Adapting and Reconfiguring Human Figure Motion Capture Data through the Application of Inverse Kinematics and Biomechanics-Based Optimisation

Michael J. Meredith

Doctor of Philosophy
Department of Computer Science
The University of Sheffield
September 2005

Adapting and Reconfiguring Human Figure Motion Capture Data through the Application of Inverse Kinematics and Biomechanics-Based Optimisation

Michael J. Meredith

Abstract

This thesis investigates the issue of modifying motion capture data, specifically the reconfiguration process which includes retargeting and individualisation. To perform modifications, a series of novel algorithms are introduced, where the first is grounded in the domain of inverse kinematics and the second is in dynamics. By applying the algorithms to existing motions, it is shown how the tasks of simple retargetting problem, individualisation and injury simulation can be achieved. These are the limit of the inverse kinematics technique. In contrast, the dynamics-based algorithm also provides the ability to add in plausible environmental or force-based changes.

Aside from the algorithms themselves, the reconfiguration of motions demonstrates the most significant portion of this work in that it is possible to take a single piece of motion data from a source actor and spawn many different versions of it in order to produce motions that better portray the build and biomechanical structure of a target character. This addresses the issue of using the same motion for each and every character regardless of its shape and size, which looks unrealistic. The reconfigured motions are produced using an example motion of a source actor and the biomechanical information of the target actor. Comparing the reconfigured motions to the real motions of target actors provides a validation for these techniques.

In addition to the two main threads of work that come from the inverse kinematics and dynamics-based modification algorithms, a new method of processing positional motion capture marker data to result in an animated hierarchical data structure is presented.

Acknowledgements

I would like to start by thanking Infograms UK, in particular Ian Badcoe, who provided the initial funding to get this project underway, without which the project would have never taken place. I would also like to very much thank Nic Chilton from Simula, Bradford University, who made it possible for me to capture the motions that were subsequently vital to the evaluation phase of the thesis. To that end, I would like to acknowledge the self-sacrifices of the 3 performers who had to wear the rather unendearing motion capture suits so that we could record them. Given that the biometric data of each of the actors is included in this work, I give no ordering in the names of these actors when I say thank you to Ahmed BinSubaih, Miguel Sales, and Steve Maddock (however, the actor labelling, i.e., A, B & D, could have gone in alphabetical order of their forenames, with me, Mike, included in the mix. Furthermore, if anyone wishes to see any of the actors wearing their fantastically appealing motion capture suits, I still have the video files on demand).

During the overall course of this work, I would like to recognise the listening and constructive comment services provided by James Edge and Manuel Sanchez, and indeed the whole computer graphics research group at the University of Sheffield, for their invaluable feedback during those moments of posing sensible, yet more often than not, unintelligent questions.

I would also like to say a big thank you to Steve Maddock who provided guidance, support (and grammar checking skills) as my PhD supervisor. I would also like to thank Steve for having faith in me from the outset and giving me the opportunity to undertake the research work – thank you for everything, it is greatly appreciated.

Research Publications

Refereed Publications

M. Meredith, S. Maddock, "*Adapting Motion Capture using weighted Real-Time Inverse Kinematics*", ACM Computers in Entertainment, Jan/Mar 2005

M. Meredith, S. Maddock, "*Individualised Character Motion Using Weighted Real-Time Inverse Kinematics*", GAME-ON 2004 (Best paper of the conference), pp.57-64, 2004

M. Meredith, S. Maddock, "*Adapting Motion Capture using weighted Real-Time Inverse Kinematics*", GDTW 2004 (Best paper of the conference), pp.120-129, 2004

M. Meredith, S. Maddock, "*Using a half-Jacobian for real-time inverse kinematics*", CGAIDE'04, pp.81-88, 2004

Technical Reports

M. Meredith, S. Maddock, "*Real-Time Inverse Kinematics: The Return of the Jacobian*", Department of Computer Science Research Memorandum CS-04-06, University of Sheffield, 2004

M. Meredith, S. Maddock, "*Motion Capture File Formats Explained*", Department of Computer Science Technical Report CS-01-11, University of Sheffield, 2001

Contents

Appendices

List of Figures

Any figures that are marked with a camera symbol (🎥) indicate that there is an accompanying animation file for that figure on the included thesis CD. Associated animation files on the thesis CD are named the same as the figure numbers.

List of Tables

Chapter 1:
Introduction

The animation of artificial characters was first seen in Winsor McKay's “Gertie the Dinosaur”, 1914[1], and has since grown into a very active area of interest with the popularisation of the entertainment industry. The reproduction of character movements was first achieved using traditional animation techniques, such as keyframing, where a key animator drew specific frames of the animation that defined important points. The remaining frames of the animation are subsequently drawn by inbetweeners based on the keys.

To aid the traditional animation techniques when used in the field of character motions, Max Fleischer introduced rotoscoping. Similar to the study of movement by Eadweard Muybridge, who used multiple cameras to capture the motions of animals and people [Muyb55, Muyb84], rotoscoping is based on the observations of consecutive frames of recorded real motions. Rotoscoping considers each frame of motion in turn and, by tracing the live action movements, the motions of artificial characters are recreated, thereby producing very lifelike motions. The technique was successfully used to produce many early cartoon animations including Betty Boop, Popeye and Superman.

Rotoscoping is still considered an effective method of extracting motion [Wagg04] or layering on special effects to that of a live action video, and the technology has evolved into bluescreen and motion capture techniques (mocap). Modern motion capture devices attempt to automate the process of extracting the motion from the real world using either markers (in the case of optical systems) or input sensors (for magnetic and mechanical systems), which are attached to the object or actor whose motion is to be recorded and tracked over time. More recently markerless motion capture has been used where the motion is recorded from live actors without the aid of markers and sensors. In mainstream capture studios, optical, magnetic and mechanical systems are currently preferred over markerless systems.

As opposed to animating with keyframes, rotoscoping considers the motion on a per-frame basis, which is the same as modern motion capture devices, where the postures of a real actor are recorded at sufficiently regular intervals to provide data for every frame of a motion. Effectively, rotoscoping and motion capture can be considered as providing keyframes for each frame in the animation. One of the biggest differences between rotoscoping and motion capture is that the latter captures complete 3-dimensional information from the actor, whereas rotoscoping only represents the 2-dimensional view from which the picture is taken (although this difference is being somewhat eroded with recent developments [Groc04]). The acquisition of 3-dimensional data is an important development in computer character animation, where models are created and postured in 3-dimensional environments. Furthermore, the data captured from modern devices are much more accurate than rotoscoping and therefore depict the subtle movements within the gross motion of the character, thus advancing another step towards even more realistic moving virtual characters.

[1] Source: Wikipedia, http://en.wikipedia.org/wiki/Character_animation

Each of these predominate types of motion capture techniques (optical, magnetic and mechanical) introduce an invasive aspect to them, which, unlike rotoscoping and markerless motion capture, can inhibit the motion of an actor. Because optical-based motion capture systems are the most general and present the least amount of intrusion on the actor, they tend to be the preferred technology in modern times to capture human movement for the use in video games and film special effects. However, the raw output from optical motion capture devices requires the most amount of post-processing to structure the data into a usable form which can be used to animate a virtual character. One contribution of this thesis is a novel technique that reliably converts positional marker data into a hierarchical data structure that can be used to animate a skinned computer character.

The ability to capture very realistic motions from human actors is the big appeal of motion capture devices. However, this is also where the main problems of using such a technology are manifested. When a motion has been recorded from a live actor, it is very desirable to reuse that motion as much as possible, especially when considering the expense and time required to record new motions. However, motion capture reuse is not a trivial problem for two reasons: the high amount of data produced and how to actually realistically modify a motion.

Due to the high sampling frequencies used to capture the actor's subtle gestures and movements, there is a huge amount of data that becomes impractical to manually adjust for anything but simple cleanup operations, especially when it is vital to maintain the subtleties of the original motion. Therefore techniques are required that allow an animator to more easily edit a motion without considering each joint orientation of each frame within the motion.

With tools to make the editing of high-density motion captured data easier the problem of how to modify motions still persists because any modifications to an existing motion should still appear realistic. One of the most basic types of modifications arises because of the dissimilarities between the real actor and the target virtual character, which result in the virtual character not correctly interacting with its environment. This is called retargetting [Glei98a] and used to reassert any incorrect interaction of, for example, the feet or hand positions.

Motion capture data modifications can be classified into three types: reconfiguration, adapting and additive. Reconfiguration includes the process of retargetting and extends it to include the ability to individualise the motion to take into account the build of the target character, i.e. a larger character would be expected to move differently to a smaller character. Adapting motion capture data is concerned with looking at ways of blending together multiple motions to give a new movement and hence make better use of existing motions. Additive motion capture modifications are concerned with introducing a new effect within an existing motion that was originally not present, for example to simulate an injury or respond to an environmental influence in a physically plausible way. The principles behind this terminology are further reviewed and explored in Chapter 2.

The area that is the primary focus of this thesis extends the concept of retargetting characters to include individualisation, which styles the resulting motion. Whereas retargetting ensures that the virtual character's interaction with the environmental is spatially correct, individualisation recreates variances between different physical builds of characters performing the same motion. For example, the naïve reuse of recorded motions results in all the characters moving in a visually identical manner, regardless of their biomechanical definition, whereas individualisation produces subtly different motions for each character, thereby affording extra depth to a character's motion. Complete reconfiguration is thus achieved when both the aims of retargetting and individualisation are met.

Character individualisation has previously been attempted [Urta04, Hsu05, Liu05], however each of these techniques requires a sample motion from the actor who is the target of the individualised motion. In contrast, this thesis presents two different novel techniques that allow the motion capture data from one actor to be mapped to that of another actor based only on the target actor's biomechanical information. The first of these techniques is based on a real-time inverse kinematics solution and an indirect interpretation of a character's biomechanical data. The second approach makes use of a rigid body dynamics generation process, which directly considers the biomechanical structure of the target character.

In addition to presenting reconfigured motions using both inverse kinematics and dynamics-based solutions, additive modifications are demonstrated using the same algorithms. The additive motion that is considered in this thesis demonstrates the ability to simulate an injury into an existing motion capture clip that previously illustrated no such infliction.

1.1 Thesis Structure

A review of motion capture hardware technology starts Chapter 2 by contrasting the main types of data acquisition devices. The standard hierarchical structure that the motion data is usually converted into is subsequently presented along with the process of how the data is used to visually animate virtual characters via skinning. The chapter continues with the presentation of a new technique that can be used to convert the positional marker data from optical-based devices to the standard hierarchical data structure, which is based on the inversion of the skinning algorithm. Chapter 2 concludes by reviewing the current start-of-the-art in the field of adapting existing motions, which further elaborates on the need for modifying them, thus defining the problems that the algorithms of this thesis address.

Chapter 3 presents a collection of 4 different-sized motion-captured actors, each performing sets of similar motions. These motions are used throughout this thesis to demonstrate and evaluate the techniques presented.

In Chapter 4, the first of the novel motion modification techniques is discussed. This focuses on the way in which motions can be kinematically adjusted through the application of inverse kinematics (IK). During this chapter a review of the mathematical concepts and techniques that have previously been used in the area are presented. Thereafter, an innovative interpretation of the Jacobian-based inverse kinematics is presented in terms of the half-Jacobian, which assists in reducing computation costs compared to the traditional approach. A further extension to the optimised inverse kinematics is subsequently described, called weighted inverse kinematics. Using weighted inverse kinematics it is possible to yield more control over the outcome of the solver by placing a bias towards a particular solution configuration. The effect of this added control allows many different motions to be spawned from a single example movement in which the generated motion portrays different styles. The weighted inverse kinematics-based technique has the ability to generate a motion similar to that of a real actor using the motion of a completely different actor and an inverse kinematics weighting vector – no physics or biomechanical information are exploited in the making of these motions. Chapter 4 also explores how injuries can be simulated into the resulting motion using weighted inverse kinematics. The inverse kinematics techniques of Chapter 4 do have limitations, which are subsequently addressed in Chapters 5 and 6.

In Chapter 5 the concept of dynamics for modifying existing motions is reviewed. This starts with a mathematical review of the rigid body dynamics that are utilised in an optimisation-based process to alter existing motions in a physically plausible manner. After the dynamics mathematics review, the work that has previously been conducted in this area by other researchers is discussed. Thereafter, Chapter 5 discusses some of the considerations that are necessary for tuning the theoretical physics into a practical solution, whereupon novel contributions to the design of the overall algorithm are made.

Chapter 6 discusses the potential of applying the dynamics-based system of Chapter 5 to the field of motion capture reconfiguration. This demonstrates the unique ability to accurately transfer the motion from one actor to another using the biomechanics of the target character. This is similar to the work presented in Chapter 3 for the weighted inverse kinematics algorithm, however the results demonstrated by the dynamics process show more realistic results because of the more accurate model used. Furthermore, the biomechanical-based motion reconfigurations are evaluated for correctness by comparing the dynamics-based reconfigured motion for the target actor against their real motion. Through the further exploration of the capabilities of the dynamics modification technique in Chapter 6, it is shown how injuries can be simulated into an existing motion that portrays none. The theory behind dynamics-based injury simulation is much the same as that shown for the inverse kinematics technique in Chapter 4, however implemented in a very different manner because of their very different approaches to modifying motion capture data.

Chapter 7 compares the two different forms of motion modification that this thesis has introduced, i.e. between inverse kinematics and dynamics algorithms. The comparisons between the two techniques focus on motion retargetting and full reconfiguration, primarily comparing their accuracy and realism. Based on the comparative advantages of the two techniques, this chapter suggests applications in which each technique is best suited.

The conclusions of this work are presented in Chapter 8.

1.2 Thesis Contributions

The novel contributions introduced in this thesis include:

- A new method for processing the raw marker position information from an optical motion capture device into an animated hierarchical data structure, which can then be used to animate a computer character.

- An analytical and empirical comparison between the Jacobian-based inverse kinematics technique, with and without an orientation component, is undertaken, which this thesis terms full- and half-Jacobian respectively in recognition of their respective matrix sizes. This leads to the novel introduction of specific constraints to convert a traditional full-Jacobian problem into the domain of a half-Jacobian solution and hence benefit from the computation speed up. This work has been published in [Mere04a].

- A novel weighting vector is introduced into Jacobian-based inverse kinematics to give Weighted Inverse Kinematics. This inclusion affords the ability to reliably control the rate of change along the inverse kinematics chain. The visual manifestation of this work results in a novel method of individualising (or reconfiguring) a character’s movements. By adjusting the weighting vector, the appearance of injuries can also be simulated. This work has been published in [Mere04b, Mere05], where procedural models of motion are considered as well as motion capture data.

- The weighted inverse kinematics is used to reconfigure the motion of one actor to another using a weighting vector based on the biomechanics of the target actor. The evaluation of the process is achieved by comparing it against the real motion of the target actor.

- The implementation of a dynamics-based optimisation algorithm, which permits physically plausible motion modifications. The implementation of the system itself introduces methods of dealing with impulse and discrete occurrences within a continuous domain, and hence contact and friction. Furthermore, the issue of ill-resolutioning within the system representation is highlighted and addressed.

- Using the dynamics-based optimisation process, the motion from one actor is successfully reconfigured to another using biomechanical information. This is substantiated through an evaluation of the technique that compares the simulated motion with the real movement of the target actor.

- The dynamics-based optimisation process is demonstrated to simulate injuries into the example motion, using the innovative process of restricting muscle forces and adjusting inter-muscle ratios.

Chapter 2:
Motion Capture Data and its Applications

With the aid of motion capture techniques, where a natural motion is captured directly from a real-life actor, much of the laborious posture configuration is eliminated from traditional keyframing. Once an initial calibration process is undertaken, hours of activity can be quickly and easily recorded, with frame rates up to 2000 fps. This effectively provides complete sets of keyframes at such a high resolution that there is no need to interpolate in-between, and if anything frames are dropped during playback.

Unfortunately, the process of capturing the motion from a real world actor, or object, and mapping it to a computer environment is not a straightforward process. Usually, large amounts of data processing are required. The motion capture process can be summarised into two categories: the first is to capture the raw data, while the second is to present this data in a meaningful structure. The data acquisition stage is described in the section 2.1 for the predominate kinds of motion capture technology, along with a brief description of what a meaningful structure for the resulting data may look like. Section 2.2 demonstrates how the structured data from motion capture devices are used in the process of skinning to animate the meshes of virtual characters. This section further serves as a mathematical basis for the novel data conversion process to convert the raw optical marker positions into a suitable structure, which is described in section 2.3. The process described in section 2.3 performs the exact opposite of the skinning algorithm, which gives the novel algorithm the name inverse skinning.

When a structured dataset is obtained from the motion capture process, it may still be desirable to adjust these motions. The possible types of motion modification are classified in Section 2.4. This is followed in section 2.5 by describing a collection of mathematical techniques that can be employed in the process of many of the different areas of modifying motions. The techniques of section 2.5 are subsequently linked back to the types of modifications (section 2.4) in section 2.6 by providing a review of the previous work that has been undertaken in the application of modifying motion capture data. The review of the current state-of-the-art techniques in section 2.6 further highlights some of the areas lacking in suitable motion modification techniques. This provides the grounding for the novel modification techniques presented through the continuation of this thesis and outlined in the summary of section 2.7.

2.1 Motion Capture Data Acquisition

Motion capture technologies work by tracking the positions and orientations of sensors, which have been strategically placed on real-world objects, over time. There are several types of sensory devices that can be used to capture this information, however the predominate technologies of modern motion capturing fall into one of 3 categories: optical, magnetic or mechanical.

Optical capture devices track the motion of real objects through the use of small markers that are attached to the tracked body, which reflect back infrared light that is emitted and captured by high-resolution cameras. Figure 2.1a and Figure 2.1b illustrate the markers and cameras used in optical motion capture, where potential marker placements are illustrated in Figure 2.1c and Figure 2.1d. Given the camera inputs, it is then the job of the capture software to triangulate the markers in space and produce a data stream of positional coordinates for each marker.

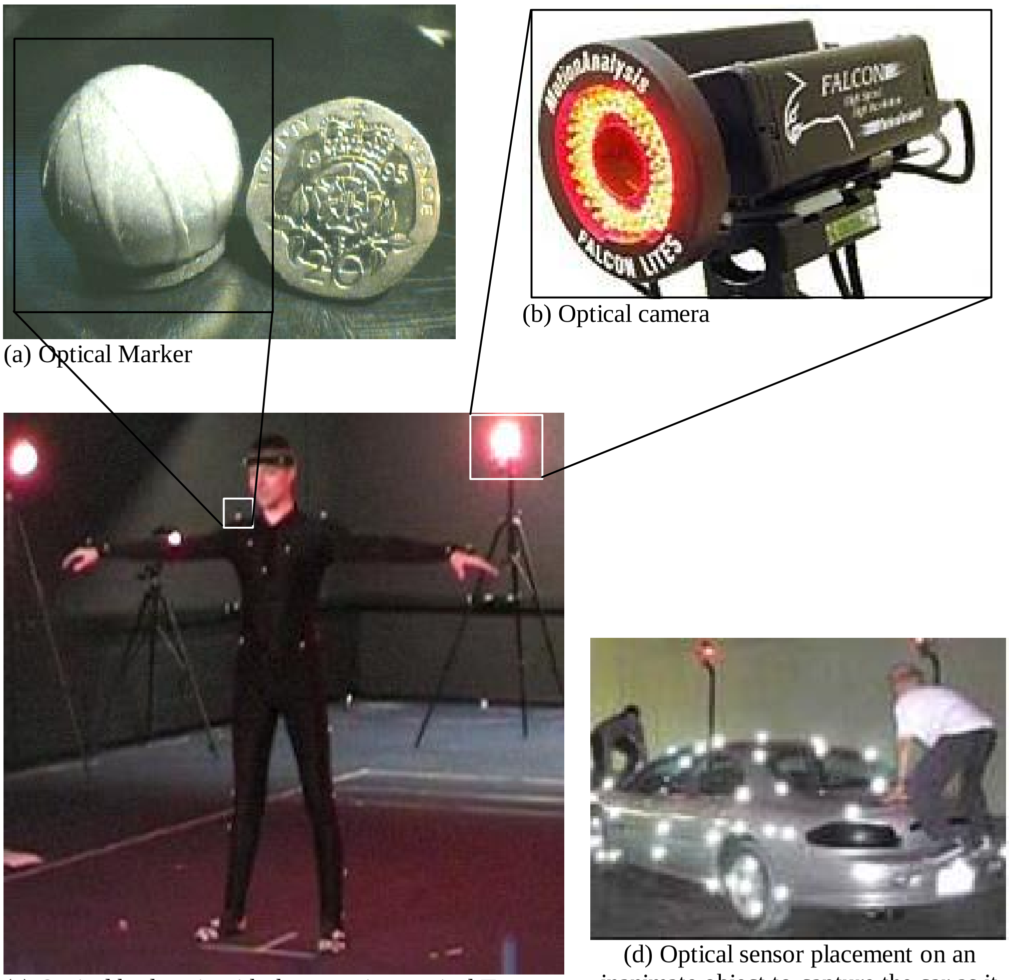
In the case of magnetic devices, the sensors used are sensitive to polarised electromagnetic fields that are emitted from a central transmitter. When the sensor readings are conveyed back to the software, they are converted into location and orientation metrics, however this requires a degree of cabling to connect the sensors to the computer. This is achieved by threading the individual sensor cables into a special suit, such as that illustrated in Figure 2.2, which are centrally collated, usually in a backpack worn by the actor, and transferred to a computer through either a central cable or wireless technology.

Unlike both optical and magnetic devices that rely on an emission and detection process, mechanical capture devices measure angular and positional differences between mechanically connected points. This is accomplished using a system of styluses that are fixed at specific locations on an object, which is illustrated in Figure 2.3 for a human actor. However, the styluses introduce a more intrusive capture than either optical or magnetic devices and are also less flexible with regards to what they can be attached to.

Once the actor (or object) has been suited up with markers or sensors, there is a degree of initial calibration required before the captures can commence. In the case of optical system, this involves calibrating both the position of the cameras and also the marker locations on the body. The former of these steps is only required when the cameras are moved. In order to identify markers in the scene, at the start of each actor’s capture session they assume an agreed base pose, such as that illustrated in Figure 2.1c, and performs a range of motion cycle. The resulting posture and motion data is thereafter used during a post-processing phase to help distinguish between markers and to create a hierarchical data file that records the animation details such as joint length, offsets and angles.

Similarly, magnetic systems also need to be calibrated when first installed with the aim of compensating for any magnetic interference in the area. Once this process is done, since the receivers are clipped onto the magnetic body suit and hence assume a fixed location, no further calibration needs to be done. Furthermore, each sensor is uniquely identified through its cable connection, which eliminates the T-pose calibration step as well as reduces the post-processing demand of differentiating between markers as in the case of optical systems. Conversely, mechanical systems require virtually no calibration because the styluses movements can be directly measured without the fear of interference and because each sensor is uniquely identifiable, there is no post-processing required.

Although there are no interference problems for mechanical devices, both optical and magnetic devices reply on a transmitted signal and are therefore are more prone to erroneous data. Optical devices are more susceptible to error than magnetic devices because they rely on markers being visible to the cameras, which may not always be the case, thus resulting in an additional occlusion problem. However, the introduction of additional cameras to capture the scene can help to reduce the problem of marker occlusion.



(b) Optical camera

(a) Optical Marker

(c) Optical body suit with the actor in a typical T-pose posture

(d) Optical sensor placement on an inanimate object to capture the car as it bounces up and down

Figure 2.1: Optical Motion Capture Hardware; (a) marker – *Courtesy of Infogrames, UK*, (b) falcon camera, (c) body suit with marker placement, (d) inanimate object with markers: Images b & d are *Courtesy of Motion Analysis Corporation*

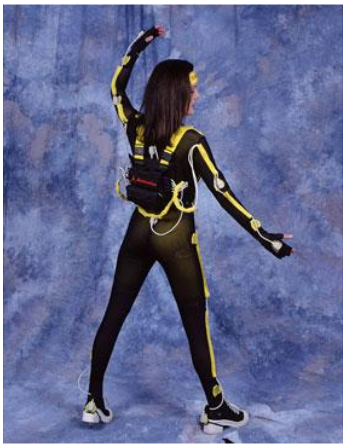


Figure 2.2: Magnetic Cyber-Suit for magnetic motion capture, *Courtesy of Ascension Technology*

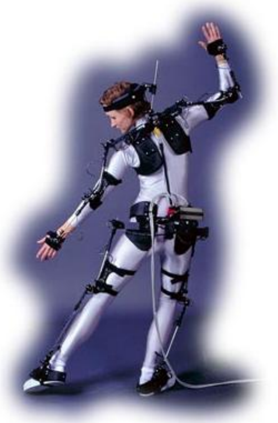


Figure 2.3: Mechanical Gypsy body suit, *Courtesy of Animazoo*

The three main types of motion acquisition are all popular because the disadvantages of one device are complemented by advantages of another and so each type of device has its own niche. For example mechanical devices are extremely well suited to real-time puppetry, while optical devices are more suited to capturing natural, unrestricted object interaction. Table 2.1 provides a comparison of these devices over some key aspects of motion capturing.

Motion Capture Device Type	Optical[2]	Magnetic[3]	Mechanical[4]
Maximum Performance Area	*20m x 20m x 10m*[5]	*Radius of 3m (single transmitter)*	*½ mile (outdoors)* *180m (indoors)*
Maximum Frame Rate	*2000 fps (only 484fps at full resolution)*	*120 fps*	*120 fps*
Maximum Number of Tracking Sensors/Markers	*500+*	*90*	*20*
Real-time Playback	*At the lower end of capture frame rate*	*Yes*	*Yes*
Relative Cost	*High*	*Medium*	*Low*
Sources of interference	*Light sources & other reflective objects*	*Metallic objects*	*None*
Relative level of intrusiveness	*Low*	*Medium*	*High*
Flexibility in capturing different types of objects	*High*	*Medium*	*Low*
Relative Calibration Required	*High*	*Low*	*None*
Relative amount of post processing	*High*	*Low*	*None*

Table 2.1: Comparison of key aspects of motion capture devices

The devices that have been discussed thus far all require expensive hardware to capture the motion so increasing work has been made toward capturing motion from more basic devices such as off-the-shelf home video cameras. Although such techniques are still technically an optical-based system, they were excluded them from the earlier discussions because in many cases they work by tracking the silhouette of a character [Wagg04] or specific feature points [Zhao04] as opposed to markers. Unfortunately, despite the promising results that have been demonstrated, their lack of specialised hardware and their relative immaturity at this time has had a detrimental affect on the accuracy of the results produced in comparison to the other techniques. Subsequently, such techniques are not currently used in capturing high-fidelity motions.

Despite being the most expensive motion capture solution, in both the gaming and movie industry the optical medium tends to be the dominant device because of its non-intrusive hardware. Optical devices also provide better handling of inanimate object interaction since the markers can be placed on virtually any object unlike the sensors required for magnetic and mechanical devices.

[2] Optical device details are taken from ViconPeak, http://www.vicon.com, 2005

[3] Magnetic device details are taken from Ascension Technology, http://www.ascension-tech.com, 2005

[4] Mechanical device details are taken from MetaMotion, http://www.metamotion.com, 2005

[5] Optical maximum performance area is dependant on the number of cameras, however a 10m distance pickup is what is suggested by Phasespace, http://www.phasespace.com, 2005

2.1.1 Motion Capture Data Handling

Once the raw data has been obtained there is normally a large amount of time dedicated to post-processing that data especially with optical capture devices. The post-processing stage often requires the repositioning of marker points such that they smoothly flow through time, between individual frames and thereby helping to eliminating any erroneous marker or sensor readings. Teams of skilled artists perform this job and depending on the degree of noise present in the raw data, it can take significantly more time than the capture itself even with the aid of tools such as FilmBox[6].

Given the cleaned marker data, it is normal to represent and store the result in a hierarchical data format, which is especially useful if the motion is to be subject to further modifications. A hierarchical (or articulated) figure consists of a series of limbs that are connected though joints, where the length and direction of the limb are defined locally with respect to its immediate parent limb. Each limb in turn inherits its parent's orientation, which will eventually result in a global position when the last parent limb is the root limb, as illustrated in Figure 2.4 for a humanoid hierarchical structure. A novel technique for mapping 3D marker points into a hierarchical structure is presented in section 2.3.

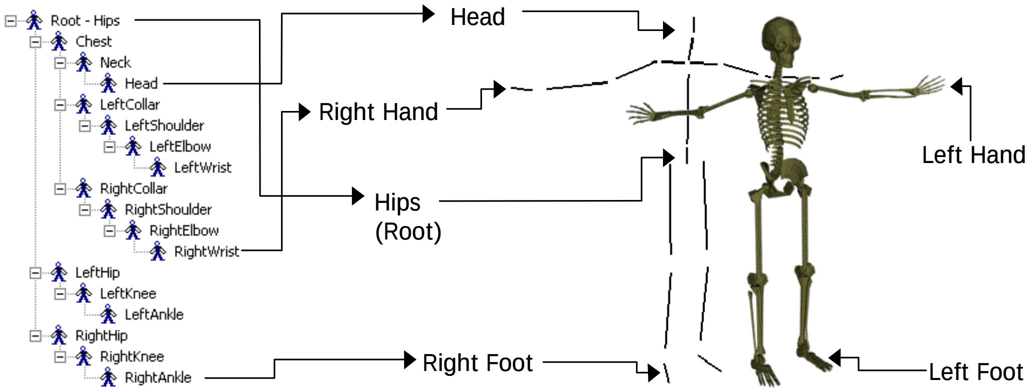


Figure 2.4: Hierarchically Defined Humanoid Character

The transformation of raw mocap data into a hierarchical format imposes a very rigid structuring and many motion capture houses have their own way of representing this data within a file. For example, some formats include a base pose that is altered with additional frame data while others just have absolute transformations and the measurement units are rarely the same across different file formats. A review of some of the more predominate motion capture file formats is presented in [Mere01], which further explains how to decode specific formats.

Figure 2.5 illustrates an example of the complete process for the motion capture data of an actor jumping. The gold spheres represent the original optical marker data, whereas the colour hierarchical character is the result of performing the post-process stage to obtain an animated articulated structure.

[6] FilmBox, Kaydara Inc. FilmBox http://www.kaydara.com

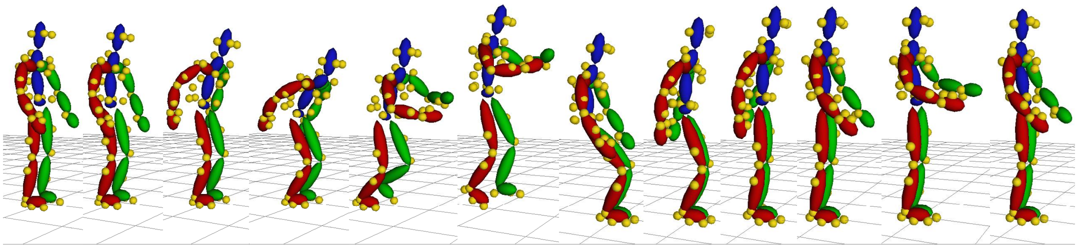


Figure 2.5: The result of extracting an animated hierarchical structure from optical motion capture marker locations 🎥

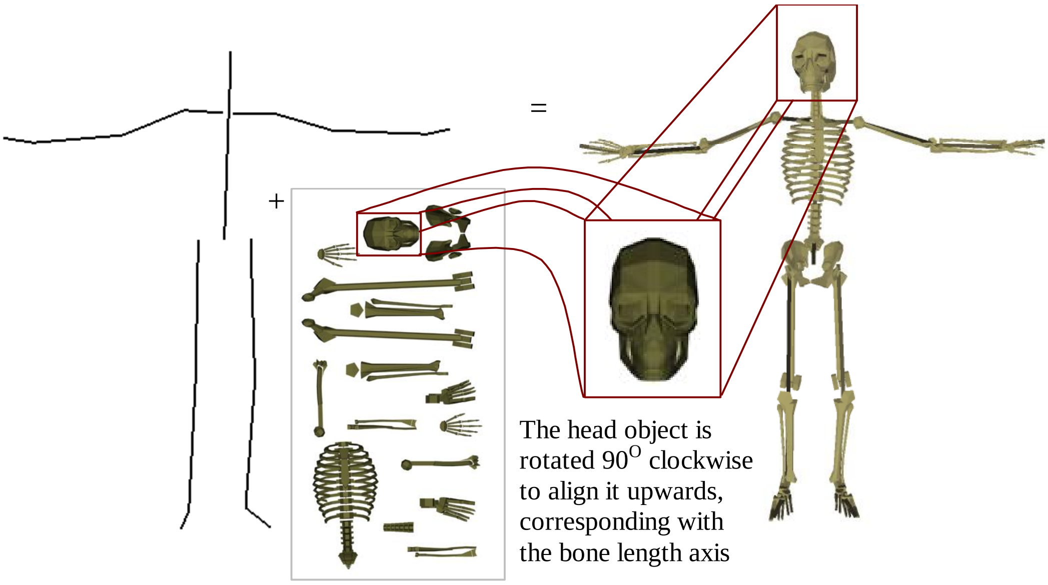
2.2 Character Skinning/Playback

The display of a virtual character is normally achieved by rendering a polygonal mesh. Instead of directly manipulating this representation to produce the animated mesh, the standard approach abstracts away the virtual representation to an articulated endoskeleton. Specifying joint angles from motion capture data, for example, subsequently animates the endoskeleton. However, the hierarchical structure needs to be mapped to the polygonal mesh so that it takes on an appearance as if there were a real endoskeleton underneath the mesh deforming the body. The term “skinning” is used to describe this process [Watt03]. Three different approaches to this will be presented in the following subsections.

2.2.1 Object Mapping

Starting from a basic articulated data representation of the animation, the easiest and most obvious process of skinning it is to attach an independent object to each of the hierarchical nodes, where each object is defined with respect to its local coordinate system. This process assumes that each object is correctly aligned to the bone direction of the articulated structure, where the frame of reference for each is taken to be their local coordinate systems. For example, if in the hierarchical structure all the bone lengths are measured along the y-axis, then you would have to provide transforms for each object such that when multiplied by their local reference frame, the length of the 3D object also aligns along the resulting y-axis. An example object mapping skinning is illustrated in Figure 2.6.

The process of object mapping to skinning a character is the simplest form and requires each hierarchal node to have an associated 3D object that is independent from the rest of the body parts. Consequently, by directly mapping each object to a node allows us to directly apply the same joint rotations to the 3D objects without any further work (with the exception of the small amount of pre-processing required to align each of the 3D objects, but this is a one-off process per skin model). Using the mapping of Figure 2.6 as a basis for the pre-processing, the resulting animations produced using such an approach are illustrated in Figure 2.7.



The head object is rotated 90^{O} clockwise to align it upwards, corresponding with the bone length axis

Figure 2.6: Skinning via Object Mapping

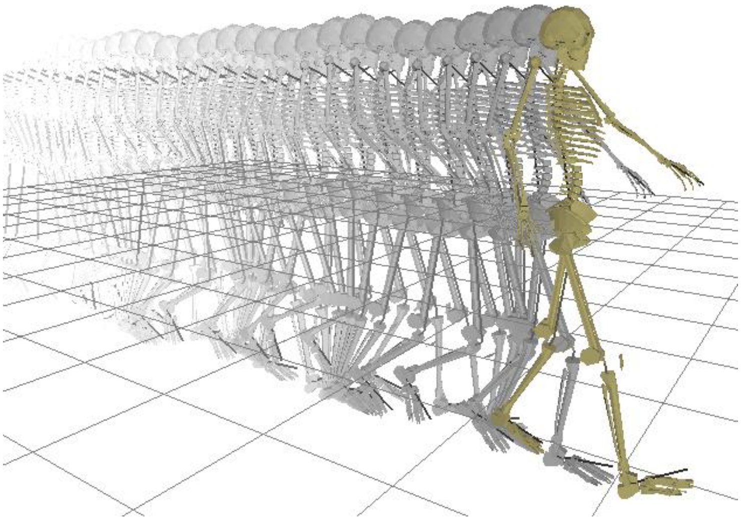


Figure 2.7: Animation using Object Mapping Skinning. The lighter grey skeleton figures are earlier frames of animation than the darker grey and coloured characters

Despite its simplicity, the method of mapping objects to articulated nodes has the fundamental drawback of requiring the whole model to be broken down and represented as discrete object parts, where in some cases, this proves impractical, especially when joint cut-offs are not easily defined. The approach is therefore more suited to models that have well defined nodes, such as skeletons and mechanical looking robots. However, in simplified cases, the process of segmenting the model into independent objects can result in holes being left in the mesh once it is divided up. Consequently, when we have a character’s skin defined with a mesh that is either a complete object, or made up of parts that do not easily break down into the independent objects, we turn to more general techniques to perform the skinning.

2.2.2 Simple Skinning

This section considers a basic algorithm that provides a technique for animating a single mesh object that represents the complete character. There is no lose of generality by assuming there is only one meshed object to represent the entire body because a collection of objects can easily combined together to form one global one, and it is this global model that is of interested. Consequently, this process provides a more general approach to skinning, which encapsulates object mapping, and as the results will shortly demonstrate, effectively give the same effect, but with the absence of mesh gaps.

The starting elements are therefore a single object mesh that is defined in a local coordinate system and a hierarchical animation representation, which is also defined in a local coordinate system. These two entities need to relate. The first step that is taken towards this goal is to fit a similar hierarchical structure to the mesh, effectively giving the object a pseudo-endoskeleton, which for the moment is disjoint from the mesh itself and “floats” inside it. Figure 2.8 exemplifies this process on the skin of a skeletal model, where the hierarchical pseudo-endoskeleton is represented as sphere-connected lines.

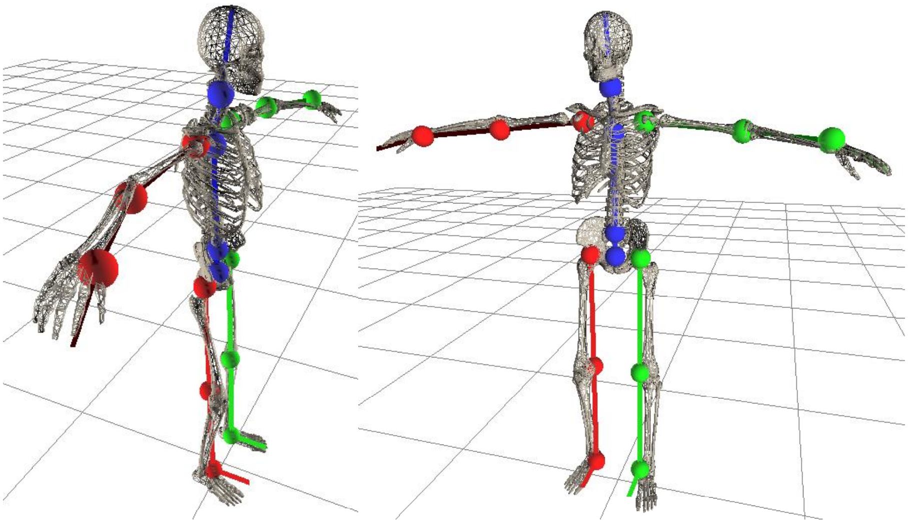


Figure 2.8: Creating a pseudo-endoskeleton that fits a meshed object, illustrated with the red, green and blue lines connected with sphere joints

The second stage in this process is to associate the mesh object to the pseudo-endoskeleton. This is achieved by associating mesh vertices to the pseudo-endoskeleton nodes, however this is not quite as simple as finding the closet node for a given vertex, especially when dealing with the upper torso area. Therefore it is useful when fitting the pseudo-endoskeleton to the mesh, to give the articulated bones dimensional information. This allows the construction of an influence box around each bone and any vertices falling within the volume belong to it. The vertices that completely lie outside the bounding boxes of all the bones are simply mapped to the closest bone. It should also be noted that for this incarnation of the skinning algorithm, each vertex can only attach itself to one bone. An illustration of the vertex attachment is given in Figure 2.9.

