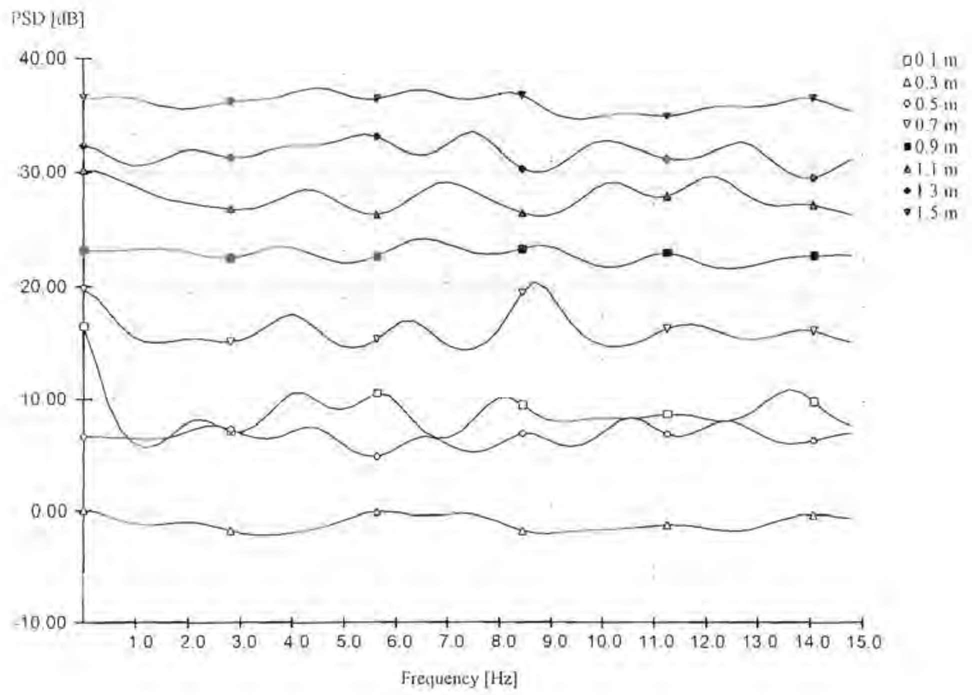


Figure 4-2b: Normalized PSD of sensor rotation noise with distance.

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

$$out = \frac{raw_{val} - raw_{\min}}{raw_{\max} - raw_{\min}}, \tag{4-2}$$

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.

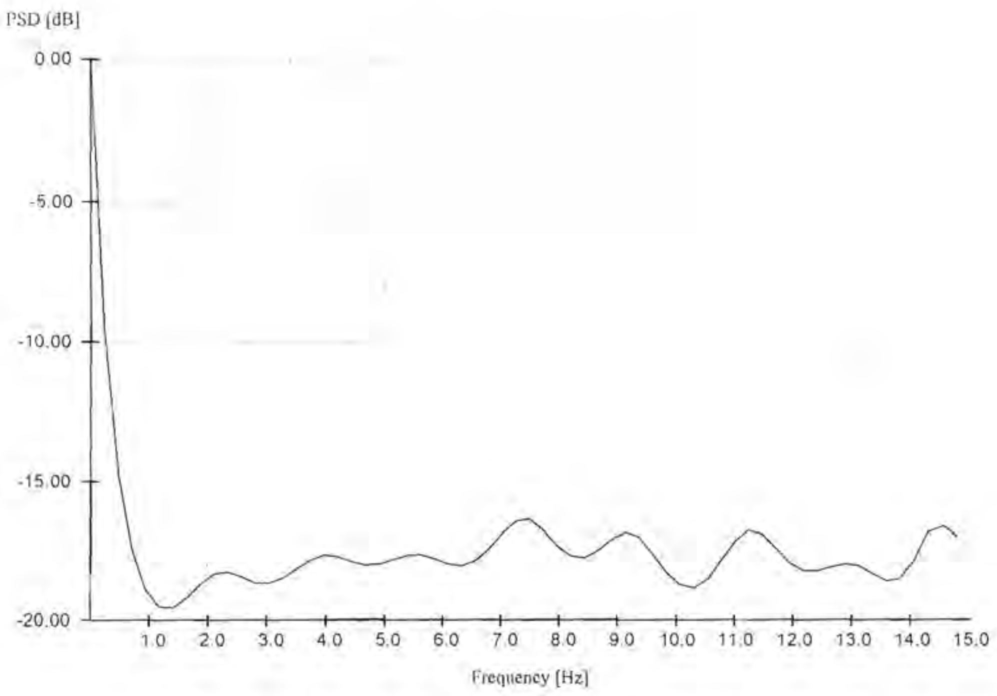


Figure 4-3: Normalized PSD of glove sensor noise.

