Digital Marionette: Augmenting Kinematics with Physics for Multi-Track Desktop Performance Animation

by

Sageev Oore

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

DIGITAL MARIONETTE: Augmenting Kinematics with Physics for Multi-Track Desktop Performance Animation

Sageev Oore

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2002

We have developed a novel desktop performance animation, DIGITAL MARIONETTE, which provides an interactive interface for animating graphical characters in real time using a minimal input configuration of just two motion trackers.

By embedding the trackers in a physical tubing, we establish a tangible interface with the various coordinate frames inherent to the character. This provides the user with the kinesthetic, visual, tactile and inertial-based feedback for manipulating these frames.

Building on this compatibility, we develop a multi-track motion recording framework and a set of mappings that enable the two 6-DOF trackers to control a 33-DOF character by progressively layering motions.

Local physically-based motion actuators are introduced to augment an existing kinematic mapping, facilitating the animation task and enhancing the naturalness of the motion. They are designed so that the user is not burdened with the additional cognitive overhead of solving the balance control problem. Thus we integrate kinematics and dynamical principles for more effective real-time animation control.

An extensive account of the experience of learning and using the DIGITAL MARIONETTE system is given. This focuses on the perspective of a user experienced both with this system, as well as with other complex interfaces; the learning process is contextualized, and some common principles are exposed. Three novice test users spent

approximately two hours each working with our system, and were able to achieve the basics of a walk within that time. These results indicate that DIGITAL MARIONETTE's interface provides opportunity for in-depth exploration, learning and refinement in the creation of animation.

Our desktop performance animation system enables the animator to fluidly and immediately play out his or her ideas of motion into animation, giving the animator a unique instrument for creating a range of truly expressive human character animation ranging from walking to dancing. DIGITAL MARIONETTE has been used in a live theatrical performance, and the motion created with it has been described by performers as "beautiful" and "graceful".

We have developed a powerful desktop tool which, via its unique tangible interface, local-physics based models, and layered mappings, allows very efficient creation of expressive animation.

Acknowledgements

I would like to express my profound appreciation and thanks to Geoffrey Hinton, my advisor. I feel very fortunate to have had the opportunity to work with Geoff. His enthusiasm, intuition and relentless creativity have inspired me deeply, and provided a starting engine and essential motivation for this work. I have learned so much from him, and there is so much more to learn.

My great appreciation and thanks also go to Demetri Terzopoulos, my co-advisor. Demetri's strong encouragement and support led me to raise my goals and to write what became the core of this thesis. While allowing me the freedom to pursue the directions of my choice, his deep insight and vision, sometimes phrased as questions, have been and continue to be invaluable.

My committee members— Allan Jepson, Radford Neal and Michiel van de Panne— gave me comments, questions and suggestions that provided important contributions to this dissertation. I am grateful for the way in which they have broadened my perspective. I would like to thank Ken Perlin, my external examiner, for his careful reading and thoughtful appraisal. His visit here was a great completion to this thesis.

I extend a heartfelt thanks to all of the members of the DGP group; I enjoyed their company, and their feedback and input was always greatly valued. I thank Petros Faloutsos and Victor Ng for providing me with the DANCE environment, and for their continual support in the development stages. I thank Joe Laszlo for his tremendously generous help in the editing suite, and both Joe and Michael Neff for their help with my demo video.

I would like to thank Elana Freeman, Trish Leeper, Michael Neff and Chris Trendall for spending time with my system. Their interest and perceptive feedback made a significant contribution to this work, and suggested a variety of future directions as well.

It is a pleasure to be able to thank Mordecai Oore for the discussion regarding the input device.

I am indebted to Jim Ruxton for encouraging me and giving me the opportunities for live performances with my system. His warm enthusiasm is wonderful.

Bill Buxton, Eugene Fiume and Fahiem Bacchus each provided me with just the right statements of encouragement and advice near the beginning, middle and end of this project (even if at the time they did not know the importance of those key meetings for me). Thanks go to Kamran Bozorgmehr for his help, especially with the live demos, and Kathy Yen and Linda Chow for all their assistance in administrative matters.

I thank Ran El-Yaniv for his suggestions and support, and I am grateful for his friendship.

Chakra Chennubhotla's corrections and discussions provided an invaluable contribution to this work. I can't imagine how I would have finished the thesis without his encouragement and friendship. I appreciate it deeply.

It is a great pleasure to thank Mladen Nedimović for everything. Instead of even trying to summarize it, I will just have a drink to it (as soon as I get home).

I am happy to thank my sister nini (Jasmine) so much for her wonderful editing of whatever material I gave her. She is one of the best writers I know, and I would love to read more of her writing.

A special thanks goes to Elana, for her love and support and listening.

I am deeply grateful to my family— Mordecai, Irène, Jasmine, Dani, Jonathan, and Niunia— for their love and support and encouragement.

for ima, aba and niunia

Contents

Chapter 1

Introduction

"Anima" means "breath"; so "to animate" means "to give breath" or life to something. Creating human character animation, with respect to this definition, is an activity that requires skill and artistry, whether it is being done by traditional cel or using computer technology.

Our goal is to design and build a tool that allows the creation of expressive character animation, and furthermore provides a potentially efficient means of doing so. Such a tool will consist of two principal elements: the exchange of sensory information (in our case visual and tactile) between the user and the computer, and the underlying transformation from the input signal to the graphical output parameters. The technology we use for the input device is two 6 degree of freedom Polhemus motion trackers. We refer to the transformation from input to output as the *mapping*, and our output parameters are the joint angles of an articulated anthropomorphic figure. Our goal, therefore, is to provide the user with a means of manipulating the trackers to control the animation of a 3D character.

Area of Focus. This direction still allows for an enormous range of exploration, which we have therefore focused in the following ways:

(1) Range of Motion: Ultimately, we are interested in a system that is general enough

to handle a wide range of motions, however we have chosen to make "walking" our testbed. This is a familiar and complex motion; an animated walk can be assessed as "believable" or "unbelievable", and can still leave room for expressiveness in the motion. As much as possible, though, we have not made the system specially tailored for this task.

(2) Level of Abstraction of Control: our focus is on low-level, continuous control. We are not looking to create a "gesture language" for high-level scripted motions.

1.1 Overview

To achieve our goal, we have developed DIGITAL MARIONETTE, a *desktop performance animation system.* Performance animation, also known as computer puppetry, is a method of creating animation in real-time: The animator uses real-time input devices to interactively drive the motion of a computer graphic (CG) character, and is provided with immediate feedback displaying the animation as it is being created. The animator is effectively a puppeteer; the computer graphics character is the puppet; and the mapping defines how the puppet is virtually strung. The feedback loop is shown in Fig 1.1.

Our DIGITAL MARIONETTE system works by providing a tangible user interface to operate in synergy with a set of carefully designed mappings. These mappings are primarily kinematic, but the input signal is filtered and in certain cases the output signal is modulated by local physics-based motion models, enhancing the final animation. All these components are tightly integrated within a multi-track motion recording environment. In contrast to most performance animation systems, the physical setup for DIGITAL MARIONETTE is minimal, and Fig 1.1 shows a user sitting at his desk manipulating a virtual skeleton character. Our system makes the following scenario possible:

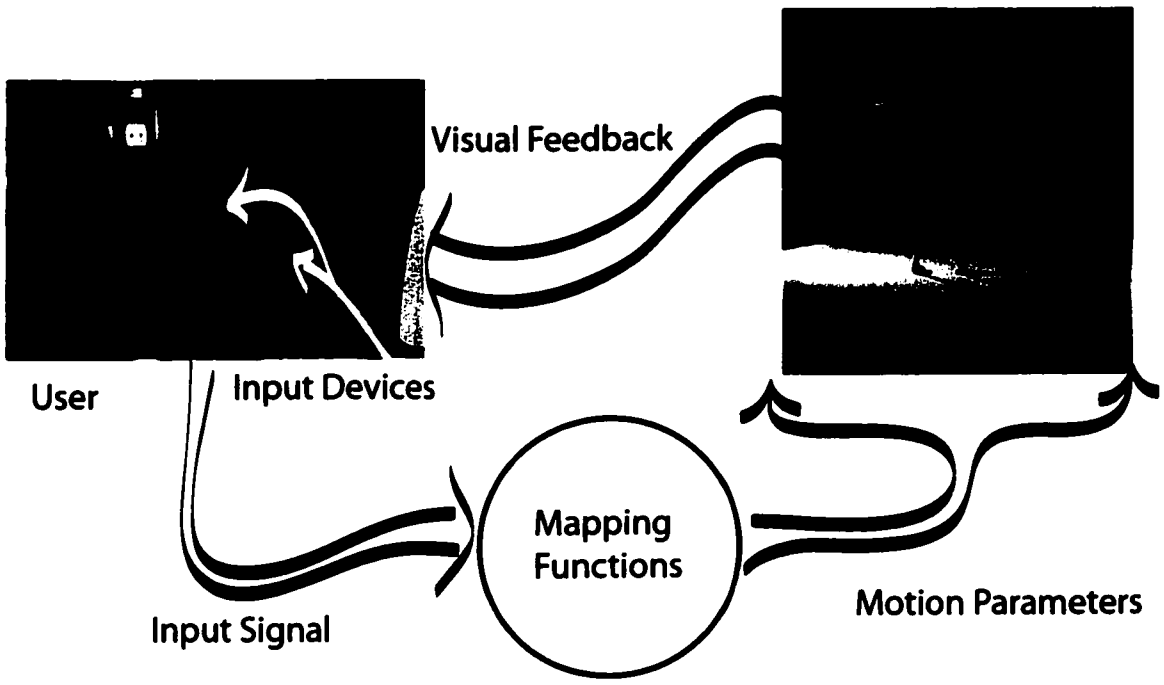

Figure 1.1: PERFORMANCE ANIMATION SYSTEM OVERVIEW

Computer puppetry enables "non-literal" relationships between what the puppeteer is doing, and what the character on the screen is doing, e.g. the puppeteer uses his arms to move the virtual puppet's legs. This diagram illustrates the tight feedback loop that occurs: The puppeteer moves the input device, and that signal gets converted almost instantaneously to make the character move. The puppeteer sees this motion, and uses it to guide his subsequent motion.

0. *User starts application: A skeleton model (virtual puppet) appears on the screen, along with several GUI's offering* **play/stop/record/...** *functionality. Each GUI corresponds to a distinct animation channel.*

1. *User selects* **channel 1** *for controlling the legs of the skeleton. Selects* **activate** *to start a live input feed from the trackers to the selected channel. Selects* **record**.
   Elapsed Time: < 10 *seconds.*

2. *User manipulates two trackers with his hands, thereby controlling the motion of the skeleton, to create* 60 *seconds of animated leg motion, so that the skeleton walks across the screen forwards and backwards.*
   Elapsed Time: 60 *seconds.*

3. *User presses* **stop** *and* **rewind** *on* **channel 1**.
   Elapsed Time: < 10 *seconds.*

4. *Repeat steps* 1. *to* 3. *for* **channels 2,3,4** *to create motion layers for the arms, wrists, spine and head.*

Total session time: *Under* 10 *minutes.*

Result : 60 *seconds of animation data of a walking figure, with* 30 *user-controlled degrees of freedom of the available* 33 *per frame, plus time code.*

The above script describes the final process of creating the short animation in the demo video using the virtual puppetry system we present in this work. One of the striking

features of this session is the relative combination of efficiency, naturalness of the motion, complexity of the motion and resolution of control that is available to the user. The entire session was accomplished in a desktop environment as shown in the left image of Fig 1.1.

1.2 Animation and Performance Animation: a Motivation

One of the essential qualities of good animation is believability. With this as *primum mobile*, even the need for realism is secondary, for an animated sequence needs to be realistic only if that is what will make it believable. There is nothing realistic about a mushroom bending its stem on its own accord, nor does the simple swaying of its cap necessarily evoke any strong response, yet the memorable Mushroom Dance in Walt Disney's "Fantasia" [Dis40] is a beautiful piece of animation in which a family of mushrooms was unquestionably brought to life and infused with personality as they danced around in circles to music by Tchaikovsky. Nor is there any implied personality in the adjustment of the base-, mid- and neck-angles of a jointed lamp, yet in the Academy Award nominated computer animation "Luxo, Jr." [Pix86], two such lamps are brought to life with very distinct personalities, desires and a familiar relationship that we immediately recognize as child and parent. In both cases, the animations are successful to a large extent because we believe them. Provided the motion is imbued with certain qualities, we are capable of attributing human or living characteristics to nearly any moving form[1], whether it is a Luxo lamp (as in "Luxo, Jr." [Pix86]) or a deformable rectangle (such as the magic carpet in "Alladin" [Dis92]).

Precisely what determines believability may be elusive, but an essential element of successful character animation— an element "at the heart of character animation"— is

[1]See [Wilb] for a discussion about anthropomorphic animation.

the character's acting ability, a quality that is often jeopardized as the animator becomes further removed from control of the character [VB86]. Yet in many computer animation techniques, a common theme is trading the resolution or granularity of the user's control for some sort of automation. This can take the form of planning motion trajectories by incorporating criteria such as energy, comfort or strength (see [BPW93]), or by interpolating through keyframes. In the former case, a difficulty pointed out by Badler et al. is that "given its range and diversity, human motion is not optimal with respect to a single criteria"[2]. Regarding the latter, Sturman and Zeltzer [SZ94] write that "Linear or spline interpolation generates the frames between keys. The relative smoothness of the interpolation tends to give these animations a subtly unnatural quality, not quite mechanical, but not quite living." Yet another automation process is procedural animation, about which Sturman and Zeltzer add that, "Programmed (or procedural) animation yields motion that is occasionally life-like, but often too regular to be a product of life itself." Frank Thomas, a great animator (who worked on such classic films as "Snow White" and "Bambi") and one of Walt Disney's "Nine Old Men", said that "surrounded with computer generated graphics, the animator has become anyone who moves an image on a monitor. [...] if it moves, it has been 'animated.' " [Tho87]. He reminds the reader that animation is giving life to things, filling them "with zest", communicating. At the same time, he points out that "Tony de Peltrie", a classic film using computer animation, "did not amaze audiences because he was made up of so many intricate moves as much as the fact that he told a poignant story." Van Baerle writes, "there is an art to creating postures and motion that can elicit an emotional response", and learning this art can take years of experience[3].

As much as acting ability is a crucial part of character animation, a crucial aspect of a character's acting ability, in turn, is *timing.* Animators use this word to refer to how

[2][BPW93], p.163.
[3][VB86]

actions are embedded in time. For example, does the character "open the door slowly then run in" or does she "open the door quickly, then pause before very slowly stepping in"? Or does she "open the door very slightly, pause, then swing it open the rest of the way and run in?". Timing is the first in Thomas and Johnston's list of "Principles of Animation" [TJ81], and is one of the most important and elusive qualities of great animation[Duf00] (as it is of acting, comedy and just about any performing art).

In methods such as keyframing, timing is typically controlled by doing things such as pushing keyframe markers (spline control points) along a horizontal axis. A major advantage of real-time low-level animation is that the animator is freed of having to map time onto a spatial representation. Timing can be exquisitely controlled by using timing.

Real-time input is not exclusive to performance animation; motion capture[4] offers the same benefit. A fundamental difference, however, is that performance animation also provides instantaneous feedback, and this is crucial for allowing the puppeteer to adapt, correct, and genuinely play with the motion as it is being controlled. Reynolds [Rey86] describes this as as follows:

> The animation system and the user form a feedback loop, each reacting to the other in real-time [...] Such a technique gives the animator/performer a very direct, intuitive and expressive control over the dynamics of the imagery.

In Jim Henson's *Waldo* animation[5], for example, "the injection of the vitality and spontaneity of puppetry gave *Waldo* an expressiveness and timing that would require a laborious effort in a standard animation environment"[Wal89]. The very essence of a puppeteer's talent is bringing objects to life by moving them; and computer puppetry allows the puppeteer to generate computer animation in the same way. By controlling motion through motion, the timing and subtlety of the human controller can be given to the animated character.

[4] A technique described in some detail at the beginning of Section 2.2.

[5] See Section 2.3 for a description as well as more examples.

Given the value of performance animation's intuitive control over aspects as important as a character's timing, why has performance animation— despite its slowly gaining popularity— not become a standard or commonplace animation tool? At least part of the answer can be found in the following excerpt from Maiocchi [Mai96] in a survey of motion capture techniques, in which he describes computer puppetry and explains why it could never be used for controlling articulated human character animation:

> [...] sometimes the results of one type of manipulation are saved and then, while that animation plays back, a second layer of motion can be added. For instance, eye movements can be recorded after the character's facial expressions have been established. [...]
>
> Digital puppetry is particularly appropriate when the characters to be animated are simple and their range of movement limited, or when special cartoon effects are to be added to the basic motion. The great number of degrees of freedom that need to be controlled for complex human motion does not make digital puppetry a viable solution for realistic looking animation[6].

It can be seen from this that although the idea of multi-tracking was already familiar at the time the above survey was written (having been used in the music industry for many years), that alone was clearly not enough to determine how to use computer puppetry to animate an anthropomorphic figure. Thus, if something as sophisticated as human character animation with many degrees of freedom needs to be controlled by real-time input acquisition, the only way of doing this is by using full motion capture. This in turn means that a motion capture studio is necessary, further limiting the use of this approach to specialized, equipped production houses.

The challenge is that puppetry has not been able to handle the large number of degrees of freedom such as those needed for human character animation. Yet, as long as

[6][Mai96, pg.28]

the actor can see the motion of the puppet, he should be able to use any input means available to control it. If we could truly capitalize on this, it could have a significant impact on the typical animation process by bringing performance animation to a more easily realizable environment. Providing a desktop interface to this high-dimensional space is the issue which our work addresses.

1.3 Contribution

This thesis makes the following contributions:

- We develop a novel desktop performance animation system, DIGITAL MARIONETTE, which enables a user to control complex human motion effectively, using a minimal input configuration. Specifically, a set of layered mappings is designed allowing two 6-DOF trackers to control a 33-DOF character in multiple passes.

- A novel tangible input device is designed which provides the user with kinesthetic, visual, tactile and inertial-based feedback necessary to achieve the real-time continuous control over the animation task.

- Local physically-based motion actuators are introduced and developed. They augment an existing kinematic mapping to facilitate the animation task and enhance the naturalness of the motion. Additionally, they are designed such that they do not entail the additional cognitive overhead on the user of requiring the balance control problem to be solved. Thus we integrate kinematics and dynamical principles for more effective real-time animation control.

- An extensive discussion and set of general principles are presented in regard to the process of learning complex, continuous interfaces, focusing on that of DIGITAL MARIONETTE, from the perspective of an experienced user.

- We have developed a powerful tool which, via its unique tangible interface, local-physics based models, and layered mappings, allows very efficient creation of human character animation in real-time ranging from walking to dancing. The animation created with DIGITAL MARIONETTE is very expressive, and has already been used in live performance.

By demonstrating a performance animation system that provides a solution for animating an articulated human character, we will have provided a fundamental starting point for further research in this direction, to explore and develop more such controllers with different qualities and underlying mechanisms.

1.4 Outline

In Chapter 2, we present a more detailed background on other animation tools, further demonstrating the importance of computer puppetry as a real-time interactive animation approach. Some existing computer puppetry systems are discussed, as well as the problem of interactive control over dynamic simulation.

Chapters 3 to 5 present the design and implementation of the DIGITAL MARIONETTE system, following a progression from the user input and interface (Chapter 3), to the mapping functions (Chapter 4), and the incorporation of physically-inspired modeling into the mapping to achieve more natural motion (Chapter 5).

The DIGITAL MARIONETTE user experience and results are considered with respect to the original goals, i.e. efficiency, quality and interactivity, in Chapter 6. The focus is primarily from the perspective of a single experienced user, although the experiences and results of several novices are also described.

A summary, future work and conclusions are given in Chapter 7.

Chapter 2

Background and Related Work

We can place computer puppetry within the more general context of animation tools/control methods as illustrated in Figure 2.1 (overleaf). The majority of discussion in this chapter is ordered by traversing the tree shown in this figure, approximately going from distantly related (but very traditional) tools, to more closely related systems. The second level in Figure 2.1 has two branches which distinguish between tools that incorporate real-time control vs those that are entirely off-line, typified by keyframing methods (Section 2.1). Among those methods that do incorporate a real-time control mechanism, we can differentiate between those that are not interactive, such as motion capture[1], which is discussed in Section 2.2, and those which do provide a continuous feedback loop, and are thus fully interactive, such as computer puppetry. This feedback loop (or lack thereof) implies a wider range of possible control mappings (e.g. more abstract or non-literal mappings) as well as differences in the overall workflow. Indeed, as highlighted by Figure 2.1, it is the interactive element of computer puppetry that provides the user with a vitally different level of control, and thus establishes computer puppetry as a fundamentally different approach. Almost all computer puppetry systems are based

[1]The term "motion capture" has been used in more than one way. See the beginning of Section 2.2 for the definition of usage in this work.

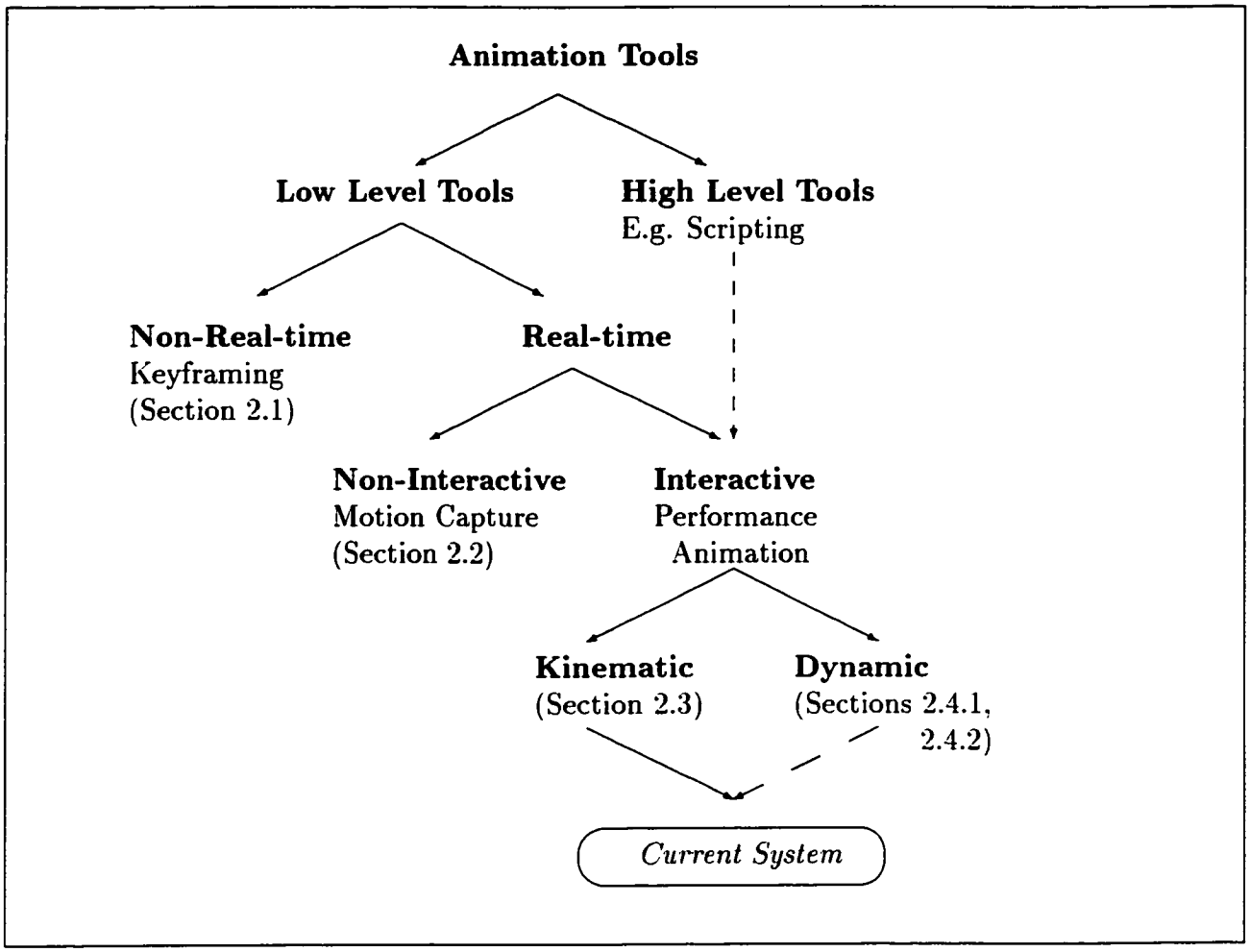

Figure 2.1: COMPUTER PUPPETRY AS A TOOL FOR CONTROLLING ANIMATION
The figure above provides a context within which performance animation appears as a fully-interactive real-time method for controlling animation. Furthermore, our system is based primarily on kinematic control, but it incorporates a hybrid approach that takes advantage of dynamic-based models. Although our work focuses on low-level control, the dotted line indicates that higher-level interactive systems are also possible (and can in fact be complementary to our work, as will be discussed in Section 2.3). The more general issue of high-level scripting tools for animation, however, is outside the scope of this review.

on kinematic control, and corresponding background is given in Section 2.3. Research has also just begun to emerge on interactive dynamic character animation, discussed in Section 2.4.

2.1 Non-Real-time Animation: Keyframing, Parametric and Kinematic Control

An idea basic to many computer animation systems is *keyframing*: specifying the character's position at discrete "key" time points (keyframes) in the animation, while the rest of the "in-between" positions get computed by interpolation. Keyframe computer animation is a natural extension of the most traditional and flexible of animation methods: drawing each frame by hand. Historically, keyframe animation began when the earliest animated film producers realized that an efficient way to use their artists was to have the senior artists draw only a portion of the frames in an animation— those frames at which the motion was somehow changing— and let the junior animators draw frames filling in the gaps. If necessary, the senior animators could also tell them about the nature of the movement between keyframes, such as the curvature of a motion or acceleration, and thereby save on the number of keyframes which needed to be specified. Keyframe computer animation works analogously, so that the animator provides a sequence of specific image frames along with the corresponding relative times at which they must occur, and the computer generates the intermediate frames by interpolation.

Hence, computer keyframing requires a way to draw or specify a keyframe, and just as in the traditional practise, a way to tell the computer what to do throughout the "in-betweens". That is, there are choices to be made with regard to:

- the control parameters by which a keyframe is specified,
- the interpolation function between keyframes,

- the design of a user interface;

and it is in these aspects that keyframing systems differ.

In early computer keyframing, the user established correspondences between line segments through which the system would interpolate [BW71, Cat78]. Later systems (such as those described in [OO81, HS85]) provided *parameterized keyframing*, where information about each image $\vec{i} \in \mathcal{I}$ could be specified in a parameterized form $\vec{p} \in \mathcal{P}$ and the interpolation would occur in $\mathcal{P}$-space such that the image could be computed from the parameters, e.g. $\vec{i} = f(\vec{p})$. $\vec{p}$ could represent joint angles, or control points for a shape defined by splines, for example. Note that interpolating through a different parameter space will generally yield a different set of in-between images, and the final motion will have a different timing. The trajectory of an animation parameter $p_j(t)$ as a function of time is referred to as a *motion curve.*

The interpolation function between these keyframes must itself be carefully selected to produce the desired motion. Linear interpolations are sometimes appropriate, but at other times will produce visible jerkiness at transitions from one interval to the next. Cubic splines are common, but cosines and more complex functions are occasionally used as necessary. In some systems [Ste83, Ber87], the animator could hand draw the parameter interpolation function. Bergeron, one of the creators of "Tony De Peltrie"[BL85, Kaw87], observed that the motion achieved by hand-drawing tended to be better than that produced by automatic interpolation.

A problem with keyframing as described so far is the large number of parameters which must be specified across many frames. One solution is to increase the level of the parameterization (ie. reduce the number of parameters). A standard way of doing this is known as *inverse kinematics*— "pull a finger to move an arm"[2]— where the animator controls the position of end-effectors, and the joint values are solved accordingly, ignoring inertial effects. In inverse kinematics, postures can be given in the form of goals (usually

[2][Rob97],page 42.

for the end-effectors), including the following [BPW93]:

- position,
- orientation,
- aiming-at,
- line or plane (ie. the end-effector must lie anywhere on the given line or plane),
- and half-space (ie. the end-effector must be on one side of a plane).

The inverse kinematic problem is usually underdetermined, so further constraints can be incorporated to arrive at a more realistic solution. For example, joint limits can be enforced [BPW93, Section 3.3.1], or balance can be handled by requiring the centre of mass to remain approximately over the support polygon [BM96]. [Zha96] describes parameterizations that decouple the human body's degrees of freedom (ie. orientation of the hand can be adjusted by wrist rotation regardless of the combined shoulder and elbow angles) to simplify the constrained minimization which is needed to compute joint angles. The efficiency of posture specification by pure kinematic approaches is at the expense of giving up some high-resolution control of timing. Sometimes, dynamically-inspired constraints have therefore been incorporated, such as optimizing comfort as a function of available strength [BPW93, Section 5.3],[KB96].

The parameters controlling the animation can be abstracted even further. For example, locomotion can be controlled by specifying footprints (which entails an underlying motion model, of course), as described in [vdP97, BPW93, KB93] using either kinematic or mixed kinematic and dynamic models (a footprint-to-locomotion system is also sold by Kinetix[3]). Alternately, a scripting language can be used, where a single command implies a set of goals for multiple end-effectors, ie. "sit","stand", "walk forward". Although efficient for specifying movement, high-level scripted systems "embed the possible motions of the model in the model description itself"[4] and therefore cannot provide the

[3]BIPED is part of their CHARACTER STUDIO package ([Rob97],page 48).
[4][Stu87]

" 'natural' motions that professional animators favor in their productions"[5].

Keyframe and kinematic approaches suffer principally from a clumsy, unintuitive handle when it comes to embedding motion accurately in the time domain, so they typically require laborious fine-tuning before they look natural. Satisfactory control of the timing of keyframed motions requires punctilious positioning of the keyframes and may often also require significant distension of the resolution (with respect to time) by the addition of more frequent keyframes. Copying or adjusting the timestamps of a keyframe can pose a number of difficulties [Stu87]. First, it may change the length of the entire animation. Second, not every keyframe needs to represent the values of all parameters, so when the key being adjusted only contains information about some of the parameters, subsequent keys can get out of sync. Third, the lengthening or shortening of a keyframe interval can require a compensating adjustment of a later interval. Additionally, since the interpolating function presumably creates in-between frames at a fixed rate, simply lengthening an interval in order to slow down a motion could cause unexpected artifacts in the motion, as new frames will be generated. If, furthermore, a spline function is being used to interpolate between keys, then moving the position of a single keyframe can have significant effects on the motion even a couple of key intervals away [Stu87]. If the interpolating functions are not splines, then slopes of two adjacent intervals must match at the keyframe which defines the boundary between them, or the motion will not be smooth. This can be handled by having selectable degrees of continuity, or allowing the specification of acceleration, or an automatic ease-in/ease-out function for matching slopes. After all interpolation appears to have been adjusted, the creation of life-like qualities and idiosyncratic motions will still require another iteration of fine-tuning of movement in the motion editor. Thus, it is only with painstaking manual doctoring that a natural looking motion can be achieved.

Given the inherent trial-and-error nature of the method, clearly the slightest ineffi-

[5][Stu87]

ciency in the specification of keyframes will become an obdurate clog throughout the rest of the process. It is therefore crucial to have a powerful interface for specifying keyframes. Some of the early keyframing systems which provided interactive tools for doing this were the BBOP system [Ste83], GRAMPS [OO81] and the EM system [HS84, HS85], all of which are described in some detail by Sturman in [Stu87][6]. BBOP allowed the direct manipulation and control over the 3D transformation matrices describing an articulated figure by using a joystick to navigate through the transformation tree, and a point in 3D could be interactively moved. There was no way of specifying constraints to prevent limbs from getting disconnected, and dragging a point in the direction perpendicular to the plane of the screen was confusing. Using a 3D model in the keyframes produced better results than 2D models due to the loss of information in the latter case [Stu87]. GRAMPS [OO81] allowed several devices to be used simultaneously (dials, tablet, joystick) for input, and for each device the animator could assign the parameters it was controlling as well as the range of its control. Assigning two dials to control the values of two joint angles, for example, provided a much more efficient way to specify the pose of a double jointed figure than was previously possible. The EM system further extended the flexibility of the animator's control to express constraints and relationships between objects (ie. arms must swing in opposite directions) which could then be controlled by multiple input devices or by a script.

Although keyframing is indisputably an essential and fundamental method for creating animation, we will now discuss how continuous real-time input technology provides a very efficient way of creating believable animation.

[6]The reference by Stern, [Ste83], was unavailable, and the information given here is based on the description in [Stu87]

2.2 Motion Capture

The animation tool most closely related to computer puppetry is motion capture. However, the term "motion capture" is loosely defined, with the most common uses being:

1. any form of animation control where continuous real-time input is used to generate corresponding animation parameters; and

2. control of character animation (typically human) by recording/measuring the motion of a corresponding "live" character, and then establishing a direct, or literal, correspondence between them, e.g. the actor's shoulder and elbow angle control the CG character's shoulder and elbow angle, etc.

According to the first definition, puppetry qualifies as an instance of motion capture. To avoid ambiguity, we use the second definition. That is, in this work, *motion capture* refers to situations where there is a direct mapping from a human controller (actor) to a human figure being animated, with no visual feedback. We reserve the terms puppetry and performance animation for the cases when visual feedback is provided.

Motion capture has its roots— like keyframing— in the earliest days of animation in a procedure known as rotoscoping. Traditional rotoscoping involved copying or tracing images from a video or moving reference, thus acquiring desired features, e.g. joint positions, straight from the sampled data. Problems such as timing control as discussed in Section 2.1 were surmounted transparently by maintaining a direct correspondence with the timing of the originally recorded motion. The limitation of rotoscoping, however, was that it required a video or film of the motion which needed to be animated, and this video had to be sufficiently accurate that it could indeed be used as a reference (with the best results obtained if it showed exactly the desired motion). The footage also needed to be from the same camera angle as would be used in the animated sequence.

Motion capture differs from rotoscoping in the sense that the points being tracked are generally well marked, e.g. either with reflective tape or sensors of some sort, but the

underlying principle common to both approaches is the idea of using a real motion as the specification for an animated one. Motion capture can be carried out with any input device capable of recording motion in real-time— optical, electromagnetic, fiber-optic or other. The classic biomechanics text by Winter [Win90] includes a thorough discussion of acquiring and processing human motion capture for biomechanical analysis. [Stu94] gives a historical account of applications of motion capture for character animation since the late 1970's, and [Mai96] surveys many of the systems and technologies used. A recent discussion which also gives some implementation details and common file format descriptions can be found in [Men99].

Motion capture offers advantages paralleling those of its analog precursor in terms of potential efficiency and naturalness of the motion, and has thus become very popular for commercial animation projects (it is far too costly for most other applications). However, motion capture is also subject to the same underlying limitations as rotoscoping; any system that uses existing motions to drive animation is necessarily limited by the quality and accuracy of the primary motion sequence (or sequences) conjointly with the system's ability to transform and manipulate motion into desired animation. The first limitation— usability of the primary motion sequence— can be quite significant. Captured data often cannot, for one reason or another, be used in its raw form for the desired application (motion capture has even been referred to as "Satan's Rotoscope" [Fla97]). Hence the second requirement— the ability to correct, transform, modify the data into a usable form— is crucial, and is an important area of research, as described below.

The first artifact which must be handled in correcting motion capture data is the noisiness of the sensor data. There are various ways of dealing with this. Early approaches used low-pass filters: Winter [Win90, Section 2.5.5] describes the use of second order linear, constant coefficient, difference equations of the form

$$y_t = a_0 x_t + a_1 x_{t-1} + a_2 x_{t-2} + b_1 y_{t-1} + b_2 y_{t-2} \tag{2.1}$$

where the y's represent filtered values and the x's are raw values. The a's and b's are constant values, summing to unity, which determine the filter characteristics. Results are shown and compared using coefficients corresponding to a Butterworth-type filter as opposed to a critically damped filter, with some discussion on selecting cutoff frequencies and using a second reverse pass to get a zero-phase-shift filter, with specific attention to locomotion data. For a comprehensive discussion of filtering techniques for general signal processing, see e.g. [RG75, OSB99].