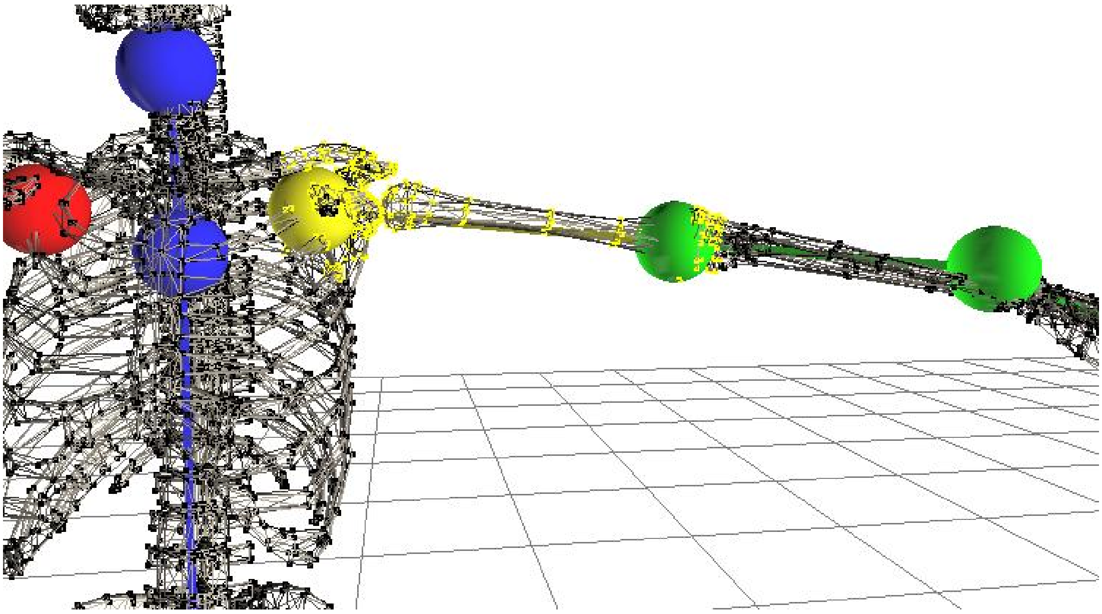


Figure 2.9: Relating mesh vertices to the underlying pseudo-endoskeleton. The yellow sphere illustrates the selected bone and corresponding line segment where the attached vertices are highlighted yellow and the non-attached vertices are black

Once the mesh vertices are attached to the pseudo-endoskeleton, the two hierarchical representations need to be related. Assuming that each node in the animated hierarchy is attached to a node in the pseudo-endoskeleton, the animation of the pseudo-skeleton can be achieved by mapping across the joint rotations. Subsequently, a mathematical relationship between the vertices of the i^{th} bone in the hierarchy, V_i, and i^{th} joint transformation, M_i, of the animation can be defined as indicated in Equation 2.1, where V_i*'* is the location of the new vertices.

$$V'_n = \left(\prod_{i=0}^{n} M_i \right) V_n \qquad \textbf{(2.1)}$$

Equation 2.1 states that the location of the vertex to be rendered is calculated by the compound joint rotations from the current node to the root, and some base location of the original vertex. The original vertex location on the mesh cannot be taken as the base location because, from Equation 2.1, it is clear that each vertex needs to be defined locally with respect to the attached bone, in much the same manner as the object mapping approach was.

However, there are two hierarchies defined; an animated one and a base posture used to map vertices to a pseudo-endoskeleton, where Equation 2.1 provides a general formulation for mapping vertices between local, V_i, and global space, V_i*'*. For the hierarchy to be animated, the vertices need to be defined in local space (Equation 2.1), however the pseudo-endoskeleton presents the reverse scenario. In this case, the hierarchical structure and the global vertex positions are known, therefore by reversing Equation 2.1 a derivation for the local vertices with respect to the global ones can be obtained. Equation 2.2 illustrates the reverse of Equation 2.1, where the original mesh vertices for the i^{th} bone are labelled V_i*''* and the joint rotation matrix of the pseudo-endoskeleton as B_i (referred to as the binding matrices).

$$V_n = \left(\prod_{i=n}^{0} B_i^{-1}\right) V_n'' \tag{2.2}$$

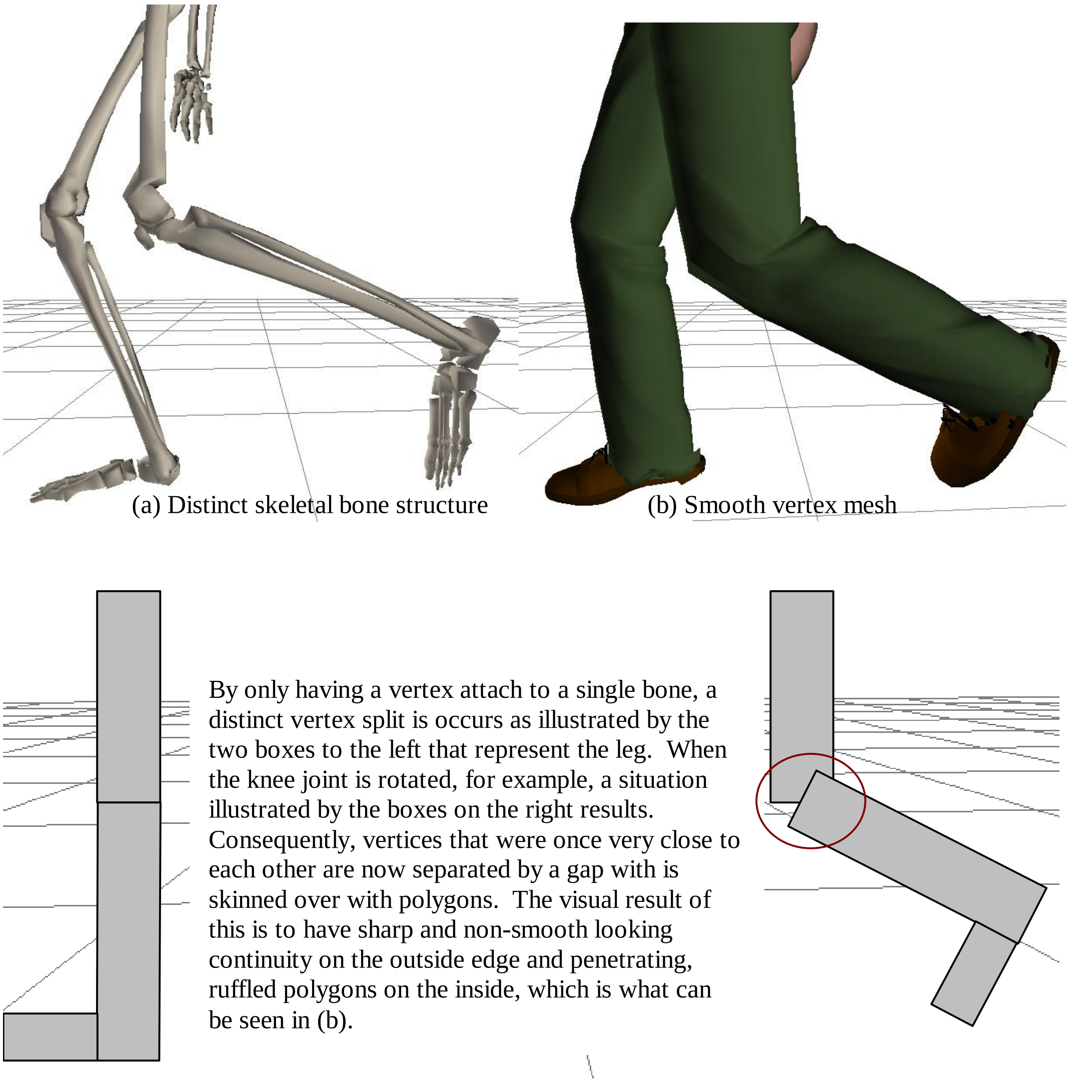
By substituting Equation 2.2 into Equation 2.1, where V_i represents the intermediate local vertices, a relationship between the original mesh vertices, V_i'', and the animated hierarchy is defined by Equation 2.3.

$$V_n' = \left(\prod_{i=0}^{n} M_i\right)\left(\prod_{i=n}^{0} B_i^{-1}\right) V_n'' \tag{2.3}$$

From an efficiency point of view, the joint rotation matrices of the pseudo-endoskeleton are constant over the animation and therefore can be pre-calculated. Consequently, in continued discussions of the binding matrices, the product formulation is simplified to B_i, which represents the transformation matrix product up to the *i*th bone. Similarly, the product animation transformation matrix for the *i*th bone will be referred to as M_i.

The formulation developed in Equation 2.3 is equivalent to dealing with independent rigid objects and once the pre-processing is done to define the binding matrices, both techniques demonstrate comparable result in terms of complexity. The only real difference between the two approaches is that because the model is not broken into separate objects, no visual gaps appear within the mesh. However, a shortcoming of these mathematically similar techniques in that we get visual distortions around joints, which are illustrated in Figure 2.10.

Figure 2.10 shows that because each vertex is attached to only one joint, when the joint bends, the vertices follow the path of one or other of the bones and hence there is no natural stretching about that region. This serves to demonstrate the limitations of the technique, which work fine for characters that have defined body parts that do not overlap, such as robots or the skeleton model in Figure 2.10a, however less suitable when it comes to modelling flowing meshes, such as human skins. Consequently, each vertex should be influenced by more than a single bone in the hierarchy, which leads to the concept of smooth skinning.



(a) Distinct skeletal bone structure

(b) Smooth vertex mesh

By only having a vertex attach to a single bone, a distinct vertex split is occurs as illustrated by the two boxes to the left that represent the leg. When the knee joint is rotated, for example, a situation illustrated by the boxes on the right results. Consequently, vertices that were once very close to each other are now separated by a gap with is skinned over with polygons. The visual result of this is to have sharp and non-smooth looking continuity on the outside edge and penetrating, ruffled polygons on the inside, which is what can be seen in (b).

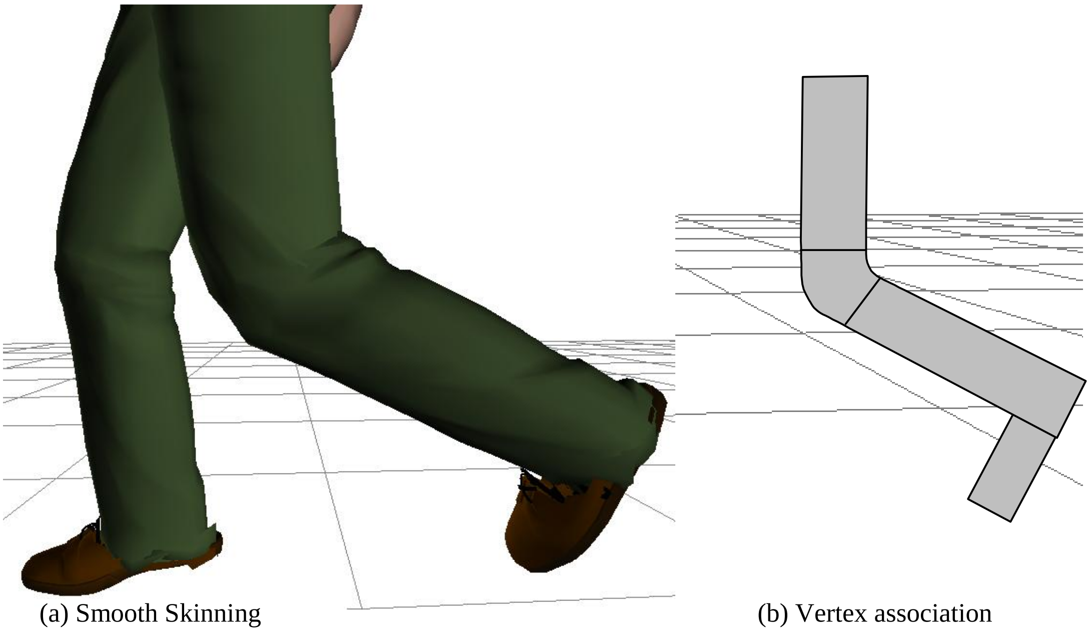
Figure 2.10: Visual distortion of simple skinning. When applied to a mesh that has distinguishable breakpoints (a), no distortion is present, however with a flowing mesh (b), the result of applying simple skinning is to introduce sharp edges and distortions at the joints

2.2.3 Smooth Skinning

The principle of smooth skinning advances the work of simple skinning. The starting point is a single mesh that completely models the character, where a pseudo-endoskeleton is put through it and a mapping between the two hierarchical structures is defined. However instead of insisting that each vertex can only be attached to one bone, this restriction is relaxed so that a vertex can be associated with many bones. This affords extra flexibility by having a specific vertex influenced by multiple bones, which was the cause of the distortions using the simple skinning algorithm. Equation 2.4 presents a modification of Equation 2.3, where the new vertex location is given by the summation over all the hierarchical bones (the representation of the original mesh vertex location in Equation 2.3 is changed from *v''* to *v* in Equation 2.4 for clarity).

$$v' = \sum_i w_i M_i B_i^{-1} v \qquad \text{where} \qquad \sum_i w_i = 1 \tag{2.4}$$

For notational simplicity, Equation 2.4 sums over each bone in the hierarchical structure, even if there is no association with the given vertex, therefore in such cases, the weighting value will be zero. A practical implementation of Equation 2.4 would only sum over the associated bones, using the appropriate transformation and binding matrices for each specific bone. The application of Equation 2.4 is illustrated in Figure 2.11.

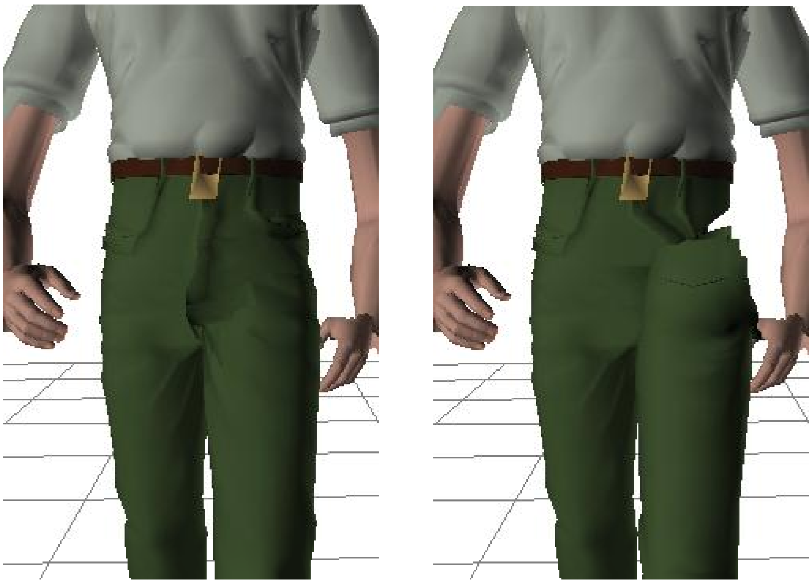

(a) Smooth Skinning

(b) Vertex association

Figure 2.11: An example of smooth skinning (a), where around the knee joint we allow each vertex to be attached to both the upper and lower leg bones which results in a smoothed region

The example of smooth skinning illustrated in Figure 2.11 demonstrates the advantage of such a technique over simple skinning, which can be seen by the increased continuity of Figure 2.11a over the simple skinned leg in Figure 2.10b. Simple skinning is actually only a special case of smooth skinning, where each vertex has only one bone attachment. However, the flip side to smooth skinning is the greater computational demands required to manually calculate the position of each vertex based on many bones. Fortunately, this can be offset through the utilisation of hardware graphics processing units and shader models [Watt03]. Determining which vertices actually have multiple bones associated and dealing with them separately can further contribute towards an efficient solution. For the vast majority of vertices, they will only have a single bone association.

One significant factor that does not initially bear out of the equations presented is the importance of defining consistent joint rotations and their orders for both the animated and pseudo-endoskeleton hierarchies. For example, assuming that the bone lengths are always measured along the y-axis, to rotate the upper leg into position, it is possible to rotate about either the x- or z-axis. However, if different axis for the different articulations are chosen, the resulting skin will appear twisted, as illustrated in Figure 2.12.



(a) Correctly matched joint orientations (b) Incorrectly match joint orientations

Figure 2.12: Twisting effects that can occur if care is not taken to maintain consistent joint rotations; (a) correctly matched joint orientations, (b) incorrectly matched joint orientation at the left hip, which results in a twisting effect in the mesh and produces visually distorted results

2.2.4 Summary

The skinning processes described above demonstrate an increasing complexity in the underlying algorithm used, which subsequently resulted in more visually correct looking characters. However, throughout the discussions, no mention has been given to any additional skin deformation that would present itself in the form muscles changing shape as they contract and relax and the appearance of tendons under strain [Karl98]. Although the addition of such properties would enhance the realism of the character, there is currently very little active research work in the area apart from that which commercial products support, such as 3D Studio Max’s Character Studio. Character Studio achieves the extra level of mesh deformation through the use of bulge angles and tendons, which, based on the underlying joint angles, deform the mesh by either bulging or shrinking areas on the mesh about the bone length.

2.3 Inverse Skinning

Optical motion capture devices record the locations of markers placed on a human actor as a way of tracking the motion of the actor. Furthermore, it has been mentioned that this marker set is transformed into a hierarchical representation of the character, but stopped short of providing any concrete basis for this process. In this section, a novel technique is introduced to perform just this operation. The process is called inverse skinning due to the starting point being a collection of spatial points positioned on the surface of a character and what is desired is an underlying complete endoskeleton representation. Posing the problem this way is exactly the opposite of the skinning process.

The process of converting marker data to articulated structures has been tackled in commercial applications that ship with the capture hardware itself, however the workings of such tools are kept secret. There has however been a couple of published works that discuss the process of dealing with motion capture data. Bodenheimer & Rose [Bode97] make use of an inverse kinematics solver to determine joint configurations based on marker locations. Zordan & Van Der Horst [Zord03] utilise a dynamics model to simulate joint trajectories using proportional-derivative servos to try and match the marker data. In the remainder of this subsection, a novel approach that can be used to construct an animated hierarchical structure from global spatial locations is presented.

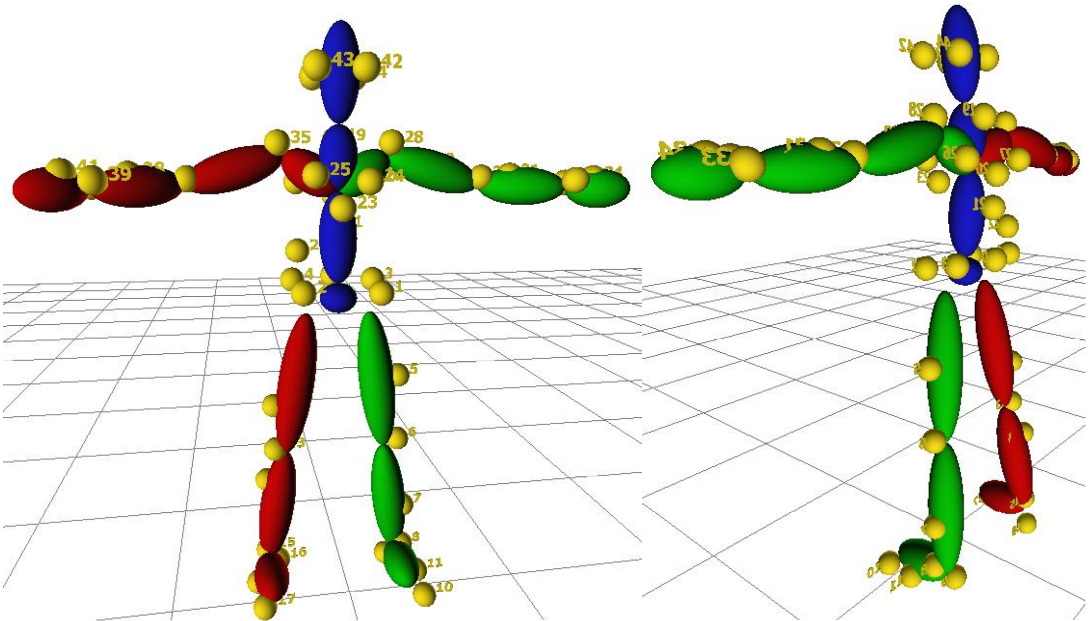
2.3.1 Reversing Simple Skinning

The inverse skinning method takes the stance that an animated articulated structure can be generated by considering the inverse of the skinning algorithms, where the marker locations are taken to be identically equal to the mesh vertices; effectively, the markers make up a very low-resolution mesh, which is deformed by the actor’s joint configuration. Consequently, going back to the skinning technique, the constructs that completely describe the system, including input and output are: 1) an original non-deformed marker mesh, 2) a pseudo-endoskeleton that passes through the non-deformed mesh, 3) an animated hierarchical structure, and 4) a deformed marker mesh based on the animated articulation.

The motion of the captured markers directly provide item 4. However, based on the fact that the markers are placed at specific locations on the body to aid tracking, it is also possible to factor out the basic articulated structure from the data in the form of bone lengths. This provides the basic structure and dimensions for both the animation and the pseudo-endoskeleton. The non-deformed base mesh object (item 1) can additionally be taken from the captured markers by taking a single frame of the motion, for which a pseudo-endoskeleton can be orientated through (item 2).

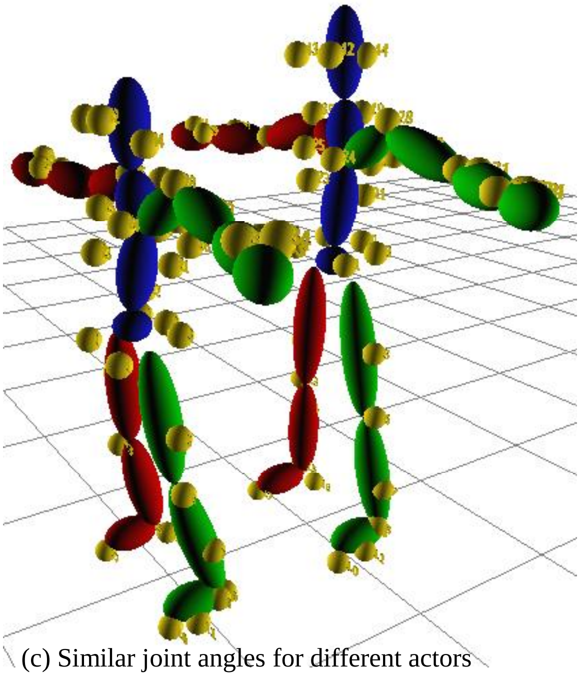
The process of obtaining item 1 and 2 determines the binding matrices and only needs to be done once per actor regardless of the number of motions being processed for them. The process can be made easier by defining the base mesh object using a T-pose posture from the original range of motion animation that is performed to calibrate the optical system for the actor. The consequence of this is that the joint configurations for the articulated structure will be very similar between actors, thereby requiring only a small amount of change for each actor. The pseudo-endoskeleton fitting of a marker mesh is illustrated in Figure 2.13.

The animated hierarchical structure (item 3) can be obtained by inversing Equation 2.3, which is based on the other three items which have been determined (items 1, 2 & 4). Based on the marker data that is converted using this technique, the inversion of the simple skinning approach is all that is required because each marker is associated with a single bone, as illustrated in Figure 2.13b. However, the method would equally well map to the smooth skinning approach, given a set of suitable weightings which would be defined at the time of constructing the pseudo-endoskeleton.



(a) Fitting a hierarchical pseudo-endoskeleton through maker data

Bone	Marker IDs
Pelvis	0, 1, 2, 3, 4
Left Femur	5, 6
Left Tibia	7, 8
Left Foot	9, 10, 11
Right Femur	12, 13
Right Tibia	14, 15
Right Foot	16, 17, 18
Thorax	20, 21, 22, 23
Neck	19
Head	42, 43, 44, 45
Left Clavicle	24, 26, 28
Left Humerus	29, 30
Left Radius	31, 32, 33
Left hand	34
Right Clavicle	25, 27, 35
Right Humerus	36, 37
Right Radius	38, 39, 40
Right Hand	41



(b) Marker/Bone associations

(c) Similar joint angles for different actors

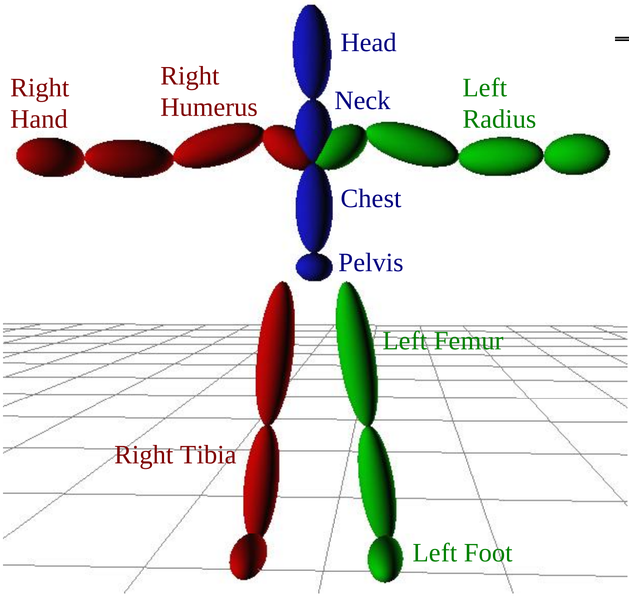
Figure 2.13: Fitting a pseudo-endoskeleton for a marker mesh (a), where the bones have the associated markers in (b). Similar postures of different size characters in the T-Pose are demonstrated in (c)

2.3.2 Solving the Inverse Skinning Problem

Unfortunately, because of the tracking error and accuracy of the optical system, the formulation of Equation 2.3 is unlikely to exactly hold, therefore a straightforward inversion would prove of little use. Furthermore, it is likely that due to the limited amount of markers used to track the body, the inverse problem is under-defined and hence has multiple solutions. Therefore, the problem is cast as an optimisation-based one in which the squared distance between the actual marker values and the predicted ones (which are defined by the current state of the optimisation process) are minimised. This is given by Equation 2.5, where o_i is the location of the i^{th} marker based on the capture data and v_i is the i^{th} marker from the non-deformed base mesh.

$$minimise\left[\left(\prod_{i=n}^{0} M_i\right)\left(\prod_{i=0}^{n} B_i^{-1}\right)v_n - o_n\right]^2 \tag{2.5}$$

The free variables of the system are the Euler rotations for each joint, which are used to construct the transformation matrix M_i. The complexity of the minimisation system can therefore be reduced by only including valid degrees of freedom (DOFs) within a human body. The standard DOF reduction of the human character used for this task, and indeed any subsequent DOF reductions in this thesis, are given in Figure 2.14. Furthermore, Figure 2.14 states the joint limits that are used in the study of human character animation, which are encode into the minimisation algorithm as linear constraints on the solution.



Bone	X	Y	Z
Pelvis	360	360	360
Left Femur	180	180	140
Left Tibia	0	0	120
Left Foot	90	0	90
Right Femur	180	180	140
Right Tibia	0	0	120
Right Foot	90	0	90
Thorax	30	10	30
Neck	20	0	60
Head	20	180	150
Left Clavicle	30	0	30
Left Humerus	180	180	180
Left Radius	0	0	160
Left hand	180	180	40
Right Clavicle	30	0	30
Right Humerus	180	180	180
Right Radius	0	0	160
Right Hand	180	180	40

Figure 2.14: Human character DOF reduction and joint ranges in the X-, Y-, and Z-axis given in degrees, where the Y-axis is the bone length axis and the X-axis comes out of the page

To solve the system of equations given by Equation 2.5 and the linear joint restriction constraints, the constrained optimisation process, given in section 5.3.1, is utilised. To effectively manipulate and represent the algebraic formulations of the inverse skinning process the PAMPERS system is used, which is described in section 5.5.1. For the purposes of clarity at this point, the constructional detail of both PAMPERS and the optimisation process is left to the more appropriate discussion in Chapter 5. All that is important at this stage is that the optimisation process takes the summation of Equation 2.5 over mesh marker points as its minimisation equation, and a set of constraints in the form joint angles, which are both mathematically represented using the algebraic symbolic system of PAMPERS. Given the input, the optimisation process returns a set of free variables that make up the joint angles, which solves the system of equations as best it can. The process is performed on a per-frame basis as opposed to considering the motion as one complete task.

From a mathematical point of view, it would naturally follow to combine Equation 2.5 over all marker points into a single minimisation function, thus covering the complete articulated structure. However, this produces a very complicated mathematical formulation, especially for the outermost nodes of the articulation. The opposite approach would be to take Equation 2.5 for each bone in turn, starting from the root and working outwards, thereby simplifying the product of transformation matrices, M_i, to a single level, namely the transformation for the particular bone of interest. Unfortunately, taking a piecewise approach reduces the accuracy of the resulting configuration, especially in low marker regions such as along the limbs of the body. Furthermore, due to the joint angle restrictions, it is possible to configure a parent node correctly but subsequently make it impossible for the child node to meet its target global configuration. For example, if the upper arm were rotated so that it aligns itself with its associated marker data but a rotation about the bone length were introduced, the possible locations that the lower-arm can take up is restricted because there is only one DOF available to be configured, which itself has joint range limitations. This is exemplified in Figure 2.15, where the upper arm is rotated such that the lower can only achieve movement that is effectively behind the character, illustrated by the red arc, and not the true location in front, as indicated by the gold markers. The upper arm is however positioned correctly according to its gold marker locations.

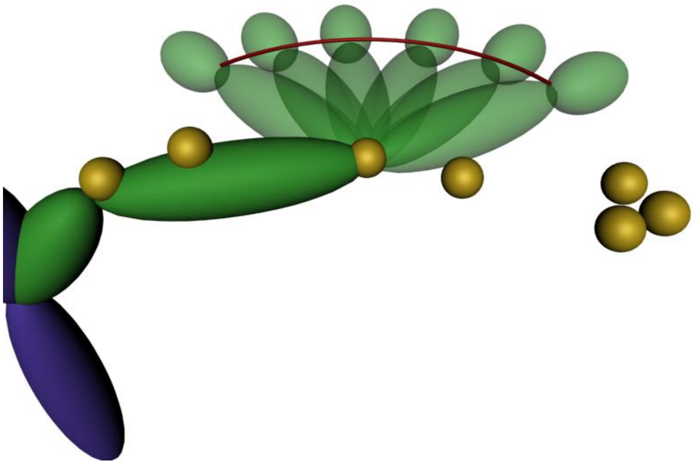


Figure 2.15: Example of a spatial restriction on articulated limbs when parent nodes are configured with no regard to the location of their children. In this example, the left humerus has been rotated backwards about the bone length so that the only possible configuration for the left radius is that marked out by the red path, where in fact we wish to align it on top of the gold marker points

Expressing the complete hierarchical structure in a single minimisation function is computationally costly. This can be reduced by taking a piecewise approach, but can result in poor results. By recognising that certain parts of the body do not affect the possible solution space of other parts a compromised solution can be derived. For example, the configuration of the left elbow in no way restricts the possible solution space of the right knee in the manner that was illustrated in Figure 2.15. Given that the character's hip (which is the root of the articulation) has a collection of 5 dedicated markers, the orientation and position is uniquely determined, which is solved first using the optimisation process (the hips could in fact be solved analytically, however it takes comparatively little time to compute using the optimisation process, therefore this slight optimisation is not made).

Once the hips are correctly positioned and orientated, the legs and the upper-body are the next hierarchically independent entities, which forms the second segmentation. From a hierarchical point of view, each leg can be considered as independent entities, thus giving a further segmentation between the two legs. Within the individual hierarchal chain of the legs, no further subdivision can be made without degrading the hierarchical fitting process, or running into the problems highlighted in Figure 2.15. Figure 2.16 illustrates the different appearance of treating each leg as a whole optimisation process as opposed to further breaking it down into individual bones, which results in the feet rarely being orientated correctly, and slight leg shaking when the foot is on the floor (this can be better seen in the included animation file).

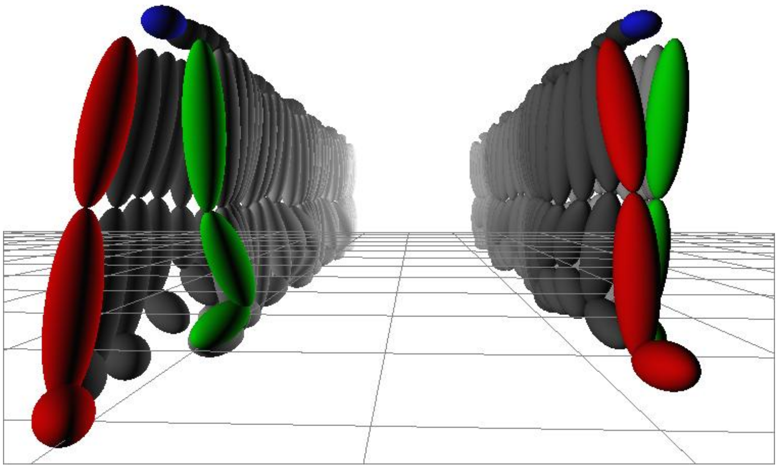


Figure 2.16: Comparison between solving the leg as a whole (right walk) compared to independent bones (left walk)

Moving up the body from the hips, the chest bone is reached, which has 3 child bones; the neck, the left clavicle and the right clavicle; thereby contributing an important factor to the orientation of the whole upper body. Assuming that it is possible to find a correct orientation for the chest, its children could be treated independently as in the case of the legs. This assumption is satisfied by considering the chest, neck and head as another complete segmentation, thereby allowing the neck and head joints to provide extra guidance for orientating the chest. The remainder of the body are the two arms, which similar to the legs, are treated as complete segmentations. Figure 2.17 summarises the segmentation of the inverse skinning process, which reduces the overall complexity of solving for the whole body. For each segmentation of the body, there is a degree of overhead required to formulate the equations. However, using PAMPERS a generalised set of equations are generated that facilitate the easily exchange of values from one frame to the next. Consequently, the complete animation of each segmentation is solved before moving onto the next.

Through the segmentation of the inverse skinning algorithm, the minimisation equations presented to the solver have been simplified, and hence reduced the overall complexity cost. Furthermore, through the use of hyperthreaded technology available in all modern desktop PCs or dual-core processors that allow two threads to operate concurrently, the independent segments can be calculated simultaneously. By running parallel threads that solve independent parts of the body, the inverse skinning process described achieved approximately a 35%[7] decrease in computation time compared to serialising the process on the same hardware.

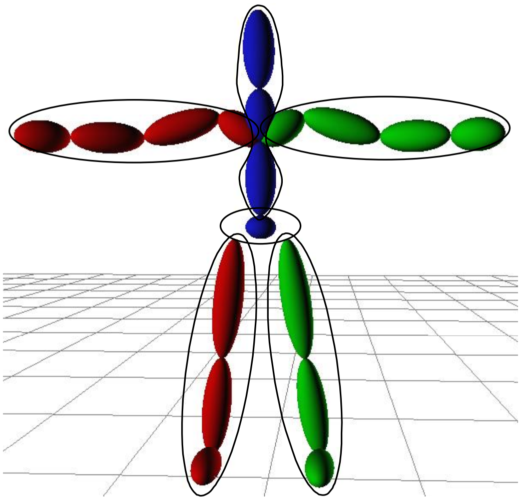


Figure 2.17: Inverse skinning limb segmentation, where the ovals encapsulate the bones solved for each kinematic chain

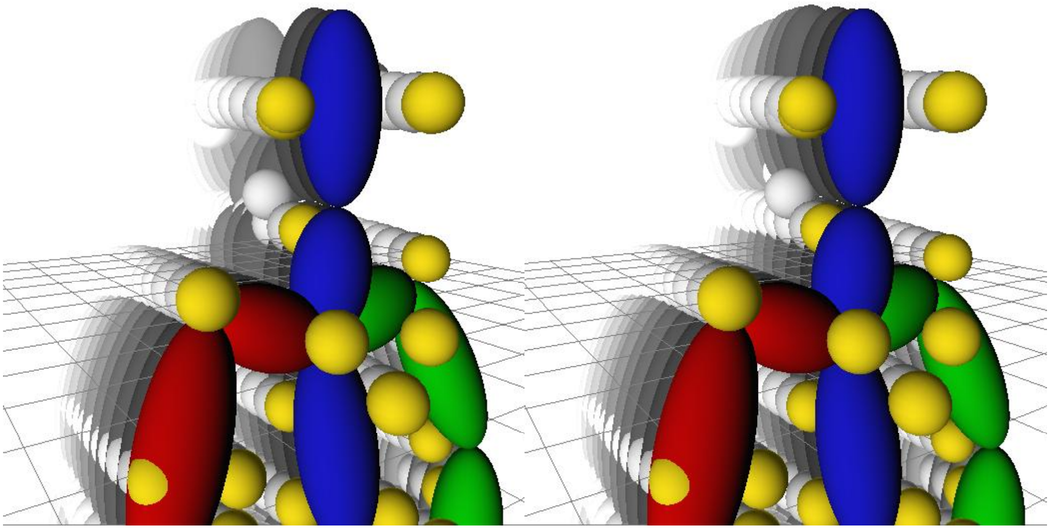
2.3.3 Dealing with Erroneous Data

The inverse skinning algorithm presented thus far has the ability to compensate for slight noise in the optical motion capture data through minimising the distance between the generated and actual marker locations as opposed to finding an exact match. However, the data can exhibit large discrepancies or even missing markers. If a minimisation is attempted here, the resulting animations will demonstrate poor results. To alleviate this problem, a distance threshold between markers associated to a particular bone is introduced. Using the original non-deformed marker mesh, the distances between markers that belong to a specific bone are calculated. During the processing of the animated marker dataset, a similar distance calculation between markers is performed. If the distance of a particular marker differs from the surrounding markers by more than a given threshold, it is removed from the optimisation equations. The difference between leaving an erroneous marker in the dataset and removing it is illustrated in Figure 2.18, where it can be seen in Figure 2.18a that because

[7] This result is obtained by averaging the time difference it takes to solve 25 different motions for the same actor on a Pentium 4 HT enabled processor, between serial and parallel implementations of body segmentation

of erroneous head marker(s), the corresponding head bone is deformed correctly to track the markers, but incorrectly for the actor’s real posture. In comparison, Figure 2.18b eliminates the incorrect posturing by tracking only the consistent marker data and dropping any markers that deviate too much from the common distance configuration.

The removal of a marker from the dataset can be achieved in one of two manners, while still being comparatively efficient. The first approach is to effectively zero out the marker in the equations, which prevents the need of reformulating the minimisation function. Zeroing out a marker is not however a simple process of setting its location to zero because this ignores the translations present. Instead, coefficients for each of the marker formulations, given by Equation 2.5, are introduced, which can be either set to one or zero depending on the whether the marker is active or not.



(a) Using all markers

(b) Dropping erroneous markers

Figure 2.18: The effect of removing erroneous marker data from the optimisation dataset, where in (a) the head bobs about as it tries to track erroneous head markers, whereas in (b) these markers are dropped and therefore the head remains upright

The problem with zeroing a marker is that if it remains erroneous for any length of time, then redundancy is being introduced into the optimisation process. This is in part eased through PAMPERS picking up multiply by zeros and not evaluating the second operand (literal variables are evaluated before compound expression). A less redundant solution is taken in the implementation of the inverse skinning algorithm, which capitalised on a feature of PAMPERS. PAMPERS is able to represent common sub-expressions and give these a placeholder (called a BaseCon) which is evaluated only once when the free variables change whereas in between such changes, it uses the result it last computed. Through this aspect of PAMPERS, each marker formulation of Equation 2.5 is represented as a BaseCon, thereby allowing them to be easily concatenate together if markers are dropped and later reinstated into the minimisation function. This does introduce a slight overhead in terms of creating new addition strings of the BaseCons however the result is a non-redundant minimisation function and the overhead is negligible in the context of a single iteration of the optimisation process.

A further problem that may arise in the course of the optimisation process is that it can get stuck in a local minimum and hence get a poor visual configuration. This issue can be suppressed by using the configuration values from the previous frames as the initial values for the current frame, thereby allowing the algorithm to better track the joint angles over time. It is however still possible for the algorithm to jump out of the optimal solution and therefore the user is provided the ability to nudge joint orientations and rerun the optimisation process over a limited time and segmentation of the body.

2.3.4 Animated Hierarchical Output from Inverse Skinning

The overall mapping process from markers to a hierarchical character is demonstrated in Figure 2.19. The inverse skinning process has successfully been applied to the optical motion marker data of 4 different actors, each performing 25 different motions, with no user intervention required. Some of these motions are used in the continuation of this thesis, especially during the evaluation of the techniques presented in this thesis.

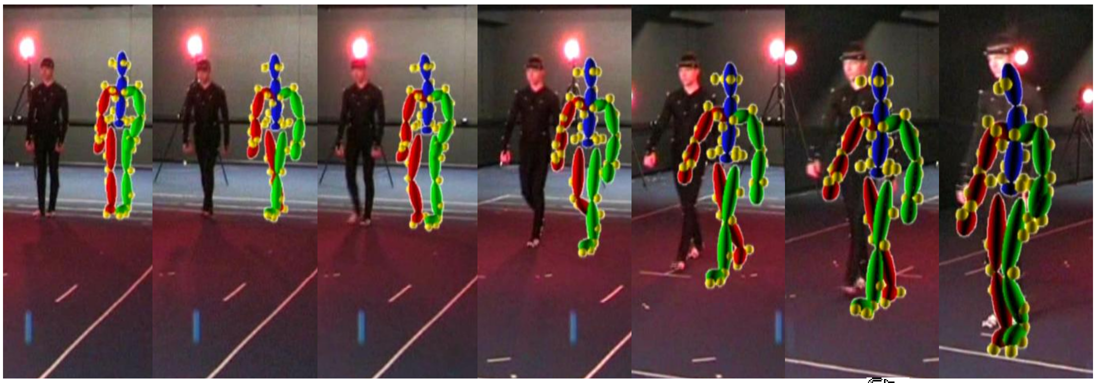


Figure 2.19: Final output from the inverse skinning algorithm

2.4 Motion Data Modifications

The driving reason for wanting to modify motion capture data is that when it is initially recorded, it is specially scripted for a particular scenario and environment. Consequently, if a motion for a different scenario were required the new motion would need to be constructed from scratch. However, this long and costly task could be reduced if tools were available to correctly modify similar motions that already exist.