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.

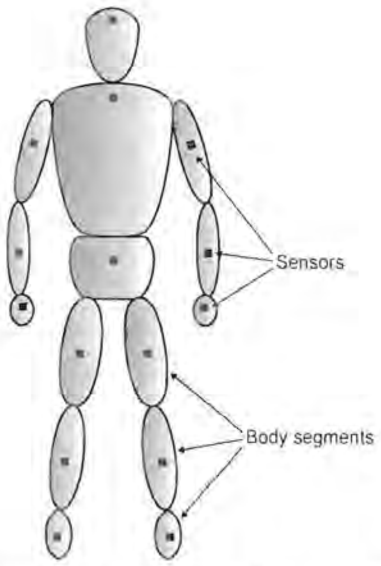


Figure 4-4: Full body sensors.

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.

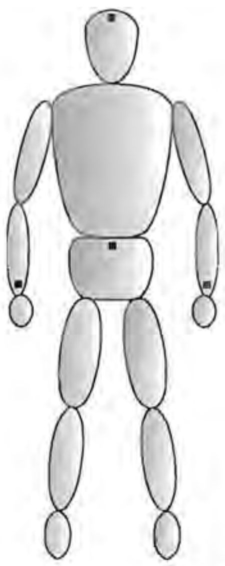


Figure 4-5: Minimal body sensors.

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.

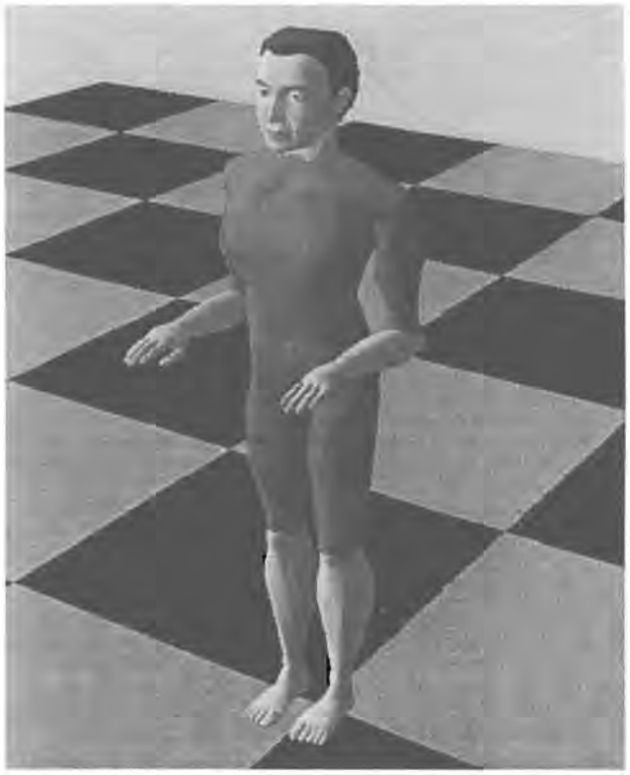


Figure 4-6: Calibration posture.

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

$$\begin{aligned}
\mathbf{A} &= \left[\begin{array}{c:c} \mathbf{I} & 0 \\ \hdashline 0 & 0 \end{array}\right]\mathbf{C}, \\
\mathbf{B} &= \left[\begin{array}{c:c} \mathbf{0} & 0 \\ \hdashline 0 & 1 \end{array}\right]\mathbf{C}, \\
\mathbf{M} &= \mathbf{A}^{-1}\mathbf{S}\mathbf{B}^{-1},
\end{aligned} \tag{4-3}$$

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.