As motion capture has become an increasingly common approach, more specialized filtering approaches are being developed. Bodenheimer et al. [BRRP97] deal with noisy measurements by using a robust statistical one-step M-estimation procedure as described in [HRRS86]. Sul, Jung and Wohn [SJW98] show how an extended Kalman filter framework can serve to make two main contributions in dealing with motion capture data: both filtering the noise inherent to the data, and also handling physical constraints such as joint angle limits. Such constraints are integrated into the filtering mechanism by treating them as special kinds of measurements with very low uncertainty, therefore pulling the state estimate towards satisfying them. They also suggest applying this filter to smooth motion transitions when concatenate motions; for sufficiently similar starting and endpoint, this seems like quite a reasonable idea.

Yamane and Nakamura [YN00] describe an on-line dynamics filter that takes as input a motion sequence, such as would be obtained by motion capture, and adapts it so that it is physically feasible. Their approach is based on dynamic computation with structure-varying kinematic chains (as described in a previous paper [YN99] by the same authors). The basic idea of creating a dynamics filter to serve this purpose is an important problem. However, this paper leaves the reader with some unanswered (or perhaps obscurely answered) questions, such as: how are they able to handle issues such as balance, especially keeping the filtering algorithm processing on-line without looking forward? They do acknowledge that to apply a normal motion captured walk to a 5-degree incline, they

needed to manually modify the data to keep the toe from hitting the ground. So there must also be some limits to the filter's robustness with respect to the noise in the data, which is reasonable, but it is not clear how constraining these limits are. For example, for how many seconds can their system follow a walking motion before the dynamically-simulated character will fall over?

Another source of error in motion capture, specific to optically-based methods, is the confusion in identifying reflective markers worn by the actors. In [HFP+00], points are tracked from multiple cameras in 2D image and 3D reconstruction, and multiple correspondences are established, all based on a sophisticated skeleton model of the actor, to end up with an automated tracking system that can be very robust against potential misidentification, such as those often due to occlusion. The approach taken by Herda et al. in [HFP+00] is built on the previous work in [SPB+98] for properly determining an accurate skeleton model in the first place based on motion-captured calibration data, which they called a "gym" motion sequence.

This latter problem— of constructing a graphical character model that matches the source data— brings up another crucial issue that invariably arises when working with motion capture input: a primary bottleneck is often correcting the data to compensate for the inevitable physiological differences between the actors and the characters they are animating[DMZ95, Dye97]. If an actor scratches his head, for example, then the same shoulder/elbow joint-angle data might cause a large-headed alien character to drive his hand straight through his skull. If that is not the desired effect, then the data has to be carefully adjusted by hand. For the same reasons, the new characters' feet often might not even touch the floor! This problem was addressed directly by Gleicher in [Gle98] wherein he proposed a method for reusing existing motion data on new characters by mathematically specifying constraints, and then minimizing the sum of the squares of the differences between the new and original motion curves while maintaining the constraints (an application of the spacetime constraints method introduced by Witkin

and Kass [WK88]). By using motion-displacement maps to represent a new motion [GL98] and using B-splines to represent the difference between the two curves, the frequency spectrum of the changes can be implicitly limited.

In [BB98], Bindiganavale and Badler describe a sensible way to treat contact-with-objects for the motion retargeting/abstraction problem. They begin by using zero-crossings of some of the acceleration of the motion parameters to narrow down possible moments of contact. Having determined these contact moments, constraints are computed to maintain the correct relationship between the character and whatever he or she is touching until the object is released, for example. A nice point is that they try to maintain a similar velocity profile by renormalizing the angular velocities of the retargetted motion based on the primary data. They also provides a good first guess for the character's gaze by checking if any objects in the "real" world fall within the line of sight of the actor, and if so, then redirecting the graphical character to look at the corresponding object in the graphical world. Admittedly, where and how the character is looking, the orientation of both his or her head or eyes, is absolutely crucial to the communication of the acting, and trying to look directly at objects all the time might be a good starting point, but there is a lot more subtlety in this aspect that cannot be automated quite so easily. It is also hard to automate this because the primary motion data might not include the gaze, but rather only the head orientation. One possible extension would be to project a cone from the character's eyes, to determine objects that fall within the region of view, and try to maintain a corresponding relationship in that regard. This could capture if the character is looking just slightly above the cup she or he is holding, but might get more confusing as the scene is filled with more objects. Nevertheless, the importance of the relationship between the character and the objects being contacted, whether belonging to the character him/herself or to the external world, is definitely important in reconstructing scaled motions, and their approach is helpful to this end.

A related question is that of obtaining a new motion given a fixed library of recorded ones. Guo and Robergé [GR96] blend two motions to create a third, different one. In particular, they use an interpolation scheme between two different walks and two different runs to create different walks. A critical step in their method is to first establish correspondences between key "events" in each sequence, ie. the point of weight-transfer in a walk corresponds to a point of no contact in a run. These correspondences are carefully done by hand. In [Per95], Perlin describes a blending mechanism (which we discuss in more detail on page 24) in which new actions can arise as a sustained transition between two or more basis motions. A remarkable example describes how, by transitioning from a "running" motion and a "standing at attention" pose, a visually realistic walk is achieved.

In contrast, Witkin and Popović [WP95] provide a method for modifying a single given recorded motion so that it satisfies new constraints (added after the recording) such as modifying a straightforward walk so that the character bends down to step through a low doorway or to step over an obstacle. They base their work on the assumption that "much of the 'aliveness' of captured motion [...] resides in the high-frequency details,"[7] so they warp the captured motion sequence in two ways such that these high-frequency details are indeed preserved. The sequence is first warped in time, $t = g(t')$, where g is a spline function through a set of time correspondences $\{t'_i, t_i\}$ specified by the user. Then, each motion curve $p(t)$ (see page 14) is warped using a transformation of the form

$$p'(t) = a(t)p(t) + b(t)$$

where $a(t)$ and $b(t)$ are splines controlled by values $\{a_i, b_i\}$ which are specified by the user as scales and offsets for selected keyframes.

Bruderlin and Williams [BW95] explicitly describe the notion of applying a wide variety of signal processing techniques to motion data. In particular, they demonstrated the use of multiresolution motion filtering, multi-target motion interpolation with dynamic

[7] [WP95, Sec. 1]

timewarping, waveshaping and motion displacement mapping.

Methods such as those used in [WP95] and [GR96] seem to offer a wide range of possible transformations, but might still be tricky to use in order to get a particular desired motion. In [GR96], for example, the effect of the interpolation is difficult to visualize based on looking at the interpolating weights. That is, one might deduce from a graph that the defined sequence will generate a motion which begins with a walk and becomes a run, but it is still hard to predict exactly how that will look. The created sequence therefore usually needs to be transformed iteratively until the desired sequence is obtained. A real-time feedback loop between user and computer potentially overcomes this problem by making it easier to get the right sequence the first time around.

In addition to signal noise, marker identification, retargeting to different skeletons and generating new motions, another important issue is stitching together independently recorded segments in sequence. Perlin [Per95] presents a natural way of blending two or more motions (not necessarily obtained by motion capture) to generate a smooth transition from one to the other. Each primary motion is given by a sequence of joint-angle values over time. The value of a joint angle in the blended transition is then obtained by taking a weighted sum of the values for that angle in the motion sequences being combined, essentially cross-fading between the two motions in joint-angle space. This also allows actions to be combined with layered starting times, so that the character can begin to make one motion while still executing a previous one, suggestive of simulating a co-articulation between motions. Dependencies are implemented to avoid averaging irreconcilable motions such as stepping onto the left foot at the same time as stepping onto the right one. In the motion warping described previously, Witkin and Popović [WP95] also use a technique similar to Perlin's for effecting smooth transitions from one warped sequence to the next. Rose et al. [RGBC96] use a nice combination of space-time constraints and inverse kinematic constraints to achieve "seamless and dynamically

plausible transitions between motion segments"[8]. In both [RGBC96] and [Per95], for any given motion there is a set of degrees of freedom that can be left undefined (e.g. a head-scratching motion does not need to define leg parameters), allowing proper layering of concurrent motions more naturally. Finally, Lee and Shin [LS00] describe a way of using multi-resolution signal processing to achieve motion blending and transitions.

2.3 Performance Animation

2.3.1 Brief History of Virtual Puppetry Applications

Sturman [Stu98] gives an informative overview of both the history of computer puppetry and some of the technical challenges that need to be overcome. He notes that one of the first real-time interactive graphics systems was SCANIMATE, developed by Lee Harrison III [Vas92], which was used for almost all flying logos shown on TV in the late 60's, 70's and early 80's. The logos were generated by a dancer wearing a datasuit built out of potentiometers and tinker toys. Its popularity declined as it could no longer compete with the slicker imagery being created, but the ensuing systems lacked the advantage of real-time control which it had afforded.

Mike the Talking Head [deG89, Rob88] was built by deGraf and Wharman, and consisted of a CG head for which all parameters— rotation, expressions, lip synch— were controlled by a single puppeteer. Control over the animated character was demonstrated by a live lip-synch to an opera aria.

A computer puppet was also first used commercially in 1989 when Pacific Data Images (PDI) created *Waldo C. Graphic*, a character for a Jim Henson production [Wal89]. The computer puppet was controlled by a single Henson puppeteer using a mechanical arm, and upper and lower jaw attachments. The animation was first performed with

[8][RGBC96]

a low-resolution rendering, and the puppeteer received his feedback on a video monitor display which overlayed *Waldo C. Graphic* with the other "real" puppets who were being performed and taped at the same time. The recorded animation data was then processed, and eventually composited with the video source.

Since then, real-time performances have continued to be used for TV, film, trade shows, industrial sales and marketing [Stu98]. Two prominent performance character animation houses include a company formerly known as Medialab (based in Paris, France), and Protozoa [deG] (now known as DotComix).

At Medialab, *Matt the Ghost* began a 4 year run in 1991 [Tar91, Stu98], for whom 3 puppeteers would create 5 minutes of material in an afternoon session— a formidable output ratio by production standards. Medialab also produced "Donkey Kong Country", a series of 26 episodes (10 hours), with 11 characters, totalling 900,000 images, and an average of 140 seconds of real-time-based animation was recorded per day [BJ97]. 70 per cent of this animation was done by computer puppetry, 30 percent by keyframe, and the total time spent on each was the same.

Some of the most interesting and creative documented explorations of computer puppetry, with really fun, artistic sensitivities, have been at Protozoa. Their examples of non-literal animation control have ranged from using "the old Vaudeville horse method to perform four-legged creatures" to puppeteering a worm-shaped character with sensors at the actor's hands and feet [dGY99]. Protozoa also sells ALIVE, a performance animation software with which it has created characters such as *Moxy* (who hosted a cartoon network) and numerous others [Rob96].

2.3.2 Interactive Character Animation Techniques

As suggested in the history given above, for many years computer puppetry systems generally existed only in the context of commercial projects. Thus much of the information about their development and engineering often remained behind the closed doors of

industry, so that very few computer puppetry systems were described in the literature, and when described, almost no details were ever given about the mappings being used. There are, however, a few exceptions, and more recently some newer research-oriented puppetry systems have been emerging, which we now describe.

In [BJ97], Benquey and Juppé explain that the monkey characters in Medialab's Donkey Kong Country series were animated by two puppeteers each. One puppeteer controlled the facial animation:

> Each of the puppeteer's fingers plays the same specific expression for all of the characters, that range for instance from happiness (left thumb fully opened) to sadness (left thumb fully bent). Anger, smiling, surprise, sadness and blinking can thus be combined to provide a full set of moods. The basics are the same for all the characters, but it takes a lot of practice to give a specific personality to a character. A little bit of surprise in *King K. Rool*'s anger will make him look crazy, just like a bit of sadness in *Donkey Kong*'s surprise will make him look really... like a chimp! The puppeteer is also in charge of the lip-synch (two fingers) and the eyes (joystick).[9]

(Exactly how all of this is implemented, of course, is not disclosed). The second actor wore 16 sensors covering his arms, torso, head and legs, as well as gloves for controlling the hands. A more literal mapping was thus used for the character's body, where:

> [...] Here again, a lot of practice is required, since a fat crocodile with a big reptilian nose, short legs and long arms can't move like a walking chimp with a flat face, long legs and arms[10].

Another relatively documented commercially-developed virtual puppet is *Cracker*, a "genetically mutated lizard cow" developed at Protozoa [Mor98]. Morasky gives a

[9] [BJ97, p.13]
[10] [BJ97, p.13]

concise but detailed description of the mapping that takes input from 5 electro-magnetic trackers worn on an actor's arms, feet and pelvis, to control the legs, head and spine of a mutated lizard cow. The foot data is left untouched in order to keep the character's feet on the floor, while the hips are translated and rotated off of the pelvis according to a specialized interpolation between the three lower-body trackers. The angle of the knees, which bend backwards, is obtained by a basic inverse kinematic computation. The torso orientation is controlled by the relative orientation between the left hand and the hip sensors, while the head is controlled by the orientation of the right hand sensor. Ankle motion is not discussed; the ankle is probably either maintained parallel to the ground, or its global orientation may be taken directly from the foot sensor data, if that is possible in this case.

In "Luxomatic" [Wila], Wilson describes a nice system which extracts the 2D position and orientation of the user's hand from a video stream, and maps this onto the end-effector of a Luxo lamp. The depth is kept fixed, and a Kalman filter smoothes the orientation information.

In [GGaQ00], Gildfind et al. describe an interesting approach in which genetic algorithms (e.g. [Koz92]) are used to evolve kinematic mappings for specific desired motion sequences. Each mapping is represented by a genetic code, and the procedure begins with a population of candidate mappings. The user evaluates each mapping in a population, and individuals considered "promising" are multiplied (via reproduction, crossover and mutation operators), while the less promising ones are culled from the population. Users can specify their feedback about any given mapping either by a single value, or by a set of values indicating their satisfaction with each component of the mapping. By repeating this procedure over multiple generations, the later populations become increasingly satisfactory, until the user eventually chooses to select a particular mapping as the final result of the search procedure. A sample demonstration could be a helpful indicator of the resulting animation, since, for example, although the authors mention animating a

walk, nothing is mentioned regarding maintenance of the ground constraint or root motion. Nevertheless, the underlying idea is interesting, and, as will be discussed in some detail in Section 7.2.2, it is both in line with some of our preliminary experiments, and makes use of elements which are enticingly complementary to our current approach.

In Figure 2.1 on page 12 we show that interactive character animation can result not only from low-level real-time control, but also from using higher-level parameterizations. For example, in mapping from an input space of dimensionality $\mathcal{N}$ to a character posture output space of larger dimensionality $\mathcal{M} > \mathcal{N}$, one natural approach is to find a lower-dimensional parameterization of the state which can span the desired range of animation. This means collapsing some degrees of freedom (e.g. establishing certain dependencies in the joint angles). If we extend this reparameterization temporally, so that entire sequences of joint angles over time are represented by only a few parameters, it may become equivalent to establishing some motion "units" for a procedural approach. In this sense, the degree of abstraction in the parameterization of the output space leads to a spectrum of possibilities that range from low-level to high-level animation control. It should be stated that the more general issues of procedural, scripted or autonomous character animation are sufficiently outside the scope of the present work that we do not review them here. However, we now describe some contrasting examples of *interactive* character animation in which different levels of parameterization can be found.

Donald and Henle [DH00] describe an interface for navigating a haptic force-feedback device (called the Phantom) to control animation, whereby trajectories in the 6-DOF input space are mapped to trajectories in the animation space. The haptic feedback allows the user to both easily "follow along" the correct input trajectory, as well as modify the trajectory by pulling or pushing on it, thereby changing the animation. An important aspect of the haptic vector field is that it depends not only on the position, but also on the velocity of the input device, and they call this a Higher-Order Vector

Field. These velocities can be used to determine the correct haptic force that needs to be provided to help a user navigate through a region where two trajectories cross (i.e. so that the user can naturally keep going along the current trajectory). The trajectory mappings themselves have been designed by hand, based on a few sample input motions, and the resulting system is really suitable for limited interpolation or variation on those few input motions. Their approach may not scale well as the desired range of target output motions increases, possibly in terms of the design of the haptic vector field itself, and almost certainly with regard to the mappings.

An example of an even more specific motion parameterization is Wilson's "The Seagull" [Wilc]. Here a corresponding feature of the input is being extracted, as Wilson uses two cameras to recover the 3D position of the head and hands of a (stationary) user in real time, allowing the user to "flap" his or her arms to control a parameter representing the bird's flapping cycle relative to two wing positions (up and down) as long as the user stayed in a fixed position and orientation. This also controls the corresponding speed of the landscape video as it scrolls by. Bobbing the head up and down triggers the sound of an authentic seagull mating call.

deGraf and Yilmaz [dGY99] point out the value of procedural animation to achieve secondary motions, allowing more parameters to be driven at once in a live performance setting. For example, one of their game characters is animated just by a mouse which is used for controlling character's speed and direction. The rest of the character's qualities are derived based on these two properties. For example, as he speeds up, "he assumes a more menacing, aggressive posture, hunkering over, opening his claws, twitching his fingers, and blurring his spokes"[11]. In another example, they describe the use of a mass-spring and gravity simulation system to animate the secondary motions of the tail, ears and belly of "Max Rodentae", a peppy little rodent creature. This gives him a nice floppiness, further enhancing his believability as a character, while still allowing the

[11][?]

frame of his body to be animated by a human controller.

Perlin and Goldberg [PG96] present the "Improv" system, a general, real-time interactive framework that can incorporate both high-level and low-level control. Improv is based on the real-time responsive animated characters introduced in [Per95], which we therefore describe first. In this earlier work, Perlin defines basic actions by specifying maximal and minimal values for each joint parameter of the articulated character, along with a corresponding function— either a cyclic sinusoidal-based function, or a coherent noise source [Per85] around a specific frequency— to interpolate between these two extrema. The cyclic functions are used as a simple but powerful way to specify motions ranging from running to a rhythmic rhumba dance motion. The noise functions serve to provide a pseudo-realism without having to specify details of the motion, such as a slight nodding and turning of the head while dancing (e.g. the character glancing around), or a slight restlessness or weight shifting from the hips while standing still. Once a set of basis motions have been defined, a blending mechanism is used to convert discrete action choices into smooth continuous transitions using a scalar weighting parameter. Furthermore, sets of actions can be layered to allow a natural composition of motions, such as "waving", and "scratching head" while "walking". The mapping from input is achieved by allowing the user to trigger these motion elements, e.g. by keystrokes or mouse-clicks.

Improv [PG96] extends this work, generalizing the framework for layering and compositing motions, and providing a system for scripting multiple interactive and autonomous characters. The architecture is based on a high-level behaviour engine and a geometry-manipulating animation engine. An English-like scripting language allows implementing decision rules, stochastic behaviour, and triggering of actions, e.g. built on the basic motions defined in [Per95]. This constitutes a powerful interface for imbuing the characters with personality characteristics which guide the character's response to both other characters' behaviour, as well as to interactive input from the user. Thus, the *author* of the system (that is, the author of the interactive animation story) can choose

the granularity of control provided to the end *user* (i.e. the person interacting with the final animation) as appropriate in different situations, ranging from high-level stochastic behaviour to fine-level motor skills.

To take full advantage of such a framework, at least two important issues must be considered: one is the specification of the basic animation components or elements (e.g. "walk", "standing posture", etc). A second is the specification of the interface itself presented to the end-user; if the user is to control fine-level motor skills at one moment (which is an inherently real-time continuous input task), and trigger high-level behaviour the next, and so forth, then the mechanisms for these different control tasks need to somehow be integrated into a single coherent handle. Our work is very well suited to the first issue of generating atomic motion components, and the second issue represents an important direction of future work in our case as well.

2.3.3 Character Geometry: Non-humanoid versus Humanoid

Typically, when a humanoid geometry is animated in real-time using either standard motion capture or performance animation, the articulation of the character's body is handled similarly in both cases, with the actor covered in a set (or suit) of sensors, and correspondence maintained between the features of the actor and the features of the CG character. The processing applied to the sensor data is primarily to handle the anthropomorphic differences between the two bodies, and puppetry allows the actor to make appropriate adjustments by having the advantage of real-time feedback [Stu98, BJ97]. Our system differs significantly in that while we do retain the complexity of the output by controlling a human articulated figure, we let the puppeteer do it using his *hands*, via a non-literal mapping. By animating a human articulated geometry to walk this way, we have chosen a very challenging testbed. The challenge is magnified because we are highly sensitive to how a human moves [Joh73], and therefore tend to remain unconvinced by unnatural motions, whereas for imaginary creatures, we may not have

such strong expectations about how they move, and hence suspend our disbelief more easily. Hodgins et al. [HOT97] demonstrate that degree of viewer sensitivity to motion variation can even depend on whether a stick figure or a polygonal representation is being used; it seems plausible that there can only be further differences in sensitivities to motion variation depending on whether the model itself is truly anthropomorphic or not. Consider that in a character such as Cracker, for example, part of her overall look is that her knees bend backwards. This is a sensible artistic choice for that character, but it also entails the advantage that the viewer is simply *not going to notice* if Cracker's walk does not exactly replicate how the back legs of a lizard move (in fact, Cracker is an imaginary character, so it doesn't need to replicate a lizard anyway). In contrast, many of the physical enhancements to our model described in Chapter 5 were developed precisely to help recreate various "natural" motions of the legs during walking.

2.3.4 Complexity of Control

In [Mor98], the author writes about *Cracker*:

> The thing I find most interesting about this character is its apparent simplicity. With only 5 EM sensors producing global position and orientation data for the hands, feet and pelvis, the performer is capable of eliciting a full range of motion and expression from such a non-humanoid character

Now for a production house, five sensors is indeed spartan (e.g. compare this with the numbers given in [Stu98] or the 16 sensors mentioned in [BJ97]). By introducing a multi-tracking system, however, we make it possible to control a humanoid character with only two sensors. Of course, multi-tracking introduces complexities of its own, so we do not claim to have a system which is, in some absolute sense, "simpler", but the trade-offs we have chosen lead to a quite different usability. For example, our system could be adapted to a "typical" home environment (see Chapter 7).

2.3.5 Animation Goals and Constraints

The choices made in designing the characters, their motions, and the virtual mappings that will be used for controlling them, are all heavily influenced by the ultimate goals of the animation at hand. This can lead to a wide variety of constraints and considerations. For example, when de Graf and Wharman were animating *Felix the Cat*, many of the challenges arose as a result of making a 3D model of an established 2D cartoon character. *Felix*'s mouth, for example, had to be animated so as to move around his face to always face the viewer, in order to match the "look" of the comic-strip character [Sør89]. This illustrates a case where the model and animation have to respect the artistic design.

The final targeted use of the puppet also has important implications. For example, Sturman classifies commercial systems into two main categories [Stu98]: those which are meant for live performance (i.e. the contents of the image buffer during animation is broadcast immediately), and those in which the performance is recorded and taken for post-processing. The former usually requires fewer puppeteers and real-time performance at more levels of the system. It is therefore more difficult for the performers as they often end up having to control more parameters. An example of a live puppet was *Mike the Talking Head.* In contrast, the recorded systems, usually with a bigger budget, have more puppeteers, can plan for a sophisticated and computationally expensive post-processing phase, but require motions which are "divisible" so that movements of individual body parts can be re-recorded on separate "tracks" as necessary. The post-processing means that the real-time feedback can be quite low resolution (as was the case with *Waldo*) and therefore more complex characters can be attempted.

2.3.6 Incorporating Other Techniques

In creating an animation, computer puppetry is not an exclusive approach, and can be used in conjunction with many other techniques to get the desired results. Except for

the case of live animation (ie. in talk shows), keyframing can be used when appropriate, or when it is easier than computer puppetry; Medialab uses keyframe for the animation of certain objects and acrobatics [BJ97]. *Moxy*, for example, has a rippling lip effect added in post-processing. Many of the Medialab characters also have additional script-driven functions running in real-time on top of the performance animation, particularly for special effects such as bubbles coming out of a fish's mouth [Stu98].

Another approach which can be incorporated is the use of physically-based animation; this is discussed in the next section.

2.4 Interactive Dynamic-Based Control

Authentic simulation of real motion can be achieved by using the classical laws of physics to compute accelerations according to the forces acting on a system in which the masses are known. Issues such as smoothness of motion, keeping proper ground contacts, and matching interpolant slopes across keyframes do not have to be addressed explicitly as they result naturally from the physical simulation. The application of physics to interactive character animation is new, and requires an understanding of some of the issues pertaining to the problem of controlling animation by dynamic simulation in general. In fact, the specific case of interactive control of dynamic character simulation has only begun to be explored, with the work by Laszlo, van de Panne and Fiume [LvdPF00] being a rare example of research in this direction. Thus, the first and primary focus of this section will be on the general issues of dynamic control, and then this latter work will be discussed. Further issues relating to interactive dynamic control, and specifically to ideas presented in [LvdPF00], are addressed in Chapter 5.

2.4.1 Background on Dynamically-based Animation in General

Some of the notable early contributions to dynamic-based graphical simulations of human motion include [AG85, AGL87, GM85, Wil87]. In some cases, such as [GM85, BC89, BPW93] dynamic and kinematic approaches are combined. In the *KLAW* system [Bru88, BC89] for example, a dynamics computation is used to calculate several main parameters during a walk (ie. angle from the vertical line at the hip to the swinging leg, the flexion angle of the knee, etc.), and the rest of the body configuration (including joints below the knee) is derived by kinematics.

The dynamics principles can be formulated in various ways, such as the Newtonian, the Lagrangian, the Gibbs-Appell [Par79], some recursive methods [Fea83, WO82, Arm79], all of which are necessarily physically equivalent (or should be) but differ computationally. Regardless of the formulation chosen, however, a primary limitation of dynamic simulations has been their computational demands, either due to the small timesteps or due to the complexity of the computation at each step. Grzeszczuk, Terzopoulos and Hinton [GTH98, Grz98] circumvent this problem by training a neural network to predict the future state of a dynamical system from its current state and external and internal forces, improving the performance by factors of one to two orders of magnitude.

Although the animation generated by dynamic simulation is indeed realistic, it is also very difficult to control the available parameters in order to achieve a desired motion. While assigning a timestamp to keyframes is hard, the task of defining appropriate control functions (such as the torque $\tau_i(t)$ at time t for each joint i) is even harder when trying to achieve a particular end-effector trajectory. The user interface in the Virya system [Wil86] provided functionality for inverting a specified kinematic trajectory to obtain dynamic control parameters using a simple but effective approximation. One catch, of course, is that this assumes that the kinematic trajectory can be defined in the first place. Ko and Badler [KB96] also use an inverse dynamics computation combined with kinematic

specifications to animate motions, making use of a dynamic balancing technique as well as limiting the required strength.

Approaches for finding physical control parameters frequently make use of optimization techniques. In Spacetime Constraints [WK88] the authors propose a system that allows a user to specify both what the character should do and how it should be done, eg. "jump from A to B, and come down hard enough to splatter when you land"[12], and then use a variant of Sequential Quadratic Programming [GMW81] to solve the resulting constrained optimization problem for the positions and forces. This method, however, requires the symbolic differentiation of the equations of motion, making it not feasible for complex models. Grzeszczuk and Terzopoulos [GT95] use simulated annealing to find low-level physics controllers that produce effective locomotion according to an objective function. They then use annealing again to find macro controllers for higher-level motions by appropriate combinations of the lower-level ones already found. The neural network in [GTH98] allows a gradient-based search through the space to find the optimal controllers for tasks ranging from controlling a two-jointed pendulum to getting a dolphin to swim.

When the objective function is difficult or impossible to define, optimization cannot be used. A physical model, whether a rigid or flexible body, might still be desireable, and this necessitates either a more intuitive parameterization or or a more interactive interface, or, when possible, a combination of both.

There are numerous ways of parameterizing a motion such as walking, and the appropriateness of each depends to some degree on the particular characteristics of the walk. Some different walking characteristics include, for example: stomping, dragging the feet, stiffness, a Charlie Chaplin walk, a Big Bird-style walk, or the way a classical dancer is taught to walk. Though most styles might be quite difficult to describe in explicit physical terms, the first few have some dynamic interpretations: stomping is (in part) related to the force of impact on the ground, dragging the feet is (in part) related to

[12][WK88]

a minimal exertion of internal torque in the joints of the swinging leg, and a stiff walk might be related to minimizing the motion in most of the joints. Fairly high level descriptions of physical principles as applied to classical ballet dance are given in [LS84]. The discussion focuses on concepts such as the location of the centre of mass relative to the points of application of force, and conservation of translational and rotational momenta. This provides a basic intuitive idea of some essential aspects, and there is a bit of quantitative analysis in the appendices at the end. Many of the details such as the necessary interactions between forces applied at different joints, constraint forces, and ground reaction, are beyond the scope of the text, which is truly intended for dancers more than for physicists, and is indeed effective in that regard. However, these details are of course as essential as any others for a full physical simulation.

The KLAW system described earlier provides control for human locomotion by three primary parameters— step length, step frequency and speed [BC89]. Further control is available by specifying up to 28 additional locomotion attributes including lateral distance between the feet, toe clearance, and maximum rotation of the pelvis. The kinematic constraints are not enough to fully determine the internal joint torques, which are therefore obtained according to heuristics from biomechanics [IRT81, Win90]. Note that the result of a simulation can then still be hand-modified by an animator— a step that should be allowed for nearly every method.

In Virya [Wil86], there are a number of modes other than the kinematic one described earlier. In "freeze" mode, for example, certain body parts can be made to remain locally stable during dynamic control by simulating a tight spring and damper clamped to the desired position. In "balance" mode, similar external forces counteract any motion for the trunk away from a desired orientation. This had only limited success, which was probably due to the fact that walking is based on falling forward [McM84a], so an external balance process makes it impossible to walk (as pointed out in [AGL86]).

A classic approach to controlling articulated bodies in physical simulation is by the

use of proportional-derivative controllers at the joint, such that

$$\tau = k(\theta - \theta_d)$$

where the applied torque τ is proportional (by stiffness k) to the difference between the current and desired angles (θ and θ_d respectively). The fact that muscles lead to this spring-like behaviour was proposed by Bernstein [Ber67][13]. In [Hin84], Hinton describes how this can be effected by "angle-torque" functions in real bodies where the internal model does not necessarily correspond exactly to the actual state of the system.

Armstrong, Green and Lake [AGL86] used angle-dependent torques both for rotating a joint to a desired angle, as well as allowing the user to specify a constraint such as maintaining the upper arm at an approximately 90 degree angle while other body parts were being manipulated (which they called a "maintain process"). In their case, the torque function used was

$$T(x) = k_1(e^{k_2(\text{abs}(\theta_c - \theta_d))} - 1) \tag{2.2}$$

where θ_c, θ_d are as before, and k_1, k_2 are set by the animator. The exponential form of this process was based on the muscle studies described in [McM84b, Hat77]. They also added a maximum bound on the torque (according to tables of maximum torques given in [Pla71]) which decreased as the desired angle was approached, and a constraint on the rate of change of the torque, to ensure a motion which was smooth and which slowed down in time. They allowed a user-specified friction constant corresponding to the viscosity described in [Hin84] to slow down joint rotation in proportion to its velocity.

Synergies are certain combinations of muscle movements relating the changes (or rates of change) in multiple joints. For example, the wrist can be moved in a straight line by changing the elbow angle at a rate twice larger and in the opposite direction to the rate of change in the shoulder angle (assuming the forearm and upper arm are of equal length).

[13]In fact, [Gre82] mentions that this could have been proposed as early as [Ber47] although the reference was not possible to obtain at the time.

That is, a side-effect of rotating the forearm around the elbow is that the wrist moves in a circular arc, while a corresponding but opposite side-effect occurs when rotating the upper arm about the shoulder in the opposite direction. Such synergies are discussed in [Hin84], where it is also pointed out that to incorporate multiple synergies, one only needs to add the individual ones. Since kinematics calculates the transformation from the space of joint angles Θ to the space of end-effectors E, a synergy in this case is a parametric solution for trajectories through Θ which lie on a specified manifold (of reduced dimensionality) in E.

A good parameterization alone, however, is still unlikely to be sufficient for a satisfying degree of control over dynamically-based animation. An interactive interface is still needed. Earlier systems attempted to provide this as much as possible under the circumstances. In the VIRYA system [Wil86], the animator could interactively choose joints and use a puck and tablet to select and move control points of the spline representing the torque function. In [AGL86], the animator could interrupt the dynamic simulation and, using a basic mouse interface, adjust either torque parameters or joint positions, or apply a torque or force to a joint while the body is in motion. Motion processes such as the maintain process which apply forces to maintain a joint near a fixed angle (using Eq (2.2)) were provided to facilitate the building of an animation one limb at a time. The principle reason for having this option was because, despite the strengths of the interface, they believed that controlling more than two or three limbs at a time would simply be beyond the capabilities of most people. Their system used a step size proportional to the square root of the values of the inertias, and with inertias approximately 10 times larger than those in a human body, their system ran only 3 to 10 times slower than realtime, depending on the complexity of the figure and of the motion.

Before discussing interactive control with dynamics, several papers are discussed: one which claims to use dynamically-based control to animate a character to dance and compares a few such approaches; another which uses motion capture to guide a dynamic

tracker; and the third of which uses a limit cycle approach to control a dynamically simulated walk.

Review of [MZW99] on Animating Upper Body Motion By Physics

Using full dynamic simulation to control complex character animation is still a very difficult research problem. Related to the kind of "acting" or expressive motions along the lines of those which we try to achieve in this work, we consider the paper by Mataric et al. [MZW99]. The title of this paper, "Making Complex Articulated Agents Dance", is perhaps slightly misleading, as

(a) "Complex" in their case refers to a model which is completely fixed from the waist down, avoiding any issues of balance, or ground contact, or locomotion of the centre of gravity of the character, etc. They also do not check for self-collision or joint-limits, though they claim this has been implemented in future work.

(b) "Dance" refers to the "Macarena" dance. Considering only the arm movements as they do, this is a dance which can be quite easily "defined" in about 12 sequential steps: e.g. stick your right arm out, turn your palm up, touch your nose, etc.

Three techniques are presented for physics-based control, which are then compared and discussed. The three methods are

1. Proportional Derivative (PD) control at the joints. This is the standard approach, and the challenge here lies in setting the target angles, setting the keyframes, setting the stiffnesses, etc. To avoid self-penetration between the arm and the head, they add in extra keyframes by hand, and use splines to interpolate, thus creating the intermediate "target" frames for the controller.

2. Joint-space force-field approach. This is ostensibly based on the work by Mussa-Ivaldi et al. [MI97, MIGB94], which suggests the existence of "basis" motions,

from experiments on spinalized frogs, which can be linearly superimposed to create a variety of desired motions. This is interesting work, but in this paper, only two such primitives are used per sub-task, where the dance has been defined as a sequence of sub-tasks. Furthermore, these primitives are chosen by hand, based on the task. So, it seems that the interesting question of how to *determine* such basis motions is not really being addressed.

3. Cartesian impedance control approach. This is an interesting representation: the targets are specified in world-coordinate-system for the endpoint, and a force F is computed based on the difference of the target and current positions of the endpoint (minus damping). This force F would be like an external spring pulling the endpoint towards target; to achieve the force, torques are applied at the joints: $\tau \propto J(\theta)^T$ where $J(\theta)^T$ is the transpose of the jacobian evaluated at the current state θ. Extra terms need to be added here to prevent self-collision and preserve joint limit constraints.

Looking at the results[14], all three motions appear fairly similar; they do look pretty smooth, but all target positions were basically "keyframed" by hand (or interpolated between manually-set keyframes).