Alterations to the base motion range from the initial issue of character mapping to the more interactive adaptations due to changes in environmental conditions. A classification of desirable motion alterations that will be considered and referenced during this research is outlined in Figure 2.20.

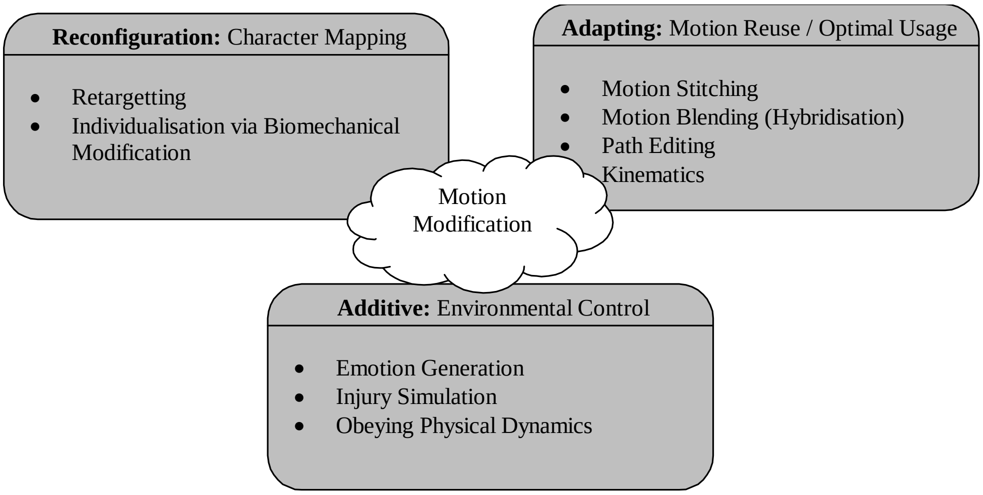
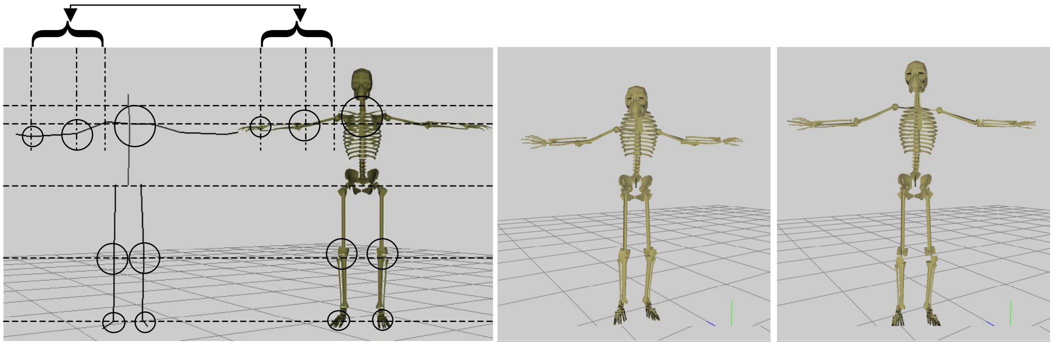

Reconfiguration: Character Mapping
- Retargetting
- Individualisation via Biomechanical Modification
Adapting: Motion Reuse / Optimal Usage
- Motion Stitching
- Motion Blending (Hybridisation)
- Path Editing
- Kinematics
Motion Modification
Additive: Environmental Control
- Emotion Generation
- Injury Simulation
- Obeying Physical Dynamics

Figure 2.20: Categorisation of motion capture playback issues

The following subsections consider the reasons for each of the modification processes that are outlined in Figure 2.20, and give grounding to the driving force behind them. As well as a review of the processes, an outline of possible solutions to some of the problems is additionally given. However, for purposes of clarity in describing the types of desirable modifications, a concrete basis to these solutions is temporarily deferred until later in this chapter (section 2.5 and section 2.6), owing to the fact that many techniques lend themselves well to several different categories.

2.4.1 Reconfiguration: Character Mapping

The process of reconfiguration concerns itself with ensuring that when a recorded motion is skinned and animated in a virtual environment, it first correctly interacts with the environment (retargetting) and secondly looks as if the motion is plausible with respect to the build/biomechanics of the character (individualisation). Retargetting is a problem that is caused by the motion data not having the same limb lengths as the virtual character and therefore, as opposed to a direct mapping, the original data needs to be scaled to match the virtual character's, which is illustrated in Figure 2.21.



(a) Comparison between the base motion (left) and virtual character (right), where the root limbs (hips) have been aligned

(b) Direct mapping

(c) Scaled version actor's base motion

Figure 2.21: Retargetting problem; (a) comparison of base motion and virtual character model; (b) problems with directly mapping the two data sets; (c) Scaling the base motion to fit the 3D model

The problem of scaling the motion capture limb lengths to correlate with those of the virtual character is to affect the final global position of the limbs. This is a problem with the childless limbs (or end-effectors) with which the majority of the environmental interaction is done, i.e. the hands and feet. Figure 2.21b and Figure 2.21c demonstrate this problem using just the base pose where it can be seen that the hands of the scaled motion are higher above the ground than those of the base motion. A similar effect can be seen in the materialisation of foot sliding as the character moves around its environment, which is illustrated in Figure 2.22. Therefore, if the character is to correctly interact with its surroundings the end-effector locations need to be repositioned.

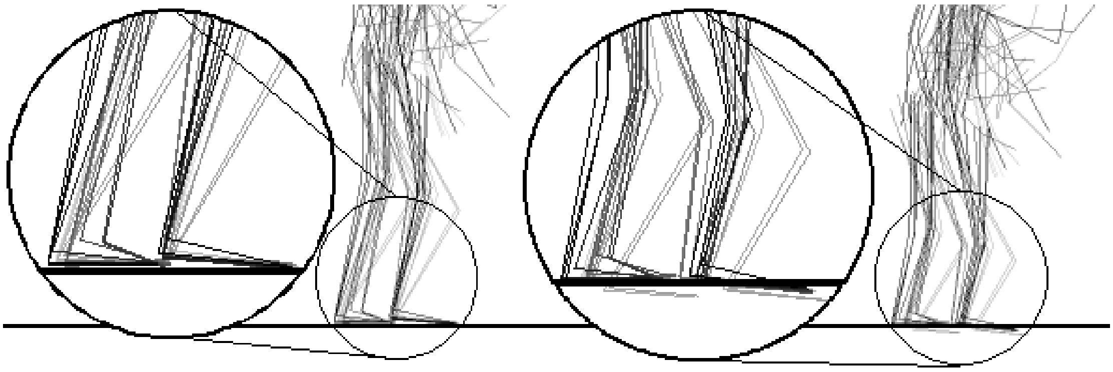


Figure 2.22: Demonstration of foot sliding with 2.7% difference in the femur and 40.7% difference in the tibia lengths; the left image is the original and the right image is the scaled version. The lighter the stick figures represent earlier frames of the motion

Under some circumstances, limb repositioning is however undesirable and attempting to reposition the end-effectors back to their original locations can result in an unnatural looking posture or motion. Therefore, this raises one of the key questions for retargetting algorithms of when the limb positions should be maintained at the cost of violating the original angular differences between limbs and vice versa.

Beyond the retargetting problem lies the subtler issue of individualisation, which is concerned with how the character “carries” itself during the playback of the motion. For example, is it acceptable to use the same motion for both a bulky character and a slim character where each character is performing the same movement? At present, in order to achieve “individualised” characters in computer games, multiple captures of the same motion are performed and during the game, the computer plays back one of the motions from the group of similar movements. The side effect of this is to increase the storage requirements for the motions and the incursion of a much greater time expense required to capture each of the similar motions. The expense is generally the predominate factor that determines the number of motion variations used and when coupled with the fact that many players will rarely be seen anyway, there is little gain from exhaustive individualisation in this manner. To put this problem into context, consider a football game for which there are over 60 teams, each consisting of 11 players. This means that to achieve true individuality you would need over 600 variations for each key movement. This is simply impractical and therefore raises the question of how to spawn off over 600 individually different motions starting with only a fraction of the variations.

One solution to the problem of individualisation is to make use of biomechanical information about the character. Biomechanical based adaptations consider the muscle forces and masses involved in a movement and through the use of Newton's Laws of motion and Lagrangian force equations, it is possible to adapt the base motion according to a parameterisation of the articulated characters, which include muscle torques and limb masses.

Reconfiguration is only concerned with the problems associated with the correct playback of captured motions in the intended virtual environment and not changing the overall motion path. Therefore, the combination of retargetting algorithms and biomechanical modifications would provide a complete set of tools with which to tackle this problem. Two novel techniques that can be used to reconfigure motions are presented in Chapter 4, using inverse kinematics, and Chapters 5 and 6, using dynamics.

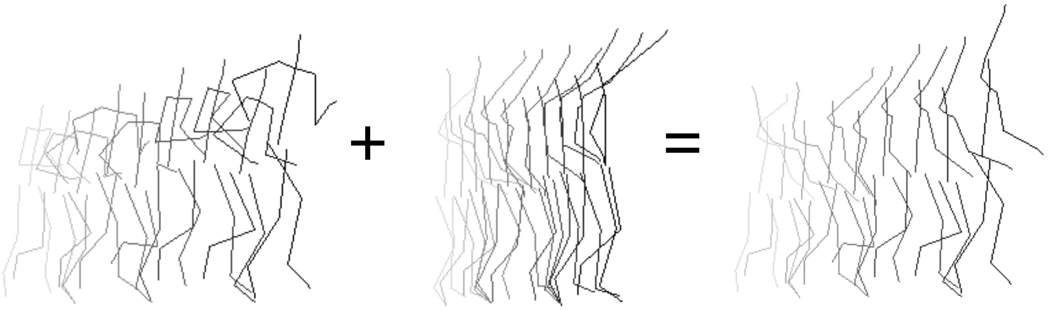
2.4.2 Adapting: Motion Reuse / Optimal Usage

The advantage that motion capture has over traditional keyframing or procedural modelling through the use of high data sampling resolutions to capture motion subtleties also proves to be its drawback when the motion needs to be modified. To modify a motion captured movement, each frame needs to be considered and changed accordingly and because of this inflexible nature, very little motion is recycled. Furthermore, when motions are captured, they are carefully considered and scripted to guarantee that the resulting motions exactly portray those required, which makes them very dependent on the intended environment and not well suited to other situations. Consequently, despite the expense, it is normal to capture completely fresh motions for new projects as opposed to reusing existing ones.

The issue of optimal usage is therefore concerned with getting the most out of existing motions as possible, thereby reducing the overall number of variations of similar movements needed for an application. As an example of poor motion usage, consider a goalkeeper in a football game in which it would be normal to record many captures of a goalkeeper diving to get to the ball which each vary only in height above the ground. The computer would then select the most appropriate motion under the circumstances during the game and play it. This approach is extremely wasteful because many motions are required that effectively perform the same movement, which is comparable to the issues of individualisation.

Motion stitching provides a solution in which motions can be strung together in order to produce longer periods of movements. Employing such techniques could lay the foundations for using smaller motion segments that represent basic building blocks of movement, say a single stride. This approach requires a good-sized set of mini-motions with some kind of selection heuristic and stitching algorithm that smoothly transitions between the different components. To illustrate the application of motion stitching, consider a character walking along a pre-defined path where the movement is being driven by a single motion, however at certain points along the path, an obstructing object is placed. With fully recorded motions, a new movement file for each obstruction would need to be recorded and stored, where the most appropriate recording would be played back during game play. Using building blocks, a walking cycle could be generated up to the offending object and upon reaching the object the computer selects an appropriate action to get round the obstruction, then continues the walking cycle once the object has been cleared. This basic principle could then be used in conjunction with path planning research [Gill99] in order to determine how an articulated character should move in a cluttered environment and best tackle different obstructing objects.

Motion blending is similar to motion stitching in that it uses multiple base motions. However, instead of appending them together, the motions are played simultaneously on the character, which allows different parts of the body to be driven by different sources (or weighted combinations of the different motions). Using the analogy of building blocks, the basic entity would be motions for individual nodes or chains of nodes (i.e. complete limbs) and a carefully selected combination of motions are chosen to give the character the desired hybridised motion. In essence, motion blending is the process of creating new hybrid motions that exhibit characteristics of each of the base motions. Figure 2.23 illustrates an application of the motion blending technique where a hybrid motion is generated by merging the leg movements of a running action with the upper body of waving movement, thus resulting in a character running whilst waving.



Base motion of character running

Base motion of character waving

Hybrid motion of a character running and waving

Figure 2.23: Hybridised motion generated from two base motions, one running and the other waving

By using a weighted combination of multiple motions for a given body segment, it is possible to generate intermediate motions that lie in the volume spanning the original movements [Rose98]. For example, the two motions of a goalkeeper diving high and low could be blended together in different proportions to result in a variety of different heights between the two original motions. Therefore, instead of recorded many similar motions, it is possible with just a few motions to generate the required range of motions.

Unlike motion stitching and blending, motion path planning only requires a single motion to operate upon. The principle behind path editing is to alter the motion data at a high enough level so as to change the overall velocity and direction of the motion while still maintaining the essence that makes the motion what it is [Glei01]. For example, consider a dance motion recorded for a particular environment where the character is to perform their routine in a circular path. Now, if a dance motion were required to follow a straight line, traditionally an entirely new motion would be required. However the tools provided by motion path editing algorithms would be able to take the high-level circle path and stretch it out into a single line.

Using the classification of Figure 2.20, the processes that tackle motion reuse and optimal usage are termed adapting algorithms because they adapt the original motion in such a way as to make it more suitable for a scenario. Motion adaptation can be achieved using the tools provided by motion stitching, blending and path editing algorithms while the field of kinematics can also contribute towards adaptation by repositioning limbs of the character in absolute space. For example, the location of a goalkeeper's hands could be made to coincide with a football using inverse kinematics as an alternative to the motion blending technique.

2.4.3 Additive: Environmental Control

Additive modifications factor details into a motion that were not present in the original in response to environmental stimuli. This could be in the form of emotion, injury or in response to a dynamical influence, for example. Emotional control considers how a character moves according to the way they feel. For example, a happy person might "spring" along, head held high, while a more subdued character would walk more slowly, looking toward the ground more [Dens97]. Such additive modifications would attempt to replicate the same kind of high-level emotions, where the character visibly expresses its thoughts and feelings through their movements. The addition of emotional characteristics would enhance the atmosphere of computer games and expose a completely new form of output medium in the form of visual body language, which can be received at a subconscious level.

Injury simulation is another concept that could be factored into an existing motion where the action is modified to take into account some kind of wound or infliction that the character has sustained. At present, computer games simulate injuries by storing motions that portray a wound and playing back the corresponding one when the character is hurt[8]. The problem with this approach relates back to the issues of optimal usage in that multiple motions for a character's movement are needed: one for the normal action and several that demonstrate some kind of injury. Consequently, the strength of simulating injuries lies in its ability to generate an assortment of possible motions when inflicting a character with different types of injuries over varying parts of the body. Subsequently, real-time simulation of additive injuries has a great deal of potential in many genres of computer games from sports to first person shooters and even role-playing games.

The final type of environmental control over base motions considers how the actual dynamics of a scene affect the motion of a character, for example impacts, changes in wind, gravity and terrain gradient. The ability to change a character's surroundings and have the character respond accordingly would enormously add to the realism of, say computer games, as it introduces an unprecedented level of feedback about the generated environment. Modifications based on the dynamics of the environment and biomechanics of the character are some of the most complicated techniques employed in character animation, as Chapter 5 and 6 demonstrate when they present a novel technique in this area. However, dynamical modifications provide the tools that ensure the resulting motion is

[8] Such games that make use of injuries in this manner include Nintendo / Rare's GoldenEye 007, http://www.nintendo.com/goldeneye007, and Perfect Dark, http://www.perfectdark.com

physically plausible, which has been something missing through the other types of motion modification that have been considered.

The field of Additive modifications covers a wide range of potential changes to base motions, however the common thread of these modifications is to adapt the motion with respect to changes or encounters in the environment. At present, little to no modifications on this level have been undertaken in computer games with the exception of a simplified approach at presenting injured characters. However incorporating algorithms that perform modifications at this level would result in a higher degree of realism over the already realistic captured motions of computer characters.

2.4.4 Summary

The problems associated with playing back existing motions range from basic character fitting to the more complex question of how the motion should behave under different environmental conditions. Up to now, the motion modification techniques discussed have been abstracted over with little grounding in how to actually go about implementing the solutions. Therefore, the following section looks at a variety of specific algorithms and mathematical techniques that will allow the implementations of the solutions to be realised. The presentation of the mathematic techniques is followed in section 2.6 by a discussion on which methods are appropriate for each solution together with a review of existing implementations and limitations of current work in the field.

2.5 Mathematical Solutions to Motion Editing

This section considers four different mathematical techniques that can be utilised as a means of tackling the problems highlighted concerning the modification of motion capture data for animating articulated characters. The techniques discussed are:

§ Blending functions
§ Forward and inverse kinematics
§ Motion warping
§ Dynamics

For each technique, an overview is provided to describe the principle of the technique along with a discussion on their advantages and disadvantages in this field.

2.5.1 Blending Functions

Blending functions provide the simplest and least computationally demanding technique for motion adaptation. This makes them excellent candidates for real-time computational tasks. Blending techniques interpolate between 2 or more values in order to get a visual result that lies somewhere between the sampled motions. This approach is comparable to the way in-between frames are generated for keyframe animation.

The way in which the motions are blended together is defined using a mathematical expression that links specific points in space and/or time of the example motions. Many different formulations of blending functions are conceivable therefore we illustrate this process with a simple mathematical representation, which is given in Equation 2.6. In Equation 2.6, the P_i values represent the positional extremities of the space/time volume that is spanned by this function, which is parameterised over some range of *u* (usually *0 £ u £ 1*). The evaluation of this function over the valid range of *u* defines the sweeping path though the space/time domain which starts at point P_0, concludes at P_n and interpolates or approximates all other points, P_i *Î { i > 0 Ù i < n)*,_along the route The B_i terms provide weightings for the corresponding point, which provides the ability to alter the degree of influence any given point has over the whole path.

$$Q(u) = \sum_{i=0}^{n} B_i u^i (1-u)^{n-i} P_i \qquad \textbf{(2.6)}$$

Practical applications of this technique include the production of smooth transitions when appending motion segments together and generating motions for hybridising multiple motion captures. Because such equations are easily evaluated in real-time, they form the basis of many of the current transitional algorithms used in this area. However since they generally consider joint angles independently, complex behaviour cannot be captured and hence the limitations of blending functions are soon realised when attempting to do anything but simple motion modifications.

2.5.2 Forward and Inverse Kinematics

Kinematics is a branch of mechanics that considers the posture of a body or system of bodies with no regard to their mass or forces acting upon them. Hierarchical structures are used to represent kinematically defined models, such as that illustrated in Figure 2.4 for a humanoid character. Within the articulation, each parental global frame of reference is accumulative defined by combining the local frame of references of each parent node that is traversed when moving towards the root.

Each node, which correlates with a character's limb, has its own local definition that is parameterised by a local rotation, a length and an offset from its parent's local centre. The limbs are non-prismatic so the offset parameter can be considered constant and hence the only varying parameters are those that define the rotation of the node. Equation 2.7 translates any attributed local description of a node, V_n, in its parent's space to give P_n, where M_n represents the child's local orientation and translation matrix.

$$P_n = M_n V_n \qquad \textbf{(2.7)}$$

This recursive nature of applying transformation from node to node as the kinematic tree is traversed upward towards the root node allows all the nodes within the tree to be expressed in a global frame. This is defined in a general sense through Equation 2.8, where P_0 is the globally defined description of V_n, for the n^{th} node in the kinematic chain expressed in a global frame of reference.

$$P_0 = \left(\sum_{i=0}^{n} M_i \right) V_n \qquad \textbf{(2.8)}$$

Equation 2.8 illustrates that the resulting locations of end-effectors are determined by the orientations and translations of all of the nodes in the kinematic hierarchy that lie along the kinematic chain between the end-effector and the root. This leads to the more recognised equation for defining forward kinematics, which is given in Equation 2.9, where q represents the complete set of orientation and translation values for a kinematic structure and X is the global position of a given end-effector in the hierarchy.

$$X = f(q) \qquad \textbf{(2.9)}$$

Forward kinematics thus allows the posture of one character to be applied to a similarly structure different character without the need of redefining the complete structure. Figure 2.24 illustrates this idea where the orientation of the limbs of the terminator model are copied from the Skeleton model with just the root positions (the pelvis node) being rotated and offset to the meet the environmental seating requirements.

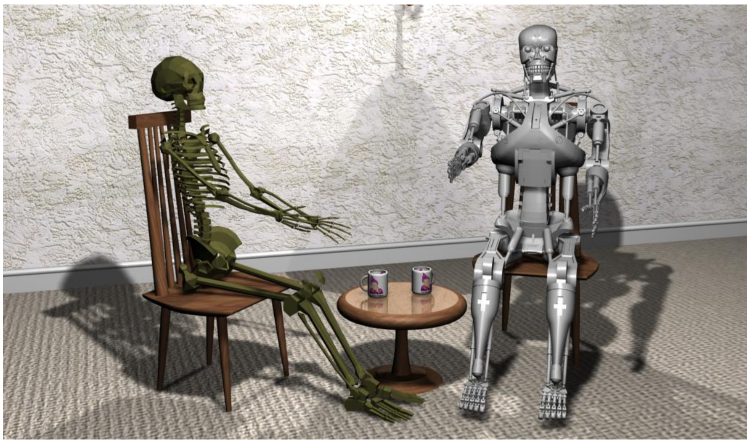
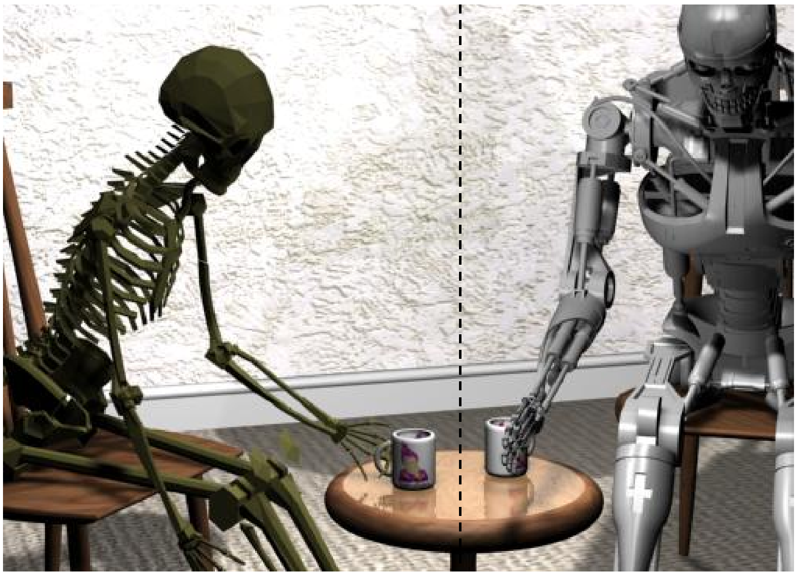


Figure 2.24: Similar looking postures using forward kinematics and matching hierarchical orientations

The problem with forward kinematics is its inability to position limbs at absolute positions in space. For short chains of nodes, this is not too much of a problem because with a little foresight, it is possible to imagine the path of the chain and thus correspondingly set the orientation values of the limbs. This is illustrated in Figure 2.25a where the upper-body of the skeleton has been re-orientated per node to grasp the mug on the table from its original posture of Figure 2.24.

However, such specific configuration data sets do not map well to different characters as illustrated in Figure 2.25b, where the skeleton's upper-body configuration state has been copied to the terminator. Since the terminator has limbs of different lengths to the skeleton in Figure 2.25b, when the joint orientations are transferred, the relative spatial location of the terminator's hand is different compared to the skeleton despite having the same orientation parameters (the retargetting problem), and hence the character does not correctly interact with the mug.



(a) Skeleton correctly grasping its mug (b) Incorrect kinematic parameters

Figure 2.25: The use of forward kinematics to (a) correctly specify an absolute position of the Skeleton's hand to grasp the mug which (b) do not (symmetrically) map across to the Terminator model to allow it to correctly pick up its mug.

Inverse kinematics (IK) provides a useful alternative to forward kinematics when it is desirable to absolutely position an end-effector without having to manually configure individual joint rotations, such as in the case of Figure 2.25. As Equation 2.10 for inverse kinematics highlights, the process is the exact opposite of forward kinematics.

$$q = f^{-1}(X) \tag{2.10}$$

Equation 2.10 can be solved either analytically [Watt92, Chin96] or numerically [Crai55, Eber01], however for our continued discussions in this chapter, an expansion of these techniques are deferred until Chapter 4, where novel work is also presented in this area.

Despite the advantages IK heralds for artists when configuring articulated structures, the technique does suffer from several problems that are inherent in the under-defined nature of Equation 2.10. As Equation 2.10 is under-defined, there are an infinite number of solutions in the orientation parameter space that could satisfy a single spatial location. Therefore, the results produced by an inverse kinematics technique can be unpredictable as the solver “chooses” one result from the many possible solutions available. This is illustrated in Figure 2.26 where the same root location and end-effector position are specified but the resulting character configurations are very different.

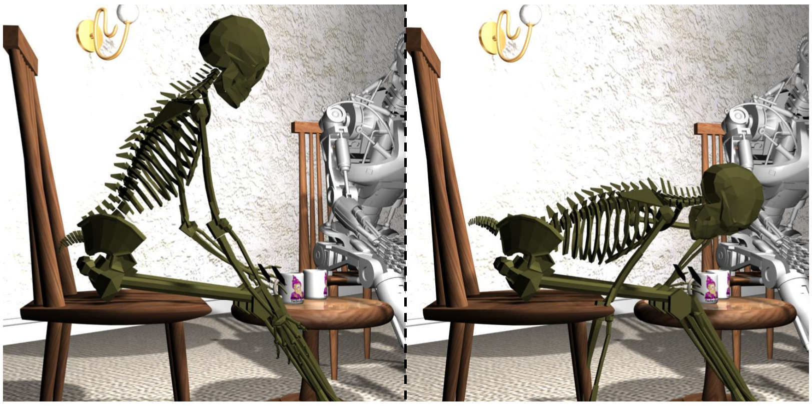


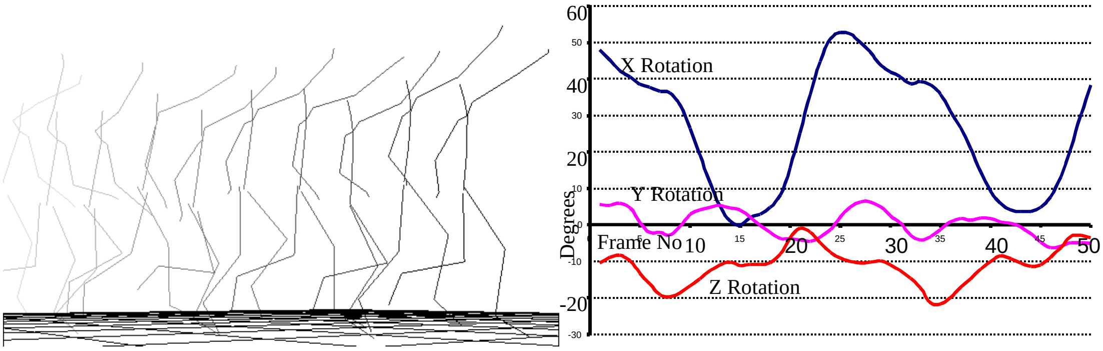
Figure 2.26: Possible different character configurations when using inverse kinematics. The pelvis of the skeleton is the root and the left hand is the end-effector of the IK chain.

In order to make the inverse kinematics techniques more usable, any implementation needs to facilitate the addition of some kind of knowledge about the hierarchy so that it can select an “intelligent” solution from the large space of all possible solutions. For example, joint angle range restrictions, such as those outlined in Figure 2.14 would explicitly prevent any implausible character configurations such as rotating an elbow backwards beyond its natural range of motion.

Despite its under-defined nature, inverse kinematics proves to be a very useful tool in modifying character motions because of its ability to completely reconfigure the joint configurations of an articulated structure with minimal data input, plus its ability to operate in a real-time manner. These points are further highlighted in the work of Chapter 4, which additionally provide concrete basis for the implementation of Equation 2.10.

2.5.3 Motion Warping

The parameterisations of animated hierarchical structures over time can be viewed as a collection of motion curves where each degree of freedom has its own independent path though time. Figure 2.27 shows the 3 degrees of freedom of a humanoid character’s femur over a short segment of animation. The idea behind motion warping is to individually modify the motion curves of a character using a small set of time-based constraints (typically less than 5 [Witk95]), similar to keyframing, in order to warp the overall shape of each motion curve yet at the same time leaving the finer detail of the motion relatively unaltered.



(a) Walking and waving motion

(b) DOF motion curves of the left femur

Figure 2.27: (a) 50 Frames of a character running and waving – stepped every 5 frames; (b) Motion curves of the 3 DOFs of the running character’s left femur

Motion warping uses two types of constraint parameterisations where the first is used to define a desirable value for a particular degree of freedom, q_i, at a given time, t_i, which is represented by the 2-tuple (q_i, t_i). The second constraint type provides the ability to scale time from the original animation frame, t_i, to a new time frame, t_i', which is also represented as a 2-tuple, (t'_j, t_j). The complete expression for motion warping a single degree of freedom is shown in Equation 2.11, where q and q' are the original and new motion curves for the DOF respectively. From Equation 2.11, the transformation function, $f(q, t)$, deforms the curve according to the (q_i, t_i) constraints, while the time warp function, $g(t')$, provides the mapping for the time warp constraints, which maps time from the new motion to the original one.

$$q'(t) = f(q, t) \qquad \text{where } t = g(t') \qquad \textbf{(2.11)}$$

The time warping function of Equation 2.11 can be represented by any continuous function that interpolates the time constraints, however the transformation function requires slightly more consideration. Since q_i are scalar values, there are only two forms of transformations that need be considered, scaling and translation. Witkin & Popovic [Witk95] expressed the transformation equation with a linear equation of the form illustrated in Equation 2.12, where *a(t)* and *b(t)* are scaling and offset coefficients respectively.

$$q'(t) = \forall_i \bullet q'(t_i) = f(q_i, t) = a(t_i)q(t_i) + b(t_i) \qquad \textbf{(2.12)}$$

The problem with Equation 2.12 is that for a given constraint point, there exist many solutions to the equation. Therefore a suitable combination of the scaling and translation components must be decided. Witkin and Popovic took the approach of keeping one component constant while varying the other to maintain the constraint, allowing the user to decide upon which to fix when a new constraint is interactively added. On determining the values of *a(t)* and *b(t)*, an interpolating spline is constructed through all the constraint points which when combined with the time warp function fully determines the motion warping algorithm.

Once valid functions for *a(t)*, *b(t)* and *g(t')* are determined, the generation of the new motion, *q'*, is a trivial task with relatively low computational demands. It is therefore possible to evaluate Equation 2.11 for each DOF of an articulated character in real-time. However, in a dynamically changing scenario, the functions *a(t)*, *b(t)* and *g(t')* cannot be pre-determined because they vary depending on the environment. Therefore the computational costs of calculating the required functions needs to be considered and thought needs to be given towards automatically selecting which coefficient to keep constant in the transformation equation (Equation 2.12). However, because the number of constraints is a sparse set and some degrees of freedom may not need modifying, the spline fitting process to determine the required functions is not excessively expensive (i.e. the technique is still possible in real-time) but the whole process is far more computationally costly then it would first appear from Equation 2.11.

Motion warping provides a simple extension to the keyframe paradigm by allowing constraints to be configured at sparse intervals, which are used to deform the original motion while keeping the overall details of the base motion intact. However, because the motion curves are based on Euler angles, which are not linear under combination, and only a small set of keyframes are used, there is scope for erroneous results in the form of unattainable postures by invalid joint angles or angle combinations. A further problem of this technique is that large deformations from the original motion can produce abnormal and unnatural results as with all warping techniques. Despite the shortcomings of motion warping to acknowledge that each warp forms a bigger picture, the technique lends itself well to the issues of basic retargetting, motion reuse and optimal usage, which is discuss in section 2.6.

2.5.4 Dynamics

The field of dynamics applied to motion editing reveals a wide scope for adaptations, which includes modifications based on the internal biomechanical forces of an articulated character and external environmental interactions. The application of dynamic simulation on top of base motions ensures that any resulting motion conforms to a set of real-world physical laws, which can be parameterised and either pre-defined or change over time. Dynamic simulation also allow motions to be generated that cannot easily be captured due to risks that the actor may be subject to while performing the motion, such as falling from a high building. Potential environmental forces that can be modelled with a view to modifying motion data include gravity, wind, surface resistance and impulse & collisions [Oshi01], while biomechanical data will provide joint torques and moments of inertia necessary to complete the dynamic calculations [Liu05].

Although dynamically simulating events for computer environments is notoriously costly in terms of computational power, the underlying principles of the technique are represented by some relatively basic concepts. For the time being, the mathematical concepts that underpin dynamical modifications are refrained from. However, Chapter 5 later expands these concepts as dynamics-based changes become the primary focus of the chapter, which also introduces new contributions to the field of character modification.

2.6 Applicability of Solutions

The applicability of the mathematical solutions for changing motion capture data in order to overcome the problems highlighted in section 2.4, are considered in this section. Table 2.2 provides an overview of which mathematical techniques are appropriate for each of the problem classifications, which are discussion in detail in the following sections.

Algorithm **Problem**	Blending Functions	Inverse Kinematics	Motion Warping	Dynamics
Reconfiguration (Character Mapping)	**O**	**P**	**P**	**P**
Adapting (Reuse & Optimal Usage)	**P**	**P**	**P**	**O**
Additive (Environmental)	**O**	**O/P**[9]	**P**	**P**

Table 2.2: Source of mathematical solutions for modifying motion data

2.6.1 Reconfiguration

Reconfiguration of motion data covers the problems of retargetting and biomechanical modifications for which several different algorithms have been applied in previous studies. The use of inverse kinematics is the obvious solution to retargetting [Choi99, Kore82] as it directly provides the tools necessary to reposition any end-effectors of an articulated character that violate spatial constraints as a result of scaling the motion to its new portions. The types of constraint violations that the IK algorithm can be used to fix include foot sliding [Kova02] and correcting the positioning of hands such that they correctly interact with environmental objects. An application of both IK corrected foot and hand positions is given in section 4.6, where a fast inverse kinematics solution is introduced

To reaffirm end-effector locations in a virtual environment, the IK algorithm is applied to each frame of the original motion. However, this means the process has no knowledge about past or future limb postures and subsequently there is no concept of easing in or out of constraints. Consequently, the resulting motions can appear to snap to positions when constraints are turned on, which introduces undesirable high frequencies into the motion. Therefore, it is useful to have a higher-level control process that either eases in and out of constraints or use filters that run over the motion to limit the introduction of undesirable high frequencies [Sul98, Lee99, Tak02, Tak05].

[9] In past solutions, inverse kinematics has not been used to provide additive motion to existing animations, however the work we present in Chapter 3 demonstrates a technique that allows us to simulate injuries, thereby demonstrating an additive technique

Monzani *et al.* [Monz00], made good use of the traditional inverse kinematics algorithms by applying a per-frame Jacobian solution to an intermediate skeleton in order to retarget base motions. The intermediate skeletal structure is used to map the base motions to different hierarchical and geometrical structures, which meant that only the modification of the single intermediate skeleton is required. Monzani *et al.* minimised any IK jerkiness by employing the solver to generate just a displacement map and smoothly ease into and out of constraints.

Shin *et al.* [Shin01] further illustrated the potential use of inverse kinematics in their importance-based computer puppetry retargetting system. The real-time, on-line system they present makes use of an IK solver that hybridises an analytical and numerical solution in order for a computer puppet's end-effectors to approach and eventually touch an object in virtual space as the live actor nears it in real space. The inverse kinematics solver used by Shin *et al.* makes use of the fact that the joint angle of the upper arm can be uniquely determined from the distance between the goal point and the shoulder (and similarly for the legs) as demonstrated by Tolani & Badler [Tola96]. However this still leaves a degree of freedom that rotates the elbow about an axis that passes through the shoulder and wrist. Lee & Shin [Lee99] demonstrate how to numerically solve the remaining degree of freedom, whereas Shin *et al.* solve it analytically with a view to minimising the deviation in the arm posture from the actors.

Shin *et al.* additionally present a technique in which automatic importance factors are determined, i.e. during the process of retargetting a character, when is it important to maintain joint angles for a natural looking posture over end-effector positions. The importance of which variables to maintain during the on-line retargetting algorithm is derived by a distance function that also considers the end-effector trajectories, thereby allowing provision for a smooth easing in and out effect. This follows on from a similar study by Bindiganavale & Badler [Bind98].

So that the potential problem of snapping within the IK algorithm may be elevated when retargetting base motions, applications based on motion warping algorithms have been used [Witk95], which demonstrate an effective alternative. Unlike inverse kinematics, motion warping considers the motion as a whole by creating a smooth displacement curve through time and consequently, the frequency content can be more closely controlled. Owing to the fact that the motion warping technique is used to deform the overall motion curve for each DOF, the resulting motion will still exhibit any characteristic high frequencies of the base motion while avoiding the introduction of new ones. This is achieved though the sparsity of the key control points used to construct the displacement curve. However the problem comes when automatically deciding upon the keyframe spacing because the addition of high frequency movements is sometimes necessary or even desirable. A further problem with this approach is that an articulated structure's degrees of freedom are considered separately. However, since Euler angles are not associative, the system of limbs is not necessarily best modelled as a linear system as it is done in this method. Studies into the use of motion warping of character retargetting have found that angular differences for retargeted motions have been small enough to allow Euler angles to be treated as linear and therefore lessen the seriousness of this problem. An additional concern with motion warping is that if the new motion is stretched too far from the base motion, unnatural effects and poses will occur which are a consequence of the method having no appreciation of body dynamics to refer to.

Thus far, only the retargetting aspect of the reconfiguration problem has been considered, primarily because the algorithms discussed ignore biomechanical and dynamic information. In Chapter 4, a novel way of cheaply introducing stylisation and individualisation effects through an inverse kinematics solution is presented, thus also retargetting a motion. However, this approach still ignores biomechanical parameters and simply approximates them, thus making a real-time application.

In order to more accurately recreate biomechanical information, more complex techniques are employed, where many such techniques utilise an optimisation process, termed spacetime constraints [Witk88]. Spacetime constraints have been used to animate non-complex objects under gravitational effects [Liu94] as well as both retargetting [Glie97, Glei98a] and apply character styles to different motions [Liu05]. For the purposes of clarity, the details of such techniques are ignored and it is enough to abstractly state that optimisation techniques attempt to solve a system of constraints while simultaneously minimising an objective function, which gives a measure of the "goodness" of the result. The mathematics behind theses techniques is expanded in Chapter 5 of this thesis, where novel work is presented in Chapter 6 that reconfigures the motion of one actor to another using only the base motion from a source actor and the biomechanical information of the target actor. This is different to the work presented by Liu *et al.* [Liu05] in that they capture the style of an actor from their motion data and then dynamically apply it to a different captured motion. In contrast, the technique later presented in this thesis does not require an existing motion of the target actor, only its biomechanical information.

The main drawback of optimisation-based techniques in the area of character reconfiguration is that of the three competing techniques discussed so far, they are the most computationally intensive, especially when defining many character and environmental dynamic constraints. This is also in part because the whole motion is consider at once as opposed to the per-frame approach taken by the inverse kinematics solutions. Consequently, it is undoubtedly the high level of processor power required to solve the equations associated with the spacetime constraints method that led to the dynamics of early studies being largely ignored when applied to the problem of character retargetting. However with the increasing performance in desktop PCs, dynamics are slowly filtering back into optimisation techniques, albeit still grounded in off-line processes.

The inclusion of dynamics to individualise motions have also been utilised in per-frame approaches. Komura & Shinagawa [Komu00, Komu01] present one such technique where they approach the problem of individualisation through the use of dynamic muscle models, in particular Hill's muscle model (which is further explained in section 5.2.1). Instead of concerning themselves with spatial constraints, Komura & Shinagawa presented a technique to individualise pre-recorded motion data by taking into account the different character muscle strengths, sizes and masses. Zordan [Zord99, Zord00] further demonstrated the principle of modifying motion data via the use of dynamic muscle models, but this time through the use of proportional-derivative control servos (see section 5.2.1). The reconfiguration of characters by Zordan is similar to that of Komura & Shinagawa in that individualisation and physical correctness was the main goal of the work and less attention was paid to spatial constraints that the character should obey. Effectively, these works only address one aspect of character reconfiguration, namely individualisation, and hence provide little provision to tackle the retargetting problem. However, it has been shown that if only small spatial changes need to be made to the final dynamics solution then a simple inverse kinematics solution can be used to affix these positions [Sofa04].