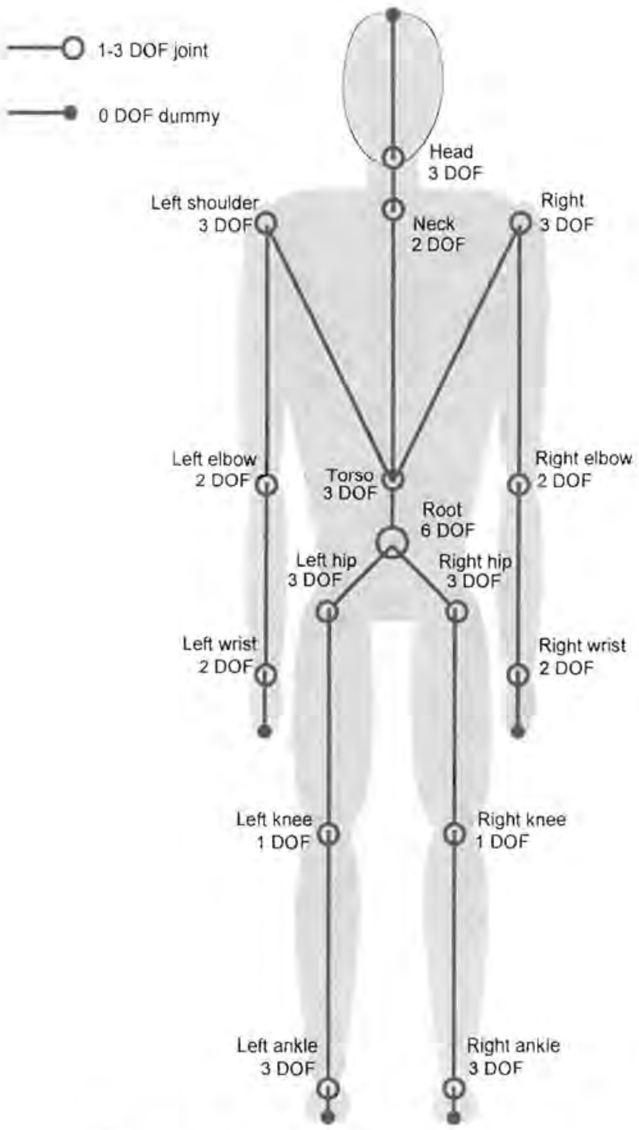


Figure 4-7: Simplified body model.

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.

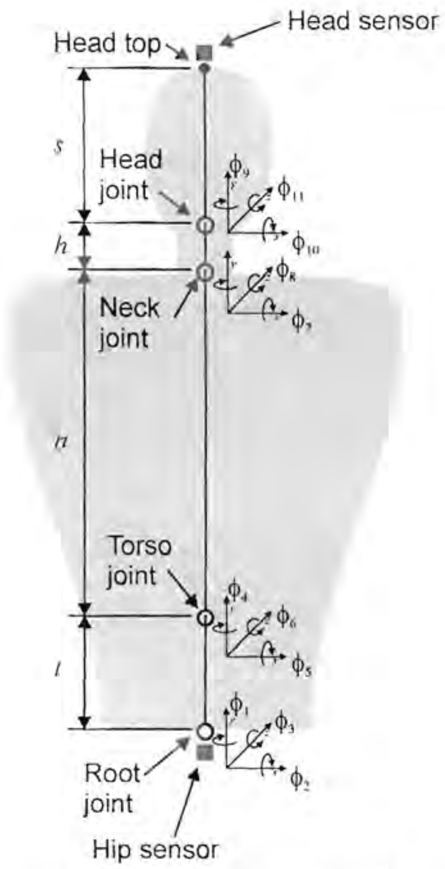


Figure 4-8: Simplified model of the spine.

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

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

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.

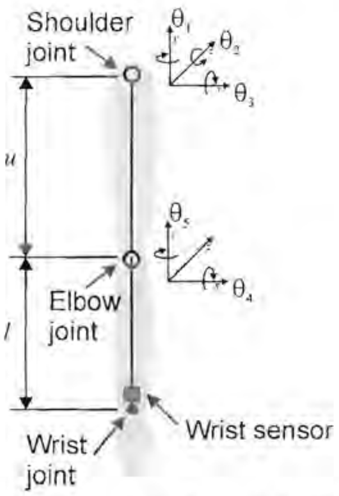


Figure 4-9: Simplified model of the arm.

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

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

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

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

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

$$\phi_3 = \tan^{-1}\left(\frac{-a_{41}\mathrm{c}_1\mathrm{c}_2 + a_{42}\mathrm{s}_2 - a_{43}\mathrm{s}_1\mathrm{c}_2}{-a_{41}\mathrm{s}_1 + a_{43}\mathrm{c}_1}\right), \tag{4-25}$$

where $\mathrm{c}_1 = \cos\phi_1$, $\mathrm{c}_2 = \cos\phi_2$, $\mathrm{s}_1 = \sin\phi_1$ and $\mathrm{s}_2 = \sin\phi_2$. 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 $\mathbf{M}$, 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 $\phi_3^{'}, \phi_2^{'}$ and $\phi_1^{'}$ 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).