The rest of the paper compares the three control methods, as well as a recorded human motion sampled at a fairly low frequency. This leads to some interesting observations: PD control gives unnaturally high hand speeds compared to human, yet produces the smoothest and most symmetric hand profiles. Also, some subtasks lead to more/less jerkier motions, consistently for all controllers. And finally, the human tends to have the jerkiest motion of all, followed by the PD control, and impedence control has the least jerky motion. A comparison is made between the difficulty of using each of these controllers, both in terms of number of parameters needed to be specified per sub-task,

[14]Results for the three controllers can be seen at:
http://robotics.usc.edu/ agents/Links/Research/Movies/mac3.mov

and also in how hard it was to determine those parameters, e.g. specifying 3D joint-angles is more complicated. Towards the end of the paper, the future possibility of being able to use the most suitable type of control for each sub-task is proposed. So, overall, the paper does illustrate some comparisons of a few existing approaches.

Review of [ZH99] on Motion Data Tracking

In [ZH99], Zordan and Hodgins make a promising step by combining the approach of physics simulation with the use of motion capture. They use the captured data from an upper body to provide the "target" or desired equilibrium angle to a PD servo-control at each rotational joint, which constitutes what they call the "dynamic tracker". The stiffnesses are set to minimize the difference between the target and equilibrium angles at each simulation time step using a gradient search. Contact reactions forces are added, as well as hand-specified constraints to be maintained by another PD controller, e.g. "maintain the wrist at an upright orientation". In the accompanying videos, the contacts look fairly good, although there is something slightly strange about some of them— they look more like the character stopped his or her hands very fast in mid-air, but didn't quite touch the object in question. Anthropomorphic differences between the actor and the computer graphics character may have to be handled on an individual basis for "behavioral contact" which needs to be preserved, but which is not embedded in the joint angle information, such as having the hands clap together. They blend motions together by interpolating joint angles and using this as the new desired sequence of angles for the PD controller. This can be quite difficult to animate properly, as "poor choices for the beginning and ending of a transition can lead to inter-body penetration and unnatural postures as well as transitions that happen too slowly or quickly."[15]

A limitation is that in tracking the motion, they do not look ahead; they only look at the current desired angle— it would appear that the system could benefit by an inverse

[15][ZH99, Sec 5.2].

dynamics approach that optimizes over the whole sequence, using the given sequence as the desired final motion, rather than as a control signal for the PD controllers. This also means that there is an inherent lag in the system, although they claim that this can be ignored in most cases. It also seems to imply that their system would be at a high risk of breaking down if any motion involving balance were to be tracked, such as walking, since there is no reason that the motion of walking constitutes the necessary equilibrium angles to control a PD-servo to actually generate a walk, under the effects of gravity.

A more fundamental difficulty— which seems like an inherently difficult problem that would have to eventually be dealt with in any such approach— is that "the control system can only track what was recorded, that is, the actual motion achieved by the human actor, not the actor's desired motion."[16] Suppose, even, that in an ideal scenario, an inverse dynamics system is used which optimizes over an entire motion sequence of walking. This approach will not handle problems such as feet sliding on the floor due to anthropomorphic difference, since the dynamic solver cannot assume, for example, that the feet were "supposed" to stay still while touching the floor (so this could potentially even lead to singularities or infinite forces attempting to correct for this sliding!) If there is no physical solution which is almost identical to the given motion, then the results are unpredictable. These are, indeed, similar questions to those left unanswered in the paper by Yamane and Nakamura [YN00], which we described earlier (page 20).

Nevertheless, the basic idea of using physics combined with motion capture for smoothing, retargeting or editing is very interesting and useful.

[LvdPF96] on Limit Cycle Control, applied to Bipedal Walking

Laszlo, van de Panne and Fiume [LvdPF96] use a finite state machine (FSM) to control the dynamic simulation of a cyclic motion such as human walking. That is, PD-controllers at the joints have target states synchronously activated by a finite state machine. Once

[16] [ZH99, Sec. 6]

the FSM is manually tuned to achieve an open-loop walking controller, estimates based on a local linear model determine the corrective forces that must be added to maintain the stability of the system. Rather than directly trying to ensure a stable limit cycle for the entire state vector, they define a basis feature of the character's posture called a *regulation variable*, e.g. the projection of the up-vector (in the local coordinate frame of the pelvis) onto a ground plane, to determine the corrective control. A key idea of limit cycle control is the ability to predict the effects of control perturbations (in this case at the hips) on these regulation variables over a step. A linear estimation procedure is used for this which does slow down the simulation process, and restricts the feasible range of target values for the regulation variables, relative to the open-loop FSM control. However, it results in a quite robust system. Sufficiently robust, in fact, that they are able to increase control over the simulated motion by adding a servo-control to the torso, letting the limit cycle controller make the necessary countering for the torso motion.

2.4.2 Interactive Control of Dynamics Simulations

The work by Troy [TV95, Tro95, Tro98] is one of the earliest examples of interactive control of the locomotion of a dynamic simulation of graphical biped character. A planar model is used, with a feedback-based 2D balancing regulator, and a state-machine-PD controller for a walking motion. The user operates the 7-link biped within a virtual environment. In more recent work [Tro00], Troy uses haptic feedback in controlling a physically simulated arm.

More recently, Laszlo, van de Panne and Fiume [LvdPF00] have demonstrated a nice system for controlling interactive 2D-dynamic character simulations for a number of example scenarios. Using 2D makes the problem more tractable, both in terms of the runtime speed, as well as by reducing the number of degrees of freedom that need to be controlled by the user. Various combinations of continuous and discrete input interfaces are tried, using the keyboard and mouse. For example, a continuous input control taking

the horizontal and vertical translation of a mouse, is used to set the target states for PD controllers at the 2-joints of Luxo lamp character. On the other hand, a discrete keyboard interface is used to control a planar bounding cat, where each of the front and back legs have 6 keys assigned to them, corresponding to 6 different equilibrium states (to a total 12 keys, as they only need to consider one front leg and one back leg in their 2D simulation). Continuous and discrete control methods are combined into a hybrid to achieve control of a biped system capable of motions such as walking, running and long jumping. A discretely-activated control is provided for "grasping" and "releasing" actions, as well as "grab the [previous | current | next] rung" in an irregularly spaced monkey-bar setup (using a type of inverse kinematic (IK) primitive).

A new application of interactive physical simulation, closely related to the approach of Laszlo et al. [LvdPF00] as described above, has recently appeared in a computer game[17] by van de Panne [vdP01]. In this game, the player uses the two degrees of freedom of a mouse to control the dynamic simulation of a 2-D skier on a variety of courses that include jumps and obstacles. The stiffnesses are set to reasonably high, constant values, and the length of the skis themselves provide the character with a relatively stable base. When the character falls and is outside the player's control, then the stiffnesses are automatically reduced to provide a more natural look. A valuable aspect of the game, from the interface point of view, consists of a series of follow-the-leader training exercises designed to teach the player the mapping from the mouse to the character. This mapping then becomes the means for the user to accomplish his or her primary goal, which is to have the character achieve certain trajectories, e.g. jumping off a cliff, doing a triple flip and landing on his feet ready for the next challenge.

[17] `www.motionplayground.com`

2.5 Summary

We have discussed some major approaches to animating a character: keyframing, motion capture and dynamics. In keyframing, we see that a common difficulty arises from the obvious but significant fact that a motion is only perceived in its true form when it is perceived in time. Looking at a non-linear motion curve (over time) of an animation parameter, it is hard to know exactly how the actual motion will look. A musician requires some years of training before he can look at a score and "hear" in his head the rhythm, and even this he does by effectively playing it mentally in time. The choreography of a human figure is not simpler, and the task of visualizing it is further complicated when a computed interpolation has to be taken into account.

While keyframe animation can be laborious, dynamic animation is simply unintuitive to control; finding the right dynamic parameters to achieve a particular desired motion is very difficult. If the desired motion is specified in great detail, then solving for the correct forces may be possible, but having specified that motion in so much detail probably required a method such as keyframing in the first place. Furthermore, if the motion is not physically possible (e.g. due to gravitational effects), then the solver cannot know what the corresponding "feasible" motion should be. To control animation without feedback is– quite literally– choreographing and performing a dance in the dark. Motion capture can work well, but is most appropriate when the motion to be animated can be accurately recorded without feedback, which only really works for certain human animations. Computer puppetry transcends the limitations of other animation techniques by allowing the animator to create motion in time and react to the created motion immediately, thus putting both the creation and the correction processes in the dimension where they occur. Furthermore, while a full dynamic simulation may be unwieldy, the use of some dynamic principles can potentially be complementary to the real-time interactive control of kinematically-based computer puppets.

Chapter 3

Input and User Interface

We now present and discuss the design and implementation of the DIGITAL MARIONETTE system, over the course of the next three chapters. In this chapter, we discuss user interface aspects of the system (not including the mappings themselves, which are given separately in Chapter 4). We begin by describing the actual input signal and our filtering approach, and its implications. We next discuss the design of the input device, and the important passive feedback it offers: both visual and tactile, as well as that resulting from its inertial physical properties. Active (e.g. computer-generated) feedback is briefly described, as well as the multi-track interface and workflow. The chapter concludes with a discussion presenting the more general concepts of "literal" versus "non-literal" interfaces, and how this relates to the DIGITAL MARIONETTE interface.

3.1 Input Parameter Space $\Upsilon(t)$

Each of the Polhemus trackers [Pol] uses a low frequency electromagnetic field to measure a 6-dimensional vector Υ. Three translation parameters $\langle v_x, v_y, v_z \rangle$ give the position of the origin of the frame of reference of the tracker, F_T, with respect to the frame of reference of the source emitter, F_E.

Three parameters, $\langle v_{yaw}, v_{pitch}, v_{roll} \rangle$ are an Euler-angle representation[1] of the rotation from the orientation of F_E to the orientation of F_T. This is computed as follows:

1) Rotate F_E by v_{yaw} radians around its **z**-axis to get an intermediate frame F'.

2) Rotate F' by v_{pitch} radians around its own **y**-axis , yielding the next intermediate frame F''.

3) Finally, rotate F'' by v_{roll} radians around its current **x**-axis to get the orientation of frame F_T.

Each of these rotations can be written as a matrix, e.g. the rotation by v_{roll} radians around the **x**-axis is given by

$$\mathrm{R}_{\mathbf{x}}(v_{roll}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(v_{roll}) & -\sin(v_{roll}) & 0 \\ 0 & \sin(v_{roll}) & \cos(v_{roll}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and these matrices[2] can be concatenated by multiplication so that the full transformation can be written as

$$M_{E \leftarrow T} = \mathrm{R}_{\mathbf{x}}(v_{roll}) \cdot \mathrm{R}_{\mathbf{y}}(v_{pitch}) \cdot \mathrm{R}_{\mathbf{z}}(v_{yaw}) \cdot \mathrm{Trans}(v_x, v_y, v_z)$$

$$= \begin{bmatrix} c_p c_y & -c_p s_y & s_p & v_x \\ s_r s_p c_y + c_r s_y & -s_r s_p s_y + c_r c_y & -s_r c_p & v_y \\ -c_y c_r s_p + s_r s_y & c_r s_p s_y + c_y s_r & c_r c_p & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where c_y, c_p, c_r represent the cosines of the yaw, pitch and roll respectively; and s_y, s_p, s_r

[1] {yaw, pitch, roll} are also often referred to as {azimuth, elevation, roll} respectively.
[2] The standard homogenous coordinate representation is being used here.

represent the sines of the same. $M_{E \leftarrow T}$ is the matrix describing the transformation such that a point $\mathbf{p}$ with respect to F_T corresponds to the point $M_{E \leftarrow T} \cdot \mathbf{p}$ with respect to F_E.

3.2 Filtering

3.2.1 Adding Viscosity

One of the challenges of using the trackers is the jitteriness of the input. This is a result both of noise in the sensor readings caused by the presence of electromagnetic devices in the environment, as well as hand jitter. As will be mentioned in Section 3.3.3, some of the hand jitter can be smoothed by the use of a weighted input device, but it is still very important to be able to filter the raw input signal in some way. Let $x_i = x(t_i)$ represent the value received for one of the input parameters at time step t_i. Then the corresponding filtered value $y_i = y(t_i) = f(x_i, x_{i-1}, \ldots, x_0)$ which will be used is given by

$$y_i = y_{i-1} + (1 - \alpha)(y_{i-1} - y_{i-2}) + \alpha(x_i - x_{i-1}) \tag{3.1}$$

We can interpret Eq (3.1) as forcing the output velocity to be an average of its previous velocity with the most recent input velocity, thus adding a "viscosity" term to the output motion. This has been found to work quite well for our purposes, though for different animations, a different filter might be more appropriate. Fig 3.1 shows a sample of raw and filtered input data.

3.2.2 Controlling the Viscosity

Adjusting the value of α between 0 and 1 in Eq (3.1) above controls the smoothness of the input signal y_i, and therefore also of the resulting motion. If α is too high (e.g. $\alpha = 1$ in the extreme case), then the input signal is essentially unfiltered, and the noise is very apparent in the animation. If α is too low, then the responsiveness is compromised, and

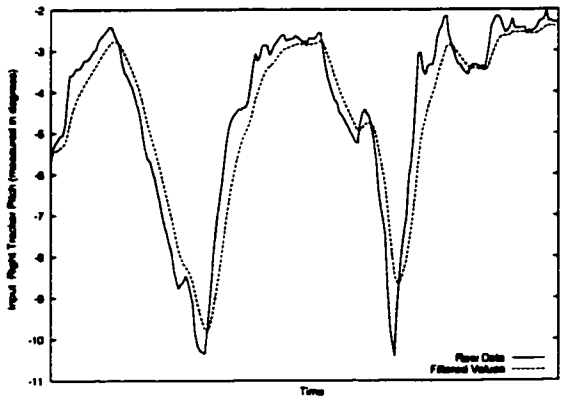

Figure 3.1: FILTERED INPUT SIGNAL

The solid line shows raw input data of one of the Polhemus tracker parameters. The dotted line shows the corresponding filtered data for $\alpha = 0.8$.

significant lag is introduced as well. In the extreme case, for example, when $\alpha = 0$, the input value does not affect the output at all.

This leaves us with a range of possible values, with which we can control the smoothness quality of the resulting motion, to advantageous effect. For example, setting a high smoothness for the hip gives the character a certain appealing quality, related to what many viewers described as "very graceful motion". The local attentuation of higher frequency components can enhance what is seen as part of the character's "style" of motion. In contrast, for other joints such as head rotation, it is effective to use a much lower-smoothness filter, allowing more responsiveness to jerky motions such as quickly glancing over his shoulder.

The choice of a particular filter value helps emphasize corresponding qualities of the character's motion. This fact led us to give the user real-time control over the filter parameter α itself. Although doing so produced some interesting results, it was quite difficult to learn to control the interaction between this and the rest of the animated parameters (i.e. direct control of joint angle values), certainly when attempting to achieving

realistically-based motion. In particular, changing the filter response in real-time can lead to highly non-linear effects, analogous to playing with delay, gain and various other parameters on effects pedals during a musical performance. Additional effects of this on the user's experience will be described briefly in Section 6.3.3.

3.3 Passive Feedback: Adding Physical Artifacts to the Input Device

The Polhemus device includes one source emitter and two sensors/trackers. The source emitter is a cube weighing under 0.5kg, about $8cm$ in length along each axis, and is assumed to remain in a fixed position while the system is being used. The Polhemus sensors are small, oddly-shaped devices, about $2-3cm$ long, and very lightweight. Both sensors and emitter are attached to the main device box by cables.

In the early stages of our system development, the user controlled the input by holding the sensors directly as shown in Fig 3.2. Manipulating the translation and orientation of the sensors continuously in this way, however, was quite awkward for a number of reasons. First, the small size and shape of the sensors made them prone to slip. Second, the cables would wrap around the user's hand, and even if this only happened partially, it was enough to quickly make them feel tangled. Also, when the cables got too curled, then their own stiffness, combined with the light weight of the sensors and the lack of a solid grip, would usually cause the sensors to flip around, slipping on the user's fingers, unless the user gripped really tightly. But gripping tightly, in turn, made it harder to manoeuver fluidly.

We solved these problems by attaching more convenient handles to the sensors. The input device redesign involved the following two main stages:

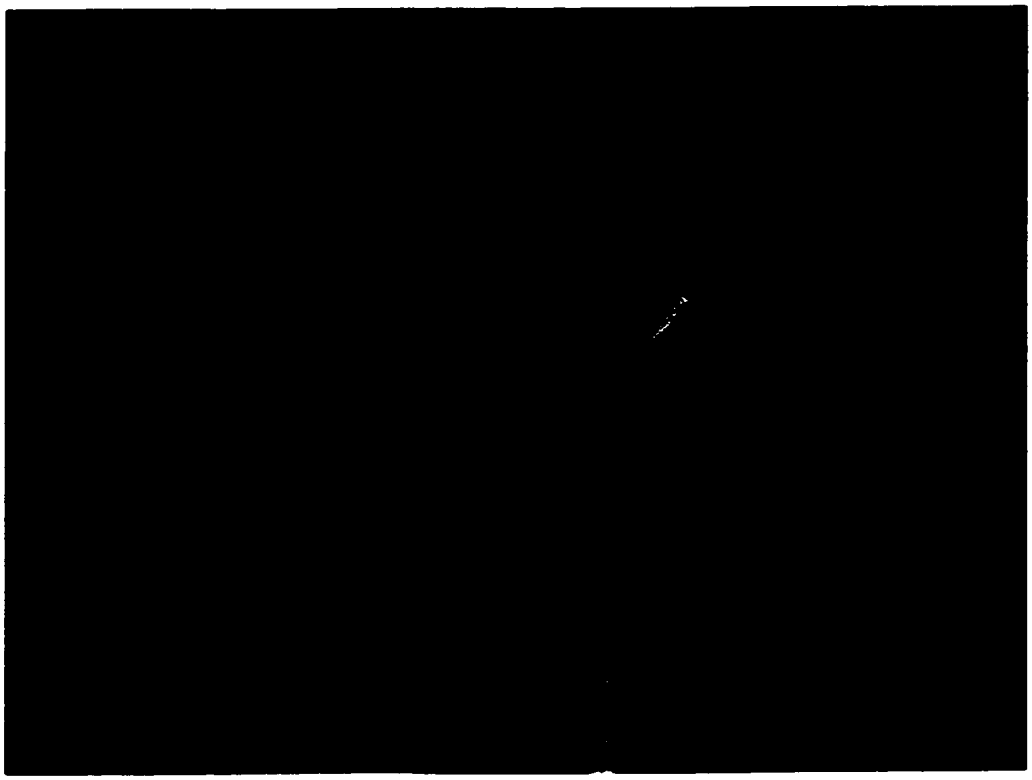

Figure 3.2: HOLDING TRACKER DIRECTLY

(1) Geometry. In the first phase, a new geometry was designed for the input hardware: specifically, we encased the tracker in a cylindrical tube about 15-20*cm* in length, and about 4 or 5*cm* in diameter— just wide enough to fit the Polhemus while still very comfortable to hold, as shown in Fig 3.3.

(2) Physical Properties. The second phase involved changing the physical properties of the input device, while keeping the same basic shape: in particular, the new cylinders were now made out of bamboo stick, thus adding a *mass* to the input device. Fig 3.4 shows the user holding onto the bamboo stick input device.

Designing an appropriate handle, in fact, was far more than a superficial modification to the system; it let us consciously redesign a crucial part of the input device itself, making the system significantly easier to use fluidly and accurately. We will now discuss some implications of this ostensibly simple new design, and put this the context of prior

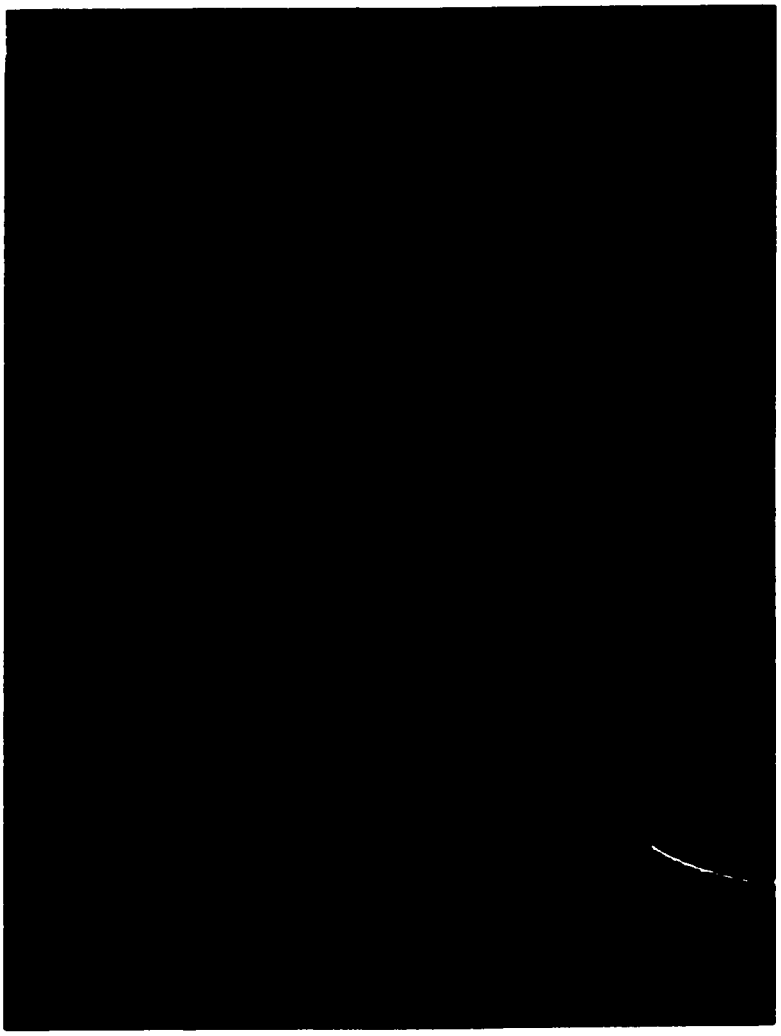

Figure 3.3: HOLDING TRACKER EMBEDDED IN FOAM TUBING

The tip of the tracker is visible at the top of the tube.

related work by Hinckley [Hin96].

3.3.1 Grip.

The lack of a suitable grip was of course the initial motivating factor in attaching the handles, so this was one problem we had to be sure to solve effectively. Indeed, our handles worked well for this purpose. The importance of grip is listed in [Hin96] as an important aspect of 3D input device design, and consideration of this across a broad range of devices seems to show that:

Figure 3.4: HOLDING TRACKER ENCASED IN BAMBOO TUBES
Although not visible, the tracker is in the top of the tube, near the user's fingers.

Many 3D input device designs encourage the user to hold the device against the palm at a fixed orientation. This is known as the *power grasp* [MI94] because this hand posture emphasizes strength and security of the grip. By contrast, the *precision grasp* involves the pads and tips of the fingers and so emphasizes dexterity and free tumbling of the input device. This issue was formally analyzed by Zhai for a 6 DOF docking task, where he found that using the fine muscle groups emphasized

in the precision grasp results in significantly faster performance [ZMB96].[3]

Our tubes allow both power and precision grasping. Interestingly, we found that the inexperienced or untrained subjects would naturally hold the interface devices in a power grasp. As the primary test subject became more advanced, the precision grasp was indeed used more often, and still quite comfortably. The bamboo sticks were sufficiently light that they could be held without needing to engage all five fingers. Note that integrating buttons onto the tube controls would typically, though not necessarily, encourage the power grasp, thus suppressing the possible advantages of using the precision grasp.

3.3.2 Visual and Tactile Cues.

In [Hin96], Hinckley describes an interface for neurosurgeons to visualize 3D volumetric brain data by providing them with a small tracker embedded in a baby doll's head to be held in one hand, and a second tracker embedded in a cutting-plane tool or stylus tool held in the other hand. In this interface, he demonstrated the effectiveness of customizing the physical characteristics of the input interface device, showing how that can provide the user with a number of important cues while using the system. Our input interface design bears some relation to Hinckley's in that, although the task goals are different, our design was also carefully chosen to be able to benefit from physically-based cues. Specifically, the cylindrical shape of the input device creates visual and tactile cues indicating a reference frame with a "long-axis," as well as axes of symmetry. By their presence, these features let us define an alignment of the axes of the input device with those of the bones in the puppet's geometry which were being controlled. For example, rotating the input tube around its long axis could correspond to rotating a virtual object such as the humerus or thigh bone around its long axis as well. Hence, by embedding the trackers in a particular shape, the cues provided by the new input device established a tangible interface to the

[3][Hin96, Section 6.9]

various coordinate frames inherent to the character. This modification was very helpful in the development and use of the virtual puppetry mappings.

Applying the same principle, if we were using an inverse kinematic mapping, and wanted to achieve a correspondence between the input device and the virtual character's hands or feet, for example, then the trackers could be encased in a flatter, rectangular object, physically representing their length and width. Additionally, a slight curve might be sufficient for the user to distinguish (again, both visually and kinesthetically) between the anterior and posterior sides of the hands and feet.

While still providing visual and tactile cues, note that our bamboo cylinders are somewhat more generic than Hinckley's baby doll head, and are even used in quite abstract ways at times, a topic discussed in more detail in Section 3.6 of this work.

3.3.3 Physical Properties and Constraints.

The cylindrical tubes were originally built using foam tubing which was extremely light, so that the weight of the trackers plus the foam was still insignificantly small. Using a heavier material was an idea partially inspired by Hinckley's thesis, in which he points out that input tools with mass provide kinesthetic feedback due to gravity and inertial properties [Hin96, Section 4.6]. This feedback— primarily the feeling that one was actually holding on to real object, rather than just waving one's hands around in space— did indeed make the input control a far more satisfying experience. The lightness of the Polhemus sensor cables relative to the weight of the heavier tubes made the cables themselves much less noticeable than before. Furthermore, the inertial properties of such an input device helped to reduce the hand jitter by filtering the raw input signal:

> The mass of the tool can damp instabilities in the user's hand motion. For example, surgeons are very particular about the *weight* of the surgical instruments, as the proper heaviness can help decrease the amplitude of small, involuntary hand

tremors.[4]

Although it has not been investigated, the natural physically-occurring filtering could potentially mean that a digital filter with a different response could be used with a smaller group delay.

In our case, the mass of the tool also helped provide kinesthetic orientation feedback, by making the tube naturally gravitate towards the vertical (neutral) orientation, when the user was holding it above its midpoint. Furthermore, by reorienting the axes so that the tracker cable was emerging from the bottom of the tube, it made it easier to hold the tube in an upright vertical orientation, and harder to hold it completely upside down, thus providing a tangible sense of the original (neutral) orientation, as well as differentiating between the upwards and downwards directions, while still visually retaining the existing symmetry along this axis.

Note that, as much as the use of bamboo stick provided a necessary mass, the bamboo stick was also still light enough to allow the user to move the input device quite freely, and without needing to exert too much force.

3.4 Active Feedback

Clearly, the crux of this work is the instantaneous visual feedback showing the animation in real-time, illustrated in Figure 3.5, and the implications of this are discussed both directly and implicitly throughout this dissertation. Nevertheless, there are less obvious, but still helpful, feedback signals that can be displayed to the puppeteer in addition to the CG character itself.

A simple and very useful such cue is colouring the left and right feet in two different colours so the puppeteer can tell them apart regardless of the view. This is especially helpful for a side view, which is often used by the puppeteer when creating a walk.

[4][Hin96, Section 4.6].

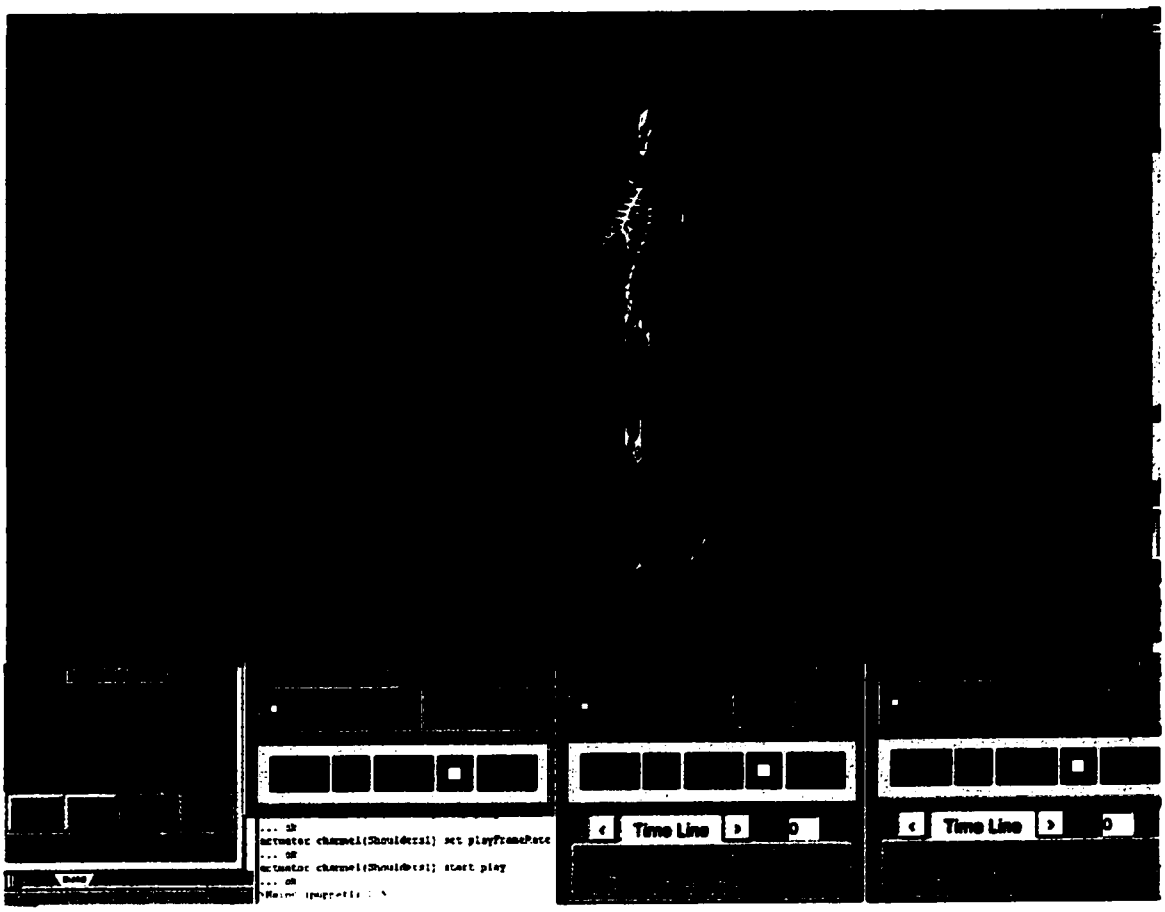

Figure 3.5: DIGITAL MARIONETTE SCREEN CAPTURE

This screen capture shows a typical working environment with the DIGITAL MARIONETTE system. Active tracks respond to the user's motion in real-time, while other tracks may play back previously recorded motion. The three colourful GUI's along the bottom right are interfaces to three of the motion tracks. They provide functionality such as track activation for live control, play, record, load, save etc. The current system has been integrated into the DANCE platform [NF].

Another useful visual signal is drawing the projection of the puppet's centre of mass (COM) on the ground. Since the user has no tactile sense of the distribution of the puppet's weight, this is a helpful indicator. In particular, the novice puppeteer can sometimes keep track of the movement of the projected COM while walking, and use it as a regulator, by trying to make sure that it is moving at a roughly constant velocity while the puppet is walking. Otherwise, like a beginner musician, he will tend to go faster during the parts that are easier to control, and slower during the harder parts, leading to an uneven performance, which typically looks (or sounds) worse than a slow but steady one, and is less useful from the point of view of practicing as well.

There are a number of other feedback channels to be explored in the future. For example, auditory feedback indicating footsteps could be quite helpful as well.

3.5 Multi-Tracking Interface and Workflow

Multi-tracking is a recording methodology that has been used for years in the music industry. To create a functional recording studio environment for the animated motions, we have implemented modular components such as "Channel", "Mapping", and "Input Device", along with corresponding Tcl/Tk graphical user interfaces these components (as shown in Figure 3.5). Our system has been integrated with the DANCE platform [NF], and the components interface to it through a plug-in architecture.

Each channel connects an input device (or file) to a subset of the character's parameters. Any of the mappings available for that subset can be reassigned at any time. A Tcl/Tk graphical user interface has been implemented for each component, and the GUI for the motion channels provides a full play/stop/record functionality. A channel is either in an *active* state, in which case the live input feed is determining the values of its corresponding output parameters, or else it is off-line, acquiring its data from recently recorded or previously saved motion.

An important practical issue which the multitracking interface has to solve in the real-time context is an intuitive and efficient way of keeping the multiple tracks properly synchronized to allow layering of tracks. Multiple channels may be playing or recording simultaneously. Furthermore, this should be designed such that the user does not have to rush unreasonably fast to press the **record** button in one track immediately before or after pressing the **play** button on another track at the risk of messing everything up. Recording four tracks should not be any harder to coordinate in the software than recording two. We solve this by providing each channel with its own built-in clock. Furthermore, the scripting interface (and therefore the GUI) provides a synchronization feature that allows the timebase of any channel to be synchronized to that of any other channel. This is crucial for layering motions in real time.

A typical workflow starts by synchronizing all channels to one "base" or master control, typically the channel for controlling the legs. Motion is then recorded for this channel. Having recorded this track (i.e. the leg motion), the user now starts recording a new track containing arm motion, while playing back the previously recorded track. Once this is done, then both tracks are played back, and the user continues to add new layers of motion.

3.6 Literal and Non-Literal Interfaces

3.6.1 Literal Interface Issues

Referring to the 3D neurological visualization interface described in Section 3.3.2, Hinckley reported that while many computer scientists and other non-surgeons suggested embedding the tracker within a skull model, no neurosurgeon made this suggestion after having tried the interface[5]. The reason the neurosurgeons saw a potential problem with

[5][Hin96, Section 3.5.1].

this is that a realistic skull model would not correspond identically to almost any of the shapes of particular patients who's data was being examined, so it would only add confusion in that regard. The transverse plane corresponding to the top of the device's eyeline, for example, would not necessarily correspond to that that of patient's data.

The concept underlying this example is that by using a well-chosen abstraction, or generic head, the input device is sufficiently suggestive to the user, yet does not suggest too much (i.e. wrong information). This indicates that the representational form of an input device— in relation to the virtual objects or functions it manipulates— can range in subtle ways between *literal* and *generic* representations. At one end of the spectrum are those interfaces which attempt to provide a very literal or specific representation, and in some cases that may be most appropriate. At the University of Virginia UI lab, for example, a virtual reality interface glove was initially used for trying to position a virtual light in a 3D space, but this interface was found to be very ineffective. Instead, a much better solution was to use a real flashlight with a Polhemus tracker embedded in it[6]. A quintessentially literal input device was the Dinosaur Input Device used to achieve the animation in the film Jurassic Park [Spi93]: a model skeleton was built to represent the virtual creature, with sensors at all of the joints [KHSW95]. This was reported to work very well and efficiently.

However, there can be problems with literal interfaces. The more specific the device, the more there seems to be room for small discrepancies between the physical and virtual models; at the same time, the more specific the device, the more specific the expectations, and thus the errors caused by the discrepancies are less tolerable. This would have been exactly the problem if a skull were used in Hinckley's visualization system, and it did happen in the Dinosaur Input Device (although the authors claim that the benefits outweighed these problems, which seems plausible given the alternatives). In fact, this explains, in essence, the same problem that occurs with literal motion capture: from the

[6][Hin96, pg. 87]. See Ref [173].

user's point of view, the motion capture input device— the body suit— extends to implicitly include the actor's body, which is then expected to be an accurate representation of the virtual character in the graphical world. However, there will almost invariably be discrepancy between the two anthropometries, leading to differences between the desired motion and the actual one produced. In a sense, this is the "happy-face effect", wherein a line drawing of a happy face is accepted without criticism as a simplistic representation, but in a more detailed drawing of a face, any flaws are quickly noticed.