The use of dynamics has also been employed to generate completely new motions without any pre-recorded motion data [Lasz96, Pann96], thereby providing a useful tool for generating difficult to capture motions. However, like motion capture data, dynamic systems for a particular character are not easily scaled to others. Hodgins & Pollard [Hodg97] demonstrated an effective technique for adapting character behaviours through the use of proportional-derivative control servos in an on-line approach where existing motions are modified for characters that have varying body proportions.

Other techniques that have been used to tackle the problem of retargetting include Inverse Rate Control [Choi00, Whit69] and Reference Modelling [Komu00, Poll99, Poll00] methods. The Inverse Rate Control method used by Choi & Ko is somewhat similar to the principle of Jacobian inverse kinematics however they make use of a closed-loop IK scheme [Chia91, Scia87]. The use of the closed-loop scheme means that any errors in positioning an end-effector exponentially converge to zero and the redundancy of the Jacobian can be used to optimise secondary tasks thereby resulting in an efficient solver. The mathematical aspect of this is further considered in section 4.1.2.

The Reference Modelling approach used by Komura *et al.* makes use of an intermediate musculoskeletal of the human body by including Hill’s muscle model as part of a spacetime constraint solver. The approach allows not only motions to be retargeted to other characters while maintaining biomechanical correctness, but also to simulate fatigue on both created and base motions, thereby overlapping into the additive category of animation modification techniques.

The main problem with the approaches taken has been their lack of ability to simultaneously retarget and individualise an existing motion. Existing retargetting algorithms work extremely well in real-time, however they do not address the issue of how the motion can be adjusted according to the characters build. Conversely, techniques that produce realistic individualisations do not necessarily preserve the desirable end-effector locations of the original motion. The complete process of reconfiguration has been best demonstrated by Hodgins & Pollard [Hodg97] and Liu *et al.* [Liu05]. Unfortunately, the former of these techniques can be difficult to control because the proportional derivative control servo variables need to be tweaked by hand to affect a result in a somewhat non-probabilistic manner. In contrast, the approach presented by Liu *et al.* (and the one used later in this thesis to achieve biomechanical reconfigurations) models the complete human articulated structure using rigid body mechanics, which affords a more intuitively controlled, albeit a more complex one too. However, the work presented by Liu *et al.* requires an example motion from an actor to obtain a set of stylisation parameters in order to apply them to different motions, which may not always be available. Consequently, the novel techniques presented in Chapter 4 and Chapters 5 and 6 of this thesis allows a motion to be reconfigured by approximating or using biomechanical information about the target character. The former of these novel approaches is a real-time inverse kinematics based approach whereas the latter uses full body rigid body mechanics and is inherently an off-line technique.

2.6.2 Adapting

The most basic technique that can be used to achieve motion adaptation (stitching and blending) is provided by the linear blending/interpolation approaches, which are quick and computationally cheap to perform. This makes them excellent candidates for real-time work. Wiley & Hahn [Wile97] and Rose *et al.* [Rose98, Rose01] considered the concept of interpolation synthesis to generate a new motion within the volume span of interpolated base motions. However, the latter work considers in greater detail where example motions should be placed in space and the interconnections between different example motions in order to reduce visual artefacts in the results. Rose *et al.* use verbs to parameterise motions into emotional expressiveness or general behaviour descriptions, for example happy or sad walks and walking left or right. Subsequently, adverbs are used to parameterise the verbs, which describes all possible variations for a particular verb. It is the adverbs that uniquely identify each verb in space, which directly correlates to an example motion. New motions are generated through the traversal of the verb graph, where the example motions are interpolated using radial and polynomial basis functions. The generation of the new motions can be performed at very high frame rates because of the ease with which the interpolation functions can be evaluated. However, this is only possible with a medium degree of pre-processing that positions the verbs in space by determining the adverb parameters. This process also needs to be performed every time a new example motion is added to the verb graph.

In order to perform viable motion transitions between existing motions, Mizuguchi *et al.* [Mizu01] takes the approach that to correctly stitch motions together, a suitable point in the second motion had to be selected based on the position of the currently playing motion. With these two points in time on the different motions, a suitably sized transition time is also required in order to produce natural looking results. However, skilled artists manually direct all of the required temporal constraints for the algorithm to operate with. This means to use such an approach a great deal of artistic labour is required during a pre-processing stage thereby making it difficult to relate the solution into a real-time, generalised transition algorithm.

Other studies into performing adaptive motion modifications have been inclined to focus on the more complex algorithms of motion warping and spacetime constraints to achieve good transitional and hybridisational effects between multiple motion segments. Witkin & Popovic [Witk95] used motion warping algorithms to overlap and blend parameter curves of different motion segments as well as reconfiguration tasks to ensure positional correctness of end-effectors. The whole process is capable of operating in real-time because of the low number of keyframe-like constraints used, which ranged between 1 and 5 per motion, however the displacement curves are manually created and tweaked as a pre-processing stage.

The application of the spacetime constraints technique in the work done by Rose *et al.* [Rose96] is aimed at semi-automatically generating motion transitions and cyclifications via a recursive dynamic formulation of the method. The spacetime minimisation function used is based on the joint torques over time since this was deemed to be a good predictor of metabolic energy. In order to solve the spacetime problem, a recursive inverse dynamics implementation of Balafoutis is used [Bala91] which is of an efficient *O(n)* recursive complexity, where *n* is the number of degrees of freedom of the system. The specialised spacetime solver is used to determine the motion of any non-supporting limbs during transitions, while other limbs are solved using an inverse kinematics optimisation algorithm over time. However, even with the use of the efficient solver, the overall algorithm is still much more computationally demanding than the blend or warping techniques.

Inverse kinematics offers a further useful tool in adapting motions, which, because it is applied in a per-frame manner, it provides a quicker solution compared to spacetime constraints and motion warping algorithms. IK has been previously demonstrated in the field of retargetting to reposition end-effectors in order to correctly interact with the environment [Glei98b] and to maintain the balance of a character [Aydi99]. In applying IK to motion adaptation, inverse kinematics can be used to stitch two motions together that don't quite align by interpolating the end-effector positions over time between the two example motions. This approach is the exact opposite to interpolating the joint angles, so similar to the problem of retargetting, it would be necessary to determine which technique would give best results for the particular scenario. For example, it might be just as well to interpolate the joint angles for arms that do not interact with the environment. Whereas, going from a standing posture to a crouching one, if joint angles were interpolated, the character's feet would inevitably sliding over the floor, hence the end-effector interpolation would demonstrate much better results in this instance.

Bruderlin & Williams [Brud95] show how applications of filtering algorithms from image and signal processing domains [Burt83, Odge85] have potential in the field of motion adapting by applying them to multi-resolution spline representations of an articulated character's degree of freedoms. Using such an approach to modify motion curves has been shown by Bruderlin & Williams to produce easily exaggerated motion characteristics by simply adjusting gains for certain frequency bands. Multitarget motion interpolations with dynamic time warping (DTW) [Demo86] are also demonstrated as a useful adaptation tool as a means of stitching and blending together motions. The interpolation is done in terms of the frequency bands and the DTW algorithm is used to best time-correlate the motions in order to avoid discrepancies such as foot planting miss-alignment between motions. Bruderlin & Williams further showed the use of waveshaping as a means of capping joint angles and displacement mapping as a path editing tool.

Gleicher [Glie01] further explores the issue of motion path editing by factoring existing motions into a path and a residual thereby allowing the path to be replaced with another while the residual serves as the detail of the motion. The technique is similar to the displacement mapping illustrated by Bruderlin & Williams however not only is the overall motion filtered out, so too are the characteristics. However, Gleicher considers the side effects associated with signal processing in more detail by looking at both the arc-length parameterisation problem and repairing any broken constraints.

2.6.3 Additive

The addition of injury simulation into base motions has been paid little attention in the field of character animation and modification, which has largely been due to the lack of appropriate tools for such operations. However with the advent of the spacetime constraints approach and motion warping techniques, researchers have experimented with their modification techniques and applied them to basic injury simulation [Brud95, Popo99]. The approaches used to simulate injuries have been rather crude in that degrees of freedom are either eliminated from global equations or simply capped. For example, removing one of the knee degrees of freedom has been shown to result in a limping motion by Popovic & Witkin [Popo99] and Liu *et al.* [Liu05]. Despite the ease with which these systems appear to simulate injury, they are not specifically designed for this purpose therefore the results are not based on how and where the character is injured. In Chapter 4, a novel technique that makes use of a fast inverse kinematics solution to simulate injuries is presented, which are illustrated by applying it to character walks in order to produce limping motions. The process operates in real-time, thereby allowing dynamic injury simulation on the fly. Chapter 6 additionally shows how the novel dynamics-based reconfiguration process presented in this thesis, can be turned to producing injured looking motions without eliminating degrees of freedom.

The concept of conveying emotional qualities in character locomotion has been paid little attention in the field of character modifications, however, early work by Unuma *et al.* [Unum91, Unum95] considers the use of Fourier principles as a method to generate emotions. The basis of the Fourier technique was that motions could be segmented into the basic motion and an additional descriptive part that describes among other details, the emotional state of the character. In order to simulate varied emotions, Unuma *et al.* extract emotional characteristics from empirical data, such as motion capture devices, which are then represented as a Fourier series expansion. Using the library of emotional parameters, more complex portrayals are built-up by extrapolating and interpolating the Fourier factors in the frequency-phase domain, which are added back into the base motion.

As an alternative to Fourier principles, Densley & Willis [Dens97] proposed the use of posturing functions to achieve emotional figure animation. Similar to the work done by Unuma *et al.*, complex emotions are deemed to be readily constructible from a basis emotion set. The posturing functions proposed by Densley & Willis consider how areas of the body are affected by each of the basis emotional characteristics, which are combined to produce more complex expressions. This leads to a high-level control system that allows a character’s emotional postures to be easily defined by tuning the different base emotions.

Rose *et al.* [Rose98] present a completely different approach of representing emotion in human locomotion as a consequence of the Verbs and Adverbs interpolation system. Instead of trying to extract the essence of an emotion, motions that express different sentiments (but performing the same basic movement) are interpolated to obtain different degrees of emotional content. Unlike the previous work undertaken by Unuma *et al.* and Densley & Willis, the emotion is not added but a large library of movements displaying various emotional content are needed. Consequently, this technique does little towards helping increase motion reuse.

Popovic *et al.* [Popo99, Popo00b, Safo04] and Oshita & Makinouchi [Oshi01] have demonstrated the practical use of dynamics in modifying character motions as being a very useful tool when factoring in environmental additive changes. However, considering dynamics is much more computationally costly than other modification algorithms that have been mentioned. The work undertaken by Popovic *et al.* looks at the problem of modifying motion capture data while accounting for dynamics, using a spacetime dynamics solver over the complete motion. By way of decreasing the computational expense of performing full-body dynamics, Popovic *et al.* simplify the humanoid structure to larger rigid bodies, that say encapsulate a complete arm, which are subsequently further abstracted away to point masses and thus simplify the dynamics used to represent the complete system. Similarly, Safonova *et al.* [Safo04] reduce the computational costs by simplifying the complexity of the full-body to a lower-dimensional representation using a principal component analysis. However, when the resulting motion is rebuilt from the simplified version, artefacts start to creep into the motion such as foot-sliding and even non-realistic dynamics.

In order to simplify the complexity of the rigid body mechanics, Oshita & Makinouchi make use of proportional-derivative control servos and simulate new motions based on an original motion using a joint angular acceleration and zero moment point tracking system. The system is intended to generate a real-time, dynamically changing motion that mimics the base motion but reacts to physical impulses or forces. At the core of Oshita & Makinouchi’s technique is the concept of comfort and balance controllers where the former is used to relieve stress on joints that exceed muscle strength and the latter is used to keep the character balanced. Driven by proportional-derivative control systems, the original joint trajectories are tracked and output as the new motion. However, when there is an environmental impulse or force acting on the body, the output angular accelerations are altered by these controllers as an additive change. In order to calculate the change in angular acceleration, active and passive control strategies are used that aim to reduce the stress by altering a number of primary joints with the goal of maintaining balance. To obtain real-time results, Oshita & Makinouchi designed their control strategy based on human-like characters in a standing double-support phase. Therefore the technique only works when the character is stood, planted to the floor, which vastly restricts the application of this technique despite its initial attractiveness.

As well as modifying existing motions for varying environments, the use of dynamic simulation has been used to completely simulate the locomotion of virtual characters [Brog98]. Despite the advantages of such a technique, the results still look somewhat mechanical and lack the personal characteristics captured with motion capture technology. A further disadvantage of completely simulating human locomotion lies with the huge computational costs associated with solving such large sets of motion equations, as unlike motion editing, there is no base motion with which modifications can be performed, thereby lacking good starting postures.

In addition to creating or modifying existing motions, dynamic simulation has been further used to generate secondary motion in response to character movements [Obri00]. This added use of dynamic simulation results in not only the character behaving correctly for the environment but also the environment responding accordingly to the character and hence an even deeper level of realism. Unfortunately with present hardware and mathematical solving techniques, the dynamic modification of character locomotion coupled with environmental response is very much an off-line process. Further usage of dynamics to simulate different types of motions and modifications are presented in Chapter 6 of this thesis, which includes a review of the area in Chapter 5 before presenting a novel dynamics-based technique.

2.7 Summary

This chapter has presented the lifecycle of motion capture data with a slight bias towards optical devices because of its predominate favour in being used in video games and films. However, before the marker data can be used to drive animated the character skins, it needs to be converted into an articulated structure. For this purposes, a novel inverse skinning algorithm has been presented in this chapter.

Despite the very realistic results produced using motion capture data, the problem with it comes when modifications are required because of its inflexible nature. The types of modifications that may be needed have been defined as reconfiguration, adaptation and additive. The goal is therefore to make use of algorithms that achieve these adjustments in such a way as to make the resulting motion appear realistic. This is the main focus of this thesis.

Existing techniques to perform such modifications have been presented in this chapter. This has led to the exposure of the lack of appropriate tools to achieve some of the desired modifications. The modifications that are the focus of this thesis primarily reside within the reconfiguration category. The current level of achievement in reconfiguration is two-fold. The first uses proportional derivative control servos to produce an on-line individualisation of an existing motion [Hodg97], whereas the second uses a complete rigid body dynamics model to extract the actor's stylisation parameters [Liu05]. This thesis complements these techniques by presenting an alternative real-time technique in Chapter 4 that approximates a character's biomechanical information. A more accurate, yet off-line dynamics-based reconfiguration process is later described in Chapter 5, which has the ability to reconfigure motions without an example motion from the target actor, which is illustrated in Chapter 6.

The techniques that are later described in this thesis also illustrate a new approach in providing additive modifications in the form of injury simulation, which contrasts with the current techniques that remove or cap degrees of freedom to perform such changes [Popo99, Liu05].

Despite this chapter's focus being on motion capture data, many of the issues and solutions that have been discussed are also applicable to other forms of data, such as keyframing or procedural models. It is therefore important to devise suitable algorithms that will provide the tools necessary to make the process of motion modification much easier than for an artist to adjusting the posture of a character for each frame of an animation.

Chapter 3:
A Motion Capture Dataset

For the purposes of demonstrating and evaluating the motion modification algorithms presented in this thesis, 4 different actors (all males) were motion captured at Simula, Bradford University, UK, using a 12-camera Vicon optical system. The collections of motions that were captured are outlined in section 3.1. Section 3.2 presents the biomechanical information of each of the actors captured. The biomechanical information is used to derive parameters for the motion reconfiguration processes, where the motion of one actor is mapped to another using only the biomechanical information of the target actor. Section 3.3 describes gait signatures. These are used in the evaluation of reconfigured motions using inverse kinematics (Chapter 4) and dynamics (Chapter 6). The gait signatures of motions are used to demonstrate the similarities of the motions of the same actor, while illustrating dissimilarities in the motions of different actors. Thus the gait signatures provide a suitable method of comparing the real motion of a target actor with a motion that has been specifically reconfigured to the target's biomechanical information using the source motion of one of the other actors.

3.1 Data for 4 Actors

Each of the 4 actors performed the motions listed in Table 3.1. The complete dataset therefore consists of sets of very similar motions performed by every actor. Figure 3.1 visually demonstrates the paths taken, where any motions in Table 3.1 that do not turn are recorded along the forward walking motion line given in Figure 3.1.

Motion Description	Number of motions recorded per actor
Walking normally in a straight line	4
Walking slowly in a straight line	4
Walking straight then off to the right	2
Walking straight then off to the left	2
Walking straight then tightly turning off to the left	2
Walking straight then tight turning off to the right	2
Jump forward from a standing posture	1
Simulated left leg limp	2

Table 3.1: Collection of identical motions performed by each of the 4 captured actors

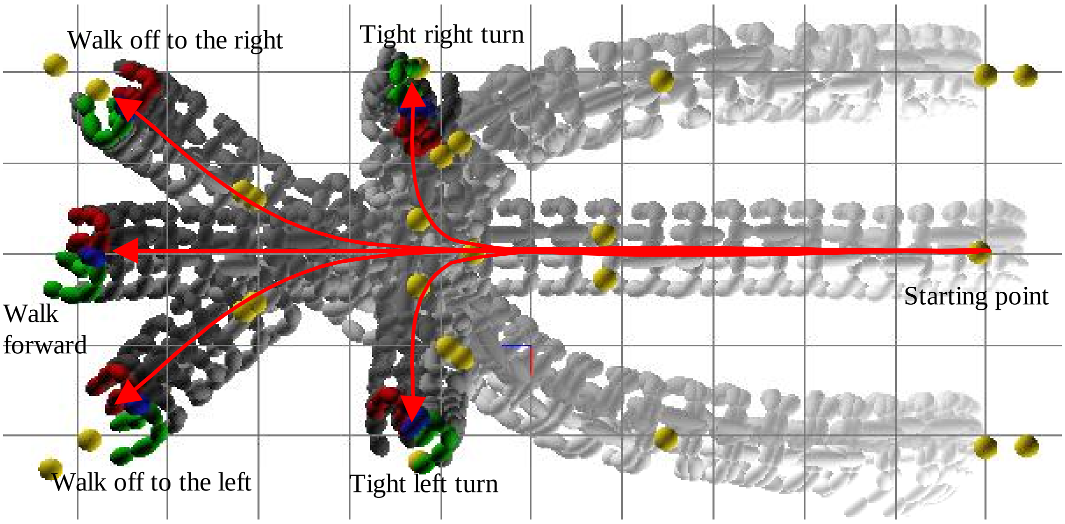


Figure 3.1: Motion paths taken during the acquisition of the motion capture data for the 4 different actors

The raw positional optical motion capture data (which was recorded at 100fps) is converted into animated hierarchical skeletons using the Inverse Skinning algorithm from section 2.3. In addition to the raw joint configuration space captured by the motion capture equipment, measurements of the actors were taken, which allows the limb masses to be extracted and used in the dynamics solver.

3.2 Biomechanical Information of the Real Actors

The limb weights are derived from the principle that the volume of the human body is directly proportional to its mass [NBDL88]. The volumes of the limbs are calculated using the measurement of the actor’s dimensions including length and circumference at various points along each of the limbs. Using the volume estimates, the actor’s total weight is segmented into its constitute limb weights. Table 3.2 provides the limb lengths and Table 3.3 outlines the calculated weights of each of the limbs, where a complete breakdown of the recorded and derived metrics for each of the actors is given in Appendix D.2.

Limb Name	Actor A	Actor B	Actor C	Actor D
Foot	0.26 m	0.27 m	0.28 m	0.27 m
Lower Leg	0.46 m	0.51 m	0.48 m	0.50 m
Upper Leg	0.32 m	0.25 m	0.41 m	0.34 m
Hips/Pelvis	0.23 m	0.21 m	0.20 m	0.24 m
Chest	0.49 m	0.42 m	0.48 m	0.44 m
Upper Arm	0.19 m	0.21 m	0.31 m	0.29 m
Lower Arm	0.24 m	0.24 m	0.30 m	0.28 m
Hand	0.18 m	0.19 m	0.20 m	0.22 m
Neck	0.07 m	0.06 m	0.08 m	0.08 m
Head	0.19 m	0.16 m	0.22 m	0.20 m

Table 3.2: Limb length breakdown of the 4 motion captured actors

Limb Name	Actor A	Actor B	Actor C	Actor D
Foot	0.96 kg	1.04 kg	1.13 kg	1.00 kg
Lower Leg	3.39 kg	4.29 kg	3.02 kg	3.47 kg
Upper Leg	6.75 kg	5.12 kg	6.94 kg	5.65 kg
Hips/Pelvis	14.09 kg	18.04 kg	9.74 kg	12.80 kg
Chest	32.70 kg	46.54 kg	28.11kg	24.15 kg
Upper Arm	1.93 kg	2.50 kg	2.05 kg	1.83 kg
Lower Arm	0.97 kg	1.02 kg	1.22 kg	0.87 kg
Hand	0.36 kg	0.42 kg	0.44 kg	0.42 kg
Neck	0.75 kg	0.78 kg	0.83 kg	0.73 kg
Head	3.64 kg	2.77 kg	4.63 kg	3.72 kg

Table 3.3: Weight breakdown of the 4 motion captured actors

Figure 3.2 illustrates the builds of the characters given in Table 3.2 and Table 3.3 where the size of the cylinders directly represent the dimensions and hence mass of the real actor. The top and bottom circumferences of the cylinders are representational of the corresponding circumferences of the real actor's limbs, which are given in Appendix D.2.

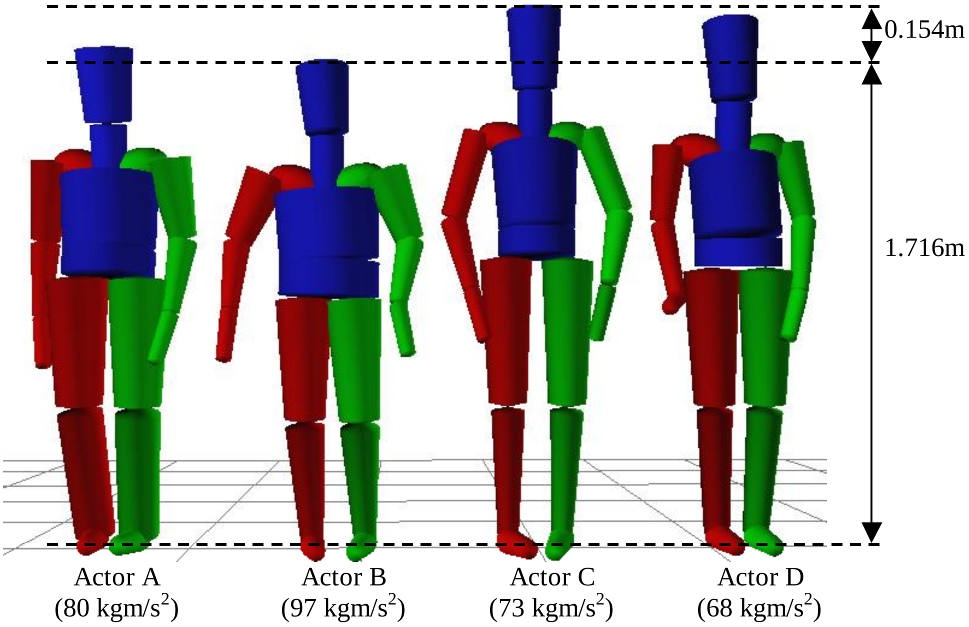


Figure 3.2: Actor builds of 4 motion captured males. The cylindrical dimensions represent the true proportions of the original actor as given in Appendix D.2

The visual representation shown in Figure 3.2 will become the standard in this thesis when presenting illustrations of motions that have been directly modified with respect to the character's biomechanical information. Conversely, any motions that have not been modified or only kinematically modified will be presented as ellipsoids representing the length of the limbs, where each ellipsoid will have a constant circumference so no character mass/build can be inferred. The non-mass representations are illustrated in Figure 3.3.

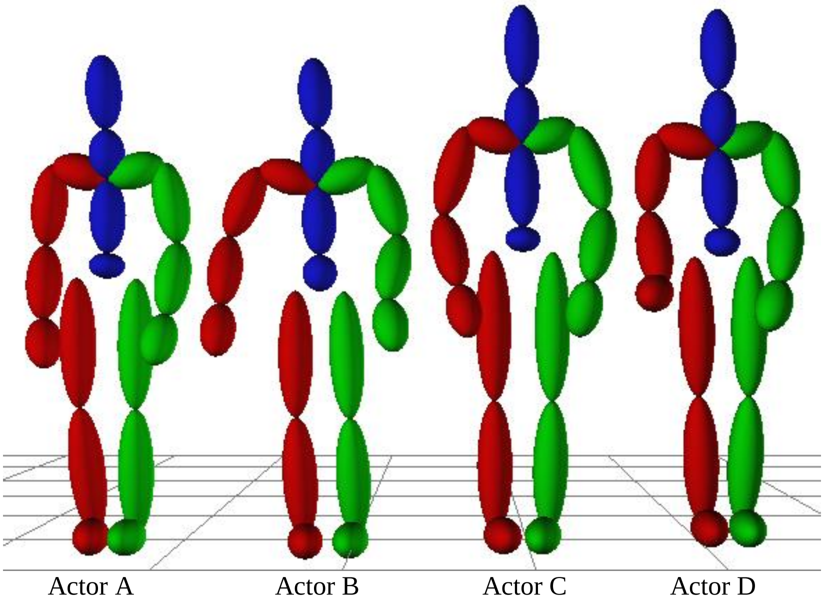


Figure 3.3: Actor non-build representations of 4 motion captured males. The ellipsoid lengths are representative of the limb lengths only of the actor as given in Appendix D.2

In terms of the actor's movements, the recorded movements of actor A and C demonstrate a similar speed in their execution where actor A demonstrates a slightly more forceful motion due to the size differences, i.e. as actor C is taller he can use his extra leg length to achieve the same speed as the shorter actor A with more ease. Actor D is slightly slower than actor A and C and demonstrates a similar ease in motion as actor C. In contrast, actor B is much slower in executing his motions, which has a much more relaxed manner about them than any of the other motions by actors A, C and D.

3.3 Comparing Between Motions

Due to the relatively short walking area used (which was approximately equal to 4 complete left leg strides of actor C), the same motion of the actors was recorded multiple times to ensure that the motions that are to be used to evaluate the modification techniques are consistent for a given actor. The consistency for a specific actor is illustrated in Figure 3.4 where the left leg gait signatures for actor C's 4 walking forward motions are given. The gait signatures of Figure 3.4 plot the relationship between the hip and the knee joint over the course of the motion, which is a metric that is frequently used in gait recognition [Gree04, Gosw98, Yoo03]. Figure 3.4 additionally includes the principal component analysis of the hip-knee joint relationship.

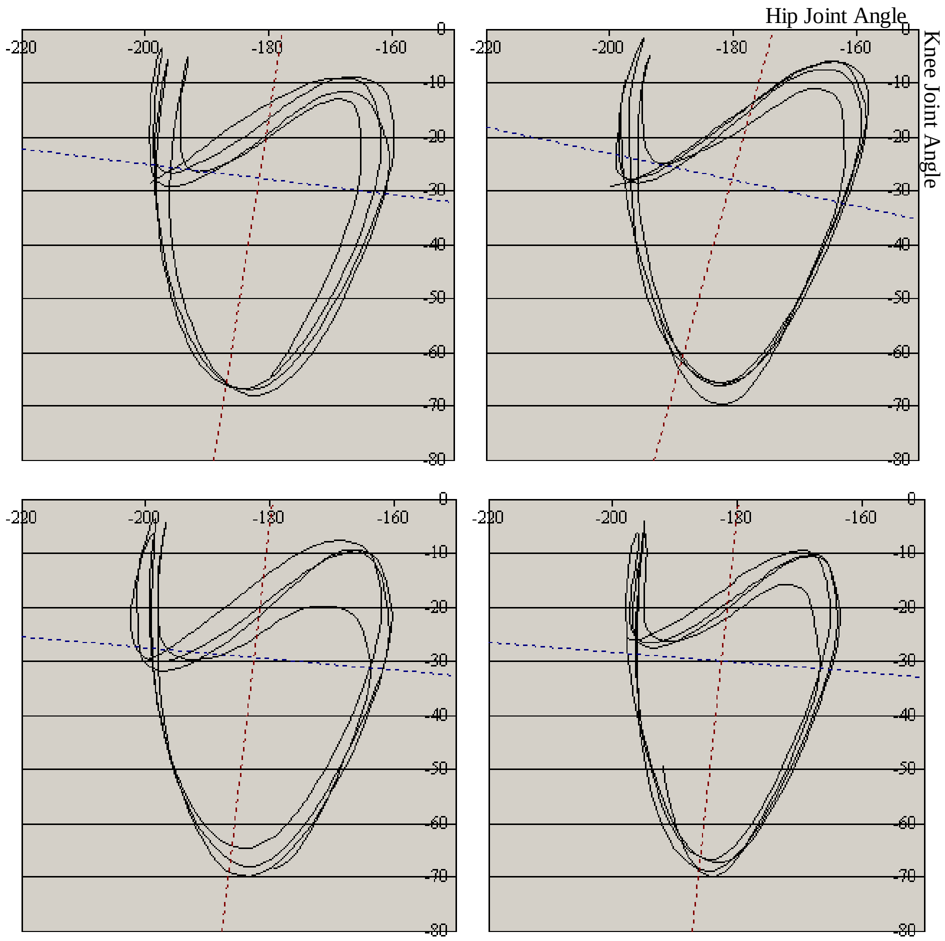


Figure 3.4: Left leg gait signatures of Actor C's walking forward motions including their principal component analysis given by the blue and red dashed lines

The consistency shown in Figure 3.4 for Actor C is similarly seen in the multiple gaits of the other actors. However, the shape of the gait signature is significantly different between actors, which is illustrated in Figure 3.5. The principal component analysis of each actor's gait signature is given in Figure 3.5 as red and blue lines. The green dashed lines in Figure 3.5, which overlay the gait signatures, indicate the local minimum and maximum joint values around the gait signature. These feature points have been averaged over the gait cycles of the diagram.

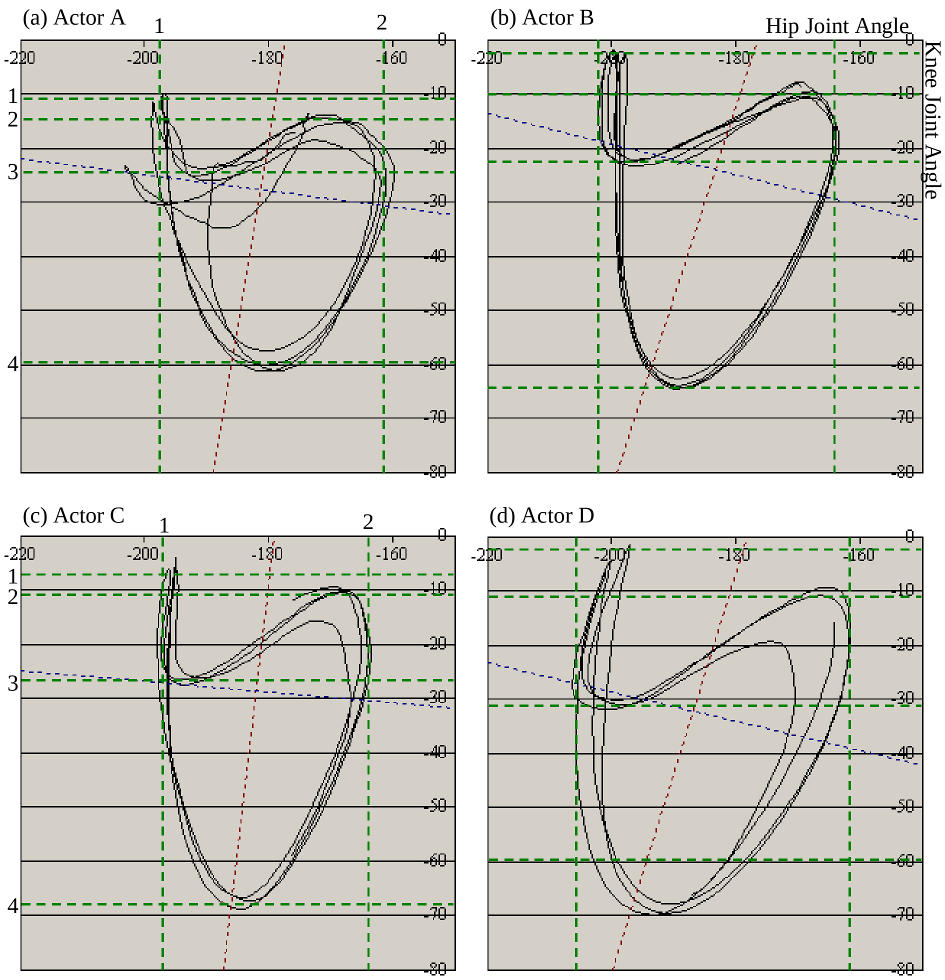


Figure 3.5: Left leg gait signature of (a) Actor A, (b), Actor B, (c) Actor C and (d) Actor D including their principle component analysis (red and blue lines) and their local minimum and maximum feature points indicated as green lines

Despite the shapes and local minimum and maximum values being very different between the actors in Figure 3.5, the principal component analysis of the gait signatures exhibit very similar eigenvectors. This is due to the gross proportions of each gait signature taking on an overall similar shape between actors, thereby failing to capture the differences in local minimum and maximum that distinguish the individual gait signatures. Therefore, in this thesis, when gait signatures are used to compare the differences in motions, the local minimum and maximum feature points (i.e. the green dashed lines) will be used because this metric better demonstrates the differences between different actors than the principal component analysis.

Chapter 4:
Modifying Motion Capture Data Using Inverse Kinematics

Using a hierarchical data structure, the job of inverse kinematics (IK) is to determine a suitable set of joint angles that allow an articulated structure to be postured with respect to meeting a specific spatial location. Usually it is the end-effectors that are positioned in space however any limb within the hierarchy can be positioning using the technique. The mathematical representation of inverse kinematics is repeated from section 2.5.2 in Equation 4.1, where q represents the complete set of orientation and translation values for a kinematics structure and X is the global position of a given end-effector in the hierarchy.

$$q = f^{-1}(X) \tag{4.1}$$

The formulation of inverse kinematics is a general mathematical solution that can be applied to arbitrary hierarchical structures. However, this thesis focuses on its application in the area of modifying motion capture data. This chapter presents novel implementations of inverse kinematics techniques, which result in original applications of the technique to reconfigure and introduce additive effects into existing motion capture data.

The problem with inverse kinematics is that Equation 4.1 is inherently underdetermined and so there is no single solution to the problem – this is both true algorithmically speaking as well as from a results point of view. Therefore, a variety of techniques have been posed over the years to solve Equation 4.1. A review of these techniques is given in section 4.1, which spans both the field of robotics, where the problem was first posed, and computer graphics in which many of the ideas are borrowed from robotics. Section 4.1 does however have a slight bias towards techniques used specifically in the field of character animation and modification.

The main focus for the rest of this chapter is based on the Jacobian-based technique for solving the inverse kinematics problem. Therefore, section 4.2 provides the foundations for a practical implementation of the technique. Section 4.3 performs an analytical complexity analysis of the algorithm given in section 4.2. This compares the difference between using an end-effector with and without orientation (position is always present), which are distinguished by their names, full- and half-Jacobian respectively.

So that the analytical results may be reinforced in a practical setting, a harness application called MovingIK is described in section 4.4. MovingIK is an IK driven simulated character walking application, which can switch between using the half- and full-Jacobian techniques. However, before MovingIK can utilise the half-Jacobian for character gait, section 4.5 discusses an innovative approach that allows traditional full-Jacobian problems, such as humanoid walking, to be converted into problems that do not consider orientation at the end-effector and hence solvable by the half-Jacobian implementation.

In addition to the conversion process from the full-Jacobian to the half-Jacobian, section 4.5 presents the empirical results of comparing these two techniques for animating a humanoid character in MovingIK. These results correlate with the theoretical ones. This conclusion is that not only is the half-Jacobian much more computationally efficient, but also that full-Jacobian problems can be mapped into half-Jacobian ones to make use of this improved efficiency [Mere04a]. This is further reinforced in section 4.6 where the half-Jacobian is used to retarget motion capture data.

Section 4.7 further builds upon the practical Jacobian solution of section 4.2 and the harness application of section 4.4 to demonstrate the novel inclusion of Weighted Inverse Kinematics into the Jacobian-based solver. Weighted inverse kinematics provides the ability to control the contributions of limbs within the inverse kinematics chain such that some move more slowly than others in an intuitive manner. The effect of applying weighted inverse kinematics to procedural humanoid gaits is illustrated at the end of section 4.7 (and has been published as [Mere04b]).

The individualisation of motion capture data using half-Jacobian weighted inverse kinematics is demonstrated in section 4.8 (and has been published as [Mere04c, Mere05]). This process demonstrates how a single base motion can be used to spawn many different motions that exhibit different characteristics that are indicative to the build of the character or even noticeable injuries. Furthermore, the weighted inverse kinematics solution is implemented at negligible computational cost to the basic algorithm, which runs in real-time.

Section 4.9 builds upon the motion individualisation of section 4.8 by mapping the motion of one actor to a completely different actor using weighted inverse kinematics. The weighted inverse kinematics parameters are based on the biomechanics of the target actor, but are not used in a dynamics sense, i.e. the approach is purely a kinematics-based technique that has no knowledge of dynamics. The weighted inverse kinematics reconfiguration process is evaluated in section 4.9 by comparing the generated motions against the real movements of the actors described in Chapter 3, who provide the basic biomechanical information.

Hence, this chapter shows that the half-Jacobian weighted inverse kinematics technique is a very computationally cheap and real-time mechanism of performing motion reconfiguration (both retargetting the character and adding in stylisation effects).

4.1 Solving the Inverse Kinematics Problem

Inverse kinematics is an under-defined problem and hence a direct application of analytical solutions rarely proves useful or even possible to contrive. However in instances where it is possible to determine an analytical solution, the approach provides the most reliable, and potentially quickest basis for generating results. Conversely, numerical-based iterative approaches are able to handle IK chains of arbitrary length, which attempt to resolve the problem via a step-wise convergence towards a solution. This classification of IK solver can be further broken down into subcategories that are characterised by the method with which they determine what the next step should be towards a solution. One variety is through the use of heuristics while another is by analysing the rate of change of the articulated structure itself, termed Jacobian based. Unfortunately however, iterative-based techniques can be less reliable than analytical applications because of problems with local minima and numerical inaccuracies that are carried through iterations.

The other main type of inverse kinematic solver is that of hybrid techniques, which make use of a mixture of IK solvers to best serve a purpose Furthermore, in addition to the techniques that directly tackle the IK problem, there are also approaches that indirectly address the problem of positioning an end-effector with only regard to the kinematics of the body. In the continuation of this chapter, these techniques are subsequently referred to as indirect methods.

The following subsections address each of these classifications in the order:

- Analytical solutions
- Direct iterative-based solvers
  - § Heuristic
  - § Jacobian
  - § The SHAKE algorithm
  - § Optimisation
- Hybridised techniques
- Indirect methods

4.1.1 Analytical Solutions

Geometric/analytical algorithms [Chin96, Paul88, Watt92] define direct mathematical relationships between end-effector locations and joint space configurations, which can be evaluated in a single step to yield the result. This is exemplified in Figure 4.1 for a two-linked system, where Equation 4.2 gives the end-effector location in terms of joint angles, and Equation 4.3 rearranges this to obtain equations for the joint angles.

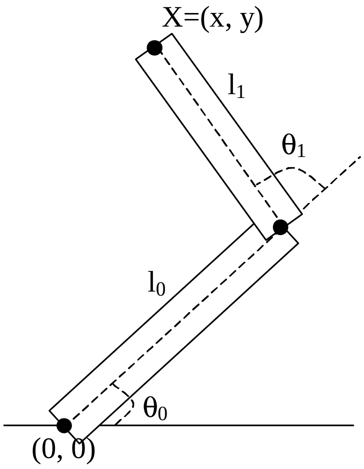


Figure 4.1: Analytical solution to a two-linked chain (based on Watt & Watt [Watt92])

$$X = \begin{bmatrix} l_0 \cos q_0 + l_1 \cos(q_0 + q_1), \\ l_0 \sin q_0 + l_1 \sin(q_0 + q_1) \end{bmatrix} \tag{4.2}$$

$$q_1 = \cos^{-1}\left(\frac{x^2 + y^2 - l_0^2 - l_1^2}{2 l_0 l_1} \right) \tag{4.3a}$$

$$q_0 = \frac{(l_0 + l_1 \cos q_1) y - (l_1 \sin q_1) x}{(l_1 \sin q_1) y + (l_0 + l_1 \cos q_1) x} \tag{4.3b}$$

Equation 4.3 illustrates how the mathematical definitions of just two joint angles becoming rather complex. Furthermore, the analytical solution fails to capture there are two unique solutions to the end-effector position, and chooses the one that enforces a joint angle between 0 and 180 degrees for the second joint angle, q_1.