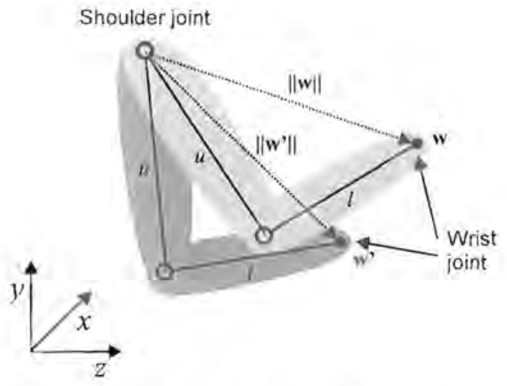


Figure 4-10: Wrist position errors.

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.

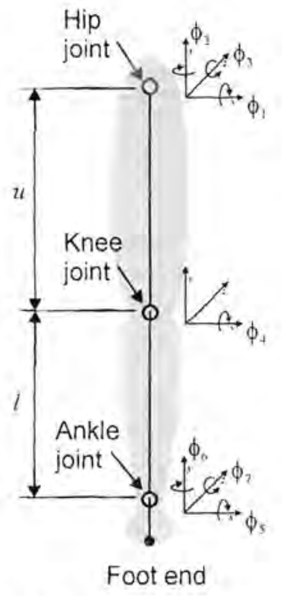


Figure 4-11a: Simplified leg model.

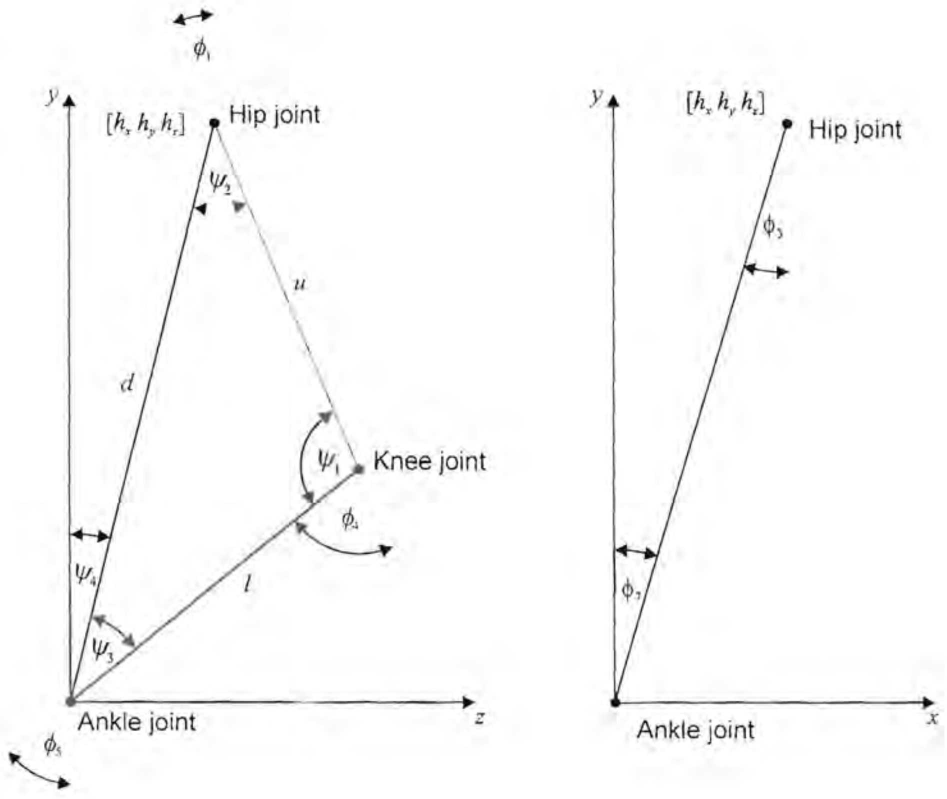


Figure 4-11b: Leg model in the *yz* and *xy* planes.

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

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

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 ψ_1 = 180 and ψ_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

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

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

$$\phi_1 = \psi_2 - \psi_4,$$
$$\phi_4 = 180 - \psi_1, \qquad (4\text{-}35)$$
$$\phi_5 = \psi_3 + \phi_4.$$

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

$$\phi_3 = \tan^{-1}\left(\frac{h_x}{h_y}\right), \qquad (4\text{-}36)$$
$$\phi_7 = -\phi_3.$$

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.