Another potential problem with a highly literal input device is that the full range of motion of the virtual model cannot be exactly realized or supported in the physical model. In both standard motion capture, and with the Dinosaur Input Device, the joint range of the virtual model may be larger than that which could have been achieved either by an actor or by the physical model. Whether this is a problem or not depends of course on the specific requirements of the animation. Another such aspect of the Dinosaur Device was that all the (virtual) joints of the graphical model were universal joints, while the physical universal joint sensors were large, and could only be used in uncluttered areas of the model such as the neck and ankles. In a similar vein, the object being manipulated— unique to a computer simulation— might also have various properties which simply could not be physically realized in an input device in any case, making this approach out of the question in those situations.

Finally, in some cases, literal control might just be the wrong approach, either too ineffective or inefficient. Consider, as an example, if a pilot were to control the ailerons of a plane by manipulating the corresponding ailerons of a small model plane in the cockpit. It seems unlikely that this would give the necessary angular resolution. Suppose, furthermore that a crucial movable wing-part is located in a place that is hard to reach without disturbing other nearby parts. Similarly, steering a car is probably easier done with a steering wheel rather than adjusting the orientation of a set of wheels of a model car. These examples are artifical, of course, but they clearly emphasize the point that

in some cases it is more important to have certain critical controls easily accessible, e.g. via large, graspable levers, or pedals, for instance. Indeed, although not mentioned in [KHSW95], according to Fitzmaurice [Fit96, Section 3.3.4]:

> In this truly direct manipulation interface [the Dinosaur Input Device], individual joints cannot be easily isolated and separately manipulated without causing surrounding joints to be altered.

3.6.2 Non-literal Interfaces

Hence we see that while literal input devices can be very useful, this does nevertheless depend very much on the context and task at hand. In fact, this is an instance of the more general principle that the appropriateness of any given input device is always highly dependent on the specific task. This latter principle is illustrated very nicely by Buxton in [Bux86]. The example given consists of two drawing devices: an *Etch-a-Sketch* and a *Skedoodle*. The *Etch-a-Sketch* uses two knobs to control vertical and horizontal motion respectively, while the *Skedoodle* provides a single joystick, into which horizontal and vertical control have been integrated, and which can therefore push the pencil-tool in various planar directions. The same two degrees of freedom are being controlled in both cases, but each is implicitly well suited to a particular type of drawing, and is harder to use for others. Similarly, various literal and non-literal interfaces each have their own tradeoffs.

In particular, manipulating a complex articulated figure is an intrinsically hard task, which can be seen as subsuming a tremendous range of sub-tasks, and no matter how the degrees of freedom are distributed for manipulation, there will inevitably be at least some challenging element in using the system. We illustrate this for an example of two different non-literal mappings for controlling the legs. In both cases, some practise is required by the user to get accustomed to what first appear as inconsistencies in the mental model of the mapping.

Consider first a mapping where the height of the tracker controls the angle of flexion at the knee in relation to the thigh (femur). Suppose an upward tracker motion extends the knee, so that when the thigh is flexed forwards, raising the tracker corresponds to the ankle moving along an upwards arc. This correlation is so strong that, as a user, it is nearly impossible not to notice it and therefore the user wrongly assumes that this is the underlying control model. However, when the thigh is extended backwards, then this same *upwards* tracker motion will correspond the ankle moving *down*. This breaks the user's mental model just described.

Suppose, then, that we create a mapping defined to be exactly the model which was assumed by the user in the preceding example. In that case, the knee bend is controlled by inverse kinematics, whereby the position of the tracker corresponds to the position in the world coordinate system (WCS) of the end-effector of the leg. But now consider the task of swinging the leg back and forth from the hip, while keeping the knee fairly straight. The difficulty here turns out to be in moving the hand in a perfectly circular arc with the right acceleration, unless one is actually swinging the arm. Otherwise, if the arc is not circular, then this will cause the leg to bend, or not accelerate properly. Indeed, a common problem with motion-captured control of IK chains is the joints' sudden acceleration or snapping into their limit values.

Thus, in creating a mapping we may have a choice as to whether the control of a particular joint or link will be given in the local coordinate system (LCS) (i.e. relative to its geometrical parent) or in the WCS, sometimes reducing to a choice between kinematic versus inverse kinematic control for the given link or joint[7]. There is no rule as to which is the more intuitive; the examples above demonstrate that whether we are focusing on relative joint angles (LCS) or on global link orientations (WCS) or positions (WCS), is

[7]This particular example is being developed for the sake of argument. Combining the two parameterizations, although possible, just leads to other trade-offs or issues, such as the need to provide an intuitive way to switch between them, with some sort of continuous blend between the two mappings. Alternately, another mapping may be possible which does not fall clearly into either of these two classifications, but for which making isolated motions may be more difficult.

very much a matter of what we are doing. This can also be observed in a different context by listening to a dance instructor, who might sometimes ask for a "straight knee" or a "pointed toe" (which are relative orientations), while other times asking for the "arms high" or "head looking towards the corner" (specified in WCS). In this regard, using a physics-based model at the joint, as will be described in Chapter 5, leads to a fundamental difference in the nature of this interface.

Chapter 4

Kinematic Mappings

4.1 Input $\rightarrow$ Mapping $\rightarrow$ Output

Let $\Upsilon \in \mathcal{H}$ denote the complete input vector at a given time and $\theta \in \Theta$ be a state vector representing the posture of the puppet. Then the mapping problem we consider here is finding a function

$$\psi : \mathcal{H} \rightarrow \Theta$$

which allows the user to control the state of the puppet[1]. In this chapter, we describe the kinematic mappings from $\mathcal{H} \rightarrow \Theta$ that are used for mapping from two 6-DOF Polhemus trackers to a CG human skeleton model. First, the information contained in the output θ is described (Section 4.2), and then we explain how a mapping ψ has been achieved (Section 4.3-4.7.1).

[1]Note that in this chapter, "state" only refers to the puppet's kinematic parameters, and not to velocities, unless explicitly indicated otherwise. Other variants of this functional form are also possible, including a feedback loop, such as in $(\mathcal{H}, \Theta) \rightarrow \Theta$, or with a time-window.

4.2 The Animated Character Output

The CG puppet we are controlling is a rigid articulated body consisting of a set of links (e.g. "upper left leg", "right forearm") connected by joints (e.g. "left hip", "right elbow") shown in Figures 4.1 - 4.4.

4.2.1 Body Local Coordinate System (LCS)

The body coordinate system is oriented so that, from the puppet's point of view

1. the **y**-vector points *up*,
2. the **x**-vector points *left*,
3. the **z**-vector points *forward*,

and the origin is approximately at the sacrum. Standard anatomical definitions can be expressed correspondingly:

1. the *median plane* is the **yz**-plane,
2. a *sagittal plane* is any plane parallel to the median plane, i.e. a set of points for which x is constant,
3. a *coronal* or *frontal* plane is any plane parallel to the **xy**-plane,
4. a *cross* or *transverse* plane is any plane parallel to the **xz**-plane.

Note that the above definitions are all with respect to the LCS of the body.

The LCS of each link is initially oriented identically to the LCS of the body, when the body is in the configuration shown in Figure 4.1.

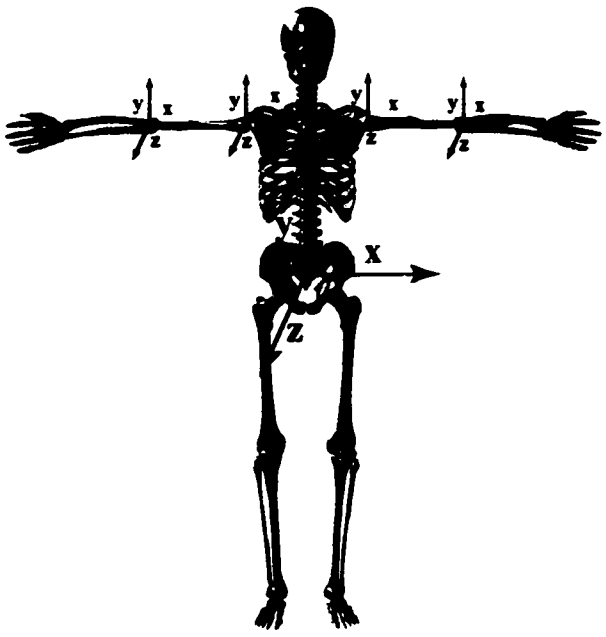

Figure 4.1: THE INITIAL (ZERO) STATE.

The large coordinate frame at the pelvis is the body's local coordinate system (LCS). Four local coordinate frames are shown for the elbows and shoulders. This is the "zero" posture of the graphical character, so all of the local coordinate frames are perfectly aligned in orientation when the puppet is in this position. For example, we can see that the long axes of the forearms and upper arms will be given by the x-axes in each of their LCS's respectively, regardless of the shoulder or elbow orientations. The long axis of the femur (thigh), on the other hand, corresponds to its local **y**-axis .

4.2.2 Joint Motion

Anatomical definitions of joint motions and their limits are given with respect to the "anatomical position" [KE93], shown in Fig 4.2. Note that this is different from the initial "zero" position (Fig 4.1) used in the graphics model. The basis joint movements

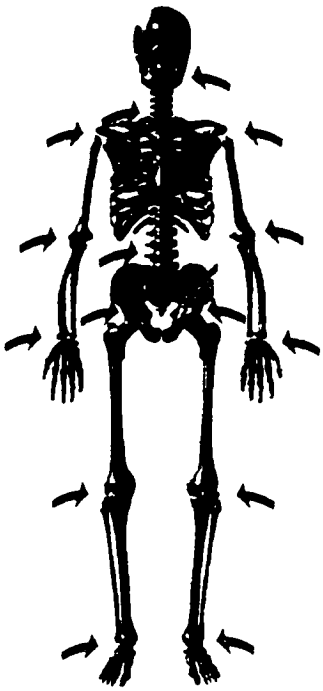

Figure 4.2: ANATOMICAL POSITION
Limbs are relaxed and extended with palms facing forward. Arrows indicate the joint articulation locations, listed in Table 4.1.

at joints are defined as:

- *Flexion/Extension*: Generally, *flexing* at a joint is bending it to bring the adjacent bones closer together, and *extending* it is straightening it. Starting from the anatomical position, flexion and extension of a joint causes bone movement along the sagittal plane. At the ankle and wrist, extension is referred to as dorsiflexion. Ankle flexion is referred to as plantar flexion. See Figure 4.3.

- *Abduction/Adduction*: From the anatomical position, abduction and adduction result in bone motion along the coronal plane. Abduction of a joint moves a bone away from the median plane (or, in the case of fingers and toes, away from the midline of the hand or foot), and adduction moves the bone towards the median plane. See Figure 4.4.

- *Lateral/Medial Rotation* or *Twist*: This turns a bone around its long axis. Medial rotation turns the front of the bone in towards the centre of the body, and lateral rotation turns the front outwards. For example, lateral rotation of the right hand, viewed from above, moves the thumb in a clockwise rotation. We mainly use the term *twist* to refer specifically to this, to avoid confusion with more general types of "rotation".

There are 33 degrees of freedom controlled in the system, and they are summarized in Table 4.1.

4.3 Multi-track Approach

The approach used for mapping from the 12-dof input to the 33-dof output space is a time-multiplexing system, where only some of the output dofs are being controlled at a given time. The clock is then "rewound", and a different track is recorded. The legs are usually recorded first, since they define the translation of the character, and all leg channels are recorded simultaneously. When the arms are being recorded, then the shoulders and elbows are sometimes recorded simultaneously, and sometimes in sequence (shoulders first). Wrists are typically recorded after the elbows have been recorded. The spine and head are usually recorded simultaneously, either before or after the arms.

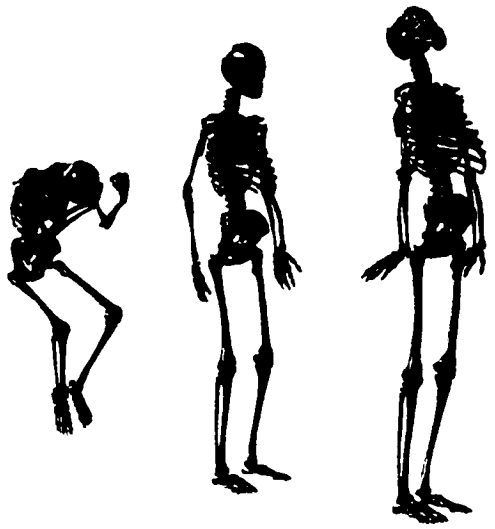

Figure 4.3: FLEXION AND EXTENSION

Mr. Bones on the left has flexed most of his joints, except for his ankle which is plantarflexed. Mr. Bones in the middle is nearly in the anatomical position, while Mr. Bones on the right has extended many of his joints (hyperextending some of them). His hands, with the palms facing the ground as shown, are dorsiflexed.

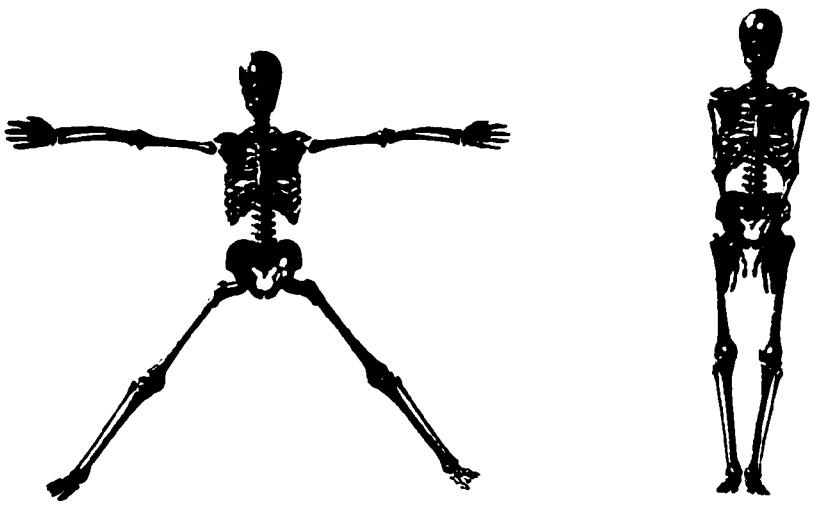

Figure 4.4: ABDUCTION AND ADDUCTION
Mr. Bones on the left has abducted his shoulders, hips and wrists, while Mr. Bones on the right has adducted the same.

Joint	D.O.F.	Child Link
Root	6	pelvis
Lower Back (L1)	3	back/torso
Lower Neck (C7)	3	neck
Head Nod (C1)	1	head
Left, Right Shoulders	3 each	upper arm (humerus)
Left, Right Elbows	1 each	forearm (ulna & radius)
Left, Right Wrists	1 each	hand
Left, Right Hips	3 each	thigh (femur)
Left, Right Knees	1 each	lower leg (tibia & fibula)
Left, Right Ankles	1 each	foot

Table 4.1: OUTPUT DEGREES OF FREEDOM: This lists the 33 degrees of freedom of the output figure which are controlled. In some cases, more d.o.f.'s are available, but are simply not being controlled, e.g. the medial/lateral rotation of the forearm from the elbow is not an active output parameter. The child links are those that share a local coordinate system (LCS) with the given joint.

4.4 Legs

The hips, knees and ankles are all recorded simultaneously, and the mappings for each have been designed to facilitate this.

4.4.1 Hips

The 3 Euler angle orientation parameters of the graspable input tube are mapped onto the quaternion representation of the orientation of the thigh bones, so that there is a direct correspondence between the tube and thigh orientation. That is, when the tube is held vertically, the hip should be vertical as well. When the tube is rotated around the vertical axis, so the hip should be as well, and so on. The orientation of the tracking receiver in the tube is physically constrained so as to make the wire run through the tube. However, we can freely reorient the source emitter in such a way that, at least for the hip joint, this computation reduces to a permutation of the axes of rotation. In particular, for input angles v_r, v_p, v_y representing roll, pitch and yaw, we apply a sequence of rotations $R_{\mathbf{x}}(v_y)R_{\mathbf{z}}(v_p)R_{\mathbf{y}}(v_r)$ corresponding to rotation about the $\mathbf{y}, \mathbf{z}$ and $\mathbf{x}$ axes in that order. Using Eqs (A.14) and (A.9) from the Appendix A, this reduces to a single quaternion:

$$\mathbf{q}_{Hip} = \langle w_0, (w_1, w_2, w_3) \rangle \tag{4.1}$$

where (4.2)

$$\begin{aligned} w_0 &= c_y c_p c_r + s_y s_p s_r, \\ w_1 &= s_y c_p c_r - c_y s_p s_r, \\ w_2 &= c_y c_p s_r - s_y s_p c_r, \\ w_3 &= c_y s_p c_r + s_y c_p s_r \end{aligned}$$

with

$$\begin{aligned} c_j &= \cos\left(\frac{v_j}{2}\right) \\ s_j &= \sin\left(\frac{v_j}{2}\right) \quad \text{for } j = y, p, r \end{aligned}$$

4.4.2 Knees

Controlling the Knee Directly Two variations are commonly used for controlling the knees. In the first case, the knee bend θ_{Knee} is controlled by a direct linear mapping from the vertical translation h of the tracker :

$$\theta_{Knee} = (\theta_{max} - \theta_{min})\, h + \theta_{min} \tag{4.3}$$

where h is expected, but not guaranteed, to be in the range $0 \leq h \leq 1$. h has not been truncated, so that larger or smaller values are possible depending on the user's motion. Joint limits are achieved by truncating θ_{Knee}, i.e.

$$\theta'_{Knee} = \begin{cases} \theta_{min} & \text{if} \quad \theta_{Knee} < \theta_{min}, \\ \theta_{max} & \text{if} \quad \theta_{Knee} > \theta_{max}, \\ \theta_{Knee} & \text{otherwise.} \end{cases} \tag{4.4}$$

Softer flexion limits can be achieved by using a function such as

$$\theta'_{Knee} = \begin{cases} \theta_{min} + 2\,\delta_{low} \left[1 + e^{\frac{2(-\theta_{Knee}+\theta_{min}+\delta_{low})}{\delta_{low}}}\right]^{-1} & \text{if} \quad \theta_{Knee} < (\theta_{min} - \delta_{low}), \\ \theta_{Knee} & \text{if} \quad (\theta_{min} + \delta_{low}) \leq \theta_{Knee} \leq (\theta_{max} - \delta_{hi}), \\ \theta_{max} - 2\,\delta_{hi} \left[1 + e^{\frac{2(\theta_{Knee}-\theta_{max}+\delta_{hi})}{\delta_{hi}}}\right]^{-1} & \text{if} \quad \theta_{Knee} > (\theta_{max} - \delta_{hi}). \end{cases} \tag{4.5}$$

This function defines regions in the output $(\theta_{min}, \theta_{min} + \delta_{low}]$ and $[\theta_{max} - \delta_{hi}, \theta_{max})$ where the relationship to the input is no longer linear, but rather changes asymptotically as the input changes. For example, as $\theta_{Knee} \longrightarrow -\infty$, $\theta'_{Knee} \longrightarrow \theta_{min}$. The transitions from the linear piece in the middle to the soft limit regions are $\mathbf{C}^2$ continuous for $\delta > 0$. For a joint such as the knee, which has a hard limit at one end, setting $\delta_{low} = 0$ is equal to the hard limit case. Note that in a real physical context, a joint may have two soft flexion maxima: one for motion produced internally by muscles, and another for motion

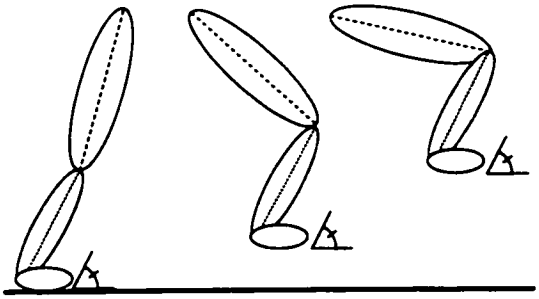

Figure 4.5: CONTROLLING WCS KNEE MOTION

Although the user is manipulating the hip orientation (with $v_{yaw}, v_{pitch}, v_{roll}$), the input vertical translation h is not being changed. As a result, the local knee orientation does change automatically so that the projected angle of incidence with the ground (or, the knee orientation w.r.t. the WCS) stays constant as well.

influenced by external forces, e.g. sitting on the ground on one's shins may produce greater flexion than could be produced by muscles alone. We currently do not make this distinction, but rather choose a single maximal value.

Controlling the WCS Orientation of the Knee. In another controller variation, the vertical translation of the tracker determines the orientation of the lower legs, or shins, relative to the world coordinate system. In this case, we control the angle which the shin forms with the ground plane, allowing movement as shown in Fig 4.5 where, if the user only rotates the leg from the hip joint, the knee bends automatically to maintain the given angle of incidence between the ground and the shin direction. So, given a hip orientation[2] $\mathbf{q}_{Hip}$ and a desired angle of incidence φ between the shin and the ground, we need to find a knee bend θ_{Knee} satisfying these constraints. This is equivalent to finding a direction vector $\mathbf{v} = \langle v_x, v_y, v_z \rangle$ which lies both on a given plane and on a given cone,

[2]By hip orientation we refer to a quaternion representing the rotation of the hip relative to its initial (zero) orientation.

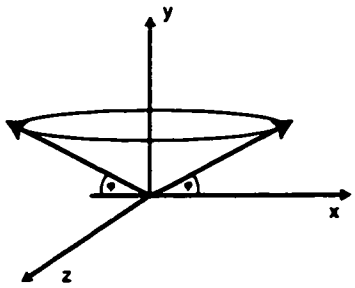

Figure 4.6: COMPUTING KNEE ORIENTATION IN WCS
The cone is made up of all unit direction vectors whose angle of incidence φ with the ground plane is fixed to the given value. Clearly, this is equivalent to having a fixed angle of incidence with the perpendicular, and, in the case of the ground plane being equal to the **xz**-plane, this is also equivalent to the slice of the unit sphere with a constant y value of $\sin\varphi$.

as follows:

1. Since the knee is a hinge joint, for a given orientation of the upper leg, the shin will move along a plane, defined by the local **x**-direction with respect to the hip's LCS, which is normal to the plane of motion. Since the hip's orientation is known, this local **x**-direction can easily be converted into the WCS, giving a vector $\mathbf{u} = \mathbf{q}_{Hip}\mathbf{x}\mathbf{q}^*_{Hip}$ in the WCS[3] . The corresponding constraint, then, is that

$$\mathbf{v} \cdot \mathbf{u} = 0 \tag{4.6}$$

2. At the same time, the set of direction vectors which have some given angle of incidence, say φ, with another plane (the ground), describe a cone of directions (Fig 4.6). If the ground is the **xz**-plane, then this cone is described by all unit direction vectors $\mathbf{v} = \langle v_x, v_y, v_z \rangle$ for which $\langle v_x, v_y, v_z \rangle \cdot \frac{\langle v_x, 0, v_z \rangle}{\sqrt{v_x^2 + v_z^2}} = cos(\varphi)$, that is, the angle between **v** and its projection onto the ground plane is φ. Equivalently,

$$\mathbf{v} \cdot \mathbf{g} = \cos\left(\frac{\pi}{2} - \varphi\right) = \sin\varphi \tag{4.7}$$

[3]This notation is defined in Section A.4 of Appendix A.

where **g** represents the perpendicular direction to the ground plane.

Combining Eq (4.6,4.7) and the fact that $|\mathbf{v}| = 1$, we have three equations :

$$v_x g_x + v_y g_y + v_z g_z = \sin\varphi \tag{4.8}$$

$$v_x u_x + v_y u_y + v_z u_z = 0 \tag{4.9}$$

$$v_x^2 + v_y^2 + v_z^2 = 1 \tag{4.10}$$

where v_x, v_y, v_z are unknown, and this can be solved by substitution. For a horizontal ground, Eq (4.8) simply states that

$$v_y = \sin\varphi \tag{4.11}$$

(which is also evident by looking at Fig (4.6)). Substituting this in Eq (4.10) gives

$$v_x = \sqrt{\cos^2\varphi - v_z^2} \tag{4.12}$$

And substituting Eq (4.11) and (4.12) into Eq (4.9) and squaring both sides gives a quadratic in v_z:

$$v_z^2 u_z^2 + v_z\left(2u_y u_z \sin\varphi\right) + \left(u_y^2 \sin^2\varphi - u_x^2 \cos^2\varphi\right) = 0 \tag{4.13}$$

Thus,

$$v_z = -\frac{u_y \sin(\varphi) \pm u_x \cos(\varphi)}{u_z} \tag{4.14}$$

and this is substituted back in Eq (4.12) to get v_x. Finally, the knee bend is computed by taking the dot product between the shin direction and hip direction (its local **y**-axis)

$$\theta_{Knee} = \mathbf{v} \cdot \left(\mathbf{q}_{Hip}\mathbf{y}\mathbf{q}_{Hip}^{*}\right) \tag{4.15}$$

Joint limits are applied using Eq (4.4) or (4.5) as desired.

4.4.3 Mapping the Ankle

The translation of the trackers along the **z**-axis is mapped to control the dorsiflexion and plantar flexion at the ankle (supination and pronation are currently ignored). As with the

knee, there is a choice of controlling the bend directly versus controlling the orientation of the foot with respect to the ground. The former is achieved analogously to Eq (4.3). The latter is achieved analogously to Eq (4.11,4.12,4.14), but where the direction vector **v** being solved for represents the ankle/foot direction, and **u** represents the ankle motion constraint provided by the shin orientation (the motion is perpendicular to the **x**-vector of the shin LCS). Eq (4.15) is replaced by

$$\theta_{Ankle} = \mathbf{v} \cdot (\mathbf{q}_{Knee} \mathbf{y} \mathbf{q}_{Knee}^{-1}) \tag{4.16}$$

where

$$\mathbf{q}_{Knee} = \mathbf{q}_{KneeLCS} \mathbf{q}_{Hip} \tag{4.17}$$

4.5 Arms

4.5.1 Direct Kinematics

The arms can be controlled similarly to the legs. That is, the tube containing the tracker is mapped to control the orientation of the humerus by rotating the shoulder. The rotational transformation for the left shoulder is accomplished by the following sequence of rotations:

$$R_{Left\_Shoulder} = \mathrm{R}_{\mathbf{z}}(-\pi/2) \cdot \mathrm{R}_{\mathbf{y}}(v_{yaw}) \cdot \mathrm{R}_{\mathbf{z}}(v_{pitch}) \cdot \mathrm{R}_{\mathbf{x}}(-v_{roll}) \tag{4.18}$$

and for the right shoulder by:

$$R_{Right\_Shoulder} = \mathrm{R}_{\mathbf{z}}(\pi/2) \cdot \mathrm{R}_{\mathbf{y}}(-v_{yaw}) \cdot \mathrm{R}_{\mathbf{z}}(v_{pitch}) \cdot \mathrm{R}_{\mathbf{x}}(v_{roll}) \tag{4.19}$$

The additional rotation term $\mathrm{R}_{\mathbf{z}}(-\pi/2)$ for the left arm would adduct the arm from pointing straight outwards to pointing straight down (and similarly for the right one). This may appear opposite to the required rotation, but it is in fact correct, in the same

way that rotating a point is equivalent to rotating its coordinate system in the opposite direction. That is, say the input orientation has been transformed so that when the arm is vertical, the **z**-rotation is considered to be 0. Then, in the LCS of the CG character, however, a "vertical arm" corresponds to a **z**-rotation of $-\pi/2$ radians, and thus an additional rotation $\mathbf{R_z}(-\pi/2)$ must be applied to represent the shoulder orientation in the correct frame of reference.

The height of the tracker controls the bend at the elbow, and the translation of the tracker along the z-axis controls the flexion and extension of the wrists. Both of these follow the same general form as given in Eq (4.3) for the knee. Also, both of these mappings have possible physical interpretations for the puppeteer, in the sense that the wrist flexion and extension can be controlled by a movement which involves the puppeteer flexing/extending his own wrists, and similarly for elbow flexion and extension, as shown in Fig 4.7.

4.5.2 Pulling the Shoulders By the Forearms

Other mappings for the arms are occasionally used. For example, the tube can be mapped to represent the position and orientation of the ulna and radius (comprising the forearm). In that case, one end of the tube, E (where the tracker device is located) corresponds to the elbow joint, and the other end, W, indicates a point on the forearm (somewhere near, though not necessarily at, the wrist), illustrated in Fig 4.8. This allows a control with properties such as the following:

- *translation*, e.g. when the user raises the tubes, the puppet's forearms are raised by rotating the shoulder appropriately,

- *orientation*, e.g. when the user changes the orientation of the tubes, the orientation of the puppet's forearms will change accordingly,

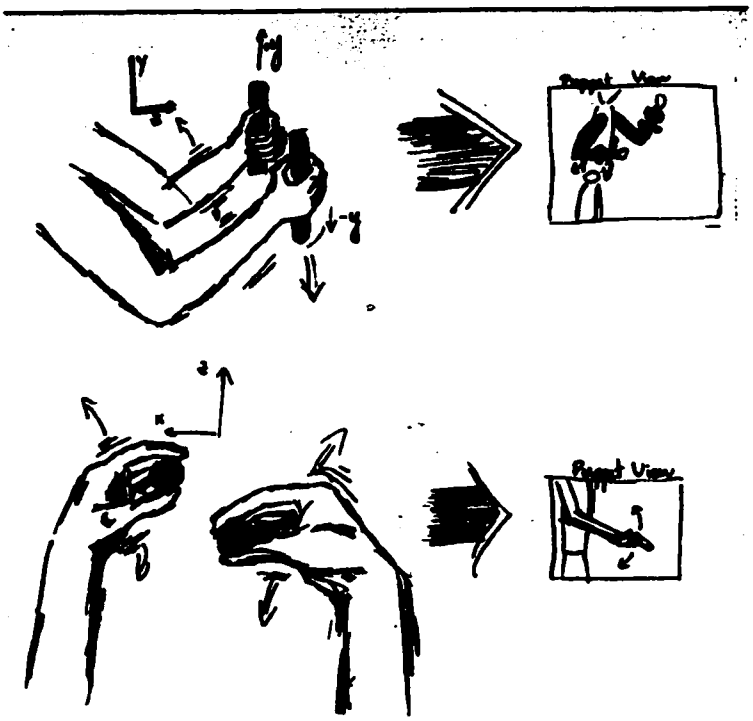

Figure 4.7: PHYSICAL CORRESPONDENCES FOR THE WRIST AND ELBOW MAPPINGS

The upper half of this diagram illustrates how the y-translation of the input, which controls the puppet's elbow flexion and extension, can be perceived by the puppeteer to be related to his or her own elbow flexion and extension. In the lower left quadrant (a view of the puppeteer's hands from above), the z-translation of the tracker input can be related to the wrist flexion and extension of the user, and controls the same parameter in the animated graphical character.

- *a fixed-point*, e.g. when the user keeps the elbow end E (of the Polhemus tube) steady and moves W, then the puppet's elbows will also stay still, and his forearms will move around.

The mapping is achieved as follows (for each arm individually):

1. **Compute the Shoulder Direction** An imaginary point O relative to the tracker source is chosen— somewhere near the chest or shoulder of the user. The vector OE from this point to the tracker position is normalized to lie in the unit sphere around O giving a direction vector $\mathbf{s} = \frac{1}{|OE|}OE$ from the shoulder to the elbow ($\mathbf{s}$ is shown in Fig 4.8). $\mathbf{s}$ is initialized to point straight downwards, and if the data received from the tracker is near 0, i.e. $|\mathbf{s}'| < \epsilon$ or otherwise invalid, then the previously used value $\mathbf{s}$ is retained.

2. **Compute the Corresponding Rotation Quaternion** This is a straightforward computation of finding the quaternion that rotates one given vector into another (see Appendix A, Sec A.4). The target vector is $\mathbf{s}$ as computed above, and the initial vector is $\langle 1, 0, 0\rangle$, the direction of the left humerus (or $\langle -1, 0, 0\rangle$ for the right one) in world-frame when its local shoulder orientation is considered to be zero.

3. **Compute Elbow Bend** The bend at the elbow is given by

$$\theta = \pm \arccos(\mathbf{e} \cdot \mathbf{s}),$$

where $\mathbf{e}$ is the direction vector in WCS for the orientation of the forearm as shown in Fig 4.8. $\mathbf{e}$ is obtained from the yaw, pitch and roll input parameters. The direction of rotation is reversed for the left and right arms.

4. **Compute the Shoulder Medial/Lateral Rotation**

This has so far specified a direction for the humerus, but the medial/lateral rotation of the shoulder is still left undetermined. This twist can be computed by noting

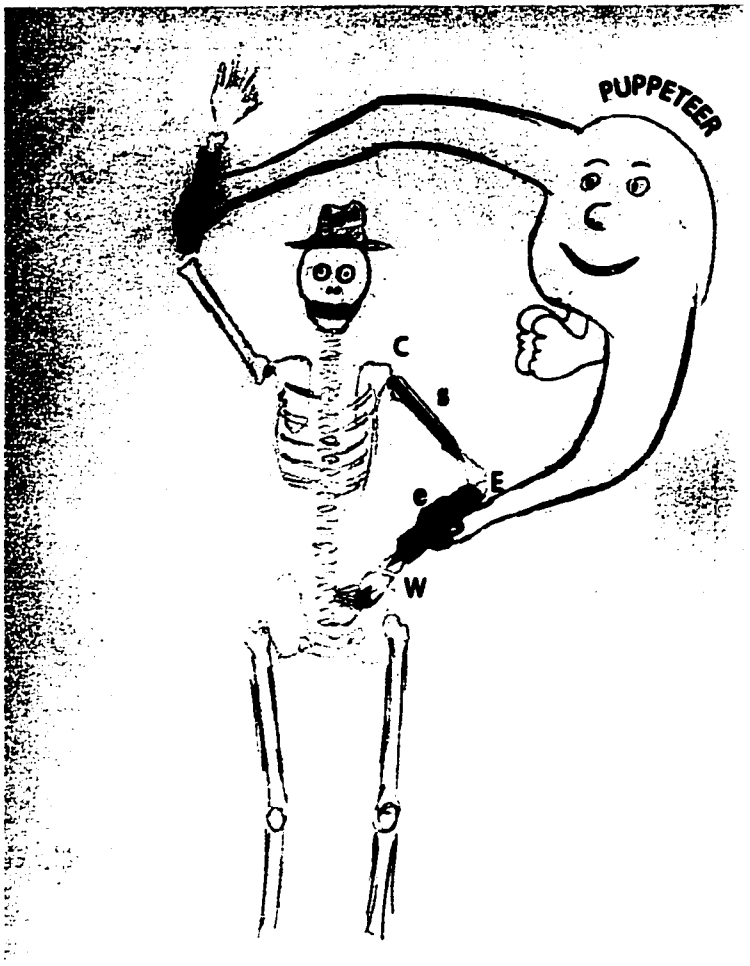

Figure 4.8: PULLING SERGEI BY THE ELBOWS— THE USER'S (MENTAL) MODEL

In this mapping, the tubes effectively correspond to the orientation of the forearms in the world coordinate system, and the trackers themselves correspond to the position of the elbows. The vector **s** is obtained from the input tracker position relative to a fixed point situated in the CS of the puppeteer (i.e. assuming the puppeteer and the source emitter stay roughly in the same relative frame of reference). The vector **e** is parallel to the direction of the tracker tube.

that the elbow is a hinge joint, so, given a shoulder direction $\mathbf{s}$, there can only be one way to simultaneously bend the puppet's elbow and twist his shoulder to achieve the user-specified orientation of the forearm, as given by the vector $\mathbf{e}$.