Due to the direct mathematical relationship, analytical solutions tend to be very quick at producing a suitable joint space configuration for a desired end-effector location. However, the limitations of this class of solver become quickly apparent as the chain size increases. In such cases, the task of reducing the problem to a single step mathematical equation is impractical or even impossible because of the under-defined nature of the problem. The current limit of practical analytical solutions has rested at a 7-DOF chain [Tola96], which has been applied to a human arm. This technique imposes a constraint that the second link exhibits only one DOF and hence is not general, although it has been successfully used as a standalone analytical technique in areas such as hand gesture retargetting [Ge05]

Kovar *et al.* [Kova02] utilised a purely analytical inverse kinematics solution to clean up motion data that exhibited foot skating. This approach however has the ability to break the rigidity of the IK chain and mildly vary the bone lengths, which potentially introduces its own artefacts, whilst clearing up the foot sliding ones. A further drawback with this approach is that, like all analytical methods, the technique lends itself well to only one specialised instance and lacks scalability or generality.

By themselves, geometric/analytical techniques tend to be less useful in the field of character animation than the other classes of solvers that shall shortly be discussed. This is due to the high level of DOFs that are required in a chain, and the generality that allows the modelling of arbitrary creations/articulations.

4.1.2 Direct Iterative-Based Solvers

When it becomes impractical to represent an inverse kinematics problem with an analytical solution, the problem can be locally approximated and thus linearised using a numerical-based technique. This comes at the cost of having to iterate towards a solution using different techniques to determine the step vector per iteration.

The following subsections discuss the different types of iterative-based solvers, which each have their own technique of determining the step vector.

4.1.2.1 Heuristic-Based Iterative Solutions

Heuristic-based iterative techniques [Eber01, Wang91, Welm93], which are also referred to as cyclic coordinate descent techniques (CCD), provide a comparatively quick determination of the step vector through the use of a heuristically determined metric. Within the main iteration of the CCD technique, there exists a sub-iteration over the joints, which are subsequently independently updated one-by-one, starting from the end-effector and moving down the chain towards the root node. The update to the current node persists as the new system state to the next node down the chain, i.e. the updates are performed immediately within the sub-iteration and not once a complete sub-iteration is complete. Through the process of dealing with each node in the chain in turn, the problem of determining how best to update it to achieve the global goal is thus simplified and consequently allows us to formulate a geometric relationship per link, which can provide relatively quick results.

The heuristic typically used in CCD methods involves minimising the spatial difference between the desired and current position and orientation of the end-effector with respect to the given link only, keeping the rest of the chain constant. Since it is only a single node that is being dealt with at a time, the minimisation process becomes linear [Welm93] and hence a desirable direct analytical heuristic is obtained. An illustration of how this technique works is given in Figure 4.2, where only the top three links are altered in order to approximate the end-effector location, which leads to an unnatural sharp kink in the chain.

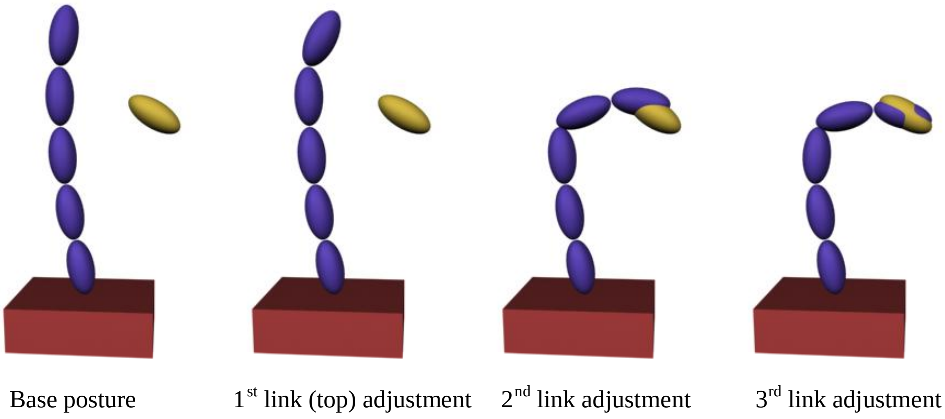


Figure 4.2: Cyclic Coordinate Descent Inverse Kinematics Solver – the purple chain is modified to meet the position and orientation of the gold spheroid

The CCD approach is a quick iterative approach because of the sub-analytical solutions, although this can on occasion come at the cost of accuracy; generally the algorithm will produce good results to a level of accuracy required, but care needs to be taken when determining the heuristic to try and reduce the resulting degree of error. However, the main problem with this technique comes when applying it to the field of character animation. The issue stems from the way in which it breaks-down the problem into smaller, more manageable parts, in that it always favours changing the earlier nodes most to reach a target. In effect, the technique will always produce a result that moves as few links as possible. Consequently, this leads to the undesirable and unrealistic result of earlier joints always moving much more than the later limbs in the IK chain, which may not even change in their orientation.

4.1.2.2 Jacobian-Based Iterative Solutions

Jacobian-based inverse kinematics solvers are well established having started out in the field of robotics [Crai55]. The fundamental principle of this technique is to iterate towards a desired solution by incrementally changing the joint configuration state, starting from a stable state. The amount of incremental change per iteration is defined by the system Jacobian, which relates small changes in joint configurations to positional offsets. Thus it is a differential-based approach. The Jacobian is defined as a matrix with dimensionality *(m $_x$ n)*, where *m* is the spatial dimension of *X* and *n* is the size of the joint configuration set, *q*, as given by Equation 4.4.

from Equation 2.9 $X = f(q)$

taking partial derivatives $dX = J(q)dq$ **(4.4a)**

where $J_{ij} = \dfrac{\partial f_j}{\partial X_i}$ **(4.4b)**

Rearranging Equation 4.4a provides Equation 4.5. This transforms the generally under-defined IK system into a linear one that can be solved using incremental iterative steps.

$$dq = J^{-1}dX \qquad \textbf{(4.5)}$$

The differential Jacobian-based technique is illustrated in Figure 4.3 where, upon each iteration, the whole chain is smoothly updated to give a more natural curve in the resulting solution. This is compared to the CCD technique of Figure 4.2. The postures of Figure 4.2 and Figure 4.3 do not overly exaggerate the comparative unnaturalness between the Jacobian and CCD techniques. However, this simple example serves to highlight the underlying problem, which becomes more apparent as the IK chain is continually perturbed to meet new end-effector goals.

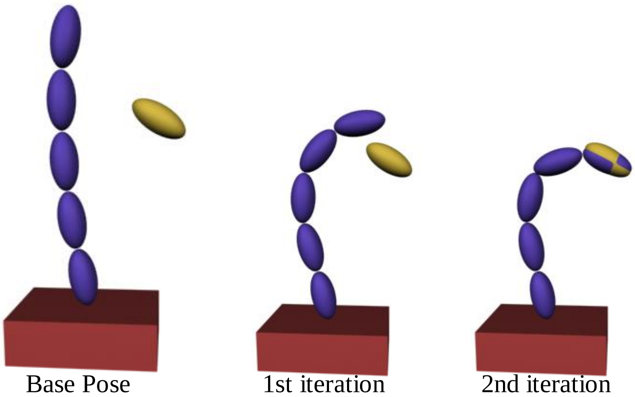


Figure 4.3: Jacobian-based inverse kinematics solver – the purple chain is modified to meet the position and orientation of the gold spheroid

Equation 4.5 now requires the inversion of the Jacobian matrix, which is very rarely square. The inversion of a non-square matrix can however be accomplished using a variety of different techniques such as right/left-handed generalised inverse or singular value decomposition [Pres92].

As the algorithm iterates towards a solution, at each step the Jacobian is determined using Equation 4.4, and all the joint angles are updated simultaneously. This dissipates the change in joint angles over the whole chain, which results in a more realistic looking posture when compared to CCD techniques.

Jacobian-based techniques have long since been utilised in computer graphics [Gira85, Madh98, Phil91] where at times its appearance of Equation 4.5, has been adjusted. One of the more predominate forms of this has been to include a null-space term that makes use of any redundancy within the Jacobian inverse:

$$dq = J^{-1}dX + (I - J^{-1}J)dZ \tag{4.6}$$

where *dZ* is the null-space update vector.

The inclusion of the null-space term allows a secondary goal to be embedded within the IK algorithm, which serves like an optimisation function as opposed to the hard-constraint of the primary IK goal. Such secondary goals have included obstacle evasion [Espi85], joint limit avoidance [Boul92], centre of mass control [Boul96], motion rate control [Teva00] and posture minimisation [Monz00]. The latter implementation is successfully used to retarget motion data between characters, which have varying geometrical or topological structures, while simultaneously attempting to remain faithful to the original per-frame motion captured posture.

Another adaptation of Equation 4.5, which has come from the field of robotics, has been presented in the form of a closed-loop inverse kinematics technique (CLIK) [Chia91, Scia96, Sici99]. The formulation of this is given in Equation 4.7, where K_x is a positive definite matrix and e_x gives an error term defined by the difference between the desired and actual end-effector position.

$$dq = J^{-1}(dX + K_x e_x) \tag{4.7}$$

The error term in the CLIK algorithm exponentially converges to a zero steady state for a fixed target position. Consequently, this allows the CLIK algorithm to steadily converge on joint angles that give rise to the joint configuration required to meet an end-effector position. The importance of this convergence property is realised when considering chains that approach a configuration that results in singularity, which is a problem for all numerical methods. The CLIK algorithm is more immune to these problems because the secondary terms can compensate for the loss of freedom in such cases.

The combination of Equation 4.6 and 4.7 are brought together by Choi & Ko [Choi00], who demonstrate an on-line motion retargetting system. The redundancy of Equation 4.6 utilises an inverse rate control process [Whit69] to provide a filtered per-frame IK solution, which attempts to match the trajectories of the original joint angles. However, the combination of the techniques traded off some of the stability given in Equation 4.7 with inclusion of the secondary goal of Equation 4.6. The result of this system is the ability to determine the IK importance factors (i.e. when node location is preferred over preserving joint angles) as an end-effector approaches a virtual object in real-time.

The Jacobian-based approach of solving the IK problem is further elaborate in section 4.2 as it forms the core of the original half-Jacobian technique in section 4.5 and the novel weighted inverse kinematics approach presented in section 4.7.

4.1.2.3 The SHAKE Algorithm

An alternative iterative-based approach proposed by Tang *et al.* [Tang99] makes use of the SHAKE algorithm [Ryck77] to achieve a fast iterative-based IK solver. This technique treats a hierarchical structure as point masses that are related by system constraints. This is in contrast to the Jacobian-based and CCD technique that encapsulates the articulated information and thereby provides us the cohesion between links for free.

In order to achieve a desired end-effector location, the mass points of the SHAKE system are adjusted per cycle until a global goal has been reached. This includes meeting a threshold of acceptable error on the constraints. However, because of the lack of node dependency of the algorithm, normally the points will lose their distance relationships between each other. To counter this issue, correcting forces are iteratively applied to each point to reassert cohesion between links. Therefore the accuracy of parent-child distances directly effects solver time. Without a reasonable level of accuracy, the appearance of rigid links moving together and away from each other would surface. This is an issue that the Jacobian-based techniques are not affected by.

The time complexity of the SHAKE algorithm is suggested by Tang *et al.* to be $O(n^2)$ with respect to the number of constraints. Therefore, since each link in a hierarchical chain requires a constraint to impose cohesion, the time to solve a system is also minimally $O(n^2)$ with respect to the number of links in the chain. The inclusion of additional system constraints such as joint angle limits has a further detrimental effect on solution time, thus making the algorithm less applicable to real-time applications as the number of links increases.

4.1.2.4 Optimisation-Based

Optimisation-based inverse kinematics solvers [Zhao94] cast the problem of Equation 4.5 as a non-linear constrained optimisation problem, whose formulation follows that given in Equation 4.8.

$$\begin{aligned} &\textit{minimise} \quad R(q) \\ &\textit{subject to} \quad a_i(q) \text{ £ } b_i \end{aligned} \qquad \textbf{(4.8)}$$

The equality/inequality constraints, $a_i(q)$, of Equation 4.8 are used to achieve secondary tasks during the optimisation process, such as enforcing joint limits, while the objective function, $R(q)$, defines the goals of the system. The objective function is thus defined through the forward kinematic relationship of the end-effector and the desired goal position, by taking their squared spatial difference. By encoding this difference in the optimisation function, the solver is more able to cope with unreachable configurations than if the difference had been directly encoded as an equality constraint.

The reformulation of the inverse kinematics problem into an optimisation process means that it is no longer as straightforward to evaluate as the other numerical techniques discussed. However suitable techniques are readily available to solve problems of the type presented in Equation 4.8 [Gill81], one of which is discuss in section 5.3. This approach however, is an iterative one, requiring a descent on the final solution from a given starting configuration.

The main disadvantage of using such a technique for performing inverse kinematics is that by the nature of minimisation formulations, a local minimum can frequently be returned as the solution to Equation 4.8, whereas there is a better global minimum that the system is unable to move toward. However, as discussed by Zhao & Badler [Zhao94], as long as the starting point is reasonably close to the global minimum, the technique will produce good results. However, as a backup, the implementation presented by Zhao & Badler allows a user to tweak the configuration to push it into a more suitable configuration space that can be used as a starting point to reapplying the optimisation process.

Komura *et al.* [Komu01a] present a technique that brings together the Jacobian- and optimisation-based techniques, where they make use of the null-space of the Jacobian solution to include muscle forces. The minimal muscle forces are first determined using an optimisation process, which requires a quadratic programming problem, similar to the technique presented by Zhao & Badler. However, the optimisation process is only with respect to the muscle models. The joint angles derived from the muscle optimisation process are subsequently used with the Jacobian of the system to update the configuration state.

Through the inverse kinematics solution Komura *et al.* use, they are able to limit the solution space that the muscle dynamics permits. The result of including muscle dynamics is to allow for more natural-looking configurations compared to the simpler inverse kinematics forms, although this comes at the cost of added computational cost required to handle the optimisation process. Komura *et al.* do not demonstrate their inverse kinematics solution with varying muscle forces to give more depth to the stylisation that might result; i.e. to distinguish between characters of different muscular builds. However, in an alternative work by Komura & Shinagawa [Komu01b], they demonstrate how to attach different muscular effects to characters. The process of applying muscular dynamics to the character is not an inverse kinematics solution and therefore an in-depth review of the work is deferred until the more appropriate dynamics section of this thesis in Chapter 5. For the immediate purposes, the information taken from the work is that the effect of applying different muscle characteristics to varying characters does limit the range of motion and hence break end-effector locations. From the equations presented in [Komu01a] and based on the latter evidence [Komu01b], it is unlikely in its present form that the dynamics grounded inverse kinematics solution presented can produce different styles for characters of different builds as this would lead to end-effector positions either not being reached or produce overly-strained unnatural postures. In effect, the technique only guides the inverse kinematics solution towards a minimal muscular dynamics change, without the power to add further refinement.

4.1.3 Hybridised IK Techniques

Analytical solutions provide a fast, single step solution to the IK problem, however they fail to scale up to complex structures. Conversely, numerical solutions scale up but are not as quick at producing results. A further problem that analytical solutions suffer from lies in their inability to incorporate additional constraints, such as joint limits, unlike their numerical counterparts. However, it is possible to marry these different solutions to develop a hybrid technique that attempts to take the best from both worlds.

Lee & Shin [Lee99] make use of a hybrid technique as part of their motion-adapting algorithm, taking motion capture data from one character and mapping it onto another. The analytical part of their inverse kinematics solution models the arms and legs [Tola96] of the character, as illustrated in Figure 4.4.

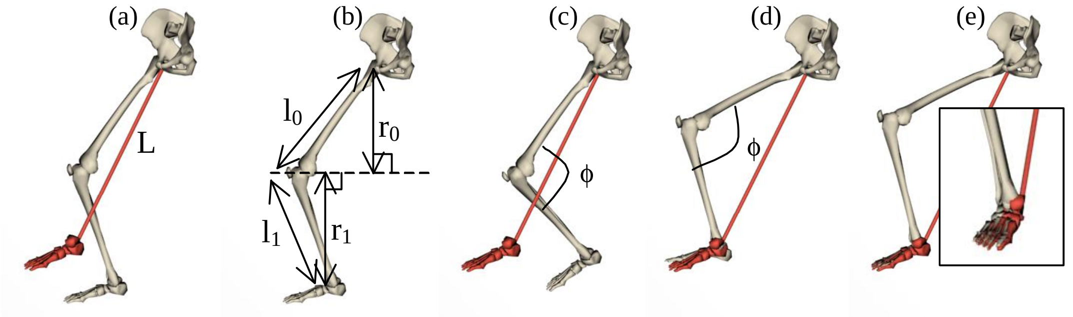


Figure 4.4: Analytical leg posturing using the constraint of only 1 knee degree of freedom

Taking the hips as the root node, the aim of the analytical component is to position the foot at the location indicated by the red foot plant in Figure 4.4a. Since there is only 1 DOF at the knee joint, the distance between the hip and foot, illustrated by the red line in Figure 4.4a, can uniquely determined this angle using Equation 4.9.

$$f = \cos^{-1}\left(\frac{l_0^2 + l_1^2 + 2\sqrt{l_0^2 - r_0^2}\,\sqrt{l_1^2 - r_1^2} - L^2}{2r_0 r_1} \right) \tag{4.9}$$

Figure 4.4c fixes the angle given in Equation 4.9, whereupon the upper leg is rotated to position the foot in the desired location, as illustrated in Figure 4.4d. The foot angles are finally adjusted in Figure 4.4d to meet the required orientation of Figure 4.4a.

This analytical representation of these limbs reduces the degree of freedom count within the system, thus presenting a simplified problem to the numerical algorithm, which is based on an iterative optimisation process.

Tolani *et al.* [Tola00] build upon their earlier work [Tola96] to demonstrate how the fixed form of a human arm or leg can be analytically represented, while additionally abiding by a set of system constraints within the solution. The constraints are cast as an iterative optimisation process that works in conjunction with the analytical representation to serve the secondary tasks. Lee and Shin [Lee99] focus on only adapting the motion, therefore the feature of constraints are absent from their work as they are not required.

Shin *et al.* [Shin01] made further use of a hybrid technique for time-critical, full-body computer puppetry. This technique attempts to use an analytical solution where possible. However, in the case where there is a large amount of body posturing required, a numerical implementation is invoked. The analytical descriptions of the IK problem are derived from the earlier work of Tolani *et al.* [Tola96], and are therefore similarly specialised to the fixed form of a 3-linked, 7-DOF structure, where the middle link only has 1 degree of freedom. As a result, the numerical solver has to be used for the IK chain defined between the root and the upper body, while the analytical solver can be used for the legs and arms of the body. If there is a large amount of posturing required, analytical solution makes way for the numerical method.

All of the hybrid techniques discussed demonstrate good approaches for performing real-time IK. In particular, Tolani *et al.* [Tola00] demonstrated a technique that was faster at producing a solution than a comparative Jacobian process. However, in each case the analytical aspect assumes knowledge about the character’s structure. This means that the overall IK technique is highly configured to a specific topology of a human arm or leg, with little scope of deviation and therefore is not general enough to be scaled up to IK chains of arbitrary structure. The other potential problem associated with the hybrid techniques are that, similar to the CCD technique, not all joint angles are updated simultaneously which means unrealistic and unproportional posture configurations could result.

Therefore, hybrid techniques have shown good results in posturing specific configurations in terms of both visual appearance and speed. However, similar to the underlying analytical techniques they make use of, hybrid methods still lack fundamental flexibility.

4.1.4 Indirect Methods

Rose *et al.* [Rose01] treat the problem of inverse kinematics as a process that interpolates between existing postures, using radial basis functions, to give a suitable configuration that meets an end-effector goal. Since blending between motions is a very quick, single step process, one of the advantages this technique offers is the speed with which a solution can be produced. In addition, stylisation aspects can be conveyed onto the motion using the corresponding base-motions. For example, using base motions that exemplified a character lifting both heavy and light objects, the corresponding set of motions, termed Verbs by Rose *et al.*, can be blended together to produce a new character picking up an object but demonstrating the appropriate effort for the object.

As opposed to using radial basis function blending, Mukai & Kuriyama [Muka05] choose to interpolate between motions using a statistical optimisation process to achieve a similar effect as Rose *et al.* This approach predicts the continuous distributions of sampled interpolated data and thus more reliably returns an appropriate blending to meet the desired target location.

Despite the quick generation of new motions, the drawback of the techniques presented by Rose *et al.* and Mukai & Kuriyama lies in the fact that they need several similar base motions to interpolate between. Generally, the sample motions need to be reasonably similar so that it is possible to determine the location of an end-effector after interpolation. Therefore, Rose *et al.* introduce pseudo-base motions, which are themselves interpolations of the original base motions, but determined off-line. These motions refine the range space of possible motions to interpolating with and thus aid the real-time inverse kinematics algorithm by demonstrating more predictable results.

Grochow *et al.* [Groc04] demonstrate an alternative approach to inverse kinematics where, similar to Rose *et al.*, they make use of existing motions to perform the task. This technique uses the motion data to train a Scaled Gaussian Process Latent Variable Model [Lawr04] (SGPLVM), which learns posture likelihoods. The SGPLVM technique subsequently relates the high-dimensionality configuration of a humanoid character to a low-dimensional space, termed the latent space, which uniquely determines a posture. The movement within the latent space therefore recreates a set of joint configurations. Keeping to the high-probability regions generates a likely posture given the data the system was trained upon.

When an artist wishes to posture the character by placing an end-effector, the SGPLVM model is searched for the most likely posture given the end-effector location and orientation. This is translated back into a complete set of joint configurations, thus resulting in a natural posturing for the complete body dependent on only a few end-effector placements. Additional user adjustment is made possible by moving about in the low-dimensionality latent space to slightly adjust the posture.

Grochow *et al.* demonstrate a very powerful tool for quickly posturing a character with only a few constraints where, similar to Rose *et al.*, the resulting configuration exhibits a stylisation that is similar to the sampled motions. However, the validity of the posture is dependant on the original motion data. Therefore, different SGPLVM models are required for fundamentally different motions so as to maintain a realistic posture. For example, if the motions of a character walking normally and a character playing an active sport were sampled by a single SGPLVM, the technique could switch between configurations for very similar end-effector postures. Likewise, the same is true in the technique presented by Rose *et al.*

The techniques of Rose *et al.* and Grochow *et al.* demonstrate a novel approach to the traditional inverse kinematics problem, which yields very realistic postures. However, both methods require specific knowledge of the motion that is to be generated, which comes in the form of motion capture data. While such data is becoming more prolific, some motions are simply impossible to capture (such as an actor falling from a tall building, as an extreme example), therefore these techniques would not easily generalise to all domains. Furthermore, for a specific domain, there is quite a large off-line overhead associated in setting up the models, which need to be regenerated each time a new kind of motion is desired, whether this is simply a new style of the same motion or a completely different motion.

4.1.5 Summary

From the research done in the field of inverse kinematics, it is apparent that analytical solutions by themselves are just not scalable enough to meet much of the demands of modern computer-based IK problems. The most advanced analytical solutions only currently extend to highly constrained 7 DOF chains, which although it proves to be a highly effective tool, are by themselves, limited in their application. This consequently leads to numerical-based techniques, which by their iterative nature take longer to produce a result but are better equipped to handle arbitrary configurations.

A bridge between analytical and numerical solutions has been presented in the form of hybrid solutions, which take advantage of the positives of both approaches. In such instances, the analytical solutions are used for subsets of the IK chain, and layered above that are the numerical techniques that operate on the simplified IK chain. The use of hybrid techniques allows the extension of analytical solutions to more complex articulated structures and because of the sub-analytical solution, suitable results are generated more quickly than using a numerical technique alone. However, the problem with hybrid solutions is that they are only as general as their constitute parts, which is persistently determined by the analytical component. Therefore, similar to purely analytical techniques, hybrid methods also fail to generalise to less specific hierarchies.

The indirect approaches to the problem of inverse kinematics have demonstrated similar results to hybrid techniques in their grounding in a specific domain, albeit at a more advanced level by introducing stylisation effects that are exhibited in their base-motions. As with the hybrid techniques, indirect methods provide a real-time solution to inverse kinematics (with a degree of pre off-line processing to formulate the sample data) and they demonstrate some excellent results, albeit specific to the sample motions. They are however more general than the hybrid techniques because it is possible to produce an alternative posture by sampling the appropriate dataset, whereas the hybrid techniques are stuck with their constrained analytical component.

The most general form of inverse kinematic solvers lie in purely numerical techniques, of which there are several complementary varieties. Fedor [Fedo03] explores the trade-off between speed, accuracy and scalability for IK solvers. One of the results from this work demonstrates that differential-based numerical solutions, although slower than both CCD and analytical techniques, provide better results for larger chains. This highlights the importance of refining numerical techniques such that accuracy and scalability are maintained but solution time is reduced.

An important fundamental bonus of using a purely numerical solution is that they can embody a whole hierarchical structure, which helps to avoid unrealistic postures. This is in comparison to the analytical or hybrid techniques where solutions are determined with respect to either smaller parts of the articulation or simplified versions. This is also a problem that the indirect methods tend not to suffer from since the posture is produced from existing real motions.

Another, more intriguing benefit that numerical techniques present lies in the opportunity to play with their inner workings to meet secondary goals or guide the solution for a given purpose. This facet has been extensively employed over the years and in the continuation of this chapter, a novel modification of a numerical-based technique is demonstrated to produce motion stylisation. It is through this extension, which is uniquely offered by numerical techniques, that inverse kinematics techniques have demonstrated a wealth of techniques that guide the IK solution towards a configuration that suits a specific higher-level goal, for example balance control [Boul96] or obstacle avoidance [Espi85].

Aside from the hybrid techniques, little work has gone into actually making the numerical approaches more computationally efficient. Alternative techniques, such as optimisation-based methods and the SHAKE algorithm, have been presented. However, in the case of the optimisation process, it is arguable whether this provides a more efficient solution for larger chains. This is because of the numerical complexity required in evaluating the minimisation problem, which non-linearly scales in complexity with respect to chain size. That is not to say the technique should be discounted because it does offer just as much scope, for affecting the solution through the simple addition of new minimisation and constraint functions for the purpose of secondary goals.

For the rest of this chapter, an optimisation to the Jacobian-based numerical technique is demonstrated for real-time application. This is further expanded in section 4.7 by illustrating a modification to the Jacobian technique that allows character motions to be controllably stylised.

4.2 A Practical Implementation of a Jacobian-Based Inverse Kinematics Solver

In this section, details concerning the implementation of a Jacobian-based inverse kinematic solver are presented, which provides the basis of the later work in this chapter.

In this implementation, a right-hand generalised pseudo-inverse is used to inverse the generally non-square Jacobian matrix, which is given by Equation 4.10. However, generating the Jacobian’s inverse in this manner can lead to inaccuracies in the resulting matrix that need to be reduced. Any inaccuracies in the inversion of the Jacobian can be detected by multiplying it with the original Jacobian and subtracting the result from the identity matrix. A magnitude error can be determined by taking the second norm of the resulting matrix multiplied by the displacement vector between the goal position and the current position of the end-effector, *dX*, as outlined in Equation 4.11. If the error proves too big then *dX* can be decreased until the error falls within an acceptable limit. Using these definitions, Figure 4.5 presents an algorithm that is used to implement an iterative inverse kinematics solution.

```
1)  Calculate the difference between the goal position and
    the actual position of the end-effector:
          $dX = X_g - X$
2)  Calculate the Jacobian matrix using the current joint
    angles: (using Equation 4.4b)
3)  Calculate the pseudo-inverse of the Jacobian:
          $J^{-1} = J^T (JJ^T)^{-1}$                          (4.10)
4)  Determine the error of the pseudo-inverse
          $error = \left\| (I - JJ^{-1}) dX \right\|$          (4.11)
5)  If error > e then
          $dX = dX / 2$
    restart at step 4
6)  Calculate the updated values for the joint
    orientations and use these as the new current values:
          $q = q + J^{-1} dX$
7)  Using forward kinematics determine whether the new
    joint orientations position the end-effector close
    enough to the desired absolute location.  If the
    solution is adequate then terminate the algorithm
    otherwise go back to step 1.
```

Figure 4.5: Iterative Jacobian-based algorithm

The computational demand of the algorithm is relatively high over a number of iterations, which is proportional to the number of IK chain nodes, so well-defined character hierarchies are advantageous. This means that each node in the articulation is defined by the minimum number of degrees of freedom (DOF) required, thereby reducing the joint configuration space, *q*, as much as possible. For example, pivot joints such as an elbow would only be modelled using a single DOF whereas a ball and socket joint like the shoulder would need 3 Euler DOFs to represent the range of possible movements.

The use of well-defined hierarchies further helps to prevent the inverse kinematics solver from producing unnatural looking postures. However this still does not cover all of the potential unnatural poses the solver can return. In order to restrict the IK solver to the orientation space of only possible character configurations, joint orientation restrictions can be enforced within the scope of the existing algorithm. The simplest way of incorporating such constraints is to crop the joint angles. This requires Step 6 of Figure 4.5 to be modified as follows:

```
6)  Calculate the updated values for the joint
    orientations and use these as the new current values:
    $q = \begin{cases} lowerbound & if \quad q + J^{-1}dX < lowerbound \\ upperbound & if \quad q + J^{-1}dX > upperbound \\ q + J^{-1}dX & otherwise \end{cases}$
```

The time to complete the IK algorithm for a given end effector is an unknown quantity because the number of iterations required is unknown. However the time to complete a single iteration is constant with respect to the dimensionality of *X* and *q*, which is unchanged under a complete execution of the algorithm. Therefore by placing an upper limit on the number of iterations, a maximum time boundary for the algorithm to return in can be set. If the solver reaches the limit then the algorithm returns the closest result it has calculated.

4.3 Complexity Analysis of the Jacobian-Based Inverse Kinematics Solver

In 3-dimensional space, the dimensionality of *X* in a Jacobian-based inverse kinematics solver of Figure 4.5 is either 3 or 6. The 6-dimensional *X* vector is normally used as it contains both positional and orientation information whereas a 3-dimensional vector only contains positional information for an end effector. It is clear that the 3-dimensional *X* vector is quicker over its counterpart and should always be used when orientation is not required. However there are times that orientation is required but it is still possible to use the 3-dimensional vector, which is discussed further in section 4.5 and a demonstration to this fact presented. For the continuation of this section, the difference in cost between the two sizes of *X* vector is considered based on the algorithm structure of Figure 4.5.

4.3.1 The Complexity of Calculating the Jacobian

If the Jacobian definition of Equation 4.4a is divided by a differential time element, the resulting equivalence provides a mapping between angular velocities in state space, *q*, and linear velocities in Cartesian space, *X*:

$$\dot{X} = J(q)\dot{q} \qquad \textbf{(4.12)}$$

In the case of a 6-dimensional *X* vector, $\dot{X}$ consists of linear velocity, *V*, and angular velocity, *W*, components, whereas the 3-dimensional *X* vector only includes the linear velocity terms. Both the linear velocity and angular velocity are with respect to a global frame of reference as too are the partial derivatives of the Jacobian. The Jacobian linking the linear and angular velocity of the end-effector, with the intermediary local angular velocities, is given in Equation 4.13, where there are *i* DOFs in the IK chain.

$$J = \begin{bmatrix} V \\ \Omega \end{bmatrix} = \begin{bmatrix} b_1, b_2, \ldots, b_i \\ a_1, a_2, \ldots, a_i \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_i \end{bmatrix} \tag{4.13}$$

The *a* components of Equation 4.13 are the local axes for a given link transformed into the global frame of reference (see [Watt92]). The *b* elements of the Jacobian are the cross products of the corresponding *a* axes with the spatial difference between the global origin of the i^{th} limb and the absolute location of the end of the articulation, P_e. The DOFs within the state space are normally ordered such that the limbs from the root are considered first, followed by their children, following this pattern to the end of the chain. Proceeding with this sequence, the orientation values of the required axes for each limb can be obtained from a transformation matrix, ${}^{0}T_j$, that converts points defined in the limb's local orientation into a global position. Equation 4.14 illustrates this for the j^{th} limb in the IK chain (note that this assumes a right-handed coordinate system), where a_{xj}, a_{yj}, a_{zj} and P_j are all 3-dimensional column vectors.

$${}^{0}T_j = \begin{bmatrix} a_{xj} & a_{yj} & a_{zj} & P_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

The parameter P_j in Equation 4.14 provides the global position of the origin of the limb. This aids the determination of the *b* components of Equation 4.13, through the relationship given by Equation 4.15 (see [Watt92]).

$$b = a_i \times (P_e - P_j) \tag{4.15}$$

Using the chaining principle for calculating the transforms of the local axes into a global reference frame, i.e. ${}^{0}T_j = {}^{0}T_1 \; x \; {}^{1}T_2 \; x \ldots x \; {}^{j-1}T_j$., the direct implementation of calculating the Jacobian in 3-dimensional space yields a constant complexity. Assuming that each link has 3 DOFs, the complexity associated with each limb in the IK chain is given by 162 flops: 98 multiplications and 64 additions & subtractions. The assumption of 3 DOFs does not add a great deal of complexity if it is an overestimate since each DOF contributes only 9 flops to the overall result (where the 9 flops is the calculation of the cross product).

4.3.2 Determining the Pseudo-Inverse of the Jacobian

During the pseudo inversion of the Jacobian matrix, the inversion of a square matrix is required. For the 3-dimensional matrix an analytical inversion of JJ^T (Equation 4.10) is readily derivable. However, the inversion of the 6-dimensional *X* vector is better suited with a numerical solution, which is performed using LU decomposition. Using the complexity of these sub-inversion routines, the complexity of the whole pseudo-inversion of the Jacobian is derived in section 4.3.3.

4.3.2.1 Analytical Inversion

The analytical inversion of a (3 x 3) matrix is given in Equation 4.16. The complexity of this is 51 flops: 36 multiplications, 1 division and 14 additions & subtractions.

$$\begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix}^{-1} = \frac{\begin{bmatrix} r_{11}r_{22} - r_{21}r_{12} & r_{21}r_{02} - r_{01}r_{22} & r_{01}r_{12} - r_{11}r_{02} \\ r_{20}r_{12} - r_{10}r_{22} & r_{00}r_{22} - r_{20}r_{02} & r_{10}r_{02} - r_{00}r_{12} \\ r_{10}r_{21} - r_{20}r_{11} & r_{20}r_{01} - r_{00}r_{21} & r_{00}r_{11} - r_{10}r_{01} \end{bmatrix}}{r_{00}(r_{11}r_{22} - r_{12}r_{21}) + r_{01}(r_{12}r_{20} - r_{10}r_{22}) + r_{02}(r_{10}r_{21} - r_{11}r_{20})} \qquad \textbf{(4.16)}$$

4.3.2.2 LU Decomposition

LU decomposition is used to determine the inverse of a square matrix by using the matrix identity, $AA^{-1} = I$, where *I* is an identity matrix. The application of LU decomposition to this equation requires matrix *A* to be split into 2 further matrices that have the form of lower and upper matrices as illustrated in Equation 4.17.

$$A = LU = \begin{bmatrix} 1 & 0 & 0 & \mathbf{L} & 0 \\ * & 1 & 0 & \mathbf{L} & 0 \\ * & * & 1 & \mathbf{L} & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathbf{O} & \mathrm{M} \\ * & * & * & \mathbf{L} & 1 \end{bmatrix} \begin{bmatrix} * & * & * & \mathbf{L} & * \\ 0 & * & * & \mathbf{L} & * \\ 0 & 0 & * & \mathbf{L} & * \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathbf{O} & \mathrm{M} \\ 0 & 0 & 0 & \mathbf{L} & * \end{bmatrix} \qquad \textbf{(4.17)}$$

The complexity of the decomposition is 151 floating-point operations (flops): 55 multiplications, 16 divisions and 80 additions & subtractions.

The decomposition of *A* into the two matrices allows the original matrix identity to be rewritten into the form of Equation 4.18.

$$LUA^{-1} = I \qquad \Rightarrow L(UA^{-1}) = I$$

$$\Rightarrow LY = I \qquad \textbf{(4.18a)}$$

$$UA^{-1} = Y \qquad \textbf{(4.18b)}$$

The first step is to solve *Y* from Equation 4.18a, which rewrites into a set of *n* linear equations where *n* is the number of columns/rows in the inverse matrix; in this case, *n=6*. Due to the nature of the lower matrix, the *n* linear equations are already in the form that allows a simple forward substitution technique to be applied. Equation 4.18b is subsequently solved to find A^{-1} using a backward substitution algorithm.

The complexity for the forward & backward substitution algorithms for a (6 x 6) matrix is 468 flops: 180 multiplications, 36 divisions and 252 additions & subtractions. Therefore, including the factorisation into upper and lower forms and the inversion of a (6 x 6) matrix using LU Decomposition, the result is a total complexity of 619 flops.

4.3.2.3 Analytical vs. Numerical Inversion

The decision to use an analytical solver for the smaller matrix and LU decomposition for the larger one is illustrated in Figure 4.6, which compares the flop complexity for both analytical and LU decomposition over a range of square matrix sizes. The results given in Figure 4.6 are obtained using a matrix with all elements non-zero so the analytical technique was unable to make use of zeros to cut off the co-factor expansions. This is a valid assumption because it would be most unlikely that the (6 x 6) matrix in the pseudo-inverse would actually contain any zeros.

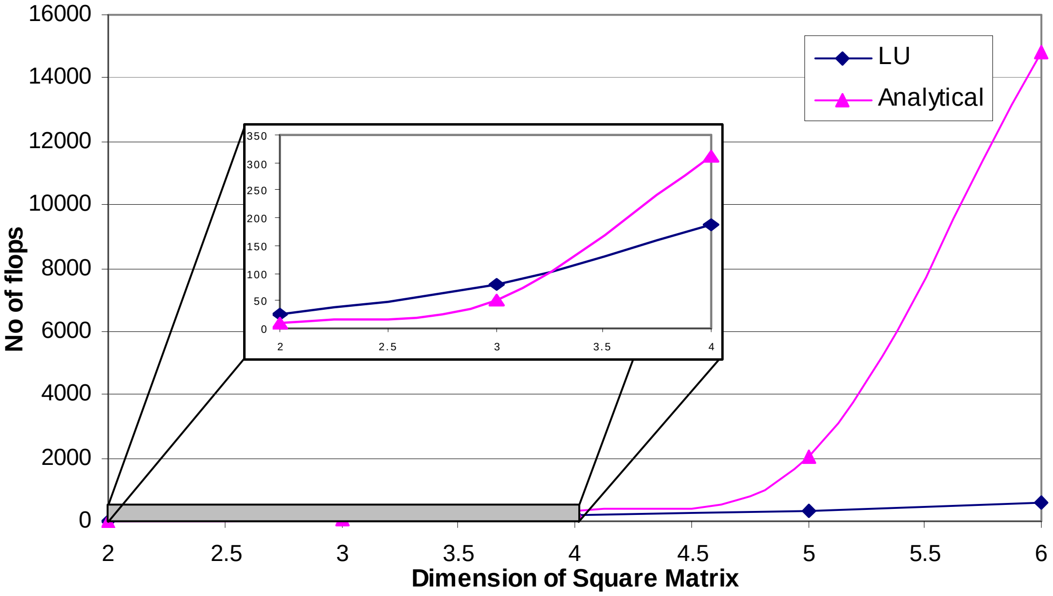


Figure 4.6: Demonstration of the complexity of solving a square matrix using an analytical technique and an LU decomposition technique

Figure 4.6 shows the analytical approach to solving matrix inversion is only better for matrices that have dimensionality equal to or less than 3. Thereafter, the number of flops required to solve an analytical inverse increases in a cubic fashion with respect to dimensionality whereas the LU technique increases at the lower squared rate. This analysis therefore justifies the use of an analytical solution for the (3 x 3) matrix while using LU decomposition for the inversion of the larger (6 x 6) matrix.

4.3.2.4 Pseudo-Inverse Complexity

Table 4.1 outlines the total number of flops required to calculate the pseudo-inverse depending on the size of the *X* vector. The variable *n* is the size of the state space, *q* (i.e. the sum of all the links' DOFs). It should also be noted that there is no inclusion of complexity to calculate the transpose of matrices when they are required as this can be handled at no extra cost be simply swapping the indexing parameters.

Size of *Matrix* ***Operation***	(3 x n) 3D *X* Vector	(6 x n) 6D *X* Vector
$A = JJ^T$	18n - 9	72n - 36
$B = A^{-1}$	51	619
$J^T B$	15n	33n
$J^{-1} = J^T (JJ^T)^{-1}$	33n + 42	105n + 583

Table 4.1: Number of flops required to calculate the pseudo-inverse of a non-square *A* matrix.

4.3.3 Complexity of the whole IK Solver

The complexity of a single iteration of the IK algorithm from Figure 4.5 can be derived using the complexity analyses of the smaller parts of the algorithm already determined. A complete compilation of the individual complexities is shown in Table 4.2 where *m* is the number of inner loops executed at stage 5 of the algorithm.

Size of X Vector **Algorithm Stage**	3	6
1. Calc. increment	3 flops	6 flops
2. Calc. Jacobian	162 flops	162 flops
3. Calc. Pseudo-Inverse	33n + 42 flops	105n + 583 flops
4. Check for convergence	18n + 15 flops	72n + 66 flops
5. Reduce dX	18m flops	72m flops
6. Update joint angles	6n flops	12n flops
7. Calculate new position	38n flops	38n flops
Total	95n + 18m + 252 flops	227n + 72m + 817 flops

Table 4.2: Complexity analysis of the Jacobian based IK solver

As Table 4.2 illustrates, the use of a 3-dimensional *X* vector appears to be about 2½ times less computationally demanding than its 6-dimensional counterpart. This equates to a theoretical speed increase of 238.9% from the 6- to the 3-dimensional *X* vector. However, the complexity of each algorithm is not only dependent on the size of the state space but also on the number of inner loops which are required to make the inversion of the Jacobian stable enough to provide meaningful results.

Therefore it needs to be shown that the use of a smaller Jacobian in the 3-dimensional *X* vector case does not adversely affect the pseudo-inverse.

To demonstrate that the theoretical reduction in computational demand has practical grounding, section 4.4 details the construction of an IK-based character walking system called *MovingIK*, which can switch between the two different *X* vector sizes. *MovingIK* also demonstrates how the smaller *X* vector can be used even when orientation is important and is further built upon to show how a novel inclusion of the inverse kinematics algorithm of Figure 4.5 can be used to stylise character motions.

4.4 MovingIK: An IK Driven Character Walking Implementation

This section describes the implementation of an IK-driven character walker, called *MovingIK*. *MovingIK* provides the basic working system for the following purposes in the continuation of this chapter:

- Demonstrate the ability to map an orientation and positional problem into a simpler positional IK one (section 4.5.2)
- Empirically compare the 3- and 6-dimensional *X* vector (section 4.5.3)
- Inverse kinematics stylisation of character motions using (section 4.8):
    - § Procedural data
    - § Motion capture data

MovingIK is comprised of three independent modules that communicate using a parameterisation that allows flexible control over the generated motions. The system's modular design is outlined in Figure 4.7, which is described during the following subsections.

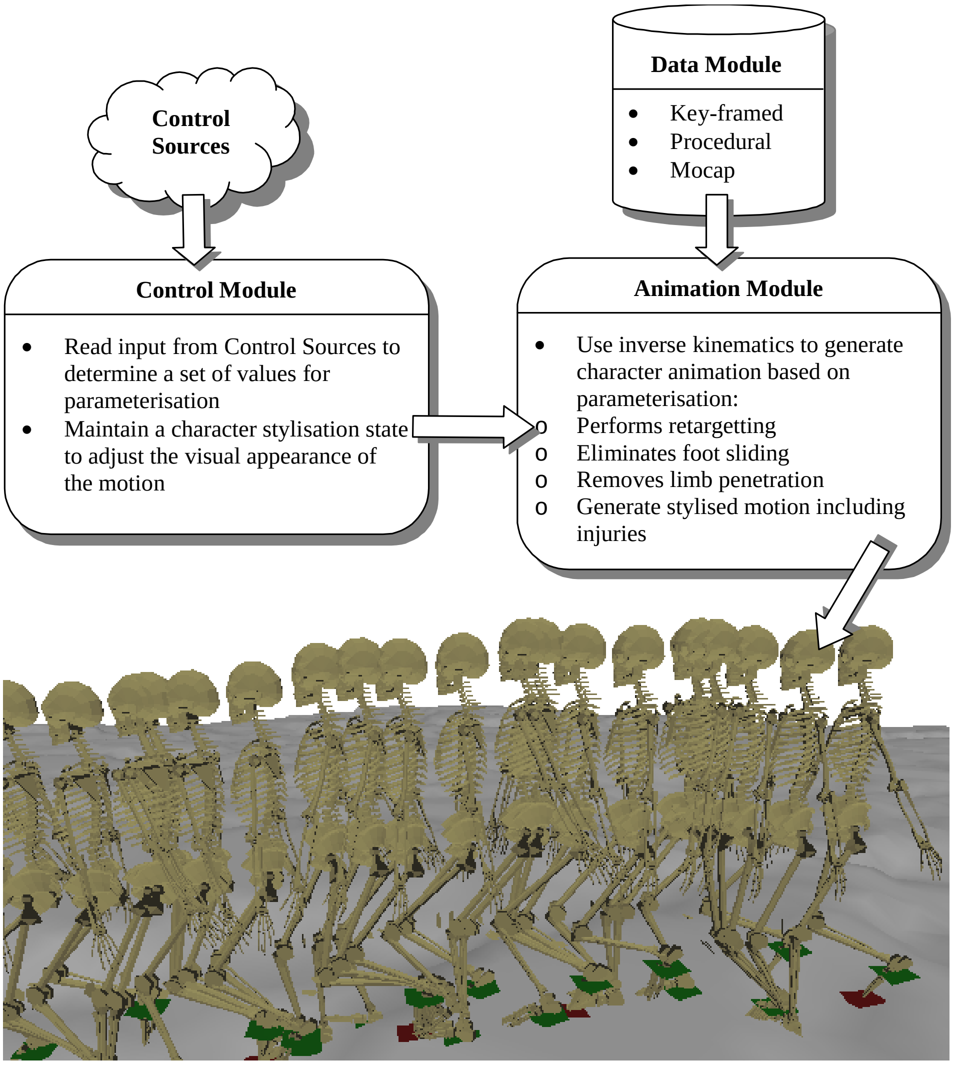

Control Sources
Data Module
Key-framed
Procedural
Mocap
Control Module
Read input from Control Sources to determine a set of values for parameterisation
Maintain a character stylisation state to adjust the visual appearance of the motion
Animation Module
Use inverse kinematics to generate character animation based on parameterisation:
Performs retargetting
Eliminates foot sliding
Removes limb penetration
Generate stylised motion including injuries

Figure 4.7: Control Structure of *MovingIK*

4.4.1 The Control Module

The Control Module sits at the top of our system and is responsible for reading input from an analogue source, which is controlled by the user, and provides the Animation layer with all of the initial parameterised values it requires to simulate the motion. The parameters include stride length, stride speed and direction of travel. The stylisation state provides the character with added individualisation, above the standard walking cycle, for the motion, which will be discussed in section 4.7.1. The parameterisation of the Control Module, which encapsulates the information required to produce a motion, is passed on to the Animation Module.

4.4.2 The Data Module

The Data Module provides knowledge, in the form of limb degree of freedom (DOF) values, about how to perform an action, which is used by the Animation Module. The output format allows the data to be modelled using a range of techniques including key-framed data, procedural models and motion-captured data, without affecting the behaviour of the other modules in the system. This affords us the ability to choose the optimal data representation for the given scenario. For example, motion capture data can be used for high detail where storage space is not an issue and procedural models can be used for background characters. The representation of a procedural model and the processing of a motion capture movement are outlined in the following subsections.

4.4.2.1 Procedural Data Model

For the basic procedural walking motion, the gait data is based on a mathematical equation. Equation 4.19 gives the mathematical location for the foot flight path. The graphical representation of the procedural stride is illustrated in Figure 4.8, where a complete cycle ranges between 0 and 3π, and the range of the function gives the relative height above the ground of the foot. This model is based on the observations of human gait motion curves.

$$Y = \begin{cases} 1-\cos(x) & if \;\; x \le p \\ 1-\cos\left(\dfrac{p+x}{2}\right) & otherwise \end{cases} \tag{4.19}$$

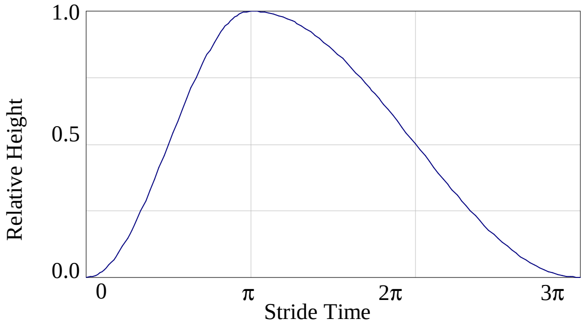


Figure 4.8: Graph of procedural stride based on Equation 4.19

MovingIK is an application that drives a character around a scene and not a retargetting solution. Therefore a flight path is used as opposed to the joint angles themselves because this is the data that *MovingIK* needs to know so that it can position the feet over time and let the IK algorithm determine the corresponding joint angles.

The procedural model used for the upper-body movement comes from a cosine curve, which animates the character's shoulders.

4.4.2.2 Motion Capture Data Model

To use motion capture data, the data modules extracts the required DOF values and end-effector positions directly from a motion capture file. The motion for the upper part of the body, from the hips upward, remains essentially unaltered from the mocap, except for a time-warping factor that is used to synchronise the upper and lower body motions.

The technique used to extract a flight path, similar to that of Figure 4.8, from the motion capture data is primarily based on a gradient analysis technique (using finite differencing because our data is discrete), where over the course of the original motion, the height values of one of the character's feet are analysed. Figure 4.9 illustrates the first target that is being looked for is a switch from a negative to a positive gradient on a frame that is within 10% of the global minimum height for the foot. This gives the start of the foot flight where the foot is just about to leave the ground. From the first target, the motion captured flight path is progressed along until a second gradient change from positive to negative is met, which is on a frame whose height value is within 10% of the global maximum. This second target is the peak of the foot flight height. Continuing along the original path, the third and final target to is located via one last gradient switch from negative to positive whose foot height is again within 10% of the global minimum.

The 10% threshold values reduce the chances of failing to find a complete stride cycle as using local gradients alone would fail where the motion data temporarily plateaus or the motion data has a little "blip". For example, in Figure 4.9, near frame 25, the motion data introduces what is almost a stationary point and, had the gradient actually changed at this point, the gradient-analysis technique would have determined this to be the third target, which is obviously not the case.

Having found all three targets, the flight curve it taken to be that that is described between the first and third target points. The height values are subsequently normalised so that they can be mapped onto the new character based on a ratio between the new and original character's leg lengths. This ratio between the leg lengths of the character is additionally used to determine the initial stride length control parameter.

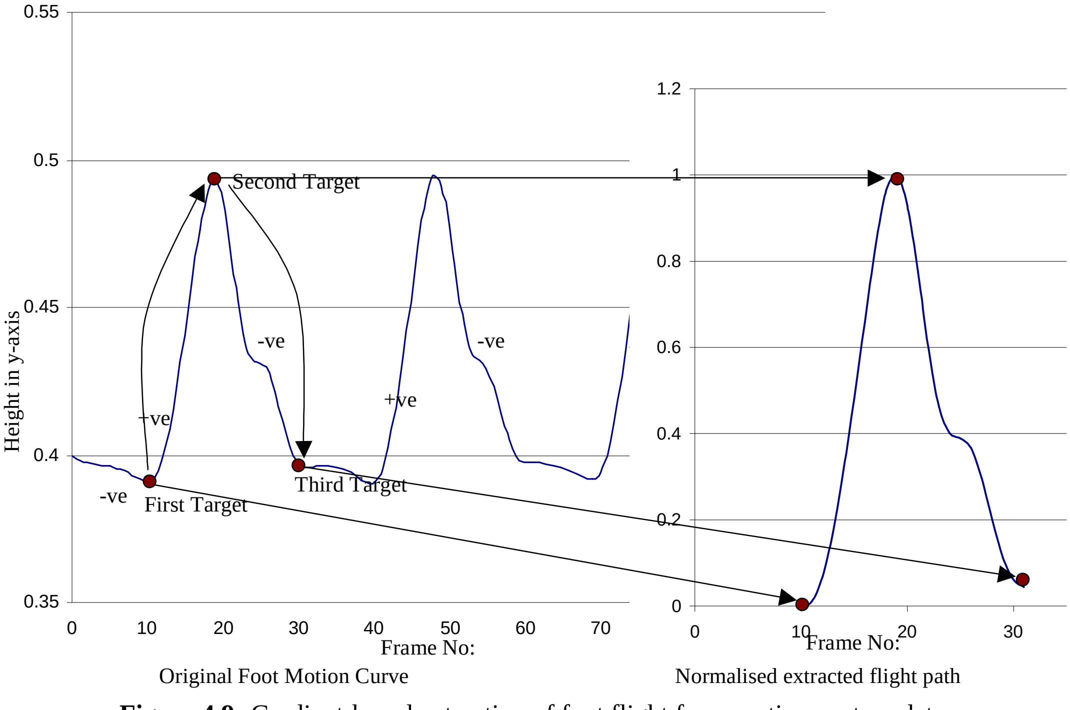


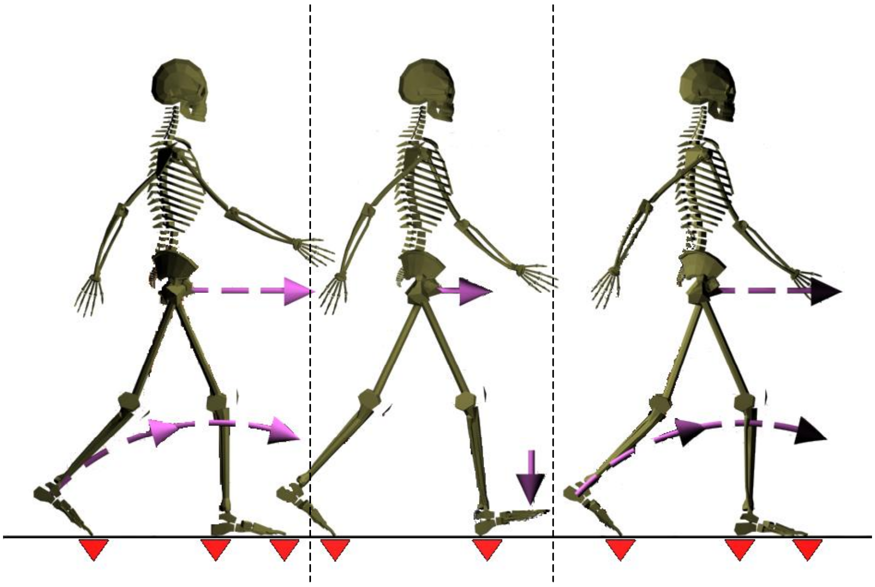
Figure 4.9: Gradient-based extraction of foot flight from motion capture data

4.4.3 The Animation Module

Through combining the parameterisation from the Control module and the examination of the surrounding terrain, the Animation module determines the extents of the motion to perform. Using knowledge from the Data module about the motion, the end-effector locations are interpolated over time between the motion extremities and thus produce the simulated motion. The end-effector locations are positioned per frame through the application of the inverse kinematic solver. In the Animation layer, the inverse kinematics component is implemented as a switchable algorithm so that either the 3- or 6-dimensional *X* vector Jacobian can be used. The implementation of the IK solver is a direct encoding of the algorithm described in section 4.2.

The foot flight path, which is obtained from the Data module, assumes a flat surface movement in that the foot lands at the same height that it originally started from. Therefore, when the character manoeuvres over uneven terrain, the source for the foot flight is rotated to make the end points match up with the height of the heel at the start of the stride and the height of the ground at the end of the walk cycle.

In addition to the data model used to drive the character's foot through the air, there is a pre-flight stage that rolls the foot from a heel supporting phase to a complete foot supporting phase [McMa84, Moch80]. This uses the inverse kinematics algorithm to simultaneously plant the heel of the character and gravitate the toes towards the ground. This extra bit of the walking cycle increases the realistic-looking nature of the resulting animation and allows the complete foot to be modelled as opposed to just the heel. An overview of this procedure is outlined in Figure 4.10 and Table 4.3.



(a) Start of walk cycle (b) Final part of walk (c) Start walk cycle again on other foot

Figure 4.10: Demonstration of the cycles implementing in our system. Each frame represents the start of the cycle with the arrows pointing in the direction of travel the node will take until it reaches the start of the next part of the cycle. The red triangles represent plants of the character's limbs

Stage *Description* Starting Configuration	(a)	(b)
Left Foot	Both heels and toes are planted on the floor	Toes are planted on the floor
Right Foot	Toes are planted on the floor	Heel is planted on the floor
Movement	1. Hips move forward, 2. Right heel is advanced forward through the air, 3. Only the left toes remain planted.	1. Hips move forward, 2. Right toes are gravitated towards the floor, 3. Left toes remain planted to the floor

Table 4.3: Description of the 2 stage walk cycle where the initial configuration is with the left foot in front and the right foot behind the body

In the subsequent subsections, a closer look at how the Animation module simulates the human gait is considered. There are two cases under which the character can move forward: walking in a straight line or turning. The Control layer uses input from either an analogue joystick or keyboard to determine the way the character moves. If there is no sideways movements then the walking forward technique is used otherwise a turning action is executed. These two techniques are thus discussed.

4.4.3.1 Move the character forwards

When the Control layer receives a joystick movement it calculates the stride length and speed based on the analogue input. The maximum stride length is half the leg length plus the distance the trailing leg is from the hips in a horizontal direction. The hips of the character only move half this distance because each leg only moves once for every 2 complete walking cycles (i.e. left leg cycle followed by a right leg cycle).

The stride time represents the temporal window used to complete stage (a) of the walking cycle of Figure 4.10. The time to complete stage (b) of Figure 4.10 is dependent on the distance the front toes are from the ground at the end of the first stage. Using the distance between the ground and the toes, the hips of the character are moved by an equal magnitude along the direction of travel. The time to perform the final stage is calculated using this distance value and the velocity of the character from stage (a).

While walking forward, the leg that will be trailing behind is fixed to the floor by the toes of the foot while the heel of the foot leading is lifted off the ground and follows the flight path given by the Data module in a vertical direction. The other dimensional components are driven in a direction parallel to the walking direction. A linear percentage of time is used to determine how far along the path the foot should be and the Inverse Kinematics solver is used to position the heel to that point, while in a separate application of the solver the toes are clamped to the floor for the trailing foot.

Once the toes of the back foot are clamped to the ground there is enough freedom within the constraints of the human body for the back heel to be lower than the toes and hence penetrate the floor. In order to eliminate this impossible pose, the position of the back heel is checked against the height of the floor at that point. If the heel is lower than the floor a separate algorithm is used to calculate the position that the heel would have to be in such that the position of the toes are maintained but allowing the heel to be repositioned onto the surface of the floor. Similarly, when the front heel is positioned along the curve, there is the possibility that the toes of the foot will penetrate the floor so this is checked and adjusted to reposition them just on the surface while maintaining the heel position.

When the appropriate amount of time has transpired for the first stage of the step cycle, the heel of the front foot will be in contact with the floor as will the trailing toes and it is time to initiate the second stage. The purpose of stage (b) is to maintain both the current heel and toe plants and gravitate the toes on the front foot towards the ground while still moving the character in a forward direction. At this stage, the front heels and back toes remain clamped to the floor while the hips continue to move. The front ankle orientation is also adjusted linearly such that at the end of the new temporal window the toes will be in contact with the floor too, thus preparing the character to start a step cycle on the other foot.

4.4.3.2 Making the character turn

The foot planting phase of making the character walk in an arc is similar to that used to make it walk in a straight line. The vertical component of the leg lifting process is identical to that of walking in a straight line. However instead of making the character walk in a straight line in the direction the character is facing, both the hips and the leading leg's motion is described as a curve in the horizontal plane. The curves that the limbs follow are described by a centre of rotation, radius and angular difference. With the turning radius determined based on the sharpness of turn, the centre of the circle can be calculated as well as an angular difference which is linearly interpolated to give the corresponding position in time.

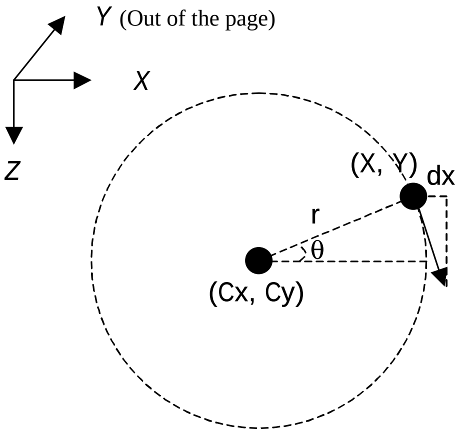
The equation for a circle whose origin is located at *(Cx, Cy)* is given in Equation 4.20, where *0 £ q £ 2p.*

$$(X,Y) = (Cx - r\cos q, Cy + r\sin q) \qquad \textbf{(4.20)}$$

The character's location must lie on this circle so *(X, Y)* can be taken to be the current location of the hips projected onto the horizontal plane. This allows Equation 4.20 to be rearranged to give Equation 4.21 thereby making the centre of the circle the subject.

$$(Cx, Cy) = (X + r\cos q, Y - r\sin q) \qquad \textbf{(4.21)}$$

q is given by the amount the character has already turned from a fixed axis. Taking the fixed axis to be the global Z-axis, *q is* therefore the *arc sin* of the *X* component of the normalised forward-facing direction vector of the character. When the character is turning in an anticlockwise direction about the Y-axis, this angle can be directly plugged into Equation 4.21. However when the character is to turn in a clockwise direction about the Y-axis, the angle needs to be negated. This process of determining the centre of rotation is illustrated in Figure 4.11.



$q = -\sin^{-1}(dx)$

(turning clockwise about Y-Axis)

$$Cx = X + r\cos q$$
$$Cy = Y - r\sin q$$

Figure 4.11: Calculating the centre of rotation for turning a character

The number of degrees through which the character should turn is determined by Equation 4.22, where *s* is the arc length, *q* is the number of radians subscribing the arc, and *r* is the radius of the arc.

$$s = rq \qquad \textbf{(4.22)}$$

With the radius fixed and setting the arc length to be the magnitude displacement that the hips would undergo if the character were walking in a straight line, the number of radians that will be covered during the walking cycle can be calculated using Equation 4.22. The number of radians to cover is linearly interpolated over the time step of the walking phase to calculate the position and orientation of the hips using Equation 4.20.

The other path that comes directly from these calculations is that of the leading heel. A similar process is used to determine the corresponding values that the heel will take during the walking cycle where the centre of orbit is used from the calculation of the hips, as illustrated in Figure 4.12. The radius of the arcs used to drive the heels of the character differs from that of the hips only by a factor of the difference between the model centre of the hips and the femur. The combination of these arc paths result in the ability to smoothly turn the character in either direction at varying rates dependent on changeable parameters such as radius and stride length.

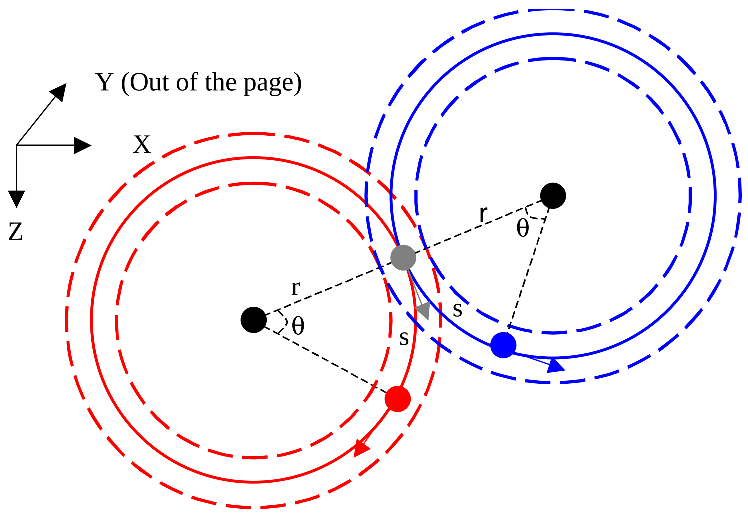


Figure 4.12: Calculation of the amount to rotate the character about based on the radius of the circle and stride length. The inner dotted circles represent the path the inside leg would take if it is the leading leg while the dotted outer path is that of the outside leg when leading. The red circles (left) are those when the character turns right while the blue circles (right) represents the character turning left

4.4.4 Other Motion Details

The character further has the ability to walk smoothly over uneven terrains. This is achieved by determining the ideal position of the hips at the end of a given walk cycle based on the height of the floor at that point and the character's leg length. This distance is linearly interpolated with the current height of the hips during the cycle along with a small sinusoidal wave that gives the effect of the hips naturally moving up and down as the character moves. This difference is also added into the heel height equation for position so that the graph illustrating the heel height in Figure 4.8, for example, is effectively skewed to take into account any varying levels in the ground.

The final addition to the basic IK walking application is the ability rotated the hips about the vertical axis to give the effect of the hips swinging. Therefore the hip cycle can be adjusted to give different behaviours – i.e. smaller or larger hip swings.

4.5 The Half-Jacobian

In section 4.3, an analysis between the 3- and 6-dimensional *X* vector Jacobian-based inverse kinematics solvers was presented. The results demonstrated that the 3-dimensional *X* vector has a theoretical performance increase of 238.9% over its counterpart. To exercise this theoretical performance difference, *MovingIK* was introduced in section 4.4 as a practical harness for both types of solvers. However, the act of animating the legs of a humanoid character would appear to exclude the use of the 3-dimensional version because orientation is just as important as position. Therefore, before presenting the empirical comparisons between the different sized X vectors in section 4.5.3, a discussion is first undertaken as to how traditional 6 DOF IK problems can be converted into the simpler positional only problems. In addition to this innovative transformation process, the 3- and 6-dimensional *X* vectors will hereafter be termed half- and full-Jacobian techniques.

4.5.1 Using the Half-Jacobian *vs.* the Full-Jacobian

An obvious application of the half-Jacobian is in applications that do not discriminate against the orientation of the final link in an inverse kinematics chain. In applications of inverse kinematics where the orientation of the end-effector has little consequence, the half-Jacobian should always be used to reduce the computation effort required. For example, when configuring a spider's legs using IK, because the spider effectively walks on the tips of its legs, the orientation of this end point is immaterial therefore only the half-Jacobian would be required.

Another, more subtle, application of the half-sized *X* vector is in situations where the penultimate link in the IK chain has unlimited and full use of all 3 DOFs (in 3 dimensional space). In this scenario the first step is to analytically calculate the position of the penultimate link based on the desired position and orientation of the final node. The half-Jacobian takes the penultimate node in the chain as the end-effector and positions it to the analytically determined global position. Once this is done the desired orientation of the final node is rotated into place such that the original end-effector coincides with the specified global position. This is similar to the later stages of the analytical 7 DOF IK solution from Figure 4.4 where the heel was analytically position as the penultimate link and the foot orientated into position. The different between this analytical solution and the half-Jacobian version is that the IK chain above the heel is unrestricted using the more general numerical technique.

Other applications where the half-size Jacobian would prove a better technique to employ over the full-size version is in situations of low resolution modelling. For example, if a complex articulated model is being animated as a background entity in a scene, it would be advantageous to switch to the quicker half-Jacobian to solve its configuration. This means that more avatars can be animated in the background of a scene than if the full-Jacobian were used.

There are many other applications where the half-sized Jacobian could substitute for the traditional full-sized version. This thesis applies the quicker, half-Jacobian inverse kinematic solver to IK-driven character walking, motion capture retargetting and individualisation by placing additional constraints on the joint ranges to limit the resulting range space. All three of these applications can easily run in real-time, as the following sections describe. The continuation of this section focuses on the former of these applications in order to provide empirical timings to correlate with the theoretical ones of section 4.3. Retargetting and individualisation using the half-Jacobian are discussed in section 4.6 and section 4.8 respectively.

4.5.2 IK-Generated Humanoid Walking

The task of positing the foot along a flight path would initially appear to require the full-sized Jacobian, inherently requiring the foot to be orientated in a forward facing direction. Without the orientation of the foot taken into account, there are an infinite number of anatomically correct positions the heel could take in order to meet a simple positional constraint. This is possible because the hip joint for a leg can rotate about the axis of the femur approximately ±90 degrees from the forward facing pose, as illustrated in Figure 4.13.

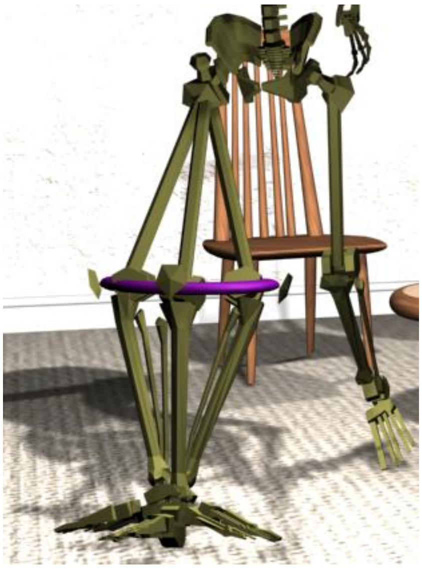


Figure 4.13: Infinite number of positional solutions to fixing a heel plant without regard to the orientation of the foot. The purple ring shows the location of all possible knee positions

However, by realising that in the course of a walking motion, any large hip joint rotations result in unnatural postures, additional constraints can be added to restrict movement to only plausible ranges. Therefore, through the original inclusion of additional constraints, the half-Jacobian can be used to calculate the position of the heel and thus simultaneously reduce the potential for orientation error and increase the performance of the solver. This effectively maps the full-Jacobian problem into the domain of the half-Jacobian technique and thus defines the manner in which the two different techniques can be used to drive the IK aspect of *MovingIK* (section 4.4).

In the following section, this process is illustrated and timing empirical timing results are obtained between the half- and full-Jacobian techniques.

4.5.3 Empirical Comparison between the Half- and Full Jacobian using *MovingIK*

The results presented in this section are obtained from running *MovingIK* on a Pentium 4 1.4GHz processor with a GeForce2 Ultra. A maximum iteration count of 200 cycles is imposed on the IK solver for the outer loop while the inner loop is subject to a 20-cycle ceiling. These limits are determined by the empirical running of the IK solver to determine over what limits a solution was very rarely found. The character driven by the user is made up of 18 hierarchical segments where only naturally-occurring DOFs within the human body were permitted. The constraints on each remaining DOF were limited to joint angles within the scope of normal human movement (see Figure 2.14).

The results given were obtained by driving the character around both flat and uneven terrains. In the case of the uneven terrain, several randomly-generated surfaces were used (including a flight of steps) and the overall results were obtained by averaging the results. Each of the uneven terrains had the same number of vertices and polygons in the model (64,082 polygons compared to 2 polygons for even terrain). The character displayed was that of either a stick figure or a 3D model consisting of 11,101 polygons.

MovingIK is not optimised to use either the half- or full-Jacobian but instead provided the ability to switch between the two techniques at run-time. There are three different configurations possible to switch between. The first two modes use only the half- or full-Jacobian to calculate the configuration of the character to position the leading foot and trailing toes. The third mode uses a hybrid approach that uses the full-Jacobian to determine the configuration of the leading foot and the half-Jacobian to anchor the trailing toes. It should be noted that during a single frame, *MovingIK* solves two IK chains – one for each leg.

The empirical results of driving the computer character within *MovingIK* are illustrated in Table 4.4, where an illustration of the results is given in Figure 4.14.

IK Mode ***Measurement***	All Half Jacobian (3D X Vector)	All Full Jacobian (6D X Vector)	Hybrid Method
Flat Floor with Stick Character	260 fps	95 fps	115 fps
Flat Floor with Skeleton	140 fps	69 fps	83 fps
Uneven Terrain with Stick Character	97 fps	54 fps	64 fps
Uneven Terrain with Skeleton Character	75 fps	42 fps	53 fps
Average time to execute each IK solver	0.24 ms	0.55 ms	- - - -
Average Number of iterations	18.15	18.0	- - - -
Average time per iteration	0.013 ms	0.031 ms	- - - -

Table 4.4: Empirical Results from *MovingIK*

The speed-up factor between the full-Jacobian and the half-Jacobian, based on the empirical average time per iteration, is 238.5%, which when compared to the analytical computed result of 238.9% (see section 4.3) reinforces the advantages of using the half-Jacobian over the full-Jacobian whenever possible.

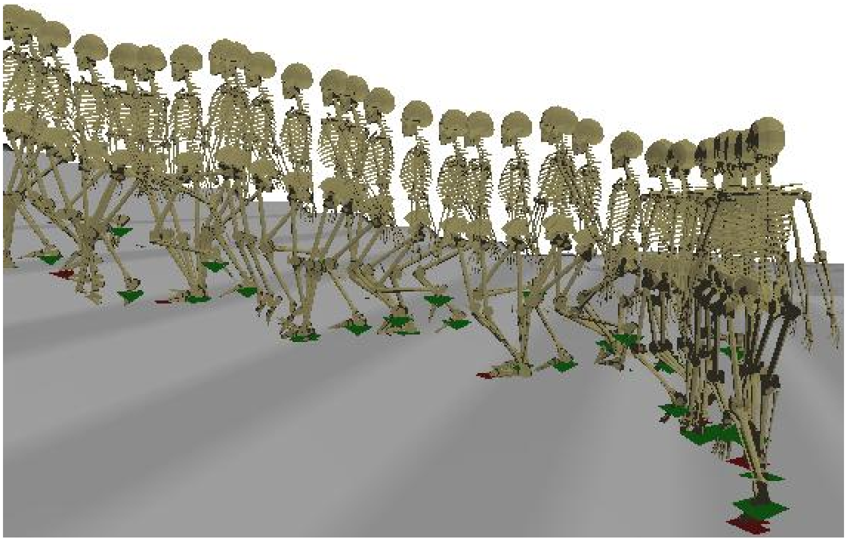


Figure 4.14: Analogue joystick-controlled real-time half-Jacobian IK over uneven terrain; green pyramids represent the intended position of the leading foot while the red pyramids indicate desired location of the training toes

A further conclusion from these results is that the use of the full Jacobian does not necessarily make the IK solver any more stable, which was a point of consideration during the analytical complexity derivations. This logical conclusion comes from the fact that the analytical speed-up factor calculated assumes that the inner loop is executed an equal number of times for both algorithms. If this were not the case then the empirical results would show a greater difference in speed up factor due to one algorithm executing the inner loop more times than the other.

Figure 4.14 additionally demonstrates the success in mapping the full-Jacobian gait generation problem into the half-Jacobian domain, where through the use of additionally constraints to limit the range of motion on the femurs, the character’s feet are always pointing in the correct direction. More examples of this will be presented during the continuation of this chapter.

4.5.4 Summary

In this section, it has been shown how there is also scope for using the quicker half-Jacobian for limited domains when orientation is required as well as position. Although the half-Jacobian has thus far only been demonstrated for a walking motion, this represents one of the most fundamental movements in computer character animation. In section 4.6 the half-Jacobian is applied to the field of motion capture retargetting with similarly successful results in both speed and visual accuracy. This is subsequently complemented in section 4.8, where motions are individualised. There are also many other conceivable domains in which the half-Jacobian can be used by placing extra dynamic constraints on joint angles to prevent the orientation from deviating too much from a natural-looking configuration. An arm, for example, would prove just as suitable a subject for the technique.

The advantage of using dynamic/additional constraints to transform an orientation and positional IK problem into a position-only task is a speed-up factor of approximately 238%. There is also no extra cost to adding in these constraints to the half-sized Jacobian algorithm because its framework already operates using joint restrictions. Effectively the dynamic constraints come for free in the Jacobian-based IK solver.

4.6 Motion Capture Retargetting using the Half-Jacobian

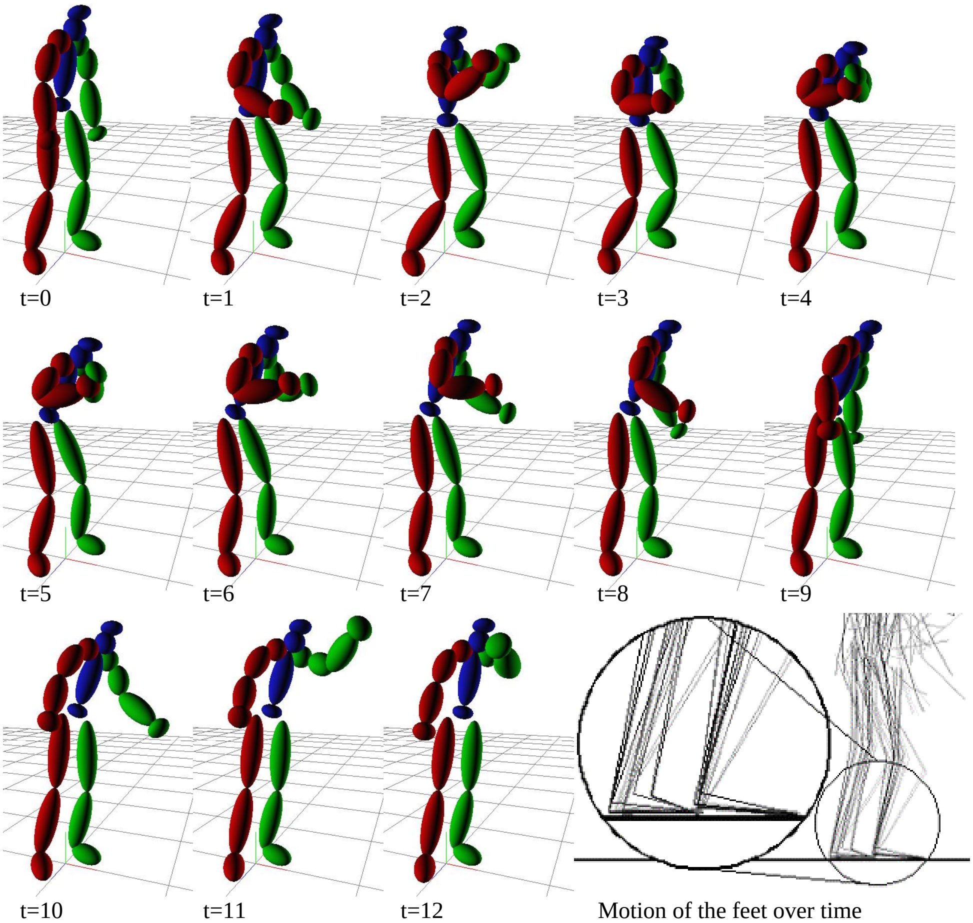
In this section, the retargetting paradigm is used as an application to further demonstrate the half-Jacobian's ability to appreciate traditionally full-Jacobian problems, using similar added constraints to those used in section 4.2 for regular joint constraints. Although this section does not really present a radically new way of retargetting a character, it serves the aforementioned purpose and later in Chapter 7 the IK retargetting process described is exercised further through a comparison with the dynamics-based ones, which is discuss in Chapter 6.

The IK retargetting solution is a per-frame algorithm, which could result in high-frequency changes being introduced. However, as the results will demonstrate, the inverse kinematics solution tends to be move favourable over time and rarely are such "snapping" problems seen. Although, if such artefacts were to creep in, a filter process could be applied similar to that in previous works [Sul98, Lee99, Tak02, Tak05] without losing any generality to the results present here: it is the aim of this section to demonstrated how the half-Jacobian could be used and not specifically the retargetting technique.

The two examples that are subsequently presented portray a character catching a football, whose feet are fixed to the spot, and a character walking along a line. The first example will illustrate two aspects of the inverse kinematics solution. The first ensures that the feet stay planted on the floor at all times, and second to retarget the arms so that the hands can be arbitrarily placed in order to meet the incoming football. The second example demonstrates the more common and generic problem of foot skating where a character that is supposedly moving about on a hard floor surface actually looks as if they are walking on ice. In this example, the upper-body joint configurations are maintained and only the legs are adjusted.

4.6.1 Football Catch

Figure 4.15 illustrates a motion capture clip that consists of a character catching a football and throwing it back out. There are two sets of end-effectors that interact with the environment, which need to be reasserted: the hands and the feet.



t=0 t=1 t=2 t=3 t=4

t=5 t=6 t=7 t=8 t=9

t=10 t=11 t=12 Motion of the feet over time

Figure 4.15: Original motion capture clip of a character catching a football and then throwing it back out

Considering the feet first, the bottom right picture of Figure 4.15 shows that there is a small amount of shuffling along the floor by the character as it reaches to meet the incoming target. This never penetrates the floor level and it is this feature that is desirable to preserve. Consequently, on a per-frame basis, the toes of the character are locked to the floor by projecting the height value back onto the floor plane. This allows the heel to leave the ground, as can happen in Figure 4.15. However, if at any time the heel penetrates the floor, the heel is taken to be the end-effector and locked to the floor. The ankle angle is mathematically recalculated so that the foot lies flat on the floor, thus ensuring the toes are also planted to the ground.

To allow the hands to be positioned at a specific height location to meet an incoming football, the original motion is subjected to a scaling factor. In order to perform this process in an on-line manner, the height of the original catch point, *c*, is determined beforehand along with its initial starting height, *h*. Given that the height value, *r*, of the retargeted hands, a scaling factor *s* can be defined as given in Equation 4.23.

$$s = \frac{r-h}{c-h} \tag{4.23}$$

During real-time play back and retargetting of the original motion, the scale factor *s*, scales the flight path of the hands in the height axis based on the original motion using Equation 4.24, where y_i and o_i are the new and original height values respectively for the i^{th} frame.