Recall that the hinge axis for the elbow is defined to be the $\mathbf{y}$-vector with respect to the shoulder LCS (see Fig 4.1). If $\mathbf{q_s}$ denotes the quaternion which rotates the zero/neutral shoulder direction into the current direction $\mathbf{s}$, then $\mathbf{y_s} = \mathbf{q_s y q_s^{-1}}$ is the rotated $\mathbf{y}$-vector in the shoulders' current frame of reference. Now, the desired value for $\mathbf{y_s}$, which we denote by $\mathbf{y}_d$ in world-frame co-ordinates, is given by the cross-product of the current humerus direction with the desired forearm direction: $\mathbf{y}_d = \mathbf{s} \times \mathbf{e}$. So we then add just enough twist ϕ to the shoulder to make the difference, which is the angular difference between the desired y-direction $\mathbf{y}_d$ and the current y-direction $\mathbf{y_s}$

$$\begin{aligned} \phi &= \arccos(\mathbf{y}_d \cdot \mathbf{y_s}) \\ &= \arccos((\mathbf{s} \times \mathbf{e}) \cdot \mathbf{y_s}) \end{aligned} \tag{4.20}$$

4.5.3 Additional Control Method: Inverse Kinematics By the Hands

A related mapping is to use the trackers to control the position and orientation of the hands, and use inverse kinematics to compute the orientation of the elbow and shoulder joints. In that case, care has to be taken to make sure that continuity is maintained. Either of these latter mappings could potentially be extended to allow the IK algorithm to work further back than the shoulders, and negotiate an orientation for the upper back depending on target positions for both arms, as well as on the orientation of the rest of the spine. If the puppet's hand is being manipulated, it would more effective to embed the polhemus in a flat box-type of shape matching the general structure of the hand rather than the tubes.

4.6 Spine

The spine is currently modeled by a 3 DOF joint at the lower back (vertebrae L5) which is attached to the root of the model, another 3 DOF joint at the upper back (vertebrae C7), and a hinge joint for the head (at C1). We use the orientation of each of the two trackers to correspond to the rotations at L5 and C7, analogously to the arm control of Eqs 4.18 and 4.19, and the height of one of the trackers to control the nodding of the head. The ball and socket joints are controlled analogously to the hip control, while the head nod at C1 is controlled using a linear relationship as for the ankles and wrists.

4.7 Pelvis: Root Motion

The translation of the pelvis is discussed in Section 4.7.1 since it is controlled indirectly. The direct mapping we use for the orientation of the pelvis, however, has depended on which of the leg parameters are being controlled simultaneously. If, for example, the hips, knees and ankles are being controlled using the mapping described in Section 4.4, then the only available input d.o.f. that is not already being used is translation of the trackers along the x-axis .

This can be done either by considering the distance of the trackers relative to one another, or relative to the source. When we only want to control the rotation of the pelvis around, say, the y-axis, then using the distance between the trackers is sufficient. Otherwise, it is possible to use the translation of one of the trackers (relative to the source) to control pelvic rotation around the y-axis while using the other one to control rotation around the z-axis. However, the puppeteer has not spent much time practising this mapping to see how hard it is.

If the leg parameters need not all be controlled simultaneously, so that all of the input translation dof's are available, then another method of control for the pelvis is to fix a line between the two hips, and bind the orientation of this line to be parallel to the

line between the two trackers. Using both hands to orient a line in this way has a very natural feeling, like a 3-dimensional steering wheel. The same method, of course, can be used to control rotation constrained to be around the y-axis as well, for example, using the x and z translation values of the input.

4.7.1 Keeping One Foot On The Ground To Move Forward

A critical issue in achieving satisfying control of the puppet is keeping it grounded. This is also the basis for locomotion. Since the Polhemus trackers translate freely in 3d space, it is virtually impossible to control the height of the pelvis directly without betraying the lack of any underlying ground constraint, since the puppet tends to hover near the ground, at best. Using a physical constraint at the input level, such as keeping one tracker (or tube) touching a flat surface is slightly better, but the trackers are very light so that even the slightest spurious motion which might raise one tracker from the table creates a noticeably un-believable effect in the graphical world.

Constraining One Foot Instead, we solve the problem by imposing the constraint that one of the puppet's feet must always be touching the ground[4]. This is achieved as follows: Suppose the puppet is in some initial state θ_i at time t_i with the left foot touching the ground at point $\mathbf{p}(t_i)$ in the WCS. Compute the new state θ_{i+1} of the puppet using any current active mappings, and compute the new global position of his foot $\mathbf{p}(t_{i+1})$. Now simply add an additional translation vector $(\mathbf{p}(t_i) - \mathbf{p}(t_{i+1}))$ to the root of the puppet, thus ensuring that the foot has not moved relative to the WCS. That is, we translate the root by just enough to compensate exactly for any translation that the fixed foot would have otherwise undergone. This is similar to the foot constraint used in [Per95].

[4]In Section 7.2.1, we suggest an approach to potentially overcome this constraint, without needing to scale beyond the desktop environment.

Alternating Feet Nailing one foot to the ground will not let the character go very far, so a mechanism is provided for switching feet. Suppose again the puppet is in state θ_i with the left foot touching the ground at $\mathbf{p}(t_i)$. However, after computing θ_{i+1}, we check whether the y component of the updated global position of the right foot, $\mathbf{q}_y(t_{i+1})$ is lower than that of the left foot by more than some threshold value, i.e. if $(\mathbf{p}_y(t_{i+1}) - \mathbf{q}_y(t_{i+1}) - \epsilon) > 0$. If that is the case, then we set the right foot to be the fixed one, and use $\mathbf{q}(t)$ in the subsequent root-translation computations until the next switch. An additional vertical translation is also applied at the transition times, to make sure that the $\mathbf{q}_y(t) = 0$. The purpose of the threshold ϵ is to discourage rapid thrashing between the two feet, as well as to allow the puppet to appear to be standing on both feet at the same time. By providing this mechanism for constraining and alternating the feet, the puppet can be made to walk forwards, as shown in the demonstration video.

Multiple Contact Points The preceding technique for alternating between feet is extended to allow for multiple contact points per foot. We use contact points at the toes and at the heels of each foot. Now, instead of just comparing the y-components of $\mathbf{p}$ and $\mathbf{q}$ (which corresponded to an arbitrarily chosen point on each foot), we compare the y-values of all 4 active contact-monitoring points. Then we compare the y-value of the lowest contact-monitoring point with the y-value of the currently fixed point. Once again, the difference must exceed a small threshold before the switch actually takes place. For larger numbers of contact points, faster collision detection algorithms can be used.

Chapter 5

Physics-Based Models For User Control

5.1 Motivating Local Physical Models

5.1.1 Introduction

The mappings described in the previous chapter allow a wide range of possible motions to be achieved with sufficient practise. The user's practise experience itself will be discussed in Chapter 6, but one observation is singled out here, as it initially motivated the direction to be described presently: A key element of learning to control the CG character is the repetition of brief, isolated motion sequences; one such motion in particular, shown in Fig 5.1, is based on a warm-up for Tai Chi. In this exercise, the character stands still and keeps the free thigh approximately horizontal while letting the hanging foot draw small circles in the air by swinging from the knee. Practising this simple exercise made it apparent that it was very difficult to control the simple swinging of the puppet's limbs. Granted, for a real person to do this exercise properly may not be as easy as it first seems, but trying to have the CG puppet do this nevertheless highlights the fact that swinging motions are very hard to "fake" kinematically, perhaps because of their very

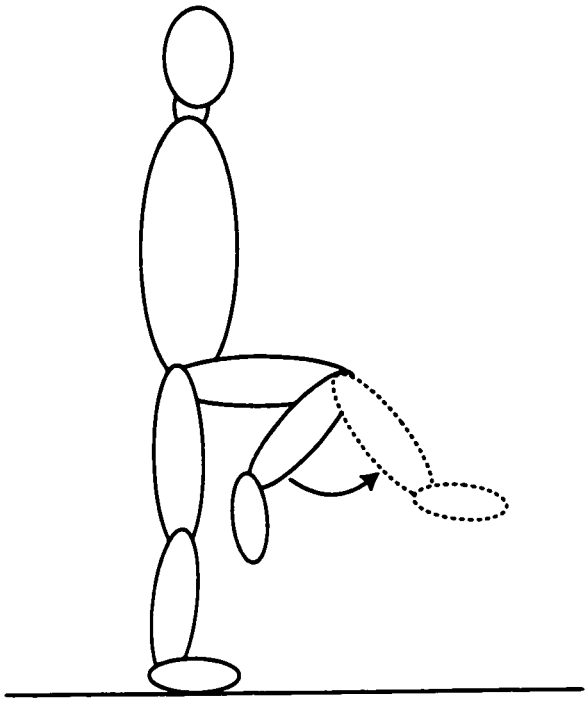

Figure 5.1: A KNEE-SWINGING EXERCISE

The character stands on one leg and most joints are kept relatively still, while the non-supporting knee swings freely.

recognizable and smooth acceleration pattern. Instead, we would like to allow motions of this sort to occur naturally, as they do in real life. When possible— and it is certainly not possible all of the time— we would like the puppeteer not to have to exert a lot of effort to achieve those motions which we achieve precisely by *not* exerting effort ourselves, such as allowing a limb to swing freely.

The reason, of course, that we do not have to work to swing our leg is because we can relax and let gravity and inertia do it for us. To give our model these benefits, one might therefore consider adding dynamics to the simulation. However, doing so is not quite straightforward: interactively controlling a physical simulation of an articulated figure is extremely difficult, primarily due to balance issues (as well other reasons explained below). Our goal, on the contrary, is to facilitate the puppeteer's control over subtleties in the motion, in this case to simplify his or her task by allowing gravity and momentum to "do some of the work" when possible, leaving the puppeteer free to control other complexities of the motion. We would like the puppeteer to focus her cognitive resources on realizing specific ideas of motion, with all their subtlety, into animation. In this light, using a full dynamic simulation might defeat our purpose, as it would add an extra layer of complexity (i.e. balance) to all motion, forcing the puppeteer to continually allocate cognitive resources to the difficult task of trying to balance the figure. We solve this problem by creating local dynamic models— in our case at the knees and ankles, but extendible to other joints. In this way, we are able to incorporate physics-based principles in as much as they facilitate the animation task, yet without creating additional overhead such as requiring the user to solve the balancing control problem in real-time. Before elaborating on these local models, we first discuss in more detail why controlling a dynamic simulation is so hard.

5.1.2 Challenges of Interactive Control of Dynamics Simulations

The use of physical simulation in interactive applications is a direction which has just begun to emerge for character animation. The challenge of interactively controlled dynamic simulation is extremely hard for many reasons— including speed, complexity, bandwidth, and parameterization considerations. One example of work in particular that addresses this problem is that by Laszlo, van de Panne and Fiume [LvdPF00], which we will use to contextualize some of the following remarks. Another source is our own set of initial experiments in trying to interactively control a full 3D simulation.

First, it is computationally difficult to achieve dynamic simulation of complex 3D systems in near real-time speeds. This is, of course, a prerequisite for interactive control. The characters used in [LvdPF00] are planar, allowing real-time simulation to be computed on a 450Hz Macintosh.

Second, dynamic balancing is hard, and inverse dynamics— finding a trajectory *sufficiently close* to the desired one, and also which subsumes balancing throughout— is therefore at least as hard. Selecting the right torques to do this offline is not easy, but for a user to do it online is near impossible. Cognitively, it is full of intricate and subtle complexities that we are simply unaware of. For example, in the early attempts at using a full dynamic simulation for the present work, every time the character took a step forward to start walking from a standing position, he would fall over before he could pick up his leg. The reason for this, it turned out, was because he had not yet shifted his centre of mass to lie above the supporting foot, and therefore bending the hip and knee did not raise the leg, but rather dropped the corresponding side. Visually, this weight-shifting can be a nearly imperceptible adjustment, perhaps not even noticeable from many 2D projections (i.e. viewing windows), and that combined with the lack of force feedback hides the causes of such problems from the user. Unlike kinematic control, a dynamically

simulated character can get "out of control", and by the time the user realizes this, there is no way to recover, so the character will fall over. Sometimes, even if the user realizes it early enough, the required correction might be surprisingly elusive, or counterintuitive from the user's "conscious" perspective (regardless of whether our bodies would have known how to react appropriately[1]). Note that using a 2D simulation context as in the work by Laszlo et al. [LvdPF00] does have the advantage of making the balance problem more tractable for interaction.

Assuming a fairly low-level control scheme, interactive dynamic simulation control also requires a high input bandwidth, since so many parameters need to be controlled simultaneously, making it unclear if and how it could be integrated with a multi-tracking system. And if it cannot, then by how much more would the kinematic control need to be expanded, in terms of the number of degrees of freedom being controlled simultaneously? Additionally, non-static balancing, and dynamic behaviour in general, are sensitive to small variations in any of the parameters such as torque values or timing. Thus, not only do many parameters have to be controlled, but they have to be controlled accurately, particularly in the time domain. A helpful trick is to use a slowdown factor, giving the user more time to react and make precise choices. Laszlo et al [LvdPF00] report a "sweet spot" for interactive play rates, such that the simulation is running slow enough to give the user reaction time, yet fast enough to provide a sense of rhythm and sufficiently responsive feedback time (the issue of feedback in dynamic simulation, is discussed more in Section 5.1.3). The most comfortable speed for a planar walking biped was 2 to 3 times slower than real-time, though real-time was achievable with sufficient practise, while a factor of about 80 was needed for a long jump for a very small light biped character.

[1]Our "body" reactions to these situations can be trained, too, as in some cases, even our body does not automatically know the best thing to do... A curious note on this point: in Tai Chi, there are a number of exercises in which two people each try to push the other off-balance. A remarkably effective technique for remaining balanced is to relax certain muscles which feel like they are "supporting" one's weight. It is completely opposite to most people's instincts, yet in some cases is even easy when a more advanced practitioner points it out.

For their cat character, slowdown rates of between 10 to 40 times were appropriate, depending on the the specific task being carried out, such as bounding or trotting. Once again, planar characters have fewer degrees of freedom to be controlled, thus significantly reducing the necessary bandwidth.

Finally, there is the difficult and open question of what exactly the control parameters should be. Any given choice of control method, of course, has an associated set of implications, some of them again quite subtle or complex. Should the user apply torques directly at the joints? Suppose for example that the user input signal is converted directly into torque τ_i at joint i by a continuous function, so $\tau_i = \phi_i(\mathbf{x})$. If ϕ_i is linear, then there would only be a single input value $\mathbf{x}_0$ for which $\tau_i = 0$, making the system very unstable. This is a simple example, and clearly is not an acceptable solution. This could be corrected by allowing a continuous interval range R_0 such that $\phi_i(\mathbf{x}) = 0$ for all $\mathbf{x} \in R_0$. But would the user be able to "find" and return to this zone easily during an animation session, and how would the user know that he or she has reached this zone for many parameters simultaneously? So should target locations be given for proportional derivative control instead? Indeed, that is an often taken approach (and will be discussed below in relation to this work). In that case, can the stiffnesses be controlled simultaneously, or do they remain fixed? Some other interesting questions include: What kind of synergies can be built in? Could the user manipulate basic motion primitives such as those described by Mussa-Ivaldi et.al. [MI97, MIGB94]? Would automatic balancing mechanisms interfere with the responsiveness of the system from the user's point of view? Although these latter questions are not explored presently, they do invite interesting future investigation. How to parameterize control of a physical simulation, in particular for real-time interaction, is clearly a very interesting and complex issue.

5.1.3 User Feedback

There are a number of unique and subtle issues that arise in regard to the user feedback for controlling full physical simulations. We originally experimented with various types of feedback. For example, to indicate the force being exerted by a PD-controller at a joint, a vector can be displayed showing the target position for the corresponding link. Assuming a constant stiffness, the angular difference between the current and target link orientations indicates the force being applied. The length of the vector can be used as an additional indicator for the same information. For a few links, this can be useful, but after that it starts getting visually cluttered and harder to read. Another issue is that since the simulation time is generally not the same as the real time (e.g. Δt might be getting incremented by $0.05s$ after every $0.10s$ of real time), accelerations are not being represented accurately. We also tried to help the user by numerically or otherwise displaying the ratio between simulation-time and real-time.

One of the fundamental difficulties is that we do not know how much net force an object is experiencing until after we have seen how much it accelerated. If, say, the goal at a given moment is to prevent acceleration (e.g. balancing), then we need to know how much resisting force to apply so that the net force is zero, but by the time we can guess this it is too late. By the same token, even in the absence of an external force, simply manipulating a PD controller can be tricky because we do not *feel* how much force we are applying at any given time in relation to the inertia of the object.

Another important observation in relation to the importance of feedback in controlling a full dynamical simulation is that when we make contact with a real object (e.g. the ground), we receive a burst of sensory information, irregularly distributed across the entire contact surface, and a proprioception of the forces as they propagate from the contact throughout our body, often accompanied by visual and auditory information as well. How exactly these sensations affect our actions is a discussion beyond the scope of this work, but it is most likely an extremely complex process. It is hard to imagine

that by visual feedback— even in combination with a few point forces (e.g. simple haptic devices), applied at locations other than the "virtual" contact surface (and therefore not propagating through our body in the same way, and not offsetting our centre of mass in the same way)— we could respond in a way nearly as well tuned as the way we naturally do in the physical world.

5.1.4 Desired Properties and Local Physical Models

Having outlined some of the major sources of difficulty in controlling a full dynamic simulation, we re-examine the need for physics in the first place, in relation to our goals: what specifically are the benefits we wish to gain, and at the same time, what are the advantages of kinematic control that we wish to retain? Recalling that our goal is to provide the user with a satisfying control interface for animation, not necessarily a realistic simulation, we explicitly list some desired properties, and see how they can be achieved by local dynamic-based models rather than by a full physical simulation.

In particular, desired motion and control properties for the virtual puppet's knee can be summarized as follows:

a1 During the swing phase of walking, the lower leg should indeed swing along, including the possibility of occasionally extending beyond its "goal" orientation just before stepping down.

a2 If the left thigh, say, is raised so that the person is balancing on the right leg only as in Fig 5.1, then the left shin should swing naturally like a pendulum (since that is just what it is).

a3 When standing, the supporting knee should have some springiness to it, unless it's fully extended (locked).

a5 Regardless of how the above properties are implemented, the puppeteer should still

be able to manipulate the knee into any desired position within the natural range of motion, and maintain it there for as long as he wants to.

Likewise, a similar set of desired properties can also be listed for the ankle:

b1 When the swinging leg strikes the ground— either with the heel or with the toes— and becomes the supporting leg, then the natural tendency should be for the rest of the foot to come down flat on the ground. The need for this was evident from trying to control the CG character, as it was both difficult to achieve ankle motion that did not look artificial, and hard to continually monitor the behaviour at the ankles when the rest of the leg motion was occupying the puppeteer's focus.

b2 The ankle of the supporting leg should have a springy quality when the weight is on the ball of the toes. (Note that we currently model the foot with a contact point at the heel and contact point at the toes; we do not model the rotation of the tarsal phalanges about the metatarsophalangeal joint).

b3 It should be easy to keep the foot of the supporting leg parallel to the ground, but also easy to bend the ankle to push off at the end of the support phase.

b4 Regardless of how the above properties are modeled, the puppeteer should still be able to manipulate the ankle into any desired position within the allowable range and maintain it there for as long as he wants to. This is to ensure that any such "automatic" behavior added to the system does not reduce the extent of the puppeteer's potential control (including those motions which happen to be unrelated to walking). For example, it should not be too hard to make sure that the toes of the swinging leg clear the ground, or to walk on tiptoes as well.

b5 Finally, and very importantly, although gravity needs to be incorporated (e.g. in helping the foot to foot to fall flat onto the ground) the puppeteer should not have to do any extra maneuvering in order to keep the puppet from falling over.

5.2 Augmenting a Kinematic Model With Local Physically-Based Properties

To see how these properties can be implemented by a local physical model, consider, as a typical example, a simple two-link structure with a hinge joint as shown in Figure 5.2, where link 1 is fixed rigidly to the ceiling at one end, and the orientation θ of link 2 is

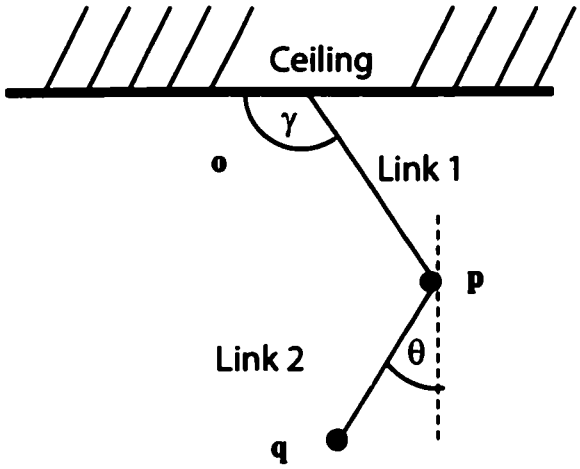

Figure 5.2: TWO LINK CHAIN

Link 1 is attached to the ceiling at an angle γ at point **o**, and Link 2 is a attached to Link 1 at an angle θ by a hinge joint. A point mass m is assumed to be at **q**.

controlled directly according to a mapping

$$\theta = \psi(v) \tag{5.1}$$

of input v. Now suppose that a simple physically-inspired property is introduced so that as link 2 rotates about point **p**, the angular velocity $\dot{\theta}$ is maintained, so that $\dot{\theta}$ at time t_{i+1} is given by

$$\dot{\theta}(t_{i+1}) = (1 - \lambda)\dot{\theta}(t_i) \tag{5.2}$$

where $0 \leq \lambda \leq 1$ is a damping coefficient. The state of the joint can then be updated by

$$\theta(t_{i+1}) = \theta(t_i) + \dot{\theta}(t_i)\Delta t \tag{5.3}$$

where $\Delta t = t_{i+1} - t_i$. This has effectively introduced behaviors of angular momentum and damping into the system, but it has also removed the user from the control loop (the motion is now parameterized only by the initial state $\theta(t_0)$ and λ). When physical properties are introduced, then a pure kinematic control mechanism is no longer applicable, and some new control must be provided within the context of physical simulation. Of course, the obvious framework within which to reincorporate interactive user control is by having the user cause accelerations by applying forces (or torques), as opposed to directly specifying (instantaneous) angular displacements. Instead of directly computing positions as in Eq (5.1), or velocity as in Eq (5.2), we use an underlying torque model, with a standard Euler-step dynamic simulation update:

$$\dot{\theta}(t_{i+1}) = \dot{\theta}(t_i) + \ddot{\theta}(t_i)\Delta t \tag{5.4}$$

$$\theta(t_{i+1}) = \theta(t_i) + \dot{\theta}(t_i)\Delta t \tag{5.5}$$

to give the state of the joint $\theta(t_{i+1})$ at time t_{i+1}, where $\ddot{\theta}$ is proportional to the sum of active torques.

Now a natural way to get a torque from a kinematic mapping ψ is by imagining an angular spring with stiffness k at joint $\mathbf{p}$ that has $\psi(v)$ as its equilibrium value:

$$\tau_{control} = k(\psi(v) - \theta) \tag{5.6}$$

The damping can be caused by a torque

$$\tau_{damping} = -\lambda\dot{\theta} \tag{5.7}$$

This, combined with $\tau_{control}$ is essentially a proportional derivative (PD) controller.

To enable the natural swinging motion suggested previously, gravity needs to be added to our model. This is done by adding the following torque (computed as in Fig 5.3):

$$\tau_{gravity} = |g|m \sin\theta \cdot r \tag{5.8}$$

where r is the length of link 2, and a point mass m is assumed to be at $\mathbf{q}$.

Combining Eq (5.6), (5.7), and (5.8), we have the total torque simulated at joint $\mathbf{p}$ to be:

$$\begin{aligned} \tau_{total} &= \tau_{gravity} + \tau_{control} + \tau_{damping} \\ &= |g| mr \sin\theta(t_i) + k\left[\psi(h) - \theta(t_i)\right] - \lambda\dot{\theta}(t_i) \end{aligned} \tag{5.9}$$

The difference between the model described here and typical PD control is that we are not really in a physical simulation: returning to Figure 5.2, suppose γ is no longer fixed, but controlled kinematically by a mapping ψ_2, so $\gamma = \psi_2(\upsilon)$. We can still use Eqs (5.9,5.4,5.5) to control joint $\mathbf{p}$ by force, while using a kinematic simulation to control joint $\mathbf{o}$ directly. Of course, the motion of link 2 will no longer be physically realistic, because rotating link 1 actually accelerates link 2, however for our purposes this is not critical. Ultimately, our goal is to map from real-time input onto a real-time graphical animation: if the animation control works better with some physically-inspired properties than with none, then we make use of this. If we were to add all of the complex physical interactions that can occur between all the joints of a human articulated figure in 3 dimensions, then the simulation— just the act of balancing let alone locomotion— would become so hard to control in real-time that the benefit would be lost. So we just add what we can use.

5.2.1 Adding a Physics-Based Model to the Knee

A model for controlling the knee can be separated into two distinct cases, based on whether the corresponding foot is in the air or on the ground. When the foot is in the air, as in Fig 5.3, then the general form of the control is as given in Eq (5.9), where r now represents the length of the shin.

For the knee of the supporting leg, gravity is ignored, leaving a PD controller. This is just enough to provide the puppet with nice "soft" knees which are not too stiff.

5.2.2 Adding a Physics-Based Model to the Ankle

An effective model for the ankle motion control is harder to achieve, because of the potential complexity introduced by ground contact, and because there are more "motion properties" we wish to obtain, and they are more subtle.

Motivated by the requirements listed in Section 5.1.4, we use a dynamically-based model, as shown in Fig 5.4, wherein the heel contact with the ground is treated as a hinge joint. The downward action of the foot which happens at heel strike results as follows (referring to Fig 5.4):

1. The force due to gravity, $\mathbf{g}$, is resolved into two components:
   (a) $\mathbf{g}_0$, which is perpendicular to the direction of the lower leg, and
   (b) $\mathbf{g}_l$, which is parallel to the leg, and has magnitude

$$|\mathbf{g}_l| = |\mathbf{g}| \cos(\alpha)$$

2. That second component, $\mathbf{g}_l$, is redrawn in Fig 5.4 at the ankle as $\mathbf{f}$, which is then resolved into two further components:
   (a) $\mathbf{f}_0$, which is parallel to the foot, and
   (b) $\mathbf{f}_l$, which is perpendicular to the foot, and has magnitude

$$|\mathbf{f}_l| = |\mathbf{f}| \cos(\beta - \pi/2)$$

This leaves us with the simple configuration shown in Fig 5.5. So the torque exerted at the ankle due to gravity is

$$\begin{aligned} \tau_g &= |\mathbf{f}_l| \cdot r && (5.10) \\ &= |\mathbf{f}| \cos(\beta - \pi/2) \cdot r && (5.11) \\ &= |\mathbf{g}| \cos(\alpha) \cos(\beta - \pi/2) \cdot r \end{aligned}$$

where r is the distance from the heel to the ankle.

This model assists in getting the foot flat on the ground since gravity will push the mass of the skeleton down to apply a torque on the supporting foot as required. Furthermore, the perpendicular component g_0 of g (in Fig 5.4) is intentionally ignored, so that the character does not risk falling over.

The user is provided with a PD-controller, once again, to manipulate the joint angle at the ankle by specifying a target equilibrium joint angle value.

5.2.3 Controlling the Stiffness Parameter

There is one snag with the control over local physical models as given so far: We want to facilitate natural swinging motion, yet we also want the user to be able to control high-acceleration motions. Natural swinging will occur when the dominant term in Eq (5.9) is the torque due to gravity, whereas for the user to control high-acceleration motions, the dominant term needs to be $\tau_{control}$. In order for $\tau_{control}$ to be large, looking at Eq (5.6) shows that either $|\psi(h)-\theta|$ must be large, or k must be large. But $|\psi(h)-\theta|$ is only really large if the user is vastly overshooting the target values, both when initiating motions as well as when stopping them, which is very difficult to do accurately. Suppose, then, that k is set to be very large. Then the joint motion will closely follow the target value trajectory $\psi(v)$ as controlled by the user. In this case, if the user simply wants the leg to swing freely, then he or she must draw just the right control path to do this. That is, the puppeteer uses an extremely high stiffness and then *pretends* to swing the leg. In fact, in the limiting case as $k \to \infty$, the model tends towards becoming kinematic control, but with additional gravitational effects[2]. But this is exactly what we are trying to overcome in the first place!

The above argument shows that k must necessarily change over time somehow. The challenge is that we do not have enough degrees of freedom to allow the puppeteer to control both the stiffnesses and the target values independently and simultaneously. We

[2]The kinematic process is much faster and more numerically stable, however.

have solved this issue for the knee and ankle control as follows.

Controlling Stiffness at the Knee

The solution currently adopted for the knee is to have the stiffness k vary depending on the angular distance of the target orientation from the world vertical axis. That is, the user input v deteremines a knee angle $\psi(v)$ relative to the thigh, which, depending on the rest of the puppet's configuration, specifies an orientation $\psi_{WCS}(v)$ in the WCS for the shin. It is this target orientation, relative to the vertical axis in WCS, that determines the stiffness, which we can now write $k(\psi_{WCS}(v))$. It should be emphasized that the stiffness is a function of the user-specified target orientation, not the current orientation. Although this is still a reduction of a degree of freedom in the user-controllable parameters, from a user's point of view it may well be more attractive than any of the fixed-stiffness solutions.

When $\psi_{WCS}(v)$ is near 0, i.e. nearly vertical, then the stiffness is very small, allowing the knee to swing almost freely. When we reduce the stiffness, we also make a corresponding reduction in the damping parameter λ. As $\psi_{WCS}(v)$ approaches a diagonal, the stiffness plateaus to a higher value. The rationale behind this is that when the shin is intended to point downwards, we let gravity do the work of keeping it there, and therefore we can relax the corresponding muscles, allowing it to swing on its own momentum. On the other hand, if it is oriented at a larger angle from the vertical, then either it is at the edge of a swing phase (which the puppeteer could achieve by making $\psi_{WCS}(v) = 0$, and consequently k is small), or else we intend it to be there, so $\psi_{WCS}(v) \approx \theta$ with $k \gg 0$. Originally $k(\psi_{WCS}(v))$ was a piecewise linear function, but this created unnatural accelerations as the leg approached horizontal target values, so a tight gaussian-shaped function is currently being used, with k approaching 0 only in the region where $\psi_{WCS}(v)$ is near 0.

For the supporting leg, the solution is simpler, in that the stiffness can remain constant.

Controlling Stiffness of the Ankle.

Regarding the ankle motion, the question is how to allow the supporting one to be loose as the rest of the body rotates around it , while allowing for much larger stiffness during the plantar flexion occurring as the supporting leg pushes off the ground . This is different from the knee, in that at one moment the ankle controller needs to be completely relaxed, while at the next moment, quite a large torque needs to be exerted.

In this case, the current ankle pitch γ (in WCS) (that is, its angular difference from the global xz-plane) is tested for whether it is within some very small tolerance of being horizontal. If that is the case, then no gravitational torque is applied, so that the foot does not rapidly oscillate between the heel and toe contact points. Furthermore, if the pitch of the supporting ankle is already horizontal, to within $\pi/20$ radians, and if the difference $|\psi_\gamma - \gamma|$ between the pitch of the target orientation and the current pitch is less than $\pi/10$ radians, then a pure kinematic simulation is used to keep the ankle perfectly level[3]. Otherwise we revert to a constant, fairly high stiffness. The ankle target angle is directly controlled relative to the ground .

Thus the user only needs to specify the ankle to remain *approximately* horizontal (once it already nearly is) to ensure that it will stay completely horizontal. It also means that the puppeteer can still override this behaviour by overshooting the target value; if he or she sets a target value which is more than $\pi/10$ radians from the current position, then the usual torque simulation will kick in.

[3]Originally another physical simulation with different parameters was used at this point to have the same effect, but it required a much smaller stepsize for numerical stability, so the momentary kinematic solution was more convenient.

5.3 Discussion and Future Work

5.3.1 Stiffness and Relaxation

The variability in the stiffness parameters, in addition to being an essential component of the control interface described above, is also illustrative of the general question of how to let a (simulated) character relax. Although it is hard to establish the exact functions that it serves, relaxing muscles appears to be crucial to natural human motion and posture, including the act of balancing. Greene [Gre82], for example, emphasizes that "we make more use of ballistic arm movements (the arm swinging mainly by its own momentum) than is common in robots"[4], and indeed, the same could be said of our leg motions, especially during locomotion. Anyone who has studied a physical-based activity— whether dance, sports, playing a musical instrument, martial arts, etc.— will know that for nearly every specific physical motion that is introduced, a good teacher will often spend a lot of time reminding the student to relax the rest of his or her body while carrying out the motion. Bernstein [Ber67] points out that while people's movements are stiffer when they first perform a new task, at the more advanced levels improvement occurs in tandem with relaxation:

> The second, highest stage of co-ordination [...] corresponds to that described by sportsmen and music teachers as "relaxing", a phenomenon which they instinctively feel but which they do not know how to describe. The economical effect of the transition to this stage is apparent; [...] it is also the case that all those mechanical-reactive forces in the complicated link systems, which, in the best cases, occurred without damage at the previous level of co-ordination, are used in a positive sense.[5]

A quintessential example of this is the way in which by relaxing, we can often let gravity work for us rather than us working against it. Relaxing seems to help us keep our

[4][Gre82, p.263]
[5][Ber67, p.109].

balance (see footnote on pg. 93), too. From a generative perspective, relaxation could be as important for animation as nearly any other aspect of physical correctness.

Yet, the issue of "letting go" or relaxing as opposed to active physical control is one which is rarely addressed in the animation control literature. This may explain why some of the physics-based animations have an unnatural look about them despite their physical integrity. Assuming, as in our abstracted model, that "relaxing" corresponds to lowering the stiffness k, then the question of how (and when) to implement relaxed motions in a physical (or even partial-physical) simulation is still quite open. Moreover, providing a user with a mechanism for varying stiffness/relaxation independently during *real-time interactive physical control* seems even more difficult, since many of the available input parameters (and corresponding cognitive load) are already dedicated to specifying the desired posture itself. At the same time, doing so would bring a natural efficiency and accuracy to the control of the simulation. In our ankle model, for example, it is in principle the temporary looseness of the joint which allows the heel and toe to stay flat on the ground during the stance phase as the rest of the leg rotates around them, rather than requiring precise control from the user to do so. Similarly, it would be nice if the whole leg could swing naturally as the user is stepping forwards, with the user not having to exert any effort at all if he or she does not want to (other than perhaps a torque at the knee to make sure that the foot clears the ground). Otherwise, the limb's motion is still effectively and visibly the result of being *pulled* towards "target" positions.

There are a number of ways to potentially squeeze in this additional parameter into the control interface. The most feasible and natural solution would probably be through the use of grip-strength controllers, having the obvious advantage that the user's "tension" gets mapped onto tension of the character, although specifying the joint on which to apply this could be tricky. Interestingly, perhaps even two discrete stiffness control values, corresponding to high and low (zero) stiffness, would still give significant advantage over a single constant value. Although a grip-strength control may be able to sense

both continuous and discrete signals, the latter could even be accomplished with simple buttons that can be held and released. A more complicated option in terms of user control, but one which does not require additional input devices, is to try to use any remaining degrees of freedom in the input that are available. For example, if the motion along the x-axis of the tracker is not already being used for the leg control, then this could correspond to stiffening or relaxing one of the leg joints. Yet another option could be to use an additional input device such as foot pedals. Finally, another possible approach might be to have intelligent controllers at each joint:

$$\left.\begin{array}{l}\text{user input [+ history]}\\ \text{dynamical state [+ history]}\end{array}\right\} \longrightarrow \text{controller} \longrightarrow k, \Delta x \longrightarrow \text{torque} \qquad (5.12)$$

where the user input still provides low-level control via desired joint angles for the character. Based on the given target values and state, and possibly some history of each, the controller should ideally come up with the appropriate stiffness and equilibrium parameters for the torque activation.

5.3.2 Equilibrium-Point Control

PD-control is a specific instance of equilibrium-point control. As such, it has properties well-suited for our task, as well as some issues that require consideration. For one, equilibrium-based control has the nice characteristic— by definition— that it can convert a kinematic or posture-based control into a dynamic control mechanism. For any gesture made by the user specifying a desired posture of the character, the difference between the character's current posture and the desired one can be used to calculate the resulting torques at each of the joints. The spring model given by Eq (5.6) could be replaced by a wide variety of functions, while preserving this characteristic. One such function that has been used for transfering targets to torque is Eq (2.2) (see pg. 39) as given in [AGL86] along with maximum bounds on the torque values according to tables given in [Pla71].