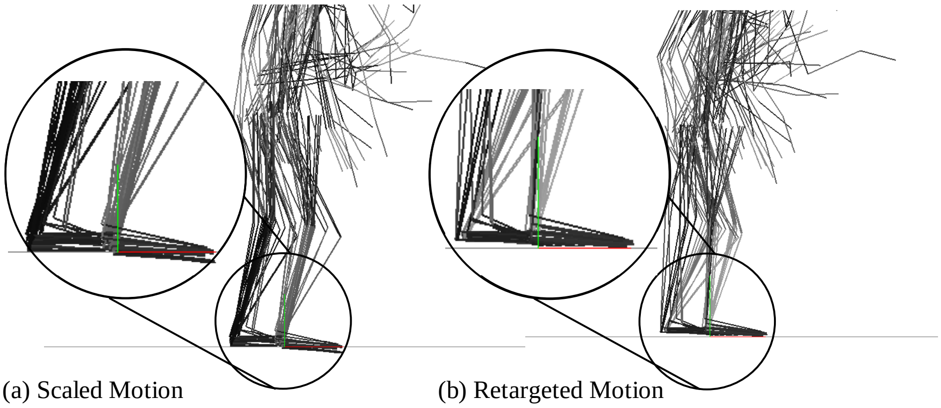
$$y_i = s(o_i - h) + h \tag{4.24}$$

The result of Equation 4.24 is plugged into the inverse kinematics solution, where the root nodes are the clavicle bones and the end-effectors are the hands. Additionally, joint constraints are enforced according to those outlined in section 4.2. This completes the control structure of the retargetting process for this motion, the result of which is presented next.

4.6.1.1 Football Catch Results

The retargetting of the movement portrayed in Figure 4.15 involves 4 applications of the half-Jacobian inverse kinematics solver per frame; one for each foot and hand. Although there is scope to parallelise these applications per frame, a serial application of the solver is used for these results. As the inverse skinning algorithm showed (section 2.3), if the IK applications were parallelise a computational time reduction of about 40% would be expected on modern hyperthreaded processors. Using a serialised application of the half-Jacobian, a retargetting frame rate of 1350 fps[10] is achieved, which closely correlates with the earlier timings given in section 4.3 for the theoretical analysis and section 4.5.3 for the empirical results.

Figure 4.16 demonstrates the retargetting of the legs, comparing the scaled motion and the retargeted motion. For this motion, the hands back are retargeted back to their original locations. Figure 4.16 illustrates how the feet of the scaled character appear to float and slide over and through the surface of the ground, whereas the retargeted version fixes this problem and correctly plants the feet on the floor when it is appropriate. This is noticeable by the more concentrated foot region in the retargeted motion of Figure 4.16b. Furthermore, it is noticeable that the feet of the retargeted character never penetrate the floor whereas those of the scaled motion do.



(a) Scaled Motion (b) Retargeted Motion

Figure 4.16: Comparison of (a) scaling and (b) retargetting the catch base motion

[10] Timing results are obtained by averaging 10 applications of the half-Jacobian on a Pentium 1.4GHz processor (no hyperthreading), where for each frame there are 4 applications of the IK solver.

Equation 4.23 and Equation 4.24 define the absolute location that the hands need to be positioned at in order to meet an incoming football at a given height. However, to provide a more meaningful indicator of the process, the illustrations of the retargeted hands are correlated with the scale factors used during the presentation of the results. Figure 4.17 demonstrates the retargetting of the original motion using a range of different scaling factors from x0.5 to x1.2 of the original motion (see Equation 4.23). Figure 4.17 additionally illustrates the full range encompassed by the different scaling factors by overlaying the scaled motions into a single image. Identical bone lengths are maintained for each retargeted motion, which are different to the original character. The scale factors demonstrated in Figure 4.17 are not symmetrically taken about x1.0 but about x0.85 instead because this factor approximately translates to be the same height of the original character scaled to its new dimension, i.e. the natural height of the new character if no upper-body retargetting were performed.

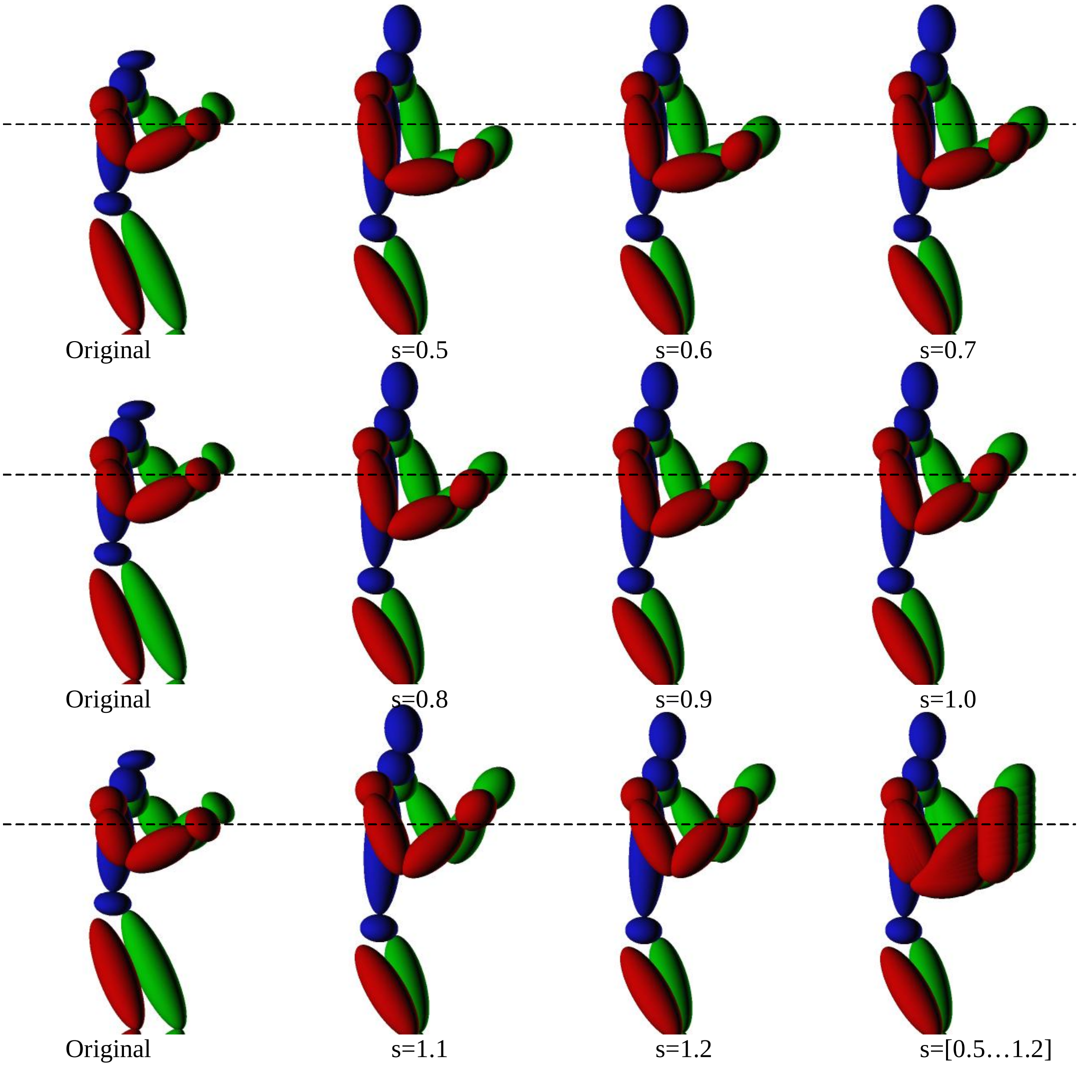


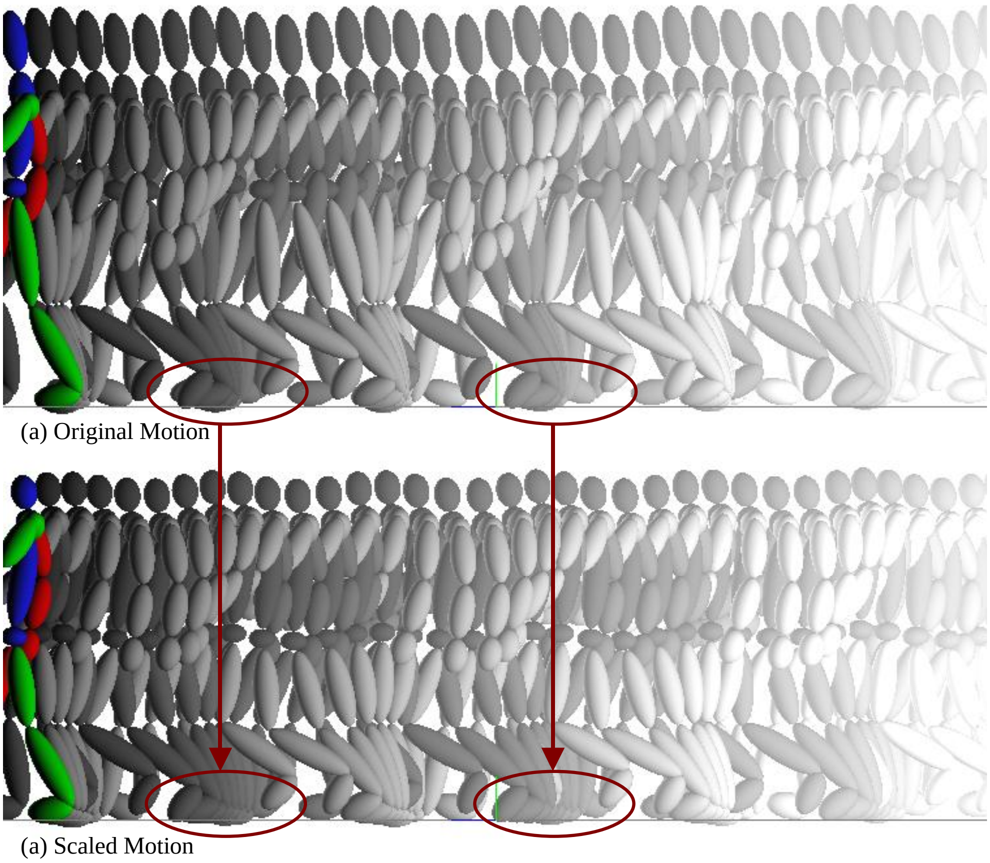
Figure4.17: Retargetting the hand end-effectors to meet different target heights

Figure 4.17 illustrates that a single base motion can be retargeted to produce a range of different motions for different catch heights, while still maintaining a realistic posture and motion, thus providing more accurate environmental interaction. Furthermore, by demonstrate the successful application of the half-Jacobian to posture both the hands and the feet over the traditional full-Jacobian in this example, the computational demands have been reduced by 240% per frame (60% for each of the 4 inverse kinematics applications). To put this into perspective, retargetting the same character using the full-Jacobian would only just have finished reconfiguring one leg in the time that the half-Jacobian has done the complete character, thereby demonstrating a huge speedup bonus for just a single frame.

By including the arms in the retargetting process in this example, the technique has been shown to work equally well for the upper body of a character as it does for the lower, which has been the primarily focus upon till now.

4.6.2 Walking in a Winter-Wonderland

In this example, the focus of the work rests with the lower body of the character by considering a normal walking character motion, which is demonstrated in Figure 4.18a. Figure 4.18b illustrates the problem when scaling the motion data to a new character without retargetting it. The ringed regions in Figure 4.18 highlight the problems of the scaled gait. Instead of the character's feet being planted on the floor, the scaled character's feet are drifting forward rather than being rooted to the spot. This can be seen by the red-ringed wide trail left by the scaled feet as the character walks as opposed to the neat and visually sharp foot plant trail of the original character.



(a) Original Motion

(a) Scaled Motion

Figure 4.18: (a) Original motion capture clip of a walking character; (b) scaled motion capture clip

The retargetting strategy taken for this motion is to leave the upper body joint angles as they appear in the original motion and just modify the character's feet. Therefore, per frame there are only two applications of the inverse kinematics solver – one for each leg. The high-level control process that determines the location of the feet is rather simple yet effective and can be applied in an on-line and real-time manner. As the new, scaled character is walking, if at any point the feet come in contact with the floor then this is taken to be a foot plant which is maintain using inverse kinematics. While the heel of the foot is planted to the floor, the toe height off the floor is replicate in the new motion, which eventually reaches the floor level. If the toes ever try and penetrate the floor, they are projected back up to the correct level.

When the scaled foot is observed to increase in altitude, the heel plant is no longer maintained and released. However, the toes are kept planted using inverse kinematics until there is a further distinct height increase in the scaled motion's toes. This process closely resembles the earlier procedural stride control model given in Figure 4.10, although in this instance, it is not a hard-coded behaviour as it simply follows what the data presents to it and uses these rules to capture possible states in the motion.

Once both the heel and the toes are released from the floor, to avoid a spatial snapping effect as the joint angles are preferred over the end-effector locations, a dampening function is introduced that eases the spatial location from where the plants are released back to that of the scaled motion over a number of frames. This introduces the in-betweening function given in Equation 4.25, which is applied to the foot using inverse kinematics. In Equation 4.25, p_i gives the current frame position based on the current scaled motion position, g_i, and the retargeted previous frame position, p_{i-1}. The parameterisation of u is linearly increased over an interpolation range that covers the number of frames that the ease out function covers. A range of 10 frames is used in the results that are subsequent presented.

$$p_i = g_i(1-u_i) + p_{i-1}u_i \qquad \text{where} \qquad 0 \le u_i \le 1 \qquad \textbf{(4.25)}$$

4.6.2.1 Walking in a Winder-Wonderland (no more) Results

Figure 4.19 demonstrates the character foot retargetting process using the control process to guide the half-Jacobian inverse kinematics solution. From the visual results of Figure 4.19, it can be seen that there are no longer wide foot trails for the foot plants, which were observed in Figure 4.18b for the scaled character. Not only are the feet successfully planted to the ground but also a realistic posture is maintained throughout (which is more visible in the included animation). This is a consequence of introducing additional joint limits that prevent the hip joint rotation too much about the bone length and hence maintain a forward facing foot, much the same as that used in *MovingIK* of section 4.4.

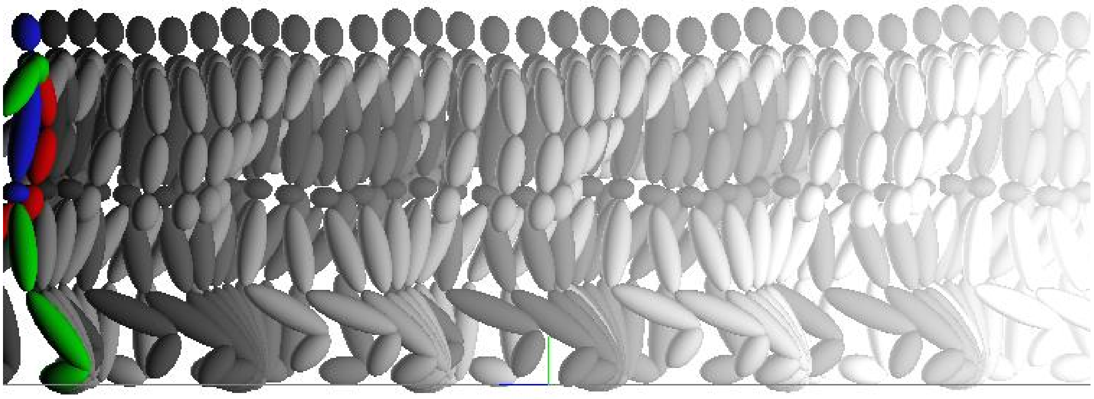


Figure 4.19: Foot plant fixing for a walking character

This example further serves to demonstrate the usefulness of the half-Jacobian over the full-Jacobian in the application of retargetting. In this case an effective cost reduction of approximately 120% is achieved (one application of the IK algorithm per frame for each of the two legs).

With these two examples, the IK retargetting algorithm is temporally ceased at this point because through the accumulative presentation of all the other results in this chapter, a wide practical scope of the half-Jacobian will be presented and proved effective without the need of labouring the retargetting issue. However, the process described in this section is put to further use in Chapter 7 where a comparison between the half-Jacobian inverse kinematics solver and the dynamics techniques is performed.

4.6.3 Retargetting Summary

The retargetting examples that have been presented in this subsection utilise the fast half-Jacobian solution to demonstrate both the computational efficiency and application of the solution in a full-Jacobian scenario. The higher-level control processes that have been presented are rather simple in their design however they do demonstrate original approaches to the specific motions considered. Since these control processes tackle specific motions, unlike the work demonstrated by Shin *et al.* [Shin01], they are not general strategies that can be used to determine when joint angles should be maintained over end-effector positions for arbitrary motions. Although because the importance factors are determined by proximity in Shin *et al.*'s work, these can lead to mistaken end-effector preference over joint angles.

The half-Jacobian retargetting results presented have highlighted the importance of finding efficient solutions to the inverse kinematics process for real-time applications because minimally two applications of the solver per frame are required - one per foot. When the hands are also repositioning, the accumulative computational efficiency gain increases to approximately 240% over the full-Jacobian process. Therefore, through the use of the half-Jacobian technique, retargetting appears as a much more attractive algorithm to be used in real-time applications, such as games, to remove current visual artefacts including foot sliding and incorrect environmental interaction.

4.7 Weighted Inverse Kinematics

The nature of Jacobian-based inverse kinematics solvers (both half- and full-Jacobian based) provides the opportunity to predictably modify how much different DOFs change when configuring an articulation. The result of applying this modification to the Jacobian-based algorithm is the ability to introduce subtle but individualised results for the same end-effector position over time. For example, the rate of angle change for the knee over the hip joint can be favoured, thus giving the visual appearance of a stiffened hip. Weighted Inverse Kinematics [Mere04b, Mere04c, Mere04d] is the term used to refer to this novel technique, which is describe in this section.

To explore the concept of weighted IK, the original Jacobian-based algorithm that was presented in Figure 4.5 needs to be considered. For increased clarity, the extra refinement that was made to include joint constraints is temporarily ignored. However, the joint constraints can be reapplied in a similar fashion after the process this section illustrates without loss of functionality.

The algorithm presented in Figure 4.5 distributes the angular changes needed to meet the desired end-effector location evenly over the IK chain. However, weighted IK modifies step 6 to include a set of dynamic weights that redistribute the contribution each DOF has in the resulting motion. Equation 4.26 replaces the angular update at step 6 of Figure 4.5, where W is a weighting vector.

$$q = q + WJ^{-1}dX \tag{4.26}$$

The weighting vector, *W*, contains real values between 0 and 1 where smaller values result in less significant changes in angle and larger values correspond to bigger angle changes. This principle is illustrated in Figure 4.20 where the IK solver is applied to a simple hierarchical structure using different weighting values. The weighting parameter for the first joint angle, which is at the root, in Figure 4.20, is reduced and compared this with an even distribution. As the illustration demonstrates, the joint angle which has a reduced weighting moves less therefore other joints in the chain have to move more to meet the desired end-effector position. This is compared to the evenly-weighted IK chain in which each of the angles involved in the chain are changed relatively equally.

Although a weighting change has only been applied to one of the angles in 4.20, the principle of relatively stiffening up joints within an IK chain equally applies when changing multiple weighting values. However, as can be seen from the graph of Figure 4.20, reducing a weighting on one joint has the effect of indirectly increasing the weights of the remaining joints because the difference needs to be compensated for. This cause and effect result needs to be considered when applying weighting values to an IK chain. Through empirical experiments it has been found that as long as these values are specified relative to each other, the results obtained from the solution demonstrate intuitive properties. If the weights are not determined in a relative manner but instead along an absolute scale, the visual results obtained would not necessarily follow that which is expected based on the weighting parameters.

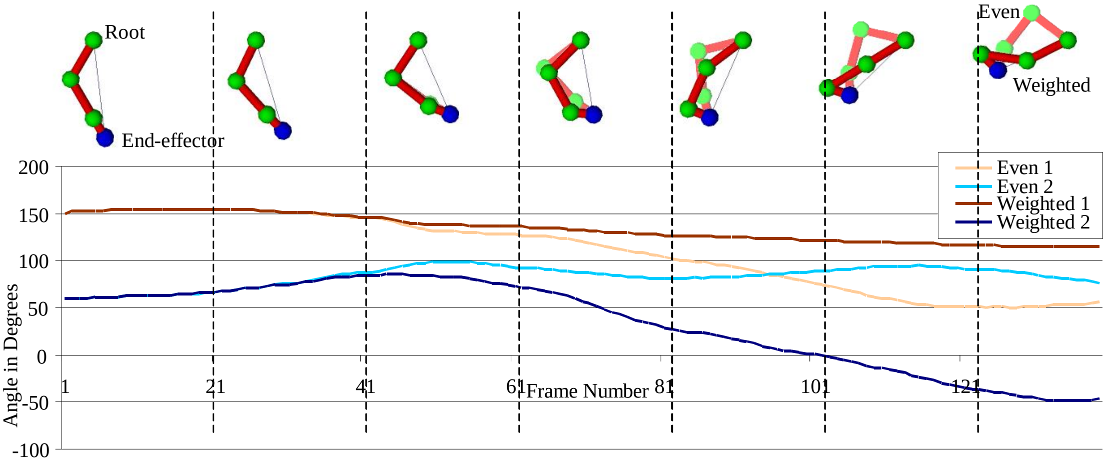


Figure 4.20: Application of weighted IK chains on a simple articulated structure. Top: Comparison of even (lighter colour chain) and weighted (darker colour chain) distribution update of angles – the root node angle has a reduced weighting value over the rest of the nodes in the IK chain. Bottom: Graph of the first two angles in the IK chain working from the root node outwards

4.7.1 Using Weight Inverse Kinematics to Individualise Characters

The property of weighted IK chains can be harness as an effective tool to produce, effectively, an infinite variation of individualised motions. This is achieved by affecting the stiffness of specific nodes within an IK chain such that over time, despite the same end-effector being used as an evenly weighted version, the character's limbs follow a different joint trajectory, as was the case in the example of Figure 4.20. Furthermore, the inclusion of the weighting factor comes at negligible extra cost to the core IK algorithm and hence works in real-time unlike many of the existing techniques that rely on complex dynamics to achieve a similar goal.

MovingIK (see section 4.4) is used as a harness to demonstrate the principle of weighted inverse kinematics. However, the basic application needs to be slightly updated and elaborated to make use of the weighted IK technique. The inverse kinematics solver within the Animation module is adjusted to include the weighting vector of Equation 4.26. The IK weighting values are supplied by the Control module and maintained as part of the stylisation parameters. The stylisation parameters additionally include states that affect the input from the control sources. These allow, for example, a linear reduction in the maximum stride length and speed in order to simulate fatigue or give the visual appearance of a limp.

The stylisation state of the Control Module is dynamically changed in response to the system's Control Sources. These can take a variety of different forms including responses to environmental events or an AI engine. An example event would be getting shot in the leg resulting in a walking motion becoming a limping one. However, the reason for changing the stylisation state is abstract away in this work where only the result of performing such as change is presented.

For subtle changes involved in individualising a character performing a normal motion, the weighting parameters are only changed from the even version slightly. This has the effect of simulating limb build within the model. For example, large limbs with low muscle tone would have low weights to simulate a sluggish movement while muscular limbs of the same size would have higher weightings to account for the strength of the muscle.

In order to simulate injuries, as well as adjusting the control parameters, the corresponding limb is stiffened by decreasing the weighting value associated with it. However, unlike the subtle changes that are associated with small deviations from the even chain, to portray injuries, large differences in the weighting matrix are introduced to increase the visual stiffness of the joint. This reflects a predominate change in the character's motion in the part of their body where a restriction is introduced by an infliction.

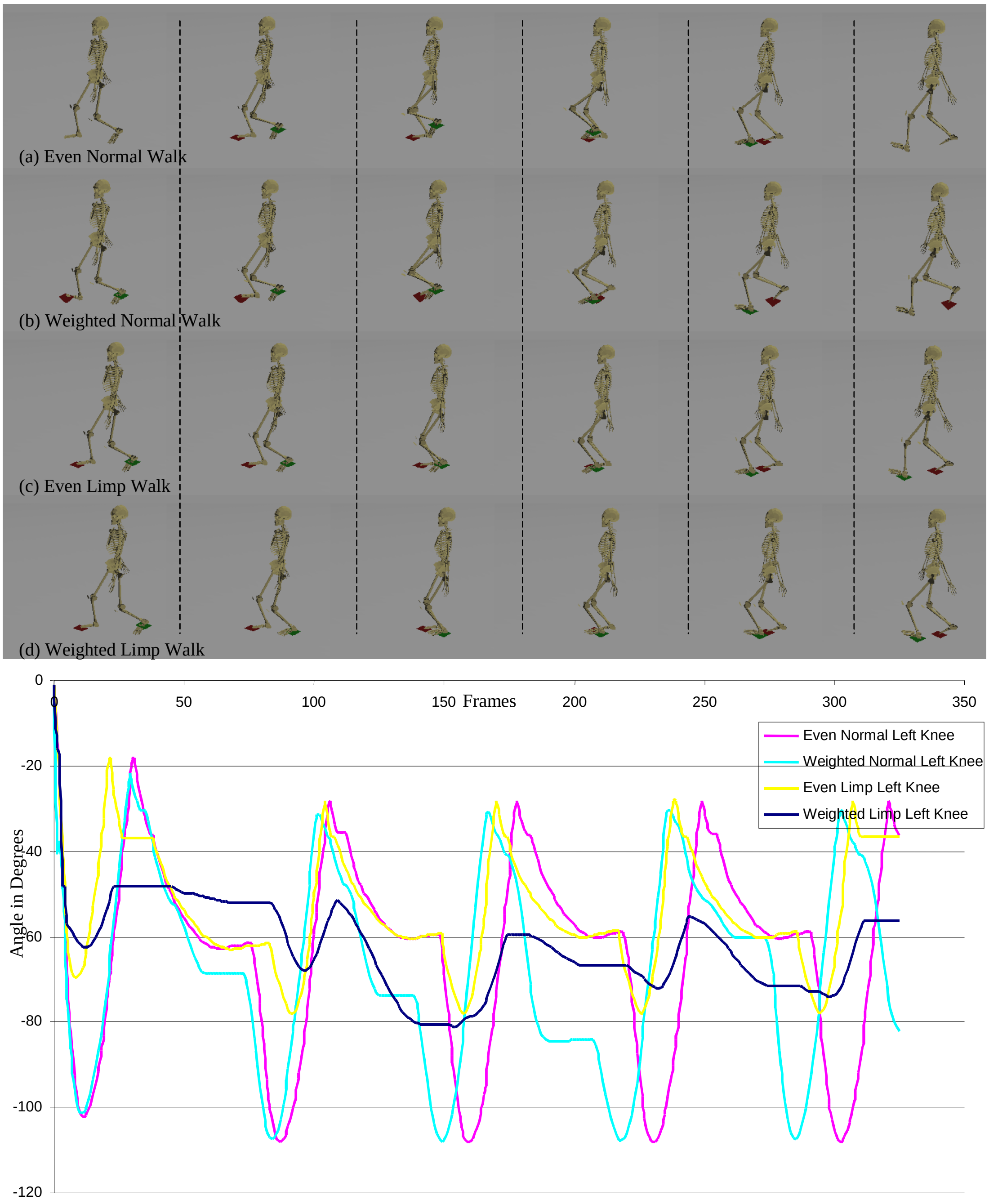
To illustrate the potential of this technique, the following section demonstrates the results obtained by changing the stylisation parameters of a procedural walk. This is further extended in section 4.8 by demonstrating how motion capture data can be individualised using weighted inverse kinematics.

4.7.2 Stylising a Procedural Gait using Weighted Inverse Kinematics

To illustrate the weighted IK technique, this section presents a range of different motions that are all produced using the same set of constants and inputs based on a procedural gait motion, with just the weighting vector being varied. All results are obtained in real-time running on a Pentium 4 1.4GHz with a GeForce2 Ultra graphics card, which demonstrate similar times to those given during the empirical study of the half-Jacobian (see section 4.5.3).

Figures 4.21a & 4.21b illustrate how the change in weighting parameters of a character generates an individualised gait by introducing a slight deviation in the normal walking motion. To produce the weighted walking motion of Figure 4.21b the weighting value at the hip joint is decreased by 20% and the rest of the weights were left the same as used in Figure 4.21a. From the graph in Figure 4.21e, the weighted chain left knee joint still reaches the same maximum and minimum angular measurement as in the evenly distributed walk, however it follows a different path over time to achieve this. This correlates with the earlier theorised results in that the effect of using the weighting reduces the speed with which the hip angle changes therefore the knee angle is changed more to compensate. This can be seen in the middle frames of Figure 4.21a and Figure 4.21b where the character's left knee is further forward in the evenly distributed solution compared to the weighted version for the same point in time.

As the IK chain starts to straighten and become singular, as it does at the extents of the walking cycle, the weighted version takes on a similar configuration to that of the evenly distributed one. This is because the possible solutions the IK solver can generate are more tightly packed into a smaller spatial configuration area so the results look similar in either case. Figure 4.21a and Figure 4.21b illustrate this where it can be seen in the first and last frames that the configurations are virtually identical.



(a) Even Normal Walk

(b) Weighted Normal Walk

(c) Even Limp Walk

(d) Weighted Limp Walk

(e) Graph of left knee joint angle for the walking characters (a)-(d) over 5 strides

Figure 4.21: Demonstration of using weighted chains for (b) individualisation and (d) injury simulation compared to the even distribution of joint changes for the same motions (a) & (c) respectively. Images of (a)-(d) are over a single stride of the left foot. (e) graphs the left knee joint of the 4 motions over time

Despite the solution looking similar at the beginning and end of the cycle, the differences during the walking phase demonstrate an individualisation of the character, which is achieved by purely applying weighting vectors to the character. The end configurations could be further diversified if the control parameterisation and skeletal limb lengths were adjusted. However, these results keep as many parameters constant as possible to demonstrate the potential of weighted IK chains. Furthermore, the Data and Animation Modules are additional contributing factors to the similar looking configurations at the extremities. This is due to the manner in which the character lands using a two-stage process (see Figure 4.10), which has been effectively fixed, thereby limiting the possible configuration space.

By changing the weighting parameters alone many individual motions can be generated. However, through experimenting with the weighting vectors, the most natural-looking motions tend to be linked with low variances in the weight vector used. Larger variances in the weight vector lead to noticeable exaggeration in the resulting motion because joint angles are not updated significantly until there is no choice in order to meet an end-effector location. Section 4.8 will provide further evidence of this correlation when the technique is applied to individualising motion capture data. The motions generated with high-variance weighted vectors could account for normal motion in defined cases however the technique can also be used to create injuries.

The generation of an injured walking motion is illustrated in Figure 4.21c & Figure 4.21d, with both the use of weighted and even distribution over the IK chains. In both of these illustrations, the control parameters have been similarly adjusted from the normal walking motion. The control parameters are adjusted for the left stride in order to decrease the maximum flight height by 50%, reduce the stride length by 50% and increase the speed with which the stride is undertaken by 30%. These values were somewhat arbitrarily chosen based on the behaviour of a real person walking to give a visible difference between the left and right motions of the body.

Comparing Figure 4.21a, which shows the evenly weighted walk, and Figure 4.21c, which shows an evenly weighted limp, it is clear that simply adjusting the control parameters is enough to visually change the appearance of the walking motion. This is most visually apparent between the middle frames of the two motions in Figure 4.21. Between these frames, the effect of reducing the flight height of the foot is clear to see, and in the latter frames of the motion it is also visible that the left foot does not travel as far forward as in the normal walking motion. An additional visual effect that is not demonstrated in the stills of Figure 4.21 is that the speed with which the stride cycle is undertaken is faster for the injured leg than for the normal walking motion. This simulates the behaviour of the character attempting to minimise the amount of time it is putting weight onto the injured leg, which was observed from a real actor pretending to limp.

Through only adjusting the control parameters of the walking motion it is possible to produce a limping motion, however the realism can be enhanced by additionally adjusting the weighting parameters. This is illustrated in Figure 4.21d where an uneven weighting is used in the IK solution to produce the limping motion. The weight parameters used to enhance the limp in Figure 4.21d impose a stiffening effect on the left knee, which is achieved by reducing the weighting value for this joint by 90%. As the DOF plot in Figure 4.21e illustrates, the effect of applying this weight is to noticeably reduce the amount of movement in the limb compared to the evenly distributed limp motion. This visually translates well into the resulting motion, which can be seen when comparing the frames of Figure 4.21c and Figure 4.21d. As these frames show, in order to compensate for the reduced movement in the knee, the hip joint of the weighted motion changes comparatively more than for the evenly distributed one. This is illustrated when comparing the absolute position of the knee joint between the corresponding frames.

The weighting vector used to simulate the motion of Figure 4.21d is tailored to produce an animation that depicts a knee injury. However when the weights are reset and the value associated with the hip joint is decreased, a hip injury can be simulated instead. This new placement of the injury is thus achieved by maintaining the same control parameters, which further demonstrates the usefulness of the weighting values in generating subtle differences between base motions. In effect, the resulting motion can be customised for a specific stylisation, which makes is possible to determine where the injury is being simulated along the leg.

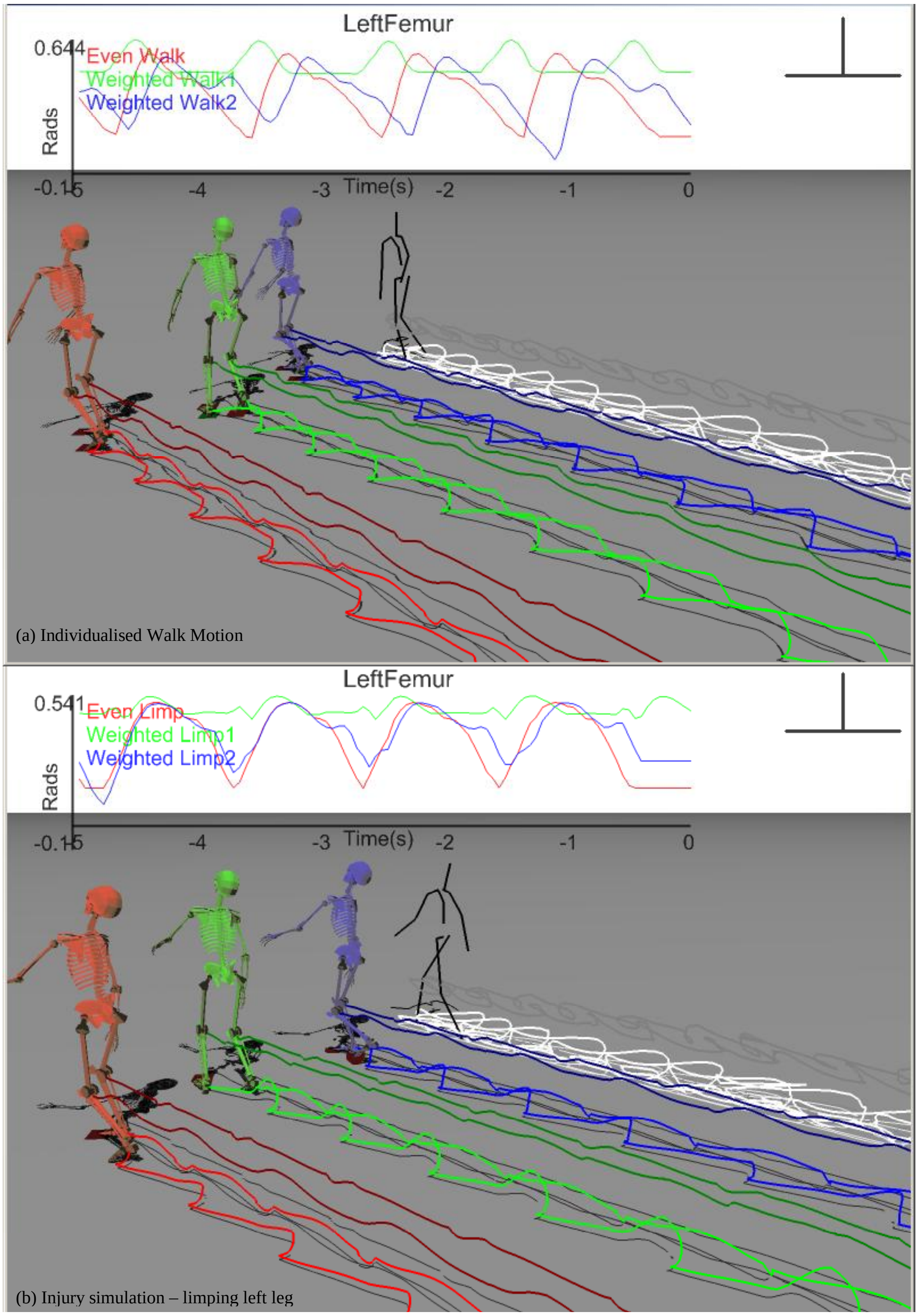
4.8 Motion Capture Individualisation using Weighted Inverse Kinematics

This section demonstrates the weighted IK approach present in section 4.7 for individualising motion capture data. The individualised motions of Figure 4.22 are portrayed with the skeleton bodies while the original motion capture data is visible as a stick character to the right of the skeletons. Each of the three skeletons in Figure 4.22a is subject to the same control parameters, but, the weighting vector applied to the inverse kinematics solver varies over the skeletons. The red skeleton has an evenly weighted distribution, i.e. all the values are 1, whereas the green and blue skeletons have weighting vectors that stiffen the hip and knee joints respectively. In the case of the unevenly weighted skeletons, the respective weightings for the joints have been decreased by 80% and 20% and the rest of the weighting values are the same as for the red skeleton.

Similar to procedural motions of Figure 4.21, from the joint angle graph of Figure 4.22a, the hip angle for the blue skeleton can still be seen to reach a similar maximum and minimum angular measurement as the evenly distributed red character, however it follows a different path over time to achieve this. The effect of using the weighting reduces the rate of angular change for the knee joint for the blue character and hence the hip angle is changed differently to compensate. In comparison, the green skeleton has a stiffened hip joint therefore, as can be seen from the joint graph of Figure 4.22a, the amount of change for this part of the body is much reduced. In this case, extra movement in the knee joint compensates for the reduced angular change that can be seen in the green skeleton's hip.

The motion capture individualisation further supports the earlier conclusion of section 4.7.2 that the most natural-looking motions tend to be linked with low variances in the weight vector used. This can be seen by comparing the green and blue skeletons from Figure 4.22a, where the green character has an 80% reduction in its weighting value compared to 20% for the blue character. It can be seen that the green character has a much-reduced range of motion due to the high weighting vector variance, which results in a more jerky movement compared to the blue skeleton, thus more applicable to injured looked movements.

The simulation of an injured walking motion on top of the motion capture data is illustrated in Figure 4.22b, where both even and non-even weightings are used across the IK chains. In the case of each of the skeletons of Figure 4.22b, the weighting parameters are the same as those used in Figure 4.22a, except that the weights have only been applied to the left leg, which is the one that is being made to limp. The control parameters are adjusted for the left stride in order to decrease the maximum flight height by 50%, reduce the stride length by 50% and increase the speed with which the stride is undertaken by 30%. These adjustments are made uniformly over each simulated character.



(a) Individualised Walk Motion

(b) Injury simulation – limping left leg

Figure 4.22: Application of *MovingIK* to adapt original motion capture data to (a) individualise and (b) simulate injury to three different characters of different IK weighting vectors

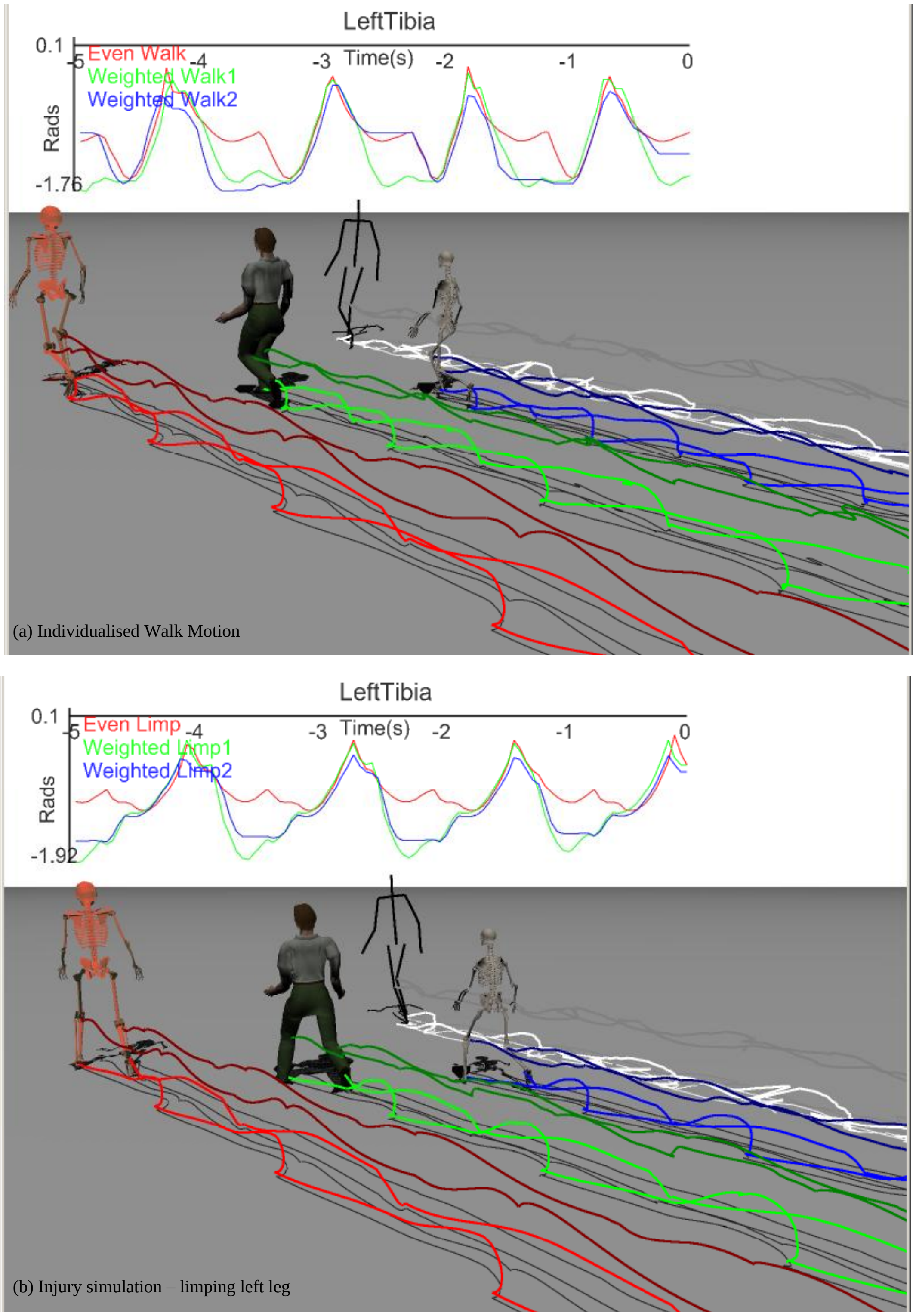
Adjusting the control parameters is enough to change the visual appearance of the walking motion to a strange limping one, which can be seen by comparing the red skeletons of Figure 4.22a and Figure 4.22b. This is most apparent when comparing the motion trails for the limping red character's feet. Here the left hand side trail barely skims over the ground whereas the right hand side trail contains much more clearance. Furthermore, the trail for the right foot is much smoother and travels further per stride than the limping left foot, a trait that can be seen in all of the limping skeletons of Figure 4.22b.