Regardless of the torque function, another property of equilibrium-point control is that the mapping from input signal to output signal (the final character's motions) is non-stationary, in the sense that an essential quality of inertia and gravity is that the character will sometimes move without input from the user, while at other times the user's motion will cause the character to stay still. This is not necessarily unintuitive, but it does have both advantages and disadvantages. The advantages, as described earlier, are that many desireable effects occur naturally. This applies to simple motions such as swinging, but also to more complex ones. If a character falls in a full dynamic simulation, for example, the user does not have to simulate the specific sequence of postures that occur during a fall, but can even continue to apply torques, possibly corrective ones, which will make the fall look more realistic. The disadvantage is the potential cognitive complexity of controlling this behaviour.

If the stiffness and damping are very high, then this "non-stationary"-ness can sometimes be neutralized, allowing the desired position to be just slightly ahead of the character's current position. External force fields (e.g. gravity) and reaction or collision forces will affect this, of course.

If the stiffness and damping are very low, however, then equilibrium-point control can possibly lead to a fundamentally different control strategy. For if a constant equilibrium point is maintained, then the applied torque will change significantly as a function of the current joint angle. This, in turn, provides the user with damped sine-like functions corresponding to any hand-input vector, just as a single push on a swing is enough to keep it going for at least several cycles. Possibly these oscillators could be coerced so as to provide good "basis" functions, which in combination would achieve the cyclic torque-functions involved in walking. This is perhaps related to the phenomenon described by Bernstein [Ber67, pg. 108]:

> [...] two successive stages of release may be observed. The first degree corresponds to the lifting of all restrictions, that is, to the incorporation of all possible

> degrees of freedom. They no longer interfere with the movements of the organism, but introduce complicating reactive phenomena, additional oscillatory frequencies, and so on. [..] The second [...] stage [...] corresponds to a degree of co-ordination at which the organism is not only unafraid of reactive phenomena in a system with many degrees of freedom, but is able to structure its movements so as to *utilize entirely the reactive phenomena* which arise.

Gravity, the stiffness of the joint controller, damping, as well as the length of each limb all contribute to the natural swinging frequency and general response of the joint to the user's control at any given time . Thus, natural frequencies of oscillation are introduced which did not exist in a purely kinematic simulation. These may present the puppeteer with certain constraints that occur over time; they may also provide a metronomic regulation to the movement, if the puppeteer is able to take full advantage of them. Currently it is not altogether clear how these natural frequencies are being used or ignored during the puppeteering.

5.3.3 Extending the Simulation - Future Work

Most of the comments of the preceding sections (5.3.1 and 5.3.2) apply equally to both full and local dynamic models. One long-term aim of this work, however, is to animate by making use of a full 3D dynamic simulation, as this has potential for the greatest naturalism of motion. The key to achieving this long term goal will be an incremental approach, meaning that functional intermediate solutions (and questions) need to be found. The present work, by creating local physical models in conjunction with kinematic control, offers the first step in opening up further directions along these lines. The question, then, will be: how to proceed? In order to be functional, the primary concern at any stage must always be to provide a good environment for puppeteering natural-looking animation, above that of achieving realistic physical simulation. Thus, at this stage it is not only legitimate but even advantageous, for example, to use gravity for the

swinging knee while ignoring it for the supporting one. Another parameter which has not yet been modeled is the acceleration at the knee due to the rotation of the thigh about the hip joint. Having an incremental approach is helpful because any physical parameter introduced requires significant amount of adjustment to tune it to work in synchrony with all other existing parameters. Furthermore, it must be tested to avoid the introduction of new instabilities or unpredictable behaviour (from the point of view of the user) which are not worth the visual gains that it provides.

The next natural extension will be adding a physical model to the hips and arms. When adding the hip model, some care may be needed once again to avoid having balancing issues, while still allowing interaction between the hip and knee motion. A full simulation for the arms, and then for the upper body should be feasible as well, where the orientation of the upper body might initially be controlled kinematically. Note that the current multi-track mapping still requires local models, as the full set of degrees of freedom cannot yet be controlled simultaneously.

As discussed earlier in Section 5.1.2, preliminary experiments in this research involved attempts to control a full 3D simulation of a two-legged character with a box for its upper body. However, the kinematic mappings themselves were not yet as refined as they are now, making the control virtually impossible, but nevertheless one lesson learned from that experience was that as the number of simulated parameters increases, it is important, if not crucial, to simplify the problem— at least initially— by enforcing various physical constraints. One such set of constraints involves the application of torques to keep the character from falling. These can be implemented in a variety of ways. For example, "contact"-type forces can be automatically applied to keep the orientation of the character's body completely vertical. On its own, this will not allow a natural walk, since walking is actually based on "falling" forward [AGL86]. Another variation is to allow the user to directly control the orientation of the character, using the orientation of the tube, for example, and applying *external* torques to achieve the

target orientation. This would be related to the work in [vdPL95] by van de Panne and Lamouret, in which external forces were used to automatically help a character balance, and were gradually removed as the character learned to balance himself.

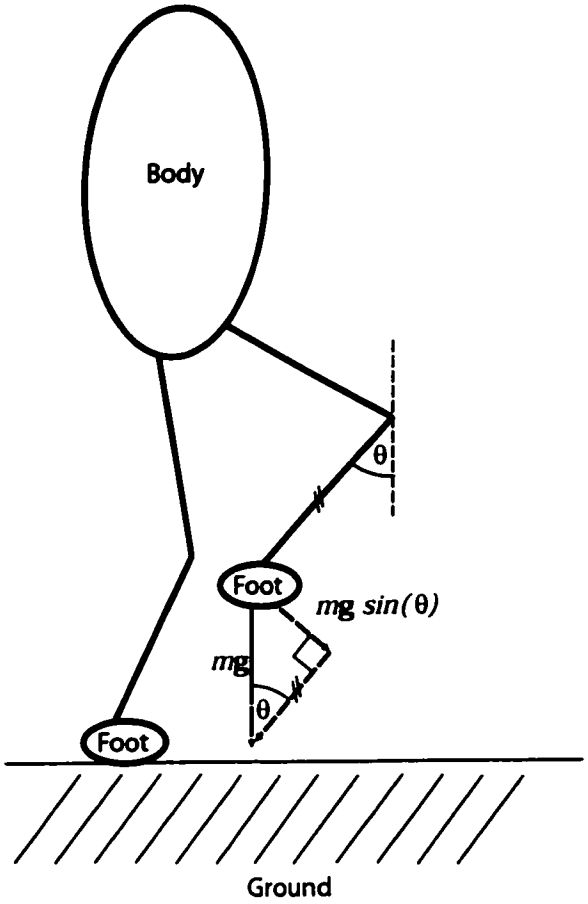

Figure 5.3: GRAVITATIONAL TORQUE AT THE KNEE

The configuration at the knee here is essentially the same for our purposes as that shown in Fig 5.2. The force due to gravity is resolved into 2 components—one parallel and one perpendicular to the shin. The latter component creates a torque of magnitude proportional to $|g|m \sin\theta \cdot r$ where r is the distance from the knee to the ankle and m is the mass of the foot.

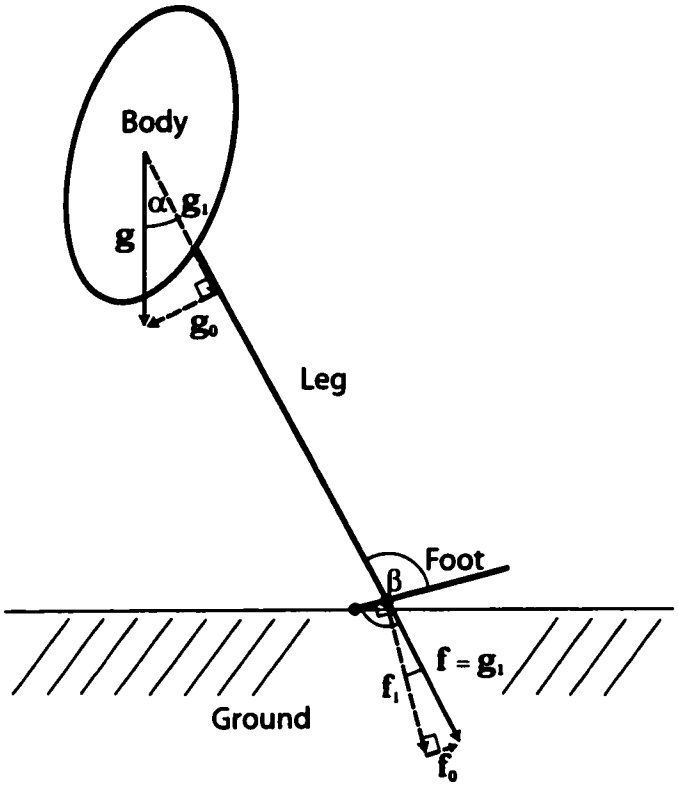

Figure 5.4: RESOLVING GRAVITATIONAL FORCES AT THE ANKLE

The leg represents the direction of the shin, below the knee. The only source of force considered in this diagram is gravity, applied on the mass of the body and acting on the centre of mass. The heel is considered a fixed hinge joint with the ground. A similar diagram can be drawn for the case when the toe is attached to the ground.

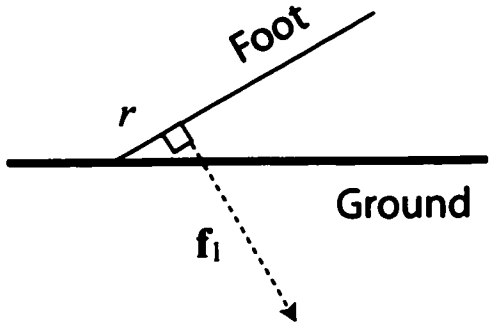

Figure 5.5: NET GRAVITATIONAL FORCE AT THE ANKLE

r is the distance from the fulcrum (where the heel touches the ground) to the ankle (the point of application of the force).

Chapter 6

User Experience and Results

A primary goal of this work has been to design a system which allows an experienced user to create an animation of good quality in a short amount of time. This goal encompasses two elements: the quality of the final animation, and the experience (and efficiency) of the user in creating it. In this chapter, we discuss how our system fares in these two domains: first, by considering animation produced by several different users (a few novices and one more experienced), and second, by describing their interaction with the system, from perspectives both of efficiency and of the learning experience. Our focus will be primarily on the case of the experienced user, as that is more representative of the design goals.

6.1 Animations Created: Demonstrations and Performances

Dock Dance A demonstration animation, "Dock Dance 1", is shown on the accompanying videotape, illustrating the character dancing alone on a dock to some music. A sequence of frames sampled from this animation is shown in Figs 6.1 – 6.6. This segment was created by a fairly experienced user, using the hip mapping of Section 4.4.1,

the WCS knee control with hard joint limits of Section 4.4.2, the ankle mapping of Section 4.4.3, the kinematic arm mapping of Section 4.5.1, and spine control as described in Section 4.6. Physics-based models were active at the knee and ankle, as described in Chapter 5. Locomotion was achieved by the foot-grounding mechanism described in Section 4.7.1. The creative process for this animation was an improvisation, in response to the chosen musical soundtrack, as shown in the video.

The Appreciator: Character Sketch In another sequence, designed by the animator in for a specific project (a character sketch for the initial stages of an upcoming art piece), the skeleton had to walk up to a painting, look at it for a while, scratching or nodding his head in appreciation, and finally clap enthusiastically. The scene would then cut to another scene showing the skeleton walking off, presumably onwards to look at the next art piece. In that sequence, of which still frames are shown in Figures 6.8 and 6.7, the most challenging aspect of the animation was creating the clapping itself. The lack of a contact model between the hands made it challenging to get just the right accelerations around the contact. As will be described in Section 7.2.1, extending the contact models is something we plan to explore in future work. Another tricky part of the clapping was getting the wrists to look sufficiently loose, in between the claps. This was feasible to achieve by recording the animation in slow motion (approximately 4-5x slower than real-time), but again, a physical model with controllable stiffness in the arms and wrists could be potentially very helpful for achieving this, and is also something we plan to explore in the future as well.

The Investigator Finally, in another test, a sample script was given to the animator, in which the character had to walk along a straight line, then stop, turn and walk at an angle to his previous direction towards a prop. Upon reaching the prop, he was to stop, bend over and wave his hand around it for a few moments, and then turn away to continue walking in his original direction. This script was achievable in a fairly straightforward

manner, and served the additional purpose of further demonstrating the suitability of the system for creating an externally-specified motion. The same mappings were used for the character's body, as outlined previously, except that this time, his root rotation was controlled by the horizontal **x**-translation of one of the trackers, as mentioned in Section 4.7. This control over the rotation of the root about its **y**-axis allowed concurrent control over the character's leg motion (and therefore translation) as well. Although not essential, it was found to be easier to synchronize turning-and-walking by having the parameters for these motions controlled simultaneously, rather than in multiple layers. In the future, we would like to have the character's orientation controlled by the action of his legs relative to the ground; this way, he could turn simply by planting one of his feet on the ground, and then rotating the corresponding hip. Whether hip rotation causes his leg to rotate, or his upper body to rotate around his leg, would therefore be determined by the kind of ground contact at that moment. For example, if he is on his heel, then his leg would rotate, but if both his toe and heel are touching, then this would cause sufficient ground reaction to allow him to rotate his body.

Live Television DIGITAL MARIONETTE has been successfully demonstrated on television (CityTV, Toronto) live to air. Within a relatively short time spot, some basic motions were shown, and a more complex animation was achieved by layering previously recorded tracks together with a live track. The whole system was brought over to the television studio and set up within a couple of hours, and the demonstration ran very smoothly. The television producer was very excited by the motion, and repeatedly commented about the "subtlety" and "naturalness" of the animation.

Theatrical Performance Additionally, DIGITAL MARIONETTE has been used in a live performance, with the skeleton character projected onto a very large movie-size screen. This was part of a live theatrical performance, and all tracks of an animation were created in front of the audience, as well as a live musical and vocal accompaniment

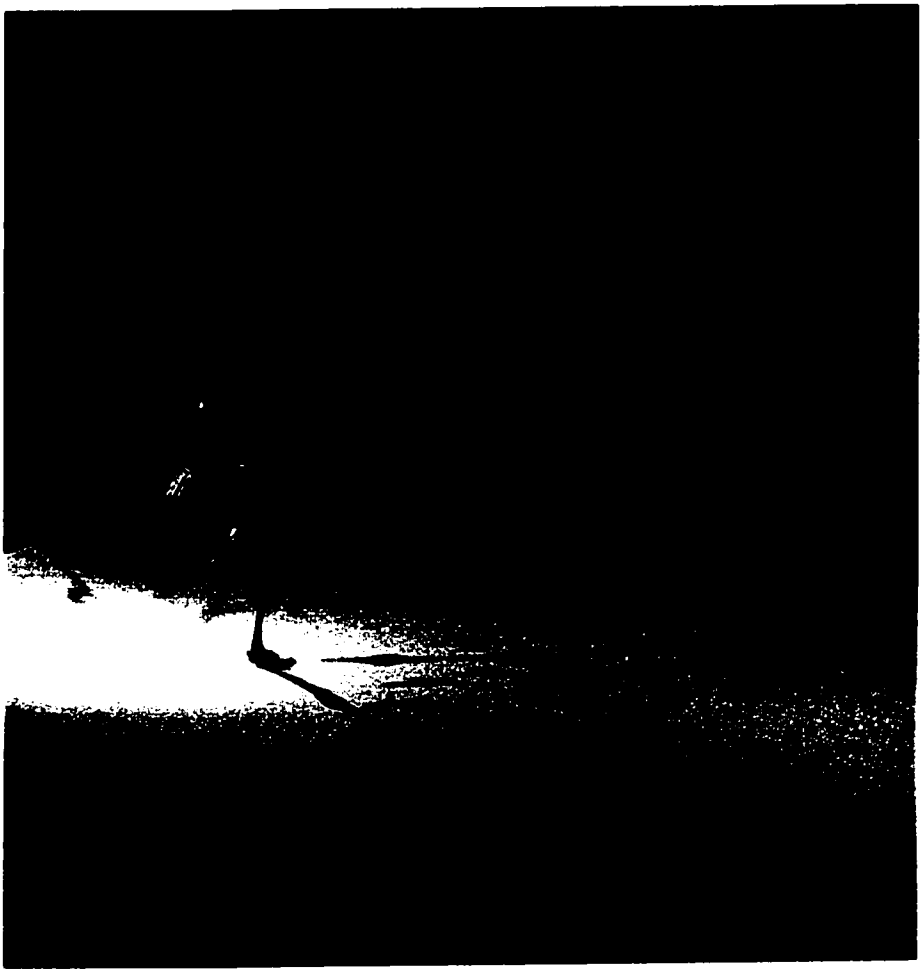

Figure 6.1: DOCK DANCE: STILL

The following sequence of images is taken from the interactively-generated animation used for the "Dock Dance 1" sequence. The frames shown here are sampled about one to two seconds apart, highlighting some of the postures achieved by the character.

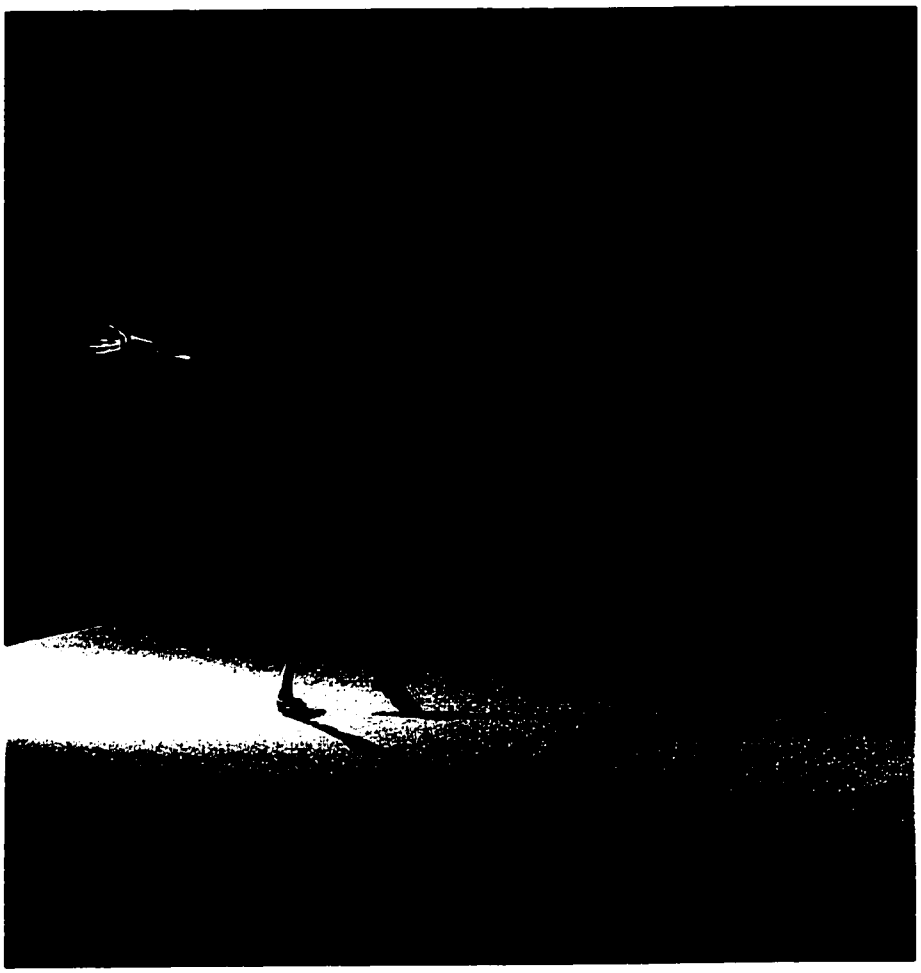

Figure 6.2: DOCK DANCE. Still frame.

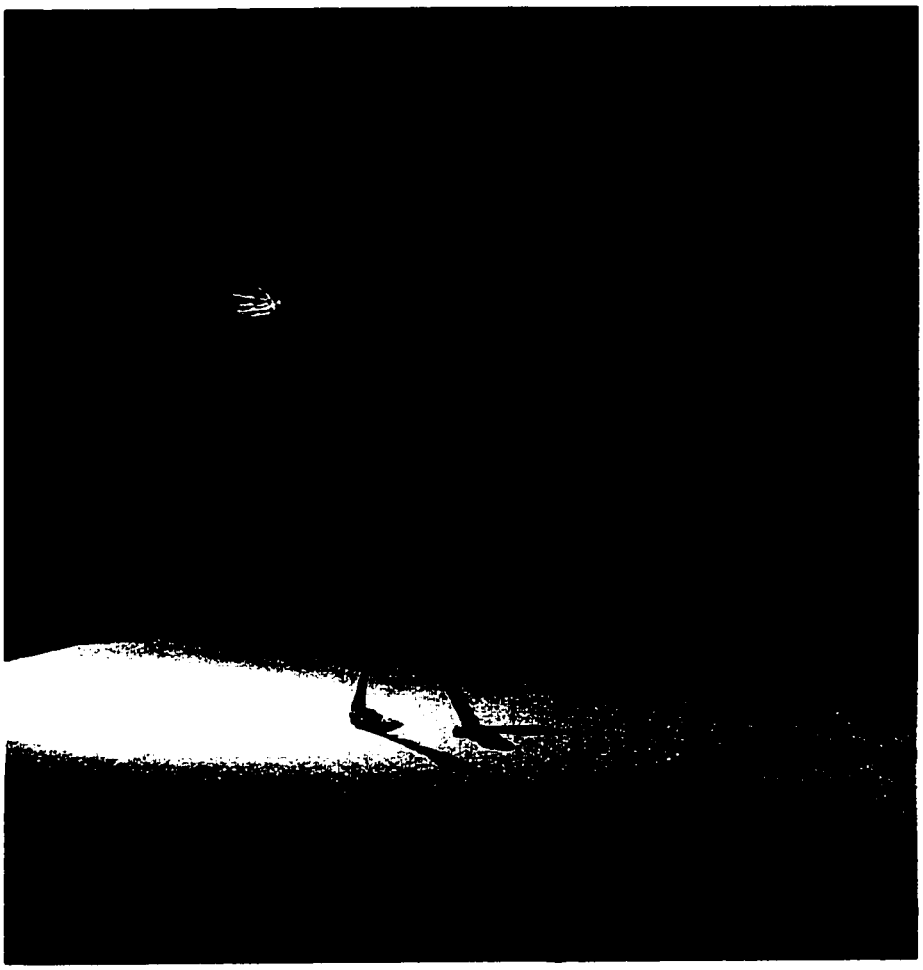

Figure 6.3: DOCK DANCE. Still frame.

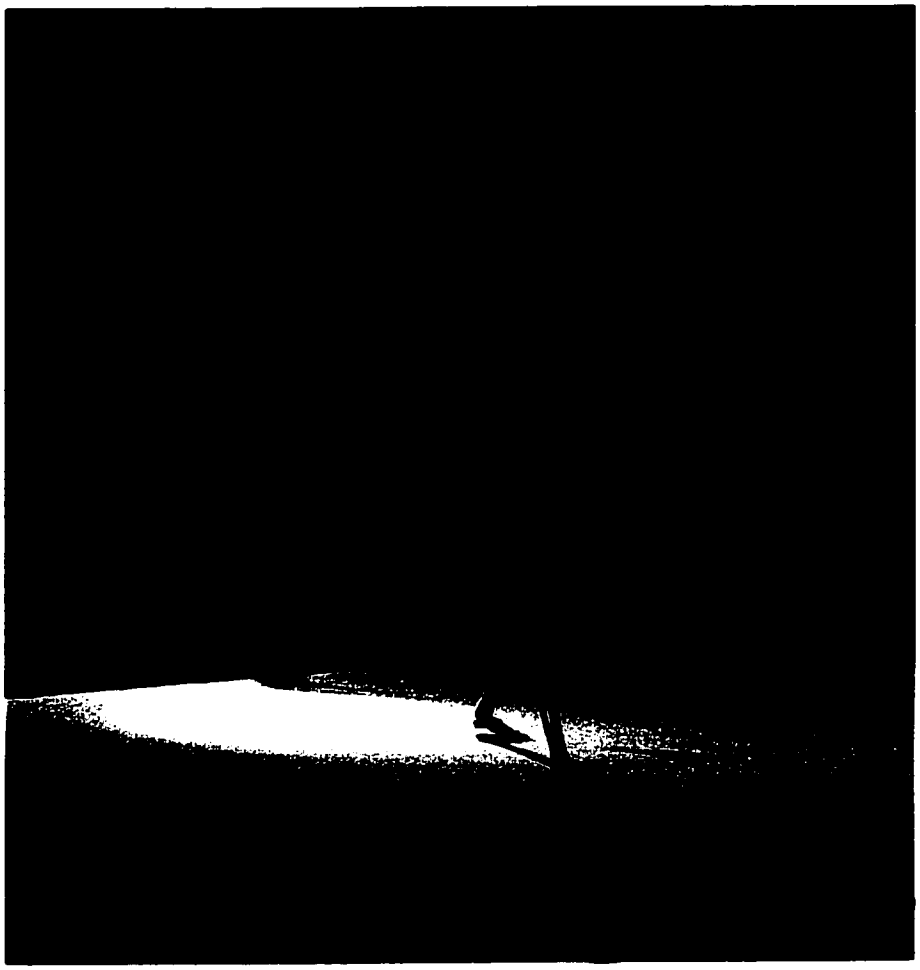

Figure 6.4: DOCK DANCE. Still frame.

Figure 6.5: DOCK DANCE. Still frame.

Figure 6.6: DOCK DANCE. Still frame.

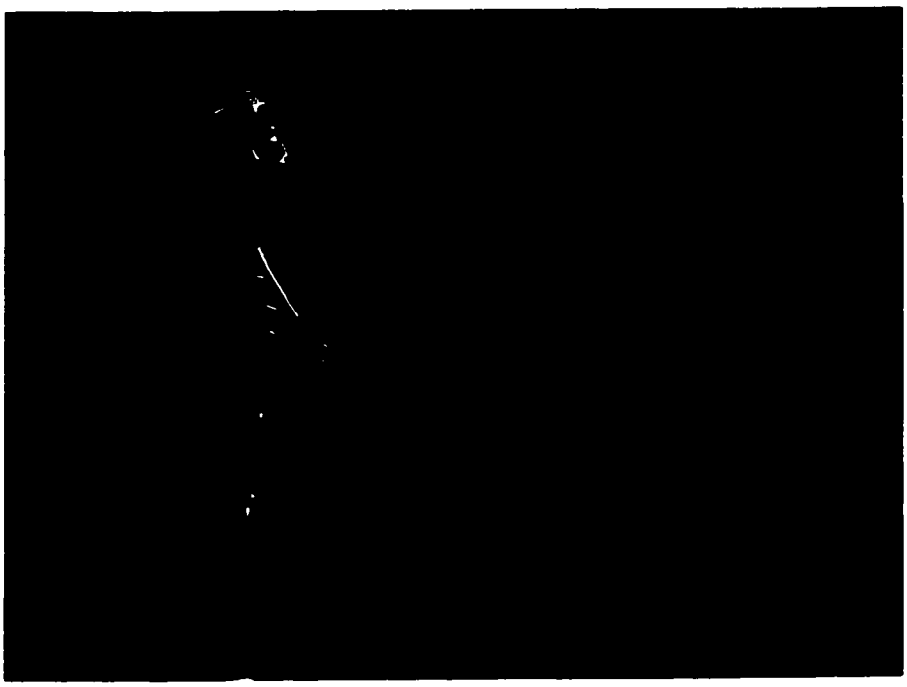

Figure 6.7: THE APPRECIATOR.

The following sequence of images is taken from the interactively-generated animation used for "The Appreciator" character sketch sequence, in which he enjoys looking at a painting, and expresses this enjoyment to the viewer. The frames shown here are sampled (irregularly) to highlight some of the postures achieved by the character.

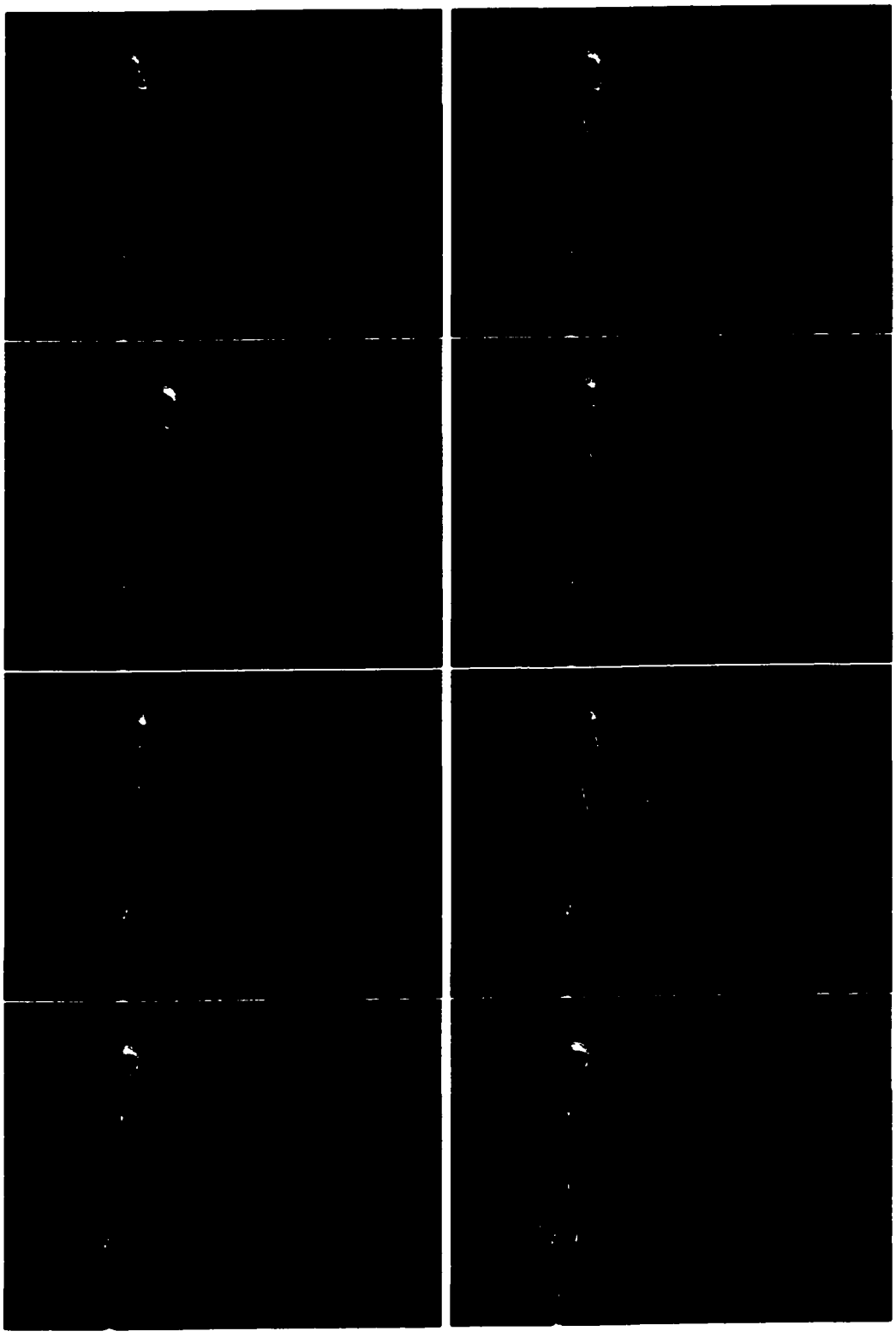

Figure 6.8: THE APPRECIATOR– A CHARACTER SKETCH.

when the fully layered animation was being played back. Projecting the character onto such a large screen involved the risk of finding and magnifying any weaknesses— perhaps subtle or otherwise unnoticeable in the screen-sized version— in his motion. However, the effect was quite strong, and audience response was very positive.

Performers' Evaluations Evaluating the quality and expressiveness of motion is inherently a subjective experience. However, among those who have seen the animation, have been several professional theatre and television performers— individuals whose skills revolve around expressiveness to a large extent— and their reactions have also been enthusiastic. One actor commented, "He's very graceful... I know you don't have facial control, but it looks like you do, when he [the character] is moving... like with a marionette." This is probably related to the expressiveness of the interactively-controlled motion.

The system was also demonstrated to a professional puppeteer. When shown just the basic walking (only the leg motion), her comment was that the motion was "beautiful". Furthermore, she immediately appreciated the fact that when his foot hits the ground there's a bit of a "bump", which she thought was very good (this results from the way the a new contact point gets bound to the ground). Interestingly, one of her first questions after watching an experienced user animating the character, was whether the foot was coming down onto the ground after heel strike "automatically", or whether the user was having to control that. She again appreciated the fact that this was indeed happening automatically (resulting from the physics-based model at the ankle).

6.1.1 Motion Constraints and Limitations

Overall, the responses to the system, and to the animation itself in particular, have been very positive. However, there are currently some limitations on the kinds of motions that can be created, and some other motions, while possible, are particularly hard to achieve.

The most significant such issues seem to be those involving contacts: both the character's contact with himself, and with objects in the world around him, such as the floor.

As mentioned in Section 4.7.1, the character's ground contact is ensured by binding the feet to the floor one at a time. This is fine when he needs just one foot on the floor, and in fact has some notable visual advantages, as described previously. It is less convenient for standing, however, since the second foot must be made to *appear* fixed to the floor by the user. This is possible to accomplish without oscillating between the feet because the actual switch from one foot to another only occurs when the foot passes a small threshold below the floor. Nevertheless, it seems that a better solution would be desirable in the future, in which the appropriate constraint would activate automatically. More importantly, the opposite case, jumping with both feet off the floor, is currently not possible. Dealing with the ground adequately is a difficult problem involving a trade-off: On one hand, the ground constraint limits the potential motions of the character, but on the other hand, it is this same constraint which automates certain accelerations that would are difficult to control otherwise (and which elicited positive feedback from the puppeteer). A possible future approach to handling this problem is suggested in Section 7.2.1.

Currently, no detection is made for self-contact. In early systems, when using a full dynamic simulation, some contacts were computed. Checking for all possible contacts slowed the system down considerably. Currently, contacts can be handled by the user, simply by simulating the appearance of a collision, but this is difficult to control accurately. This will be discussed in Section 7.2.1 as well.

6.2 Efficiency of DIGITAL MARIONETTE

The preceding section focused on DIGITAL MARIONETTE's output, i.e. the animation itself. Now, as mentioned at the beginning of the chapter, we will discuss the experience of

using the system, starting with a consideration of DIGITAL MARIONETTE's *efficiency* in creating animation. This was a principal design objective, and by nature of the continuous interactive interface, this objective is successfully fulfilled, with DIGITAL MARIONETTE indeed being very efficient. The animation clip "Dock Dance 1" discussed above is approximately 60 seconds long, and took a total of under 10 minutes to generate. For an approximate breakdown of this time, see Table 1.1 on page 4 in Chapter 1.

Generally, the time taken to create an animation can be approximately estimated by the sum

$$T \approx \sum_{i=1}^{n} k_i T_i + \sum_{j=1}^{m} S_j \tag{6.1}$$

where

T_i = the time it takes to create track i,

n = number of tracks to be recorded in separate passes,

k_i = the number of separate takes required for track i,

S_j = the duration of transition j (e.g. rewinding, switching the active track, etc),

m = the number of transitions,

T = total time to create the animation

We now elaborate on these factors, and estimate upper and lower bounds on T.

Number of Tracks In our case, although we have defined 5 tracks, $3 \leq n \leq 5$, and is typically 4, since we control the shoulder and elbow tracks simultaneously, but do the wrist separately.

Number of Takes k_i, the number of takes required, is perhaps the most uncertain factor. It is dependent on the difficulty of the desired motion (discussed in the sections below); on the complexity of its interactions with other tracks given the multi-tracking

framework; and on the available time. For all live performances, for example, there is only time for a single take. Similarly, for video Scene 1, only one take was needed per track. On the other hand, for reasons described below, video Scene 2 required 1 take for the leg mapping, but 2-5 takes for the other tracks. k_i could also depend on the desired accuracy of the motion. Due to its intrinsic efficiency, a possible future application of such a system might be to quickly put together rough "drafts" of animated sequences, in which case some inaccuracies are tolerable, as long as the result is achieved very efficiently. In that case, k could be quite small, meaning very few retakes. Establishing a general bound on k is comparably hard to assessing how many drafts need to be written of an arbitrary text: it depends greatly on both the content and the context (and the writer). In our experience so far, however, one take is often enough, and up to 5 or 6 have sometimes been necessary, though not for all tracks.

Time per Track The time taken to create a single track is usually 0.5 to 3 times the duration of the animation for that track itself, and most often the same duration. The reason for this factor, as will be explained below, is that one occasional technique is recording a motion at some fraction of the final speed, usually no more than 2 or 3 times slower. Infrequently, one might also record at a higher speed.

Transition Time The number of transitions m in Eq (6.1) is the sum of all takes over all tracks, $\sum_{i=1}^{n} k_i$. In our current system, a single transition time ranges from approximately 5 to 30 seconds. This time includes the release of one or both input devices, the acquisition of the mouse, selection of the desired command (e.g. start recording, rewind, etc), and reacquisition of the input device. For an animation segment of 1 minute in length, this can add up to be a non-negligible fraction of the time, and possibly also distracts the user from his or her primary focus. The only optimization that has been implemented in this regard is the ability to synchronize tracks so that a single "play" or "record" command can activate all relevant tracks, a feature which is quite necessary for

a single user to be able to operate the system. However, this aspect of the interface is outside the scope of our main research focus, and could be improved upon, perhaps in future work, by the use of a speech recognition system, for example.

Upper and Lower Bounds Based on the numbers given above, assuming 5 takes for every track, 5 tracks, a slow-down factor of 3x for each track, etc., the upper bound for our system can be approximated to be (75 T_a + 12.5) minutes, where T_a is the desired length of the animated sequence. So for a minute of animation, this adds up to just under 90 minutes. The lower bound (assuming no slowdown factor, but no speed-up factor either) is (3 T_a + 1.5) minutes, which works out to be 4.5 minutes for 1 minute of animation. In practise, a minute of animation such as that of "Dock Dance" took 10 minutes.

6.3 The Interactive User Experience and Learning Process

In this section we describe the interactive experience and learning process associated with the DIGITAL MARIONETTE animation system. Our discussion is based primarily on the observations of, and by, an experienced subject S—[1] (who was also the designer of the current system) as well those of some novice users. We note that S— is the same subject who learned to use GLOVETALKII [Fel94, FH95], and is also a musician, thus having prior familiarity learning complex, continuous interfaces. (In this light, some of the observations given below are not only specific to this project, but have relevance to learning such interfaces in general).

We also remind the reader that this system is designed for more experienced users; the goal is not to have an interface which makes each user an "instant animator", but rather

[1] *S*— for *Subject*

a system which allows trained animators to create nearly instant animations. We thus begin this discussion of the user experience with some preliminary comments regarding the learning curve of experts.

6.3.1 Expert Learning Process

A principal difference between systems intended for novice users versus those intended for expert users is that, in the latter case, simplicity of use has been traded for a potentially much greater range, resolution, or bandwidth of control. Even for early interactive keyframe interfaces, it was found that some which took longer to learn ultimately proved to be more efficient (e.g. [HS85] versus [Ste83]). Typical learning curves of novice-oriented systems consist of quick progress at the beginning, by definition, followed by a permanent plateau, as there is probably little else to learn (otherwise, it would be an expert-oriented system). Conversely, expert-oriented systems may have any of a variety of possible learning curves, where even quick progress at the beginning is not guaranteed.

The depth and range of possible output in systems with expert-oriented interfaces imply a great variety of choices over aspects to "focus on" during learning. These choices, in turn, are driven at least in part by the user's goals and learning style, as well as any deconstruction of the task itself, making it hard to separate an inherent learning curve of the interface from a user's experience with the system. An example of such a deconstruction is the common practise of musical scales independently of the instrument being learned. At the same time, the uniqueness of the learning process is very obvious with music as well, where the individuality of the "users" can manifest itself in their learning styles as much as in their performance.

In GLOVETALKII [Fel94, FH95], the description of S—'s learning was subdivided into eight phases. Not including system initialization, there were six phases. Of these, two can be seen as general tasks or structured exercises: passage reading, and unrehearsed speech. This leaves four remaining learning stages, corresponding to: (1) phoneme production

within simple words, (2) words and interword pitch control, (3) short segment formation with suprasegmental pitch control and singing, and (4) fine tuning (movement control and phrasing). The first three of these stages (i.e. phonemes, words, phrase segments) reflect a time-based hierarchical breakdown of the speech task itself. By providing control over phoneme production, GLOVETALKII preserved this same hierarchy, and S— allowed his learning process to be guided by this structure. The fourth phase (fine tuning) could be considered part of the user's learning style. Furthermore, it was pointed out that these phases did not occur linearly, but rather with substantial overlap. This, too, can also be explained as being part of the user's learning style (without implying that this phenomenon is unique to this user).

In addition to breaking the GLOVETALKII task down into more elementary components to be practised, S—'s approach to learning both DIGITAL MARIONETTE and GLOVETALKII (influenced to some extent by his experience as a musician) was to have a particular "piece" to be working on, from which specific issues arise, and hence for which custom-designed exercises would be developed. In controlling character animation, there is no clear candidate for a general time-based breakdown corresponding to the phoneme-word-phrase structure inherent in speech, so the learning process was based more on such custom-designed exercises. In the next sections, we discuss these exercises, and distinguish between three different aspects of the learning experience: basic control tasks, refinement of complex motions, and general principles. As we shall see, one of these general principles will also highlight the importance of the overlap between the different learning stages described above.

6.3.2 Basic Control Tasks at the Initial Stage

The main motion chosen to be learned was walking. The reasons for this choice were that walking is a motion which involves the whole body properly synchronized, it is sufficiently complicated to be challenging, and it is also familiar enough that a typical viewer can

have some intuition about, or sensitivity to, the animation quality. However, achieving a "nice-looking" walk right away was not possible, so some initial simplifications and sub-tasks were necessary. This led to two main types of initial exercises:

1. Isolated motions

2. Functional bipedal locomotion

Isolated Motions Isolation exercises are extremely valuable, almost universally across disciplines: they provide the individual with a means of understanding, and hopefully internalizing, the relationship between input and output. One type of isolation exercise involves focusing on one output variable, and observing and learning the specific set of input variables that need to work in synergy to achieve control of this output variable while maintaining other outputs constrained. Another type of isolation is constraining many of the input variables, and observing the effect on the output as only one primary input variable is changed. In music, this can mean using the various muscles in the arm and hand in synergy to play one note at a time, or focusing on the responsiveness of the instrument to the different variations in single keypresses; in GLOVETALKII, this corresponded to the practise of single-phonemes. In our animation system, isolation exercises involved keeping the character in some steady state while practising rotating from just one joint, acquiring an understanding of the effect of doing so, and the corresponding kinesthetic experience of the user to achieve this. Two important examples of isolation exercises consisted of:

- standing still and swinging one leg back and forth from the hip

- standing still, with one leg up, swinging the shin back and forth from the knee

The first one was important because a significant problem that novice users would have with the system is that the free leg would easily get stuck in the ground. Although

simple, these exercises were very important for developing the user's familiarity with the system, and a more intuitive understanding of the control model.

Another isolated motion exercise was the act of shifting from one leg to the other, without necessarily moving forwards or backwards. This further accustomed the user to the way in which the feet get bound to the ground. In one variation, the feet could be side-by-side (as in a typical standing posture), while in another, one leg could be in front and one behind, leading to a back-and-forth rocking motion.

Generally, these exercises were useful for the user to get a feeling for the kinematic interaction between joint motions[2].

Functional Bipedal Locomotion By "functional locomotion", we mean that initially, the forward motion did not have to be graceful or even resemble conventional motion. The rocking and free leg-swinging exercises described above were the precursors to the simplified bipedal locomotion; by combining these two motions, the character could be propelled forward. Having isolated the concepts of switching feet and of swinging a leg, this now allowed the user to focus on getting the proper synchronization between. Again, part of the goal of these exercises was to overcome the problem of feet sticking unintentionally to the ground. One such way of moving forward was by "waddling", wherein the knees were kept straight, ankle motion was ignored, and the only focus was to swing the free leg so that it moved out and around, landing in front of the supporting leg. The reason to move it out and around (rather than just straight through) was to ensure that the foot would clear the ground level without getting stuck.

[2]Isolation tasks were also invaluable in the design stages of the system, to assess the responsiveness of the system to various input control signals. The shin-swinging exercise, for example, was precisely what motivated the use of local physics-based models.

6.3.3 Refining Complex Motion: Walking

The exercise above for functional— but not necessarily proper— locomotion was next refined in various ways to become a walking motion. Note that once a basic bipedal locomotion was "internalized" from the user's perspective, then some of the corrections and refinements were perceived as isolated details layered on top of a nearly automatic motion. The refinement tasks are described below.

Swinging the Leg Through The first major correction was to learn to swing the leg through, parallel to the direction of motion, rather than swinging it around. This meant getting the right bend at the knee during the swing, and then straightening the knee just before stepping down.

"Falling Forward" and Swinging Just Enough Once the leg could be swung through, then the user practised controlling the step size. Some practise was necessary for the user to be able to "feel" how to do this, finding the right mix of the large swing of the free leg, with the slight rotation of the support leg around the ankle (that is the falling forward).

COM Velocity Control As mentioned in Section 3.4, one of the visual feedback signals for the user was a projection of the character's centre of mass (COM) on the ground. Once S— had gained some comfort in controlling the character in a more intuitive way, then he experimented with maintaining some awareness— and therefore control— over the velocity of the character's COM. Thus, the overall pacing could be refined, for example, by trying to keep the COM going at a more or less constant velocity while walking.

Working with the Viscous Filter Recall that in Section 3.2 the input filtering process is described, whereby a tunable parameter α controls the viscosity effect of the input

filter. This introduces a certain non-linearlity to the motion control, as well as a potentially noticeable lag. This does not present a problem to the user, but it is certainly another aspect of control that can be potential refined. A related type of issue in music might be the loudness decay of a note on the piano, and the practise of listening sufficiently carefully to relate the loudness of the next note with that of the previous note to get a good phrasing. An even closer example is learning to master the use of a variety of available digital processors (e.g. delay, reverb) during real-time performance. As will be discussed in the next section on general principles, a very helpful practise approach is to work very slowly, so that the filter still maintains smoothness in the motion but the lag has a much smaller relative effect.

Soft Knees A quality common to some styles of dance is having "soft knees", meaning a certain, sometimes subtle, motion in the knees, in synchrony with the rest of the dance. This adds an important element of fluidity to the overall motion, and there was some effort by the user to apply this concept to the animated character's motion, although it was still at an early stage of refinement.

6.3.4 General Principles for Practising

Throughout S—'s experience with DIGITAL MARIONETTE, a number of general techniques and practise principles emerged. These principles are neither inherently "basic" nor "advanced", but are ideas suitable for guiding exploration of various motions and at various levels of expertise. S— found them very helpful for continuing to make progress and discovering new subtleties or ways of using the system even when it felt as if a plateau of some sort had been reached. Some of them have a broader scope than this system, extending to general principles for learning such complex tasks. In fact, some are based on cumulative learning experiences in addition to the DIGITAL MARIONETTE system, including both GLOVETALKII and musical performance. This section effectively

provides an experienced user's perspective on learning complex, continuous interfaces, with emphasis on DIGITAL MARIONETTE as a representative example.

Slow Practise This is one of the principles which can be applied from the very first uses of the system, and doing so is highly advisable. Almost every accomplished musician is well aware of the profound importance of extremely slow, steady practise; and this applies equally strongly to real-time animation control using DIGITAL MARIONETTE. In both DIGITAL MARIONETTE as well as motor control tasks such as playing an instrument, we observe that a trademark of novices has been that they will do the easy maneuvers fast, and then necessarily slow down and stumble through the more difficult parts. One of the purposes of slow practise is to achieve *steady* and *accurate* performance, and the more difficult the user finds the sequence to be, the slower he must go in order to achieve this. Another benefit is that it allows the performer to continually relax while performing the motions, so that when they are sped up, the user can still be relaxed, which is again invaluable for steadiness and accuracy, as well as the freedom to adapt the control spontaneously. Slow practise may range from half of the target tempo, to eight or more times times slower! When it was suggested to one of the novice users to slow down, he complied, but still maintained what appeared to be a "reasonable" tempo. When S— demonstrated a *very* slow control, then the user tried it and almost immediately exclaimed, "Now you can really *feel* that you have control over it!" Going slow also lets the user explore the more subtle variations available in controlling the input.

Exaggerate/Reduce Another very helpful exercise for discovering DIGITAL MARIONETTE's responsiveness to the input parameters, and thus being able to achieve greater fluidity, is to maximally exaggerate the character's motion, and also reduce it to a minimally recognizable form. An example of exaggeration is swinging the leg very high up and around during a waddling motion, while the associated reduction is bringing the leg as close to the ground, making the smallest possible step with minimal movement. In

another form of the exercise, the user can focus on exaggerating and reducing his or her *own* motions to achieve a relatively stable output motion. This teaches the user how his or her whole body is involved in the control. For example, in moving one's hands around, it is easy to overlook the shoulder motion that is also occurring (or not occurring). Becoming aware of this can be quite helpful in determining how and where it can be minimized or used. In GLOVETALKII, this notion of reducing motion was alluded to in the fine tuning stage, in which S— tried minimizing his hand motion through vowel space. The goal of that exercise was speed, which is in fact another advantage of being able to minimize the input motion, although the exploration of this is best initially done slowly and steadily. Sometimes, even just one session of slow, reduced-motion practise, can have an immediate and noticeable effect on the user's degree of control.

Dealing With "Mistakes" A common challenge with continuous real-time interactions— music, GLOVETALKII and DIGITAL MARIONETTE included— is dealing with mistakes. Why is this challenging? Watching beginning users pick up DIGITAL MARIONETTE, whether for five minutes or longer, we observed that if the character got out of control, they would have a tendency to lose even more control when attempting to immediately correct it, or bring it to a neutral position (such as standing up straight in the initial position). The same thing happens in many other situations: especially in the case of novice users, mistakes are often accompanied by a frantic, urgent reflex to correct them immediately, which usually does not help.

What are the alternatives? This depends on the context, which is either a performance or a practise situation. In performance, the best thing the user can do is to keep going regardless of mistakes. This is not as easy as it sounds: most beginners (either in music, or in using this system) cannot always continue smoothly after making a mistake. In music it is a valuable skill to be able to keep going without losing a beat. Often, making a mistake will distract the user, making them lose their focus and get disoriented. Similarly,

it was only after a certain amount of practise that the user of this system was able to keep going steadily regardless. This ability was also quite important in making GLOVETALKII intelligible: if the user made a mistake, then trying to correct it would usually lose the listener completely, because the listener *could not know* that the user was trying to repeat something until he got it right; instead, the listener would try to parse the repeated stutter under the assumption that it was a single word or coherent phrase. By letting go of the mistake and continuing *steadily*, without missing a beat, the listener would stand a better chance of parsing the mistaken word correctly (i.e. guess the user's intention) by making use of the context provided in the subsequent speech.

In a practise situation, what the user can do is to stop on every mistake and get it right. This is different from going back to the starting position; S— found it to be very helpful to "freeze" as soon as the mistake was recognized, possibly retracing his motion only very slightly, and very slowly trying to pinpoint exactly what happened. If one goes all the way back and tries to do the same thing over again, which is a very common response, usually the same mistake is made again, too. In some cases, especially dynamically evolving systems, going back only slightly may not be enough or even possible, in which case going back nearer to the beginning is probably necessary.

Playing with Speed An interesting area of exploration lay in the fact that motions can be played back at faster or slower speeds than at which they were recorded. Some difficult motions are easier to achieve by recording at a slower speed. On the other hand, one of the challenges of recording at half-time, for example, is being consistent about making everything go slower, rather than just the tricky bits. Playing things at different speeds can also sometimes be helpful in highlighting little idiosyncracies in the motion.

Playing with a Music Track Closely related to playing with speed is the idea of using a soundtrack. This can be a prerecorded musical piece, or a piece specially written for the animation, or even just the click of a metronome. This can be helpful for multi-

track animation, where the music can provide an additional source of cues as to what specifically is happening next, and a very precise marker for when it will happen, since a piece with a steady rhythm provides a continuous stream of highly predictable phrasing information[3]. One implementation issue that could make this even more convenient would be to have the animation software also control the start and stopping of the music or sound track.

Revisit All Exercises Finally, an integral principle for learning and practising complex tasks is occasionally returning to previous exercises, regardless of whether they were simple or difficult. When a user becomes more advanced, then returning to a very basic exercise can sometimes reveal what appears to be a brand new exercise, by being able to focus on new aspects of it or due to higher, revised standards. He or she may also be able to absorb it in a different way, or have a new kinesthetic perception of it. In complex interfaces, there is often room for more refinement on a previous exercise even if the user is already on the next one, and S—'s experience with both DIGITAL MARIONETTE and GLOVETALKII reflected this.

6.3.5 Novice User Experience

Procedure

Three novice users have practised the system for about two hours each, over two or three separate sessions of 30-60 minutes in length. The primary goal of these sessions was to see how much control the users could acquire over the system within this timeframe, with the secondary goal being to achieve some basic form of locomotion. Due to the complex nature of the interface and control task, it was considered inefficient to simply give the subjects a single initial set of instructions and then have them spend the rest of the time

[3]That is, in this regard the rhythmic structure can be seen as having someone counting out loud in the background very steadily, "seven, eight, nine, ...", in which case the user can predict quite accurately just when the "ten" will arrive.

exploring without any guidance. Instead, the sessions developed more in the spirit of a "music lesson"— another example of an interface intended for long-term expert users— involving natural interaction between the novice and the experienced user. Thus the format of these sessions consisted of a combination of:

1. direct practise by the novice user (i.e. in manipulating the CG character), and

2. short breaks of 5-15 minutes each, as comfortable for the users, during which time they discussed their experience, or S— would suggest exercises for them to try out next.

S— would sometimes give the users some guidance while they were manipulating the system. Each user was informed at the beginning of his or her session that the goal was to observe their experience in these initial stages of learning to use the system, and that if a basic familiarity with it were acquired, then the next step would be to explore a task such as walking.

Results

All three users were able to achieve some form of basic bipedal locomotion within the first 20-30 minutes of practise, although after this brief practise time it was still very inconsistent, and did not yet look like a realistic walk, of course. By the end of the two hour practise period, all three test subjects had accomplished at least a couple of short walking sequences which, although not as refined as those achieved by the more experienced user, nevertheless were clearly beyond the initial stages of "waddling". In fact, some of these walking sequences even showed early signs of refinement such as the leg swinging through with some smoothness.

A difficulty at this point, however, was achieving consistency, and one subject even discussed the feeling of "getting excited that it's working, and then making a mistake right afterwards". We believe this is a phenomenon common to many motor skills tasks,

and which can be overcome with practise and experience.

Although two hours is not really enough to focus on the refinement exercises, the little time that was spent on them was useful. In the practise sessions, the users were eager to try walking as soon as possible. After having attempted to do that, they were more appreciative of the isolation and refinement exercises, and were quite pleased to be able to alternate between these exercises and the task of walking, and thus apply the newly acquired skills. This again suggests a certain coherence of control available in the system, wherein even novice users can focus on particular facets of control, and then apply them to improve on more complex tasks.

Observations

Some points and observations noted during these practise sessions include the following:

- Transitions of switching feet were a bit difficult, in that they required the users to be particularly careful with the hand motions during that time to avoid getting the first foot "stuck" back to the ground. One user observed that this led to a tendency to exaggerate some of the leg motion to ensure that the feet did clear the ground, while another said that, "smaller steps are harder, because that means the foot is closer to the ground, so you need more control"[4]. One subject said that he would have liked some more feedback indicating which foot was currently bound to the ground, and possibly even indicating the moment of contact.

- One user found the most enjoyable layer of control to be the spine, neck and head joints, as that was both intuitive to control, and added a lot of expression to the character.

- A common difficulty for all three users was the lack of depth cue provided by the

[4]These observations correspond exactly to the previous discussion of S—'s experience (described on page 137), where the same kind of observation led to an exercise of intentionally exaggerating and minimizing the motions.

system. For this reason, the foot-colouring was crucial in helping the users know which foot was in front and which one behind.

- One of the users had an interesting jump in progress after S— demonstrated a *much* slower control speed to this user, who then tried going at that slow speed (as described previously on page 137). Besides the output looking very different, the interaction itself felt like a completely different experience.

- One user wanted a physical artifact on the input device indicating the neutral orientation in the "roll" direction. Although the other two subjects did not request this, achieving the right roll was indeed observed to be a source of confusion due to the radial symmetry of the cylindrical inputs.

- A common topic of discussion for all three users was some aspect of acquiring intuition over the control. One user described his conscious attempt to try to "feel" his way through the control. Two of the users referred to moments at which they had "found a rhythm" for controlling the walking motion, at which point the control seemed easier or more fluent, one subject saying "once you get it, it feels intuitive, once you get the rhythm in your body". One user suggested looking for ways which would help the user find such a rhythm more easily, and this will be discussed in Section 7.2.3.

  Some aspects of the mapping, such as control over the head and spine, were generally felt to be more intuitive than, for example, the leg control. There was indication that this can be learned, however, as one user said "it feels so much smoother" as soon as she started her second session using the system. In particular, the isolation exercises seemed to be good for helping the users to internalize the control of certain motions.

- One of the users said that when the basic walk was happening, the control "felt really satisfying".

Thus, we find these overall results to be quite promising, in that there appeared to be clear signs of potentially refined control, even after only two hours of practise, and there was a noticeable learning curve during that time as well, with various subtle aspects getting better. Generally, it was found that the longer the subject practised, the more consistent their performance (e.g. they could animate a longer sequence of steps before making a mistake).

6.4 Summary

In this chapter we discussed performance with the DIGITAL MARIONETTE system from several perspectives.

We began by describing some of the live performance contexts in which the system has been used, and the positive audience responses in these cases.

A discussion outlined the major factors determining the efficiency of creating an animation with the system, leading to an approximate estimate for the expected time to create an animation in terms of the length of the sequence being created. Our practical experience has indeed fallen within those bounds, generally on the faster side.

We then described the learning experience from the perspective of a more experienced user, distinguishing between three aspects of learning: basic initial tasks, refinements, and general principles. The user's experience indicates that in addition to the initial learning curve, there is indeed a potential for improvement over a longer period of time, and for more refined animation control. That is, the system appears to have the depth to allow for genuine "expert" users.

The experience of three novice users indicates that although the system is designed for expert users, the initial learning curve of the system is such that after a couple of hours of practise, a user is able to get some basic, albeit unrefined, motion control over the character.

The demonstration video shows an example of motion which, when created, gave the experienced user S— a sense that the system truly allows for expressive animation.

Chapter 7

Conclusion and Future Work

7.1 Summary

We began this work by placing computer puppetry in a context relative to other animation techniques and showing how its real-time low-level interactivity provides the user with a fundamentally different control experience, both in efficiency and in quality (Chapter 2). Specifically, keyframe animation was seen to be laborious, and dynamic animation unintuitive to control (Sections 2.1, 2.4.1). To control animation without feedback is– quite literally– choreographing and performing a dance in the dark. Motion capture can work well, but only if the motion to be animated can be accurately recorded without feedback, which only really works for certain human animations (Section 2.2). It was thus argued that computer puppetry transcends the limitations of other animation techniques by allowing the animator to create motion in time and react to the created motion immediately, thus putting both the creation and the correction processes in the dimension where they occur.

We then presented a detailed exposition of the DIGITAL MARIONETTE system, beginning with the input acquisition and device interface (Chapter 3). The input signal and filter were described, noting the effects of different filter viscosity values, and their

applicability to different joint parameters. The hardware interface was discussed next, focusing on the kinesthetic, visual and tactile feedback provided passively by the input device design. This design facilitated a more intuitive understanding of some aspects of the mappings as well. Additional active feedback techniques were mentioned, and the multi-tracking interface was described, which allows the user to record and superimpose layers of motion, and in this way handle the large number of degrees of freedom of the CG character. The concepts of literal and non-literal mappings were defined and the spectrum they span was discussed.

A detailed discussion was given of the kinematic mappings which transform the input signal into the CG character's joint parameter values (Chapter 4). This included discussion of issues such as joint limits, and control relative to the LCS of a body part versus the WCS. An explanation was given for the technique by which the character's feet get bound to the ground, and therefore by which it is possible for him to walk forward by moving his legs.

Local physics-based models were introduced and applied at the knee and ankle joints to enhance the kinematic mappings (Chapter 5). This provided the user with some of the advantages of a physics-based simulation, yet without entailing the difficult challenge of requiring the user to dynamically balance a character interactively.

Having described the various components of the DIGITAL MARIONETTE performance animation system, we gave an account of users' experiences and results with the system. This discussion began by presenting animation created using DIGITAL MARIONETTE by a more experienced user, as well as a description of some live performance contexts in which our system has been successfully used. The main factors determining the efficiency in using the system were outlined, leading to a set of approximate bounds on the ratios between the duration of the final target animation and that of the corresponding creation process. Finally, an extensive account of the experience of learning and using the DIGITAL MARIONETTE system was given. As the system is designed for more advanced users, the

focus of this section was on the perspective of a user (the author) who has experience both with this system, as well as with learning other complex continuous interfaces. Thus, in addition to detailed discussion of aspects specific to DIGITAL MARIONETTE, this learning process was contextualized relative to that of other complex interfaces, and some common principles were exposed. Also included in the discussion were the results of three novice test users who spent approximately two hours each, working with our system, and were able to achieve the basics of a walk within that time. Together, these results suggested that the DIGITAL MARIONETTE interface offers "depth" of control in that continual exploration can allow continual refinement of the animation and its creation process.

7.2 Future Work

Our system raises a wide variety of new questions and provides a foundation from which numerous future directions can be explored in relation to the real-time creation of animation. Some of these main directions will include: further use of physics-based models for real-time animation, the application of learning algorithms to control more degrees of freedom simultaneously, the issue of interactive motion editing, and the use of a broader range of both input and output modalities. We briefly discuss some of these topics below.

7.2.1 Contact and Physics-Based Models For Real-time Animation

One of the key sources of difficulty in using our system, as indicated in Chapter 6, lies with the relationship of the character to the ground. Since exactly one point is always fixed to the ground, having the character appear to stand on both feet at once actually requires more practice than should be necessary, and the double stance phase of walking is not entirely satisfactory. By the same token, the character cannot yet be made to lift

both feet off the ground at the same time, without sacrificing this otherwise crucially helpful constraint. Furthermore, in the early stages of walking, the novice user needs to pay particular attention to manoevering the character in such a way that the free leg does not "accidentally" hit the ground as it swings through, causing it to become bound to the ground at that point.

All of these issues would best be handled if we could overcome the constraint of having exactly one foot touch at a time. The nature of this constraint arises from physical principles— that is, friction, gravity, location of centre of mass— so ideally it is by some application of these principles that we would also like to solve it. The problem with just using a full physical simulation is that although it will ensure realistic ground contact, the price of that might be an uncontrollably difficult system from the perspective of a real-time user, as well as presently being too computationally intensive for real-time interaction.

Nevertheless, there are a number of initial ideas to investigate that could incorporate physical principles further, building on the approach taken in Chapter 5. For example, we hope to overcome the constraint of requiring at least one foot to be on the ground, and thus allow motions such as jumping, by keeping track of the velocity of the character's centre of mass. If the COM's vertical component is sufficiently large, then perhaps both feet could automatically be released from the ground, allowing the character to follow a projectile motion until landing back on the ground. Similarly, consideration of the location of the centre of mass relative to the feet (assuming two-point support, say, at the heel and toes) might be useable in order to give the user a control that allows fluid transition between bipedal and unipedal contact. Again, care must be taken here to provide the user with a more flexible control mechanism, yet without requiring the user to engage in the very difficult task of balancing a dynamic character.

The character's relationship to the ground is a particular example of the more general issue of handling contacts, both those of the character with himself, as well as with

external objects. Here again, a physics-based friction and collision model, taking into account factors such as impact velocity, could be valuable in computing directions for sliding, sticking or bouncing, for example.

Ultimately, we would like to make use of as realistic a physical simulation as possible, without compromising the animator's freedom, flexibility, and granularity of control. Some approaches to extending the application of dynamic simulation in the control were discussed in Section 5.3.3.

7.2.2 Adaptive Algorithms for More Control

We are interested in how to integrate control over more parameters concurrently, including non-separable mappings from multiple inputs to multiple outputs. One of our early directions of exploration, and a method recently described by Gildfind et al. [GGaQ00], was the use of adaptive algorithms for this purpose. The challenge in this approach is maintaining a system which is both *usable*, in that it must both provide the necessary flexibility, as well as "understandable" so that the user can achieve desired motion trajectories.

As a particular example, we describe a preliminary experiment of ours in creating a mapping for controlling an articulated walk. In this case, the user wore a cyberglove, and a pre-animated walk cycle was used. As the walk cycle was played on the screen, the user moved his second and third fingers to imitate, or "walk along" with his fingers to the motion being shown. This generated a data set $\mathcal{D}$ consisting of pairs of vectors $\{\Upsilon_t, \theta_t\}$ at each time step, representing the joint angles of the user's hand and the joint angles of the animated character, respectively. A model $F_\xi(\Upsilon)$ parameterized by a vector ξ, was trained to predict θ by minimizing the sum squared error:

$$\min_{\xi} \left[\sum_t (F_\xi(\Upsilon_t) - \theta_t)^2 \right]$$

In these preliminary runs, F_ξ was chosen to be linear, but various non-linear models will

be explored in future work. Even with this model, however, a total of only about 5-10 minutes was required both to generate the necessary data and to train the function from the input vector to the joint angle state. The resulting mapping worked very well for the specific purpose of easily recreating a similar walk by repeating this twiddling finger motion[1].

The problem, however, lies not in (re)creating a very specific desired motion, but in generating a mapping which allows the creation of a desired *range* of motions. What happens, for example, if the user wants wants certain features of his or her hand motion (e.g. fingers spread apart more, or straighter, or thumb up, etc) to correspond to certain features of the walk (e.g. stride length, degree of forward leaning, etc)? Or, how about having the character walk while keeping his arms stiff at his sides, or even with one arm raised in front as if carrying a torch? With the mapping as described above, the only such parameter that was somewhat controllable was flexing and extending the fingers further to control the stride length, but even this was not very refined in its sensitivity.

We can distinguish between at least two main approaches to be taken in solving these problems adaptively. In the first approach two large data sets are obtained: one representing the feasible input domain that the user can create, and the second representing the variety of desirable output motion. Assuming the desirable range of output motion lies in a manifold which is a proper subset of the full joint angle space, and likewise for the feasible input domain, then one goal is to use an adaptive process to maximize the degrees of freedom of the output that the user can control. One of the initial designs for the GLOVETALKII system was based on this strategy, but at the time it was not possible to achieve a usable mapping in this way (discussion is given about this in [Fel94]). To point out one of the many difficulties with this approach, getting data sets that accurately

[1]This is in significant contrast to Gildfind et al.'s approach [GGaQ00], where the user, wearing two VPL data gloves, spends roughly one and a half hours testing and evaluating the results of nine generations of a genetic algorithm, to achieve a walk by alternately opening and closing the fingers on each hand.

represent "feasible" domain and range might nearly be an infeasible task in itself.

In the other type of approach, the mapping is essentially specified in advance, and corresponding data is created during the training sessions. In our example given above, this would mean that the relationships between hand-features and walk-features, or any other such behaviours, are determined, either by the designer or the user, and training data is created exemplifying these relationships. The hope in this case is that the adaptive mapping will capture all of these desired relationships, and interpolate between them in a sensible way. The final GLOVETALKII system worked in this way. Specifying the training data is much more practical by this approach, and the power of the adaptive algorithm serves to tune the parameters of this mapping to make it usable. However, the design of the mapping is an issue that remains to be addressed. If the user just "moves along" with pre-animated sequences following his or her instincts, but without having planned the motions in advance, then the problem will likely be that similar input configurations are going to end up being used for different output configurations, depending on the context of the motion, making the final learning task much more difficult or impossible[2].

Thus, as the complexity of the desired output range increases, generic approaches begin to break down, possibly unable to converge on a correct solution. For this reason, we believe that although our preliminary experiments with adaptive mappings were of limited general use, our current work in fully specifying an interactive system for general character motion provides a platform from which adaptive algorithms could now prove to be a very powerful and interesting tool.

Predicting Motion Layers

In addition to the questions and issues addressed above, we can also extend our current system with adaptive algorithms in another interesting way, by *re-contextualizing* the

[2]Alternately, if a sufficiently flexible model is used that can accomodate these examples, then the resulting mapping will be so sensitive to subtle variations in the input that it may be nearly impossible to control with any consistency for this reason itself.

problem as follows: Can we augment the multi-tracking control by providing a "guess" of what the other tracks might be doing, to give more context to the animator, even if different choices are made later on and recorded over them?

This could be designed in advance or possibly accomplished in an on-line manner. Consider as an example a situation wherein an animator wants to produce a sequence with some roughly-outlined structure A. She provides the kinematic data for a rough first pass of A, using three separate channels. Now, having seen how it all looks together, she wants to re-record the channels, but possibly changing some of the timing and details. She still has to do it in three separate tracks, say starting with Track 1. Now, however, rather than being *bound* to Tracks 2 & 3 as previously recorded, she would like the software to automatically use them as a template, and create new versions in real-time which correspond to his new recording. If the system "doesn't know" what to do, it just doesn't do anything; the objective of these tracks is to give her a better *context* by adaptive means while recording Track 1; she still always has the option of going back and re-recording Track 2 or Track 3 if the automatic version was not exactly what she wanted.

Note that, in comparison to the more general initial question of non-separable multi-dimensional mappings, this has the additional benefit that even if the user were unable to fully anticipate the software's "guess" for the appropriate motion, it would hopefully not throw the user off by quite as much, since that was not what she was directly trying to control in the first place. In contrast, if a general mapping, possibly achieved by adaptive means, does not have a clear intuitive relationship understood by the user, then the user might easily be thrown off by various surprises or unexpected motions during the control. The stability in the multi-tracking variation as described lies in the fact that at least *some* part of the motion was well-behaved (i.e. predictable) relative to the input from the user's point of view. We note furthermore that this would provide a convenient context for ongoing testing of such non-separable mappings, and, as "provider-

of-auxiliary-motion", the system may even be able to have the opportunity to adapt to a user over a longer period of time.

7.2.3 Additional Feedback

One of the challenges in establishing the CG character's contact with other objects, especially the ground, is due to the limited effect of the pure visual interface in this regard. When we make physical contact with an object, there is a burst of sensory input information, primarily tactile, as well as acoustic and visual. Furthermore, the physical presence of the objects automatically constrains or effects our movements on it or around it. It would be very interesting to extend DIGITAL MARIONETTE's modalities of interaction by exploring both auditory and haptic feedback.

7.2.4 "Home" Performance Animation

Interestingly, DIGITAL MARIONETTE could be adapted to become an inexpensive "home" performance animation system: Noting that the virtual characteristics of the input are very simple, with minimal occlusion effects, we believe that the electromagnetic-based sensors could be replaced by one or two small desk cameras to track the motion of two well-marked cylindrical tubes. This would be a significant economization from the traditional very high-end studios that have been a prerequisite for creating performance animation.

7.2.5 Extending the Input Device

Alternatively, we can consider ways of expanding the input device to achieve even more sophisticated levels of control. For example, the input sticks could potentially be enhanced with buttons or with a ring of pressure sensors. One application of this would be for controlling secondary motion characteristics. In particular, it would seem quite intu-

itive to use an input parameter such as grip strength to control stiffness parameters in a physically-based simulation. This would make use of an input degree of freedom that is perceptually (and "proprioceptually") quite separate from the positional and orientation input control parameters.

Other input device arrangements include the use of a microphone for verbal control over the multi-tracking interface, so that the user does not have to let go of an input device in order to press play or record. A more complex approach would also process the real-time audio input signal to drive the animation itself in a continuous fashion.

7.2.6 Film Motion Previsualization

Because creating animation with our system is so fast, it could also be applied in production contexts to create motion previews before fleshing out a full animation. Such motion previsualization (including for live action scenes) using animation is becoming a more frequently used technique in film production. Examples include "Crouching Tiger, Hidden Dragon" [Lee00], "Starship Troopers" [Gre98, Hay98], and "Star Wars: Episode I", all for which previsualization artists were hired to animate previews of various action scenes. That is, there is increasing value in creating "animatics", or moving storyboards; something which is fast to create, but also carrying more information than a storyboard, by actually showing relevant aspects of the movement, yet without spending any more time or effort on extraneous details than necessary to give a useful picture.

7.3 Conclusion

Our primary goal was to provide an efficient, powerful and satisfying interface for expressive character animation. We have achieved this goal and, moreover, we have brought the power of the performance animation tool to the desktop environment. To further

contextualize this result we recall the quote[3] from [Mai96] that, despite an awareness of multi-tracking layering techniques, "the great number of degrees of freedom that need to be controlled for complex human motion does not make digital puppetry a viable solution for realistic looking animation." Contrarily, in this work we demonstrate a performance animation system which enables a user to control complex human motion very effectively, using just two trackers.

One of the challenges in achieving this was indeed handling the large number of degrees of freedom of the output. By embedding the trackers in a physical tubing, we established a tangible interface with various coordinate frames inherent to the character, providing a valuable compatibility between the physical input device and the underlying transformations from input to motion. Building on this compatibility, we constructed a multi-track motion recording framework and a feasible set of layered mappings from the two input trackers to the thirty three degrees of freedom of the graphics character. We were then able to augment this kinematic-based approach by adding local physical models in such a way as to *simplify* the animation control rather than make it harder, and at the same time enhance the naturalness of the motion. Thus we have integrated kinematics and dynamical principles for more effective real-time animation control.

By synthesizing a variety of carefully designed techniques, including a tangible input device, local physically-based models and multi-tracking, we have introduced a novel system which extends the range and finesse achievable by real-time animation to allow control of complex natural motion of an articulated human geometry. An intrinsic benefit of the real-time interactive approach of our system is that creating such animation is extremely efficient.

This efficiency and directness of control can be very satisfying for the experienced user. Our desktop performance animation system enables the animator to fluidly and immediately play out his or her ideas of motion into animation, giving the animator a

[3]Given in full on page 8 of this work.

unique instrument for creating a range of truly expressive human character animation ranging from walking to dancing. DIGITAL MARIONETTE has been used in a live theatrical performance, and the motion created with it has been described by performers as "beautiful" and "graceful"— words that, although subjective, at the very least suggest an authenticity or naturalness of motion, and at best lie within the realm of the higher, elusive goals of animation as described in Chapter 1 of this work. The experience of both novice and advanced users further appears to indicate that DIGITAL MARIONETTE's interface provides opportunity for a depth of exploration, learning and refinement in the creation of animation.

By virtue of the multitracking interface and the control interface designed for each of the motion layers, we have been able to scale the required motion input configuration to a desktop environment, anticipating a new usability for virtual puppetry that needs a considerably smaller setup than existing systems. This leads to a great variety of possible applications, ranging from film motion previsualization to home computer puppetry and entertainment, in addition to performance itself. The simplicity of the input device design could eventually allow the motion tracking to be achieved by using camera-based rather than electromagnetic tracking, further increasing the accessibility of the system.

The strength and originality of our system is not just in the efficiency, or the naturalness, or the potential complexity of the motion, but also in the union and combination of these elements.

Having established a mapping and input interface that succeeds in achieving expressive control over a fully articulated graphical human character (including for complex tasks such as walking), we have provided a valuable solution and benchmark which will be helpful in guiding the design of increasingly complex control approaches. In a system such as GLOVETALKII, for example, early experiments using unsupervised learning proved to be unsuccessful for automatically generating a mapping from hand motions to speech, and furthermore gave no indication of whether such a mapping could even exist.

However, once a mapping was carefully hand-designed, then supervised neural networks were very helpful in achieving and refining the required gesture classification by adapting to the user's hand motions. In our case, initial experiments were inconclusive as to whether methods such as physics or adaptive algorithms would achieve useable results. However, we have provided a fully functional solution for desktop character animation, and this will provide both a *reference point* for exploration using tools such as adaptive algorithms or physical simulation, and a *foundation* that should make it possible to carry these approaches much further and with better results than would have been achieved otherwise. While early full 3D physical simulations were not possible to control interactively, our local physics-based models have shown us that some dynamics can, in fact, be integrated into the control mechanism. Furthermore, it can be done so as to help the real-time animator, rather than make his or her job more difficult. The question of how to extend such application of physical principles to the real-time animation task is another new direction to pursue.

We have demonstrated a performance animation system that provides a solution for animating an articulated human character, combining the expressiveness available by human low-level input with the efficiency of real-time interaction, achieving a new level of animation control via performance animation. In doing so, we have provided a fundamental starting point for further research in using real-time interactive animation techniques for sophisticated motions such as character animation. We hope this will encourage exploration and development of more such controllers with different qualities and underlying mechanisms.

Appendix A

Quaternions for Orientation

Some basic properties and definitions are given here; more information can be found in, e.g. [PW82, Han00, WW92].

A.1 Basic Properties

Quaternions originated as a 4-dimensional generalization of complex numbers, with complex basis directions $\mathbf{i}, \mathbf{j}, \mathbf{k}$, for which an associative, non-commutative multiplication is defined, with:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{A.1}$$

The set $\{\pm 1, \pm\mathbf{i}, \pm\mathbf{j}, \pm\mathbf{k}\}$ forms a group with respect to multiplication, with $+1$ being the left and right multiplicative identity.

For any scalars $a \in \mathbb{R}$, and for any basis directions $\mathbf{d} \in \{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$:

$$a\mathbf{d} = \mathbf{d}a \tag{A.2}$$

$$a_1\mathbf{d}_1 + a_2\mathbf{d}_2 = a_2\mathbf{d}_2 + a_1\mathbf{d}_1 \tag{A.3}$$

$$(a_2 + a_3)(a_1\mathbf{d}_1) = a_2a_1\mathbf{d}_1 + a_3a_1\mathbf{d}_1 \tag{A.4}$$

$$a_3(a_1\mathbf{d}_1 + a_2\mathbf{d}_2) = a_3a_1\mathbf{d}_1 + a_3a_2\mathbf{d}_2 \tag{A.5}$$

Furthermore, for any two sums, $\mathbf{u}_1 = \sum_i a_i \mathbf{d}_i$ and $\mathbf{u}_2 = \sum_j a_j \mathbf{d}_j$, multiplication is defined by:

$$\mathbf{u}_1 \mathbf{u}_2 = \sum_i \sum_j (a_i \mathbf{d}_i)(a_j \mathbf{d}_j) \tag{A.6}$$

A quaternion $\mathbf{q}$ represents a sum:

$$\mathbf{q} = w_0 + w_1 \mathbf{i} + w_2 \mathbf{j} + w_3 \mathbf{k} \tag{A.7}$$

where the coefficients $w_0, w_1, w_2, w_3 \in \mathbb{R}$ are scalar.

Although not a vector, quaternions are often written in vector form:

$$\begin{aligned} \mathbf{q} &= \langle w_0, w_1, w_2, w_3 \rangle \\ &= \langle w_0, \mathbf{w} \rangle \end{aligned}$$

A.2 Adding and Multiplying Quaternions

Let $\mathbf{q_a}, \mathbf{q_b}$ and $\mathbf{q_c}$ be quaternions, where $\mathbf{q_a} = \langle a_0, \mathbf{a} \rangle = \langle a_0, a_1, a_2, a_3 \rangle = a_0 + a_1 \mathbf{i} + a_2 \mathbf{j} + a_3 \mathbf{k}$ and similarly for $\mathbf{q_b}$ and $\mathbf{q_c}$. Quaternion multiplication follows from Eq (A.6), and using Eq (A.1)–(A.5) to simplify we get that

$$\mathbf{q_a} \mathbf{q_b} = \langle a_0 b_0 - \mathbf{a} \cdot \mathbf{b}, a_0 \mathbf{b} + b_0 \mathbf{a} + \mathbf{a} \times \mathbf{b} \rangle \tag{A.8}$$

where $\mathbf{a} \times \mathbf{b}$ is the cross product of vectors $\langle a_1, a_2, a_3 \rangle$ and $\langle b_1, b_2, b_3 \rangle$. Using this twice gives us associativity of quaternion multiplication

$$\begin{aligned} \mathbf{q_a}(\mathbf{q_b}\mathbf{q_c}) = (\mathbf{q_a}\mathbf{q_b})\mathbf{q_c} \quad = \quad & \langle a_0 b_0 c_0 - a_0 \mathbf{b} \cdot \mathbf{c} - b_0 \mathbf{a} \cdot \mathbf{c} - c_0 \mathbf{a} \cdot \mathbf{b} - \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}), \\ & a_0 b_0 \mathbf{c} + a_0 c_0 \mathbf{b} + b_0 c_0 \mathbf{a} \\ & + a_0 (\mathbf{b} \times \mathbf{c}) + b_0 (\mathbf{a} \times \mathbf{c}) + c_0 (\mathbf{a} \times \mathbf{b}) \\ & - (\mathbf{b} \cdot \mathbf{c}) \mathbf{a} - (\mathbf{a} \cdot \mathbf{b}) \mathbf{c} + (\mathbf{a} \cdot \mathbf{c}) \mathbf{b} \rangle \end{aligned} \tag{A.9}$$

Notice that when $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are the $\mathbf{x}, \mathbf{y}, \mathbf{z}$ direction vectors, then Eq (A.9) simplifies considerably:

$$\mathbf{q_x} \mathbf{q_y} \mathbf{q_z} \quad = \quad \langle a_0 b_0 c_0 - 1, (b_0 c_0 + a_0) \mathbf{x} + (a_0 c_0 - b_0) \mathbf{y} + (a_0 b_0 + c_0) \mathbf{z} \rangle$$

$$= \langle a_0 b_0 c_0 - 1, b_0 c_0 + a_0, a_0 c_0 - b_0, a_0 b_0 + c_0 \rangle$$

A.3 Quaternion Properties

The conjugate of a quaternion $\mathbf{q} = \langle q_0, \mathbf{q} \rangle$ is defined as

$$\bar{\mathbf{q}} = \langle q_0, -\mathbf{q} \rangle \tag{A.10}$$

and the magnitude is defined by

$$\|\mathbf{q}\| = \left(\sum_{i=0}^{3} w_i^2 \right)^{\frac{1}{2}} \tag{A.11}$$

Note that $\mathbf{q}\bar{\mathbf{q}} = \langle \|\mathbf{q}\|^2, 0, 0, 0 \rangle$.

For the quaternion of unit length (where $\|\mathbf{q}\| = 1$),

$$\mathbf{q}^{-1} = \bar{\mathbf{q}} \tag{A.12}$$

A.4 Rotations With Quaternions

A vector $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$ can be represented by a corresponding quaternion with a 0 scalar component:

$$\mathbf{q_v} = \langle 0, \mathbf{v} \rangle \tag{A.13}$$

A rotation R by angle θ about a normalized axis vector $\mathbf{n}$ can be represented by the quaternion $\mathbf{q}_R = \langle r_0, \mathbf{r} \rangle$, where

$$\begin{aligned} r_0 &= \cos \frac{\theta}{2} \\ \mathbf{r} &= \left(\sin \frac{\theta}{2} \right) \mathbf{n} \end{aligned} \tag{A.14}$$

The rotation of vector $\mathbf{v}$ by R is then given by

$$\mathbf{q_w} = \mathbf{q}_R \mathbf{q_v} \mathbf{q}_R^{-1} \tag{A.15}$$

where $\mathbf{q_w}$ represents the rotated vector as per Eq (A.13). A shorthand notation for this is simply

$$\mathbf{w} = \mathbf{q}_R \mathbf{v} \mathbf{q}_R^{-1}$$

A.4.1 Rotating v into u

Given a unit vector $\mathbf{v}$ which is to be rotated into unit vector $\mathbf{u}$, one needs only to compute the corresponding angle θ and axis of rotation $\mathbf{n}$. The simplest way to do this is by letting the axis of rotation be $\mathbf{n} = \mathbf{v} \times \mathbf{u}$, and $\theta = \arccos(\mathbf{v} \cdot \mathbf{u})$, and then substituting this in Eq (A.14) above. If we additionally want to keep some vector $\mathbf{w}$ fixed under the rotation, so that $\mathbf{w} = \mathbf{q}_R \mathbf{w} \mathbf{q}_R^{-1}$, and if additionally we know that the angle $\angle(\mathbf{w}, \mathbf{u})$ between $\mathbf{w}$ and $\mathbf{u}$ is the same as $\angle(\mathbf{w}, \mathbf{v})$, then this is equivalent to rotating from $(\mathbf{w} \times \mathbf{v})$ to $(\mathbf{w} \times \mathbf{u})$ about axis $\mathbf{w}$.

A.4.2 Compositions of Rotations

Suppose we want to rotate vector $\mathbf{v}$ by rotation R_1 followed by rotation R_2, corresponding to quaternions $\mathbf{q}_{R_1}$ and $\mathbf{q}_{R_2}$ respectively. Then from Eq (A.15) we have that

$$\mathbf{q_w} = \mathbf{q}_{R_2} \left(\mathbf{q}_{R_1} \mathbf{q_v} \mathbf{q}_{R_1}^{-1} \right) \mathbf{q}_{R_2}^{-1} \tag{A.16}$$

which, by associativity (Eq A.9), is equivalent to first multiplying the the two rotation quaternions:

$$\begin{aligned} \mathbf{q_w} &= \left(\mathbf{q}_{R_2} \mathbf{q}_{R_1} \right) \mathbf{q_v} \left(\mathbf{q}_{R_1}^{-1} \mathbf{q}_{R_2}^{-1} \right) \\ \mathbf{q_w} &= \left(\mathbf{q}_{R_2} \mathbf{q}_{R_1} \right) \mathbf{q_v} \left(\mathbf{q}_{R_2} \mathbf{q}_{R_1} \right)^{-1} \end{aligned} \tag{A.17}$$

Thus composition of rotations corresponds to quaternion multiplication.

References

[AG85] William Armstrong and Mark Green, *The dynamics of articulated rigid bodies for the purposes of animation*, The Visual Computer **1** (1985), no. 4.

[AGL86] William Armstrong, Mark Green, and R. Lake, *Near-real-time control of human figure models*, Graphics Interface '86, Canadian Information Processing Society, May 1986.

[AGL87] William Armstrong, Mark Green, and R. Lake, *Near-real-time control of human figure models*, IEEE Computer Graphics and Applications **7** (1987), no. 6, 52–61.

[Arm79] William Armstrong, *Recursive solutions to the equations of motion of an n-link manipulator*, Proc. Fifth World Congress on the Theory of Machines and Mechanisms, American Society of Mechanical Engineers, 1979, pp. 1343–1346.

[BB98] Rama Bindiganavale and Norman I. Badler, *Motion abstraction and mapping with spatial constraints*, CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments (Geneva, Switzerland) (Nadia Magnenat-Thalmann and Daniel Thalmann, eds.), November 1998, pp. 70–82.

[BC89] Armin Bruderlin and Tom W. Calvert, *Goal-directed, dynamic animation of human walking*, Computer Graphics **23** (1989), no. 3, 233–242.

[Ber47] N. A. Bernstein, *O postroyenyii dvizhniy [on the construction of movements]*, Medgiz, Moscow, 1947.

[Ber67] N. A. Bernstein, *The co-ordination and regulation of movements*, Pergamon Press, London, 1967.

[Ber87] Philippe Bergeron, *3-d character animation on the symbolics system*, SIGGRAPH 87 : Course Notes, vol. 5, 1987.

[BJ97] Véronique Benquey and Laurent Juppé, *The use of real-time performance animation in the production process*, ACM SIGGRAPH 97 : Course Notes (Los Angeles, California), vol. 1, ACM Press, 1997.

[BL85] P. Bergeron and P. Lachapelle, *Controlling facial expressions and body movements*

in the computer-generated animated short Tony De Peltrie, Siggraph 85 Advanced Computer Animation Seminar Notes (New York), ACM Press, July 1985.

[BM96] Ronan Boulic and Ramon Mas, *Hierarchical kinematic behaviors for complex articulated figures*, Interactive Computer Animation (Nadia Magnenat Thalmann and Daniel Thalmann, eds.), Prentice Hall Europe, London, 1996.

[BPW93] Norman I. Badler, Cary B. Phillips, and Bonnie Lynn Webber, *Simulating humans : Computer graphics animation and control*, Oxford University Press, New York, 1993.

[BRRP97] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella, *The process of motion capture: Dealing with the data*, Computer Animation and Simulation '97 : Proceedings of the Eurographics Workshop in Budapest, Hungary (NY) (D. Thalmann and M. van de Panne, eds.), Springer Computer Science, Springer, September 1997, pp. 3–18.

[Bru88] Armin Bruderlin, *Goal-directed, dynamic animation of bipedal locomotion*, Master's thesis, Simon Fraser University, Vancouver, Canada, 1988.

[Bux86] W. Buxton, *There's more to interaction than meets the eye: Some issues in manual input*, User Centered System Design: New Perspectives on Human-Computer Interaction (D. A. Norman and S. W. Draper, eds.), Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986, pp. 319–337.

[BW71] N. Burtnyk and M. Wein, *Computer generated key frame animation*, Journal of SMPTE **80** (1971).

[BW95] Armin Bruderlin and L. Williams, *Motion signal processing*, Computer Graphics, 1995, Annual Conference Series, ACM, pp. 97–104.

[Cat78] Edwin Catmull, *The problems of computer-assisted animation*, Proceedings of SIGGRAPH 78, vol. 12, August 1978, pp. 348–353.

[deG] B. deGraf, *Protozoa*, Company.

[deG89] B. deGraf, *Notes on human facial animation*, SIGGRAPH 89 : Course Notes (Boston, Massachusetts), vol. 22, 1989, pp. 10–11.

[dGY99] Brad de Graf and Emre Yilmaz, *Puppetology: Science or cult?*, Animation World **3** (1999), no. 11.

[DH00] Bruce Randall Donald and Frederick Henle, *Using haptic vector fields for animation motion control*, Proc. IEEE International Conference on Robotics and Automation (ICRA) (San Francisco, CA), April 2000.

[Dis40] Walt Disney, *Fantasia*, film, 1940.

[Dis92] Disney, *Aladdin*, film, 1992.

[DMZ95] Scott Dyer, Jeff Martin, and John Zulauf, *Motion capture white paper*, Tech. report, Windlight Studios, Alias—Wavefront, December 1995.

[Duf00] Greg Duffell, *Timing in animation*, Lecture (unpublished), 2000.

[Dye97] Scott Dyer, *Introduction*, ACM SIGGRAPH 97 : Course Notes (Los Angeles, California), vol. 1, 1997.

[Fea83] R. Featherstone, *The calculation of robot dynamics using articulated-body inertias*, International Journal of Robotics Research (1983), 13–30.

[Fel94] Sid Fels, *Glove-talkII: Mapping hand gestures to speech using neural networks - an approach to building adaptive interfaces*, Ph.D. thesis, University of Toronto, 1994.

[FH95] Sid Fels and Geoffrey Hinton, *Glove-talkII: An adaptive gesture-to-format interface*, Proceedings of CHI'95 Human Factors in Computing Systems, ACM Press, 1995, pp. 456–463.

[Fit96] George W. Fitzmaurice, *Graspable user interfaces*, Ph.D. thesis, University of Toronto, 1996.

[Fla97] Flaherty, *Satan's rotoscope in practice*, ACM SIGGRAPH 97 : Course Notes (Los Angeles, California), vol. 1, 1997.

[GGaQ00] Andrew Gildfind, Michael A. Gigante, and Ghassan al Qaimari, *Evolving performance control systems for digital puppetry*, Journal of Visualisation and Computer Animation (2000).

[GL98] Michael Gleicher and Peter Litwinowicz, *Constraint-based motion adaptation*, Journal of Visualization and Computer Animation **9** (1998), no. 1, 65–94.

[Gle98] Michael Gleicher, *Retargetting motion to new characters*, Proceedings of SIGGRAPH 98, ACM SIGGRAPH, 1998, pp. 33–42.

[GM85] M. Girard and A.A. Maciejewski, *Computational modeling for the computer animation of legged figures*, Proceedings of SIGGRAPH 85, vol. 20, 1985, pp. 263–270.

[GMW81] Phillip Gill, Walter Murray, and Margaret White, *Practical optimization*, Academic Press, New York, New York, 1981.

[GR96] Shang Guo and James Robergé, *A high level control mechanism for human locomotion based on parametric frame space interpolation*, Computer Animation and Simulation '96 : Proceedings of the Eurographics Workshop (Austria) (Ronan Boulic and Gerard Hégron, eds.), Springer-Verlag/Wien, 1996, pp. 95–108.

[Gre82] Peter H. Greene, *Why is it easy to control your arms?*, Journal of Motor Behavior **14** (1982), no. 4, 260–286.

[Gre98] Colin Green, *Previsualization for "starship troopers": Managing complexity in motion control*, Conference Abstracts and Applications: Technical Sketches, ACM SIGGRAPH, 1998, p. 294.

[Grz98] Radek Grzeszczuk, *Neuroanimator: Fast neural network emulation and control of physics-based models*, Ph.D. thesis, University of Toronto, Toronto, Canada, May 1998.

[GT95] Radek Grzeszczuk and Demetri Terzopoulos, *Automated learning of muscle-actuated locomotion through control abstraction*, Proceedings of SIGGRAPH 95 (Los Angeles, CA), August 1995, pp. 63–70.

[GTH98] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton, *Neuroanimator: Fast neural network emulation and control of physics-based models*, (to appear in) Proc. SIGGRAPH 98, 1998.

[Han00] Andrew J. Hanson, *Visualizing quaternions*, SIGGRAPH 2000 : Course Notes, 2000.

[Hat77] H. Hatze, Biological Cybernetics **25** (1977).

[Hay98] Craig Hayes, *Starship troopers*, Conference Abstracts and Applications: Technical Sketches, 1998, p. 311.

[HFP+00] L. Herda, P. Fua, R. Plänkers, R. Boulic, and Daniel Thalmann, *Local and Global*

Skeleton Fitting Techniques for Optical Motion Capture, Computer Animation, 2000.

[Hin84] Geoffrey H. Hinton, *Some computational solutions to bernstein's problems*, Human Motor Actions - Bernstein Reassessed (H.T.A. Whiting, ed.), Elsevier Science Publishers B.V., The Netherlands, 1984, pp. 413–438.

[Hin96] Ken Hinckley, *Haptic issues for virtual manipulation*, Ph.D. thesis, University of Virginia, 1996.

[HOT97] Jessica K. Hodgins, James F. O'Brien, and Jack Tumblin, *Do geometric models affect judgements of human motion*, Graphics Interface '97, 1997, pp. 17–25.

[HRRS86] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: The approach based on influence functions*, John H. Wiley, New York, 1986.

[HS84] P. Hanrahan and D. Sturman, *Interactive control of parametric models*, Proceedings of SIGGRAPH 84, 1984.

[HS85] P. Hanrahan and D. Sturman, *Interactive animation of parametric models*, The Visual Computer **1** (1985).

[IRT81] Verne T. Inman, Henry J. Ralston, and Frank Todd, *Human walking*, Williams and Wilkins, Baltimore, MD, 1981.

[Joh73] G. Johansson, *Visual perception of biological motion and a model for its analysis*, Perception & Psychophysics **14** (1973), no. 2, 201–211.

[Kaw87] Toshifumi Kawa, *Tony de peltrie - symbol of a new era of computer animation*, SIGGRAPH 87 : Course Notes, vol. 5, 1987.

[KB93] Hyeongsuk Ko and Norman I. Badler, *Intermittent non-rhythmic human stepping and locomotion*, Tech. Report IRCS-93-13, 1993.

[KB96] H. Ko and Norman Badler, *Animating human locomotion with inverse dynamics*, IEEE Computer Graphics and Applications **16** (1996), no. 2, 50–59.

[KE93] Wynn Kapit and Lawrence M. Elson, *The anatomy coloring book*, 2nd ed., Addison-Wesley Educational Publishers, Inc., 1993.

[KHSW95] B. Knep, C. Hayes, R. Sayre, and T. Williams, *Dinosaur input device*, Proceedings

of CHI '95 Conference on Human Factors in Computing Systems (New York), ACM, 1995, pp. 304–309.

[Koz92] John R. Koza, *Genetic Programming*, MIT Press, 1992.

[Lee00] Ang Lee, *Crouching tiger hidden dragon*, 2000.

[LS84] Kenneth Laws and Martha (photographs) Swope, *The physics of dance*, Independently published (can be ordered from author's website), 1984.

[LS00] Jeehee Lee and Sung Yong Shin, *Multiresolution motion analysis and synthesis*, Tech. Report CS/TR-2000-149, 2000.

[LvdPF96] Joseph F. Laszlo, M. van de Panne, and E. Fiume, *Limit cycle control and its applications to the animation of balancing and walking*, Proceedings of SIGGRAPH 96 (New Orleans), August 1996, pp. 155–162.

[LvdPF00] Joseph F. Laszlo, M. van de Panne, and E. Fiume, *Interactive control for physically-based animation*, Proceedings of SIGGRAPH 2000, ACM SIGGRAPH, 2000.

[Mai96] Roberto Maiocchi, *3-D Character Animation Using Motion Capture*, Interactive Computer Animation (Nadia Magnenat Thalmann and Daniel Thalmann, eds.), Prentice Hall Europe, London, 1996.

[McM84a] T.A. McMahon, International Journal of Robotics Research **3** (1984), no. 2, 4–28.

[McM84b] T.A. McMahon, *Muscles, reflexes, and locomotion*, Princeton University Press, Princeton, N.J., 1984.

[Men99] A. Menache, *Understanding motion capture for computer animation and video games*, Morgan Kaufmann, 1999.

[MI94] C. Mackenzie and T. Iberall, *The grasping hand*, Advances in Psychology, vol. 104, Amsterdam, 1994.

[MI97] F. A. Mussa-Ivaldi, *Nonlinear force fields: a distributed system of control primitives for representing and learning movements*, Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997, pp. 84–90.

[MIGB94] F. A. Mussa-Ivaldi, S. F. Giszter, and E. Bizzi, *Linear combinations of primitives*

in vertebrate motor control, Proc. Natl. Acad. Sci. USA, vol. 91, August 1994, pp. 7534–7538.

[Mor98] Mike Morasky, *Wiring cracker: The mechanics of a non-anthropomorphic real-time, performance animation system*, Siggraph 98: Conference Abstracts and Applications (New York), Computer Graphics Annual Conference Series, ACM SIGGRAPH, 1998, p. 310.

[MZW99] Maja J. Matarić, Victor B. Zordan, and Matthew M. Williamson, *Making complex articulated agents dance*, Autonomous Agents and Multi-Agent Systems **2** (1999), no. 1.

[NF] Victor Ng and Petros Faloutsos, *Dance: Dynamic animation and control environment*, Software system, www.dgp.toronto.edu/software/dance/dance.html.

[OO81] T.J. O'Donnell and Arthur J. Olsen, *Gramps - a graphicsl language interpreter for real-time, interactive, three-dimensional picture editing and animation*, Proceedings of SIGGRAPH 81, vol. 15, August 1981, pp. 133–142.

[OSB99] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck, *Discrete time signal processing*, 2nd ed., 1999.

[Par79] L.A. Pars, *A treatise on analytical dynamics*, Ox Bow Press, Woodbridge, Conn., 1979.

[Per85] Ken Perlin, *An Image Synthesizer*, Proceedings of SIGGRAPH 85, vol. 19, 1985, pp. 287–296.

[Per95] K. Perlin, *Real time responsive animation with personality*, IEEE Transactions on Visualization and Computer Graphics **1** (1995), no. 1, 5–15.

[PG96] Ken Perlin and Athomas Goldberg, *Improv: A system for scripting interactive actors in virtual worlds*, 1996.

[Pix86] Pixar, *Luxo Jr.*, film, 1986, Created by John Lasseter.

[Pla71] S. Plagenhoef, *Patterns of human motion: A cinematographic analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[Pol] Polhemus, `www.polhemus.com`.

[PW82] Edward Pervin and Jon A. Webb, *Quaternions in computer vision and robotics*, Tech. Report CMU-CS-82-150, 1982.

[Rey86] Craig W. Reynolds, *Real-time animtion*, SIGGRAPH 86 : Course Notes, 1986.

[RG75] L. R. Rabiner and B. Gold, *Theory and application of digital signal processing*, Prentice Hall, Englewood Cliffs, N. J., 1975.

[RGBC96] Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen, *Efficient generation of motion transitions using spacetime constraints*, Proceedings of SIGGRAPH 96 (New Orleans), August 1996, pp. 147–154.

[Rob88] Barbara Robertson, *Mike, the Talking Head*, Computer Graphics World (1988), 15–17.

[Rob96] Barbara Robertson, *Cyber acting*, CGW (1996).

[Rob97] Barbara Robertson, *Making motion easy*, Computer Graphics World **20** (1997), no. 4, 42–48.

[SJW98] ChangWhan Sul, SoonKi Jung, and Kwangyun Wohn, *Synthesis of Human Motion Using Kalman Filter*, CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments (Geneva, Switzerland) (Nadia Magnenat-Thalmann and Daniel Thalmann, eds.), November 1998, pp. 100–112.

[Sør89] Peter Sørenson, *Felix the cat - real time computer animation*, Animation Magazine (1989), 13–14.

[SPB+98] Marius-Călin Silaghi, Ralf Pläankers, Ronan Boulic, Pascal Fua, and Daniel Thalmann, *Local and Global Skeleton Fitting Techniques for Optical Motion Capture*, CAPTECH '98: Workshop on Modelling and Motion Capture Techniques for Virtual Environments (Geneva, Switzerland) (Nadia Magnenat-Thalmann and Daniel Thalmann, eds.), November 1998.

[Spi93] Steven Spielberg, *Jurassic park*, Film, July 1993.

[Ste83] Garland Stern, *Bbop - a program for 3-dimensional animation*, Nicograph '83 Proceedings (Tokyo, Japan), December 1983, pp. 403–404.

[Stu87] D. Sturman, *Interactive keyframe animation of 3-d articulated models*, SIGGRAPH

'87, Course Notes, July 1987, Also available in Siggraph '90 Course Notes On 'Human Figure Animation: Approaches and Applications' (pages 32–41).

[Stu94] David Sturman, *A brief history of motion capture for computer character animation*, SIGGRAPH 94: Course Notes for Course 9: Character Motion Systems, 1994.

[Stu98] David J. Sturman, *Computer puppetry*, IEEE Computer Graphics and Applications **18** (1998), no. 1, 38–45.

[SZ94] David J. Sturman and David Zeltzer, *A survey of glove-based input*, IEEE computer Graphics and Applications (1994), 30–39.

[Tar91] H. Tardif, *Character animation in real time*, http://siggraph.org/publications/panels/siggraph91/p11.html, 1991.

[Tho87] Frank Thomas, *The future of character animation by computer*, SIGGRAPH 87 : Course Notes, vol. 5, 1987.

[TJ81] Frank Thomas and Ollie Johnston, *Disney animation– the illusion of life*, Abbeville Press, New York, 1981.

[Tro95] James J. Troy, *Dynamic balance and walking control of biped mechanisms*, Ph.D. thesis, Iowa State University, 1995.

[Tro98] James J. Troy, *Real-time Dynamic Balancing and Walking Control of 7-Link Biped*, Proceedings of ASME Design Engineering Technical Conferences, 1998.

[Tro00] James J. Troy, *Haptic control of a simplified human model*, http://www.cs.sandia.gov/SEL/conference/pug00/papers/pugtroy.pdf, 2000.

[TV95] J. Troy and M. Vanderploeg, *Interactive Simulation and Control of Planar Biped Walking Devices*, Workshop on Simulation and Interaction in Virtual Environments, July 1995, pp. 220–224.

[Vas92] W. Vasulka, *Lee harrison III*, Pioneers of Electronic Art, exhibition catalog for Ars Electronica 1992 (D. Dunn, ed.), Linz, Austria, 1992, pp. 92–95.

[VB86] Susan Van Baerle, *Character animation : Combining computer graphics with traditional animation*, Graphics Interface '86, 1986.

[vdP97] Michiel van de Panne, *From footprints to animation*, Computer Graphics Forum **16** (1997), no. 4, 211–223.

[vdP01] Michiel van de Panne, *Motionplayground*, www.motionplayground.com, 2001.

[vdPL95] Michiel van de Panne and Alexis Lamouret, *Guided optimization for balanced locomotion*, Computer Animation and Simulation '95 : Proceedings of the Eurographics Workshop (Netherlands) (Wien, Austria) (Demetri Terzopoulos and Daniel Thalmann, eds.), Springer-Verlag, 1995, pp. 165–177.

[Wal89] Graham Walters, *The story of waldo c. graphic*, SIGGRAPH 89 : Course Notes, vol. 4, 1989, pp. 65–79.

[Wila] Andy Wilson, *Luxomatic : Performance animation with a sock puppet*, http://white.media.mit.edu/ drew/luxomatic/luxomatic.html.

[Wilb] Andy Wilson, *Natural modes of human form and motion*, http://www-white.media.mit.edu/ drew/9.34/text.html.

[Wilc] Andy Wilson, *The seagull: A performance animation system*, http://www-white.media.mit.edu/ drew/seagull/seagull.html.

[Wil86] Jane Wilhelms, *Virya - a motion control editor for kinematic and dynamic animation*, Graphics Interface '86, Canadian Information Processing Society, May 1986.

[Wil87] Jane Wilhelms, *Using dynamic analysis for realistic animation of articulated bodies*, IEEE Computer Graphics and Applications **7** (1987), no. 6, 12–27.

[Win90] David A. Winter, *Biomechanics and motor control of human movement*, 2 ed., Wiley Interscience, New York, NY, 1990.

[WK88] Andrew Witkin and Michael Kass, *Spacetime constraints*, Computer Graphics **22** (1988), 159–168, Proceedings of SIGGRAPH 88.

[WO82] M.W. Walker and D.E. Orin, *Efficient dynamic computer simulation of robotic mechanisms*, Journal of Dynamic Systems, Measurement, and Control (1982), 205–211.

[WP95] Andrew Witkin and Zoran Popović, *Motion warping*, Proceedings of SIGGRAPH 95, ACM, Inc., 1995, pp. 105–108.

[WW92] Alan Watt and Mark Watt, *Advanced animation and rendering techniques*, ACM Press, Addison-Wesley, Harlow England, 1992.

[YN99] Katsu Yamane and Yoshihiko Nakamura, *Dynamics computation of structure-*

varying kinematic chains for motion synthesis of humanoid, Proceedings of the 1999 IEEE International Conference on Robotics & Automation (ICRA) (Detroit, Michigan), May 1999.

[YN00] Katsu Yamane and Yoshihiko Nakamura, *Dynamics filter— concept and implementation of on-line motion generator for human figures*, Proceedings of the 2000 IEEE International Conference on Robotics & Automation (ICRA) (San Francisco, CA), April 2000.

[ZH99] Victor B. Zordan and Jessica K. Hodgins, *Tracking and modifying upper-body human motion data with dynamic simulation*, Computer Animation and Simulation '99 : Proceedings of the Eurographics Workshop in Budapest, Hungary, 1999.

[Zha96] Xinmin Zhao, *Kinematic control of human postures for task simulation*, Ph.D. thesis, 1996.

[ZMB96] S. Zhai, P. Milgram, and W. Buxton, *The effects of using fine muscle groups in multiple degree-of-freedom input*, Proc. ACM CHI'96, 1996, pp. 308–315.