The green and blue skeletons of Figure 4.22b demonstrate the enhanced realism obtained by introducing weighting parameters on top of the modified control parameters. Similar to the weightings applied to the normal walking motion, the weightings that have been applied to the green skeleton stiffen the left hip joint while the blue skeleton's left knee joint is made stiffer. As the joint angle graphs of Figure 4.22b illustrate, the effect of applying these weightings is to noticeably reduce the amount of movement in the limb compared to the evenly distributed limp motion. This is most noticeable for the hip joint of the green skeleton, which is the angle that is graphed in Figure 4.22b.

Figure 4.22b further illustrates how it is possible to adjust the weightings for the IK chain such that an injury can be simulated in a different part of the leg. The weighting vector used to simulate the motion of the blue skeleton is tailored to produce an animation that depicts a knee injury. Conversely, when the weight value associated with the hip joint is decreased, a hip injury is simulated as illustrated with the green skeleton. This is achieved by maintaining the control parameters and only adjusting the weighing values.

As further example of the potential of the weighted inverse kinematics individualisation process, Figure 4.23 is presented. The original motion capture data is illustrated by the stick figure, while the individualised motions of Figure 4.23 are portrayed by the two different sized skeleton characters and the human-looking skinned one. The red skeleton of Figure 4.23a uses an evenly weighted IK vector, whereas the human and grey skeleton characters have a 20% reduction in their hip and knee joint weightings respectively. Unlike the uniform control parameters of Figure 4.23a, the control parameters used in the production of the limping motions of Figure 4.23b are adjusted for the left stride in order to decrease the maximum flight height by 50%, reduce the stride length by 50% and increase the speed with which the stride is undertaken by 30%. An even IK weighting is used for the red skeleton in Figure 4.23b, whereas a reduction in the weighting vector of 80% is used for the hip and knee joints of the human and grey skeleton characters respectively.

The simulated motions illustrated in Figure 4.23 illustrate the same behaviour to that which has been discussed. Thus, as the results demonstrate, the additional factor of weighted IK chains provides a good level of differentiation between the changes in joint angles within the character, which visually introduces subtle changes for motions that have the same control parameters. This allows many different motions to be spawned, each individualised towards a specific character's attributes, at no extra computational cost to the core IK algorithm.



(a) Individualised Walk Motion

(b) Injury simulation – limping left leg

Figure 4.23: Alternative application of *MovingIK* to adapt original motion capture data to (a) individualise and (b) simulate injury to three different characters of different IK weighting vectors

4.8.1 Weighted Inverse Kinematics Character Individualisation Summary

The use of real-time inverse kinematics to adapt existing motion-captured animation has the primary advantage of reducing visual artefacts associated with such animations, i.e. the motion is successfully retargeted to the new character and environment. Through the addition of weighted inverse kinematics chains, an enhancement to the technique has been demonstrated that produces richer visual realism at no extra computational cost. Using weighting values, it is therefore possible to take a single motion-captured animation and adjust the motion to different sized and stylised characters. This demonstrates a computationally cheap mechanism for producing new retargeted and individualised character animations from a single motion-captured data file. Taking the technique a step further, it has been shown how the same base representation can be adapted to simulate injury stylisation by adding in discrete visual differences.

From the results, is has been discovered that weighting vectors which have small variance values over their elements (i.e. small difference in the weight values) produce normal but subtly different motions suitable to the individualisation of character motions. This level of individualisation allows the relative build of the character to be controlled by assigning comparatively smaller weights to those joints that would be expected to change less than others due to muscle structure or limb mass. This is due to the direct relationship between the amount of angular change performed during the IK algorithm and the weighting values. This will be returned to in section 6.2.4, which looks at the affects of changing the biomechanics of a character in a dynamics-based simulation.

The use of larger variances within the weighting vectors generates exaggerated motions that have been used to depict an injured motion. The simulation of injuries is enhanced through the use of changes in the control parameters and although this has the effect of fundamentally changing the resulting motion, the weighting values give further control in producing a good-looking motion. This is further explored in the following section where weighted inverse kinematics is use to map the motion of one actor to a completely different one, which is further evaluated.

4.9 Mapping the Motion of one Actor to another Using Weighted Inverse Kinematics

Based on the results presented in section 4.8 for generally individualising motion capture data, this section discusses the potential of using weighted inverse kinematics to map the motion of one actor to another using only a weighting vector, thus forming a complete reconfiguration solution. The motions that are used as the base motion of the reconfiguration process are taken from the dataset presented in Chapter 3. The weighting vectors used during the reconfiguration process are based on the biomechanical information of the actors within the dataset. This allows the reconfigured motions to be evaluated against the real motions of the actors who provide the biomechanics for the weighting vector.

4.9.1 Mapping the Gait Motion of Actor C to Actors A, B, and D

In this subsection the normal gait motion of actor C is reconfigured to the actors A, B and D, changing only the limb lengths and the weighting vector. All other control processes and variables are kept constant throughout the reconfigurations of actor C to the target actors. The final weights that are used to affect the outcome of the weighted inverse kinematics reconfiguration process are outlined in Table 4.5, which additionally summarises the mass distribution of the actor's legs for comparison. Initially a weighting vector for each actor is determined by considering the relative mass distribution within the leg, where a higher relative ratio component for the leg is given a lower overall value in the weighting vector to correspond with a damping affect due to the higher amount of energy required to move it. However, the final weightings values present in Table 4.5 are arrived at by a degree of trial-and-error, starting from the initial guess, to give the best matching generated gait – the weighting vectors in Table 4.5 provide the best results for the target actor, where slight deviations gives less good results.

Node Name	Actor C (Control Weights)	Actor A	Actor B	Actor D
Femur weight	6.95 kgm/s^2	6.76 kgm/s^2	5.12 kgm/s^2	5.66 kgm/s^2
Tibia weight	3.02 kgm/s^2	3.40 kgm/s^2	4.30 kgm/s^2	3.48 kgm/s^2
Foot weight	1.14 kgm/s^2	0.96 kgm/s^2	1.04 kgm/s^2	1.01 kgm/s^2
Relative Ratio	10:10:10	11:14:10	10:20:12	10:14:10
Weighting Vector	N/A	[1, 0.1, 1]	[1, 0.05, 1]	[1, 0.2, 1]

Table 4.5: Weighting vectors used to individualise the gait of actor C to actor A, B & D, where the weighting vector corresponds to the weightings applied to the femur, tibia and foot respectively

Due to the high correlation between the relative ratio and the weighting vectors for each actor in Table 4.5 it would be possible to mathematically link the two sets of attributes. However, this is refrained from because of the small number of actors in the dataset and hence any such equation would be superficial.

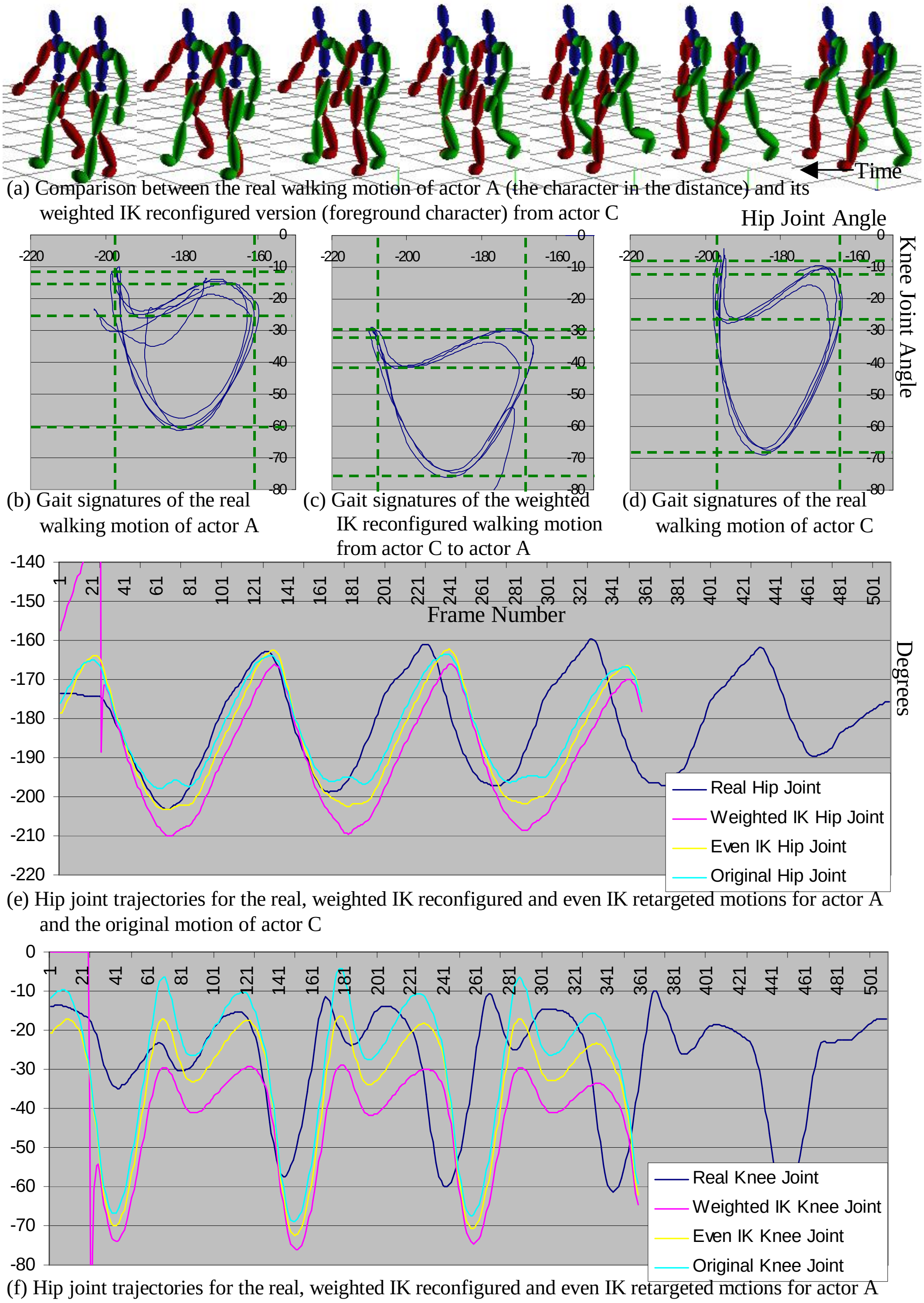
The motions that result from the weighted inverse kinematics reconfiguration process using the weighting vectors of Table 4.5 are give in Figure 4.24 through to Figure 4.26 inclusive. The joint trajectories of these figures include the following motions:

§ The real walking motion of the target actor,

§ The weighted inverse kinematics reconfigured walking motion for the target actor using actor C's gait as the example motion,

§ The even inverse kinematics retargeted walking motion for the target actor using actor C's gait as the example motion (i.e. the weighting vector has elements all set to 1 for even IK),

§ The original walking motion of actor C



(a) Comparison between the real walking motion of actor A (the character in the distance) and its weighted IK reconfigured version (foreground character) from actor C

(b) Gait signatures of the real walking motion of actor A

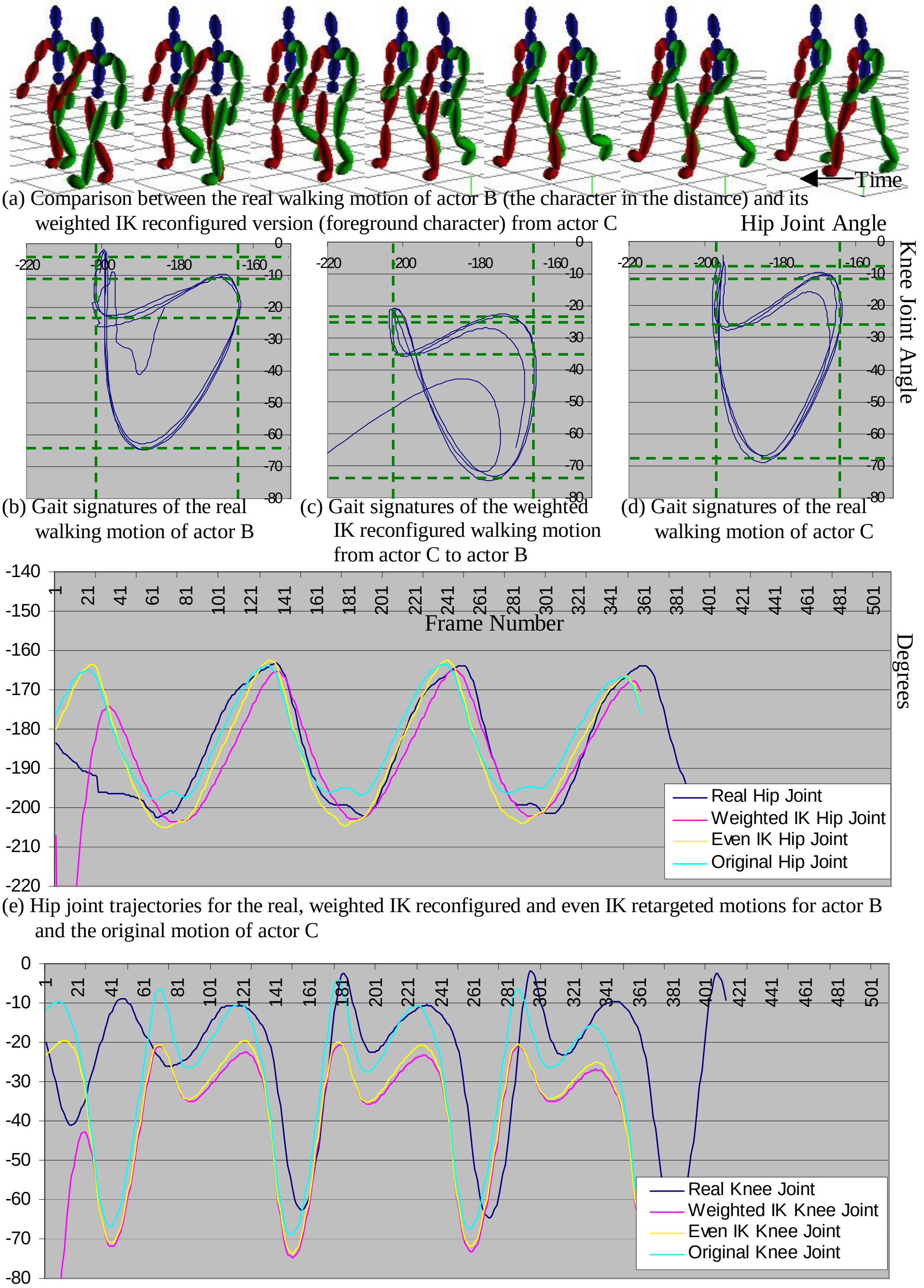
(c) Gait signatures of the weighted IK reconfigured walking motion from actor C to actor A

(d) Gait signatures of the real walking motion of actor C

(e) Hip joint trajectories for the real, weighted IK reconfigured and even IK retargeted motions for actor A and the original motion of actor C

(f) Hip joint trajectories for the real, weighted IK reconfigured and even IK retargeted mctions for actor A and the original motion of actor C

Figure 4.24: Actor C to Actor A: Weighted inverse kinematics mapping of the normal gait motion of actor C to actor A using the corresponding weighting vector of Table 4.5. (a) Illustrates the reconfigured motion compared to the real one for actor A, (b) gives the gait signature of the real walking motion of actor A, (c) gives the gait signature of reconfigured motion from actor C to actor A, (d) gives the real gait signature of actor C and (e) & (f) graph the hip and knee joint trajectories respectively of the real, weighted IK reconfigured and even IK retargeted motions for actor A and the original base motion of actor C



(a) Comparison between the real walking motion of actor B (the character in the distance) and its weighted IK reconfigured version (foreground character) from actor C

(b) Gait signatures of the real walking motion of actor B

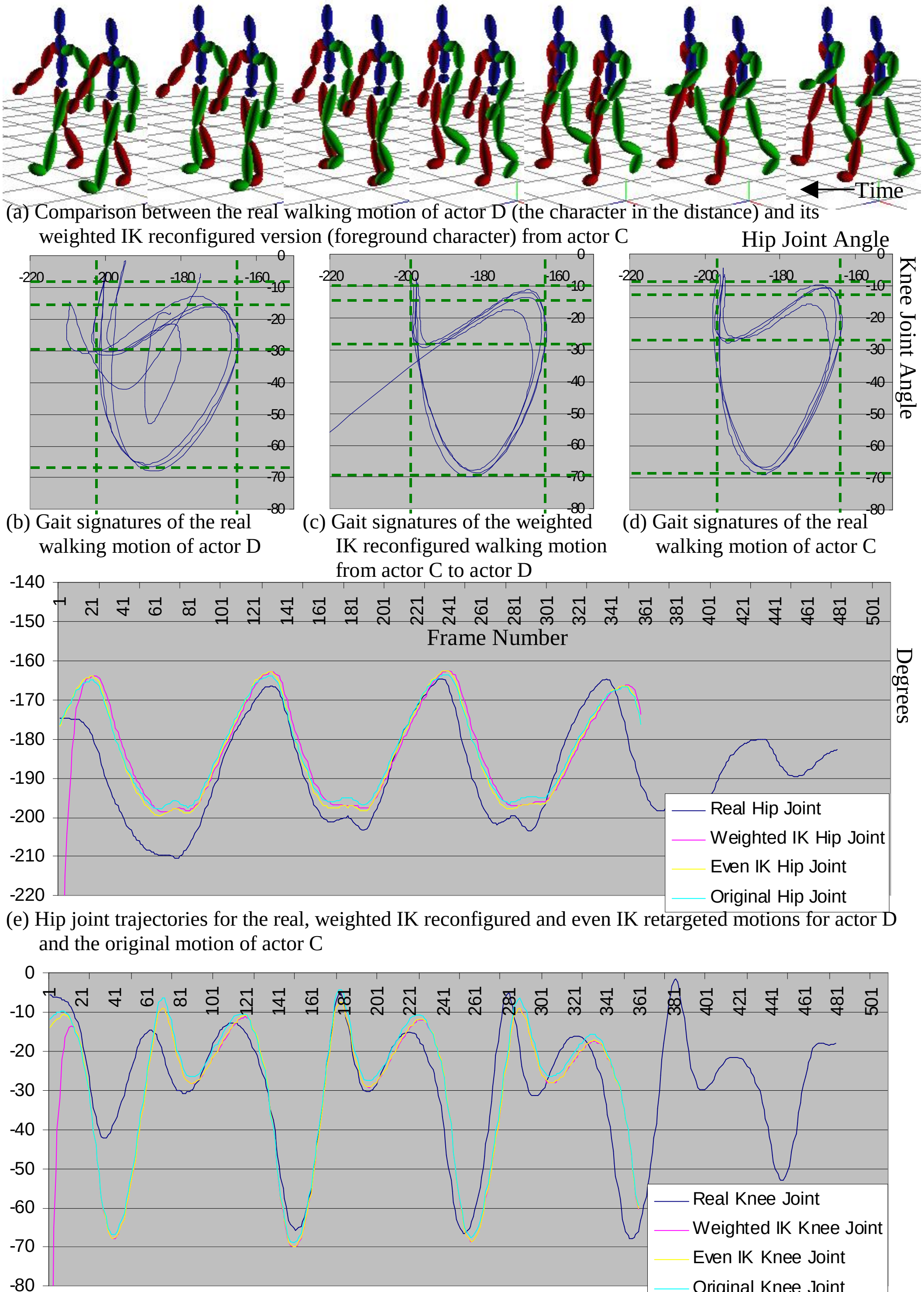
(c) Gait signatures of the weighted IK reconfigured walking motion from actor C to actor B

(d) Gait signatures of the real walking motion of actor C

(e) Hip joint trajectories for the real, weighted IK reconfigured and even IK retargeted motions for actor B and the original motion of actor C

(f) Hip joint trajectories for the real, weighted IK reconfigured and even IKretargeted motions for actor B and the original motion of actor C

Figure 4.25: Actor C to Actor B: Weighted inverse kinematics mapping of the normal gait motion of actor C to actor B using the corresponding weighting vector of Table 4.5. (a) Illustrates the reconfigured motion compared to the real one for actor B, (b) gives the gait signature of the real walking motion of actor B, (c) gives the gait signature of reconfigured motion from actor C to actor B, (d) gives the real gait signature of actor C and (e) & (f) graph the hip and knee joint trajectories respectively of the real, weighted IK reconfigured and even IK retargeted motions for actor B and the original base motion of actor C



(a) Comparison between the real walking motion of actor D (the character in the distance) and its weighted IK reconfigured version (foreground character) from actor C

(b) Gait signatures of the real walking motion of actor D

(c) Gait signatures of the weighted IK reconfigured walking motion from actor C to actor D

(d) Gait signatures of the real walking motion of actor C

(e) Hip joint trajectories for the real, weighted IK reconfigured and even IK retargeted motions for actor D and the original motion of actor C

(f) Hip joint trajectories for the real, weighted IK reconfigured and even IK retargeted motions for actor D and the original motion of actor C

Figure 4.26: Actor C to Actor D: Weighted inverse kinematics mapping of the normal gait motion of actor C to actor D using the corresponding weighting vector of Table 4.5. (a) Illustrates the reconfigured motion compared to the real one for actor D, (b) gives the gait signature of the real walking motion of actor D, (c) gives the gait signature of reconfigured motion from actor C to actor D, (d) gives the real gait signature of actor C and (e) & (f) graph the hip and knee joint trajectories respectively of the real, weighted IK reconfigured and even IK retargeted motions for actor D and the original base motion of actor C

The gait signatures and joint trajectories of Figure 4.24 through to Figure 4.26 show that similar ranges of motion between the real and reconfigured movements have been achieved. In the case of the knee joint actor A has a range of 45 degrees, actor B has a range of 55 degrees and actor D has a range of 60 degrees, which corresponds to the ranges of the real actors. This is in contrast to both the base motion and retargeted motions where the knee range of motion is significantly large for the target actors A (Figure 4.24) and B (Figure 4.25). An analogous closeness in the hip range of motion additionally follows this pattern where between the real and reconfigured motions more similar overall ranges are achieved compared to the original and retargeted motions.

However, despite affecting the desirable changes in the joint ranges of motion, there is still considerable difference in the resulting motions. This is illustrated by the large differences in angular offsets demonstrated in the gait signatures and joint traces of Figure 4.24 through to Figure 4.26. Although a good joint range is obtained, the starting, and hence the end point, of the range varies a great deal between the original and the reconfigured motions. For example, the real knee joint of actor A ranges between -10 & -60 degrees, whereas the generated knee joint shifts this down to -30 & -80 degrees; a 20 degree shift. A similar effect is visible in the hip joint trajectories, but perhaps to not such a great extent.

The reason for the joint range offsets is due to the manner in which the motion reconfiguration is applied using the retargetting process to provide the control processes, as described in section 4.6. The control processes phase in and out of the foot plants to smoothly traverse between the end-effector locations and the original joint trajectories. The effect of this is to rigidly determine the foot location throughout most of the motion (and not just the foot plants) using the higher-level control process. Therefore, given the starting configuration for the IK algorithm for a specific frame is determined from the previous frame’s configuration a compound dampening effect occurs over time. This pushes the joint range of motions downwards and is never compensated for because of the relatively high restrictions on the foot and toe plants by the control processes. Hence the overall range of motion stays within its effected dampened range that is pushed downwards compared to the original and real motions. Effectively, the dampening influence of the weighted inverse kinematics algorithm has a tendency to shift the joint ranges involved when both the hip and heel/toe plants are specified for the majority of the motion.

A further difference when comparing the gait signatures of the generated motions is that they have a tendency to slant to the right near the bottom of the pattern (most negative), which is most noticeable in the motion of actor B (Figure 4.25). This is caused by the continual dampening affect of the weighted inverse kinematics algorithm that occurs throughout the individualisation of the motion, where the hip joint is free to move more relative to the change in the knee joint because of the corresponding weightings. This indicates that the complex interactions that occur during the real motion of an actor do not consistently dampen the motion of a given joint. However, this is what is happening with the weighted inverse kinematics approach. It might be possible to dynamically adjust the weighting vector as the motion reaches different stages within the walk cycle, which at the moment is constant per actor, thereby producing different dampening conditions over the course of the stride to generate a better fitting motion. However, this would require a more detailed study of human locomotion and biomechanical interactions and therefore would constitute an area of future work to build upon the weighted inverse kinematics technique.

4.9.2 Mapping Motion from Actor C Discussion

The reconfiguration of actor C's gait to the correct dimensions of the other three actors using an appropriate weighting vector to approximate the mass distribution of the leg, have shown that at a high level there are some similarities towards the real motions. The visually passable motions that have been generated by the weighted IK algorithm, which accompany Figure 4.24 through to Figure 4.26, further complement this result. However, because of the control processes used, the weighted inverse kinematics technique also introduced a couple of undesirable properties into each of the motions, i.e. the range offset and slanting gait signatures.

The undesirable changes in the joint trajectories, and hence gait signature, are due to the control process employed to drive the inverse kinematics technique. It is therefore conceivable that other, more complex processes can be designed to better take advantage of the weighted IK approach and result in more comparable reconfigured motions to the real movements of the target actor. The basis for this future work has at least been proven in this section by the technique's ability to minimally approximate the desirable ranges of the real actor from a base motion and weighting vector. Furthermore, as Chapter 6 will subsequently discuss, the application of weighted inverse kinematics to dampen joint movements during motion has a correlation with the adjustment of biomechanical limb masses in a fully dynamics-based motion reconfiguration technique. Thus using the correct control processes it should be possible to generate more plausible reconfigured motions using the very computationally cheap weighted inverse kinematics approached.

4.10 Conclusions

In this chapter, an analytical breakdown of the Jacobian-based inverse kinematics solver has been presented, where the computational cost of the half- *vs.* the full-Jacobian is compared. In correlation with the analytical results, the empirical results supported the fact that the half-Jacobian is the less computationally demanding approach by approximately 60%. The empirical results additionally demonstrated that the half-sized Jacobian proved just as computationally stable as the full version. This consequently illustrated how a computational speed up is found using a Jacobian-based solution, without loss of generality to large chained articulations.

It was further demonstrated how the half-Jacobian could be used where normally the fully-Jacobian would be used. The process of utilising the half-Jacobian in a full-Jacobian domain is achieved through the use of additionally constraints. The purpose of the added constraints is to prevent the inverse kinematics solution from getting into unnatural postures or body alignments that are undesirable, where such alignment issues becoming a problem when eliminating the orientation aspect from the full Jacobian. Fortunately, the addition of the further constraints does not hinder the performance of the original algorithm, as a set of joint limit constraints are already being maintained, which are dealt with in exactly the same manner. Therefore, through the use of some added constraints, a full-Jacobian problem can also be reformulated to make use of the more efficient half-Jacobian approach. This two-fold inference thereby demonstrates a novel perspective on the Jacobian-based inverse kinematics solution, which has been published in [Mere04a].

By further extending this work on the half-Jacobian, but not limited to this subclass of Jacobian-based solver, a novel approach of enriching character motions generated using per-frame inverse kinematics is introduced in the form of weighted inverse kinematics. This technique is able to individualised motions using the same control parameters. Therefore from the same base motion many different movements can be spawned, each exhibiting subtly different stylisation traits that can characterise the build/bulk of the character.

The evaluation of mapping the walking motion of actor C to actors A, B and D from this thesis’s dataset (described in Chapter 3) using weighted inverse kinematics demonstrated promising correlations between the reconfigured motions and real motions of the target actor. Indeed, the reconfigured motions were much more of a match to the real actor’s real movements than either the original base motion or the IK retargeted motion. However, the evaluation also demonstrated the limitations of the technique in its inability to completely capture the complex dynamical nature of real human motion. These limitations of the weighted inverse kinematics approach demonstrate a fundamental problem of using a purely kinematics-based modification technique.

However, one of the key benefits of using weighted inverse kinematics is that the resulting overhead is negligible and fits into the existing Jacobian-based solution. This means that when coupled with the half-Jacobian technique, a very quick and computationally inexpensive dynamic character individualisation is achieved, albeit not completely accurately.

The use of weighted inverse kinematics additionally provides the ability to dynamically simulate injuries. This is a natural consequence of using large variance weighting vectors, but further serves to illustrate the usefulness of the novel individualisation technique presented in this chapter. This is in contrast to previous approaches that simulate injury by removing complete DOFs [Popo99, Liu05], whereas the approach in this chapter just inhibits their comparative speed of motion. Furthermore, the previous approaches have been grounded in a dynamics paradigm, which are comparatively more computationally demanding than the weighted inverse kinematics approach.

Although weighted inverse kinematics has been demonstrated specifically on character gaits, the principle can apply to any form of posturing using IK. At the heart of this technique, the effect of applying weighting values within the IK chains is to directly affect the rate of change in joint angles. This is a mathematical adjustment on the IK solver itself therefore anything that makes use of the algorithm can utilise this work. Within the field of character animation, weighted IK has most potential where computational costs need to be kept minimal but enhanced realism is desirable.

The techniques presented in this chapter have considered how to make the inverse kinematics technique efficient and at the same time approximate limited dynamics, without actually modelling them. The results have shown promising advances in this area with real-time results. However, because dynamics are ignored in this process, it is possible to generate physically implausible motions. In the following chapters, dynamics will be used to modify motions, thus addressing this concern.

Chapter 5:
The Mathematical Dynamics of Articulated Structures

The culmination of Chapter 4 resulted in the application of weighted inverse kinematics technique to individualise a single piece of motion capture data to many different styles/builds of character. However, with the solution having no concept of forces, the results are not biomechanically driven and a suitable weighting vector was only estimated to best represent the target actor. This chapter looks to improve the individualisation process by directly taking account of an actor's biomechanical information through the encoding of a dynamics-based motion modification technique. Previous works have attempted similar objectives [Urta04, Hsu05, Liu05], however these techniques require an existing motion to extract stylisation parameters that are used to generate subsequent motions. The work in this chapter, and illustrated in Chapter 6, proposes a novel method of taking the biomechanics of a target actor and modifying the motion of a different actor to represent the motion of the target actor had they performed this motion – no motion of the target actor is sampled.

The dynamics-based system used to achieve biomechanically correct modifications is driven by a complex interaction of a collection of mathematical constructs that are described in this chapter. The results of applying the technique are given in Chapter 6. The description of the mathematical constructs starts in section 5.1 by providing a review of the rigid body mechanics used to model the dynamics of the environment and character. This review only covers standard rigid body material, however it is presented for completeness because it is subsequently referred to later in this chapter. Furthermore, the derivations presented in section 5.1 are more complex than the normal equations quoted in standard texts [Gold80, Kibb96] because unlike the standard texts, in this work, the centre of mass does not usually coincide with the origin of the body. The consequence of this is illustrated in the equations of section 5.1.

Once the basic mechanical principles have been established, section 5.2 reviews the previous work in the field of dynamics-based algorithms and their application in motion capture modification techniques. Thereafter, section 5.3 explores how the rigid body mechanics of section 5.1 can be used to facilitate the simulation of motion using an optimisation-based process – it is this that drives the dynamics-based modification process described in this chapter. Section 5.3 additionally details the remaining theoretical constructs of the dynamics based system that is used in this thesis.

The theoretical components of the dynamics-based system, which are discussed in section 5.3, do however lead to problems when attempting to map the constructs into a practical framework. With the aid of the mathematics reviewed in section 5.1 and section 5.3, section 5.4 addresses these issues. During these discussions, original solutions to the problems covered are highlighted and considered. Such issues include ill-resolutioned equations and dealing with discrete occurrences within the continuous domain, which is used in the dynamics-based motion modification technique described in this chapter.

Section 5.5 continues on from section 5.4 by providing a further review of practical considerations in implementing the dynamics-based motion modification technique. However, the focus of section 5.5 is above the equation level given in section 5.4 and considers more general practical issues such as programmatically representing the system of equations defined by the earlier sections of this chapter. Details of how the optimisation-based technique is initialised and solved are additionally given in section 5.5.

This chapter concludes with a summary in section 5.6 of the novel dynamics-based motion modification technique that is used in chapter 6 to reconfigure motions from one actor to another, using only biomechanical information from the target actor.

5.1 Rigid Body Mechanics

Mechanics is a branch of mathematics that considers the effect of forces acting on a body. It is generally recognised that the field of mechanics can be subdivided into three, more specific topics that are related to the type of body that is being considered: rigid body, deformable body and fluid mechanics. Within the field of mechanics, the term dynamics is used to describe the change in motion of a body as opposed to static bodies that are either at rest or moving with constant velocity. In general, this chapter will be talking about bodies in a dynamic sense.

Due to the underlying endoskeleton of a skinned computer character, the degree of deformation is highly constrained and therefore, from a mathematical point of view, it is arguably insignificant when compared to the forces acting on the body as a whole. Therefore it is appropriate to model the human form using rigid body mechanics applied to the hierarchical endoskeleton where mass distribution and dimensions of limbs are taken as constant. This has been the standard approach for researches modelling human dynamics, which is highlighted in section 5.2, and indeed is the method used in this thesis.

In this section, a review of the more important rigid body mechanical principles is presented. This facilitates the construction of a novel dynamics-based motion capture reconfiguration and additive modification process, which is presented in Chapter 6. Much of what is described in this section is a review and therefore more details can be found in any classical mechanics text [Gold80, Kibb96]. All formulations and equations will be assumed to be in 3-dimensions unless otherwise stated, using a right-handed coordinate system and post-multiplying operation strategy (i.e. the matrix stack should be read right to left). Similarly, unless otherwise stated, when discussing angular rotations, the convention of applying a rotation about the Y-axis first followed by the X-axis, followed by the Z-axis is used. This gives the matrix stack $R_zR_xR_y$, where R_i is the rotation matrix about the i^{th} axis.

5.1.1 Dynamics from Simple Particles to a Rigid Body

The simplest formation of a body in mechanics is a single particle that has mass but no size, which can be described using a scalar for mass and a 3-dimensional position vector (orientation does not need to be represented as the particle has no size). The result of applying directional forces to a single body becomes a trivial task of resolving the forces into a single resultant one. The behaviour of the particle is completely determined by Newton’s Second Law of Motion, given in Equation 5.1.

Force = **M**ass x **A**cceleration **(5.1)**

More complex mechanical bodies can be viewed as a composite of particles that are held together by intra-particle forces. In the case of rigid bodies, the intra-forces can be largely ignored because it is the motion of the body as a whole that is of interest. What becomes important in the case of many-particle bodies is the concept of a centre of mass (COM), which is given by Equation 5.2, where m_i and r_i are the mass and positions of the i^{th} particle respectively and M is the sum of all the particle masses. Using the COM, the motion of a rigid body can be determined as if treating the body as a particle based at the COM, whose mass equals the sum of its constituent particles. However, to determine the resultant forces acting on the body, the shape additionally, and hence the orientation, need to be considered.

$$\text{COM} = \frac{1}{M}\sum_i m_i r_i \qquad \textbf{(5.2)}$$

In addition to a positional vector, P, relating to the COM, an orientation matrix, R, is also associated with a rigid body, which allows the rigid body to be rebuilt around the COM, where P and R are with respect to a fixed frame of reference. This is achieved by realising that at a base orientation, there is a constant vector, d_i, from the COM to each of the particles as illustrated in Figure 5.1 and Equation 5.3. Similar equations can be derived for the velocity and acceleration of each of the particles by taking first and second differentials of Equation 5.3 respectively, with respect to time.

$$p_i = P + Rd_i \qquad \textbf{(5.3)}$$

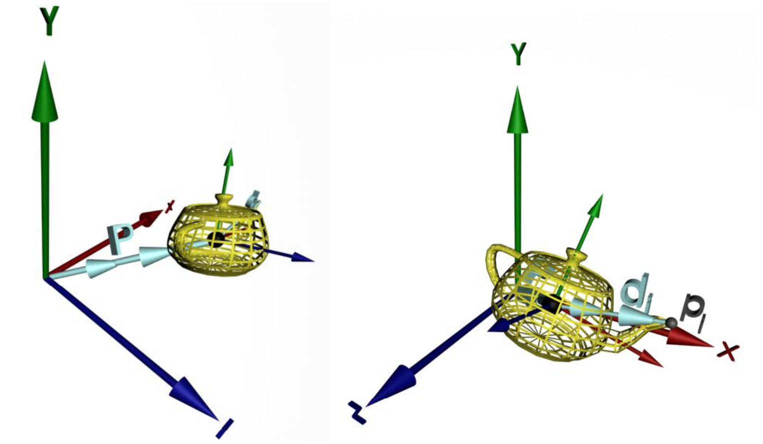


Figure 5.1: Determining particle locations based on the motion of a rigid bodies COM, where P is the offset from the origin to the COM of the teapot and d_i is a constant vector from the COM to a particle on the teapot, p_i

At this point it becomes convenient to introduce the concept of generalised coordinates. If the intra-particle constraint forces of a multi-particle body were expanded, the system would contain a large number of variables, including individual particle positions and constraint forces. By recognising any variables within the system that have a constant mathematical relationship and replacing these with a new single variable (a generalised coordinate) and a mathematical constraint function, the overall number of system variables can be reduced. The effect of generalising the system in such a manner is to produce a set of linearly-independent variables, which simplifies resulting equations of motions. This process has already been used in Equation 5.2 and Equation 5.3, where the motion of an arbitrary number of rigidly connected particles is represented using a single rotation and orientation vector for the COM and constant offsets to the individual particles. When rigid bodies are discussed, the generalised coordinates are usually the position of a root rigid body and orientations for each of the connected multi-particle bodies.

With the rigid bodies defined using generalised coordinates, the dynamics of the mechanical system can be expressed using the principle of conservation of energy, through the Lagrangian. The Lagrangian, *L*, is defined as the difference between the kinetic energy, *T*, and potential energy, *V*, of the system where the energy equations are expressed in terms of generalised coordinates, *q*, as illustrated in Equation 5.4.

$$L(q,\dot{q}) = T(q,\dot{q}) - V(q) \tag{5.4}$$

The equations of energy for a mechanical system expressed with the Lagrangian can be defined using a set of *m* Lagrange's equations, where *m* spans the set of generalised coordinates. Lagrange's equations of motion are given in Equation 5.5, where Q_i represents a generalised force acting from outside the system onto the *i*th generalised coordinate.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} - Q_i = \frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} - Q_i = 0 \tag{5.5}$$

The advantage of expressing the motion using Lagrange's equations is the complete system can be defined using a minimum set of equations, *m*, which correspond to the minimum number of variables used to represent the system. Through the rearrangement of equation 5.5, it can be further demonstrated that this is a reworking of Newton's second law of motion, Equation 5.1, in terms of generalised coordinates. Therefore, nothing has been lost through expressing the motion via Equation 5.5, however the ability to express the motion using a minimum set of equations through the kinetic and potential energy of the mechanical body has been achieved. This allows more complex systems, such as hierarchical rigid bodies, to be more readily represented.

To define the energy equations of Equation 5.5, the generalised coordinates are taken to be that of the position, *P*, and orientation, *R*, about a fixed point on the rigid body. From the basic definitions of potential and kinetic energy, Equations 5.6 and 5.7 can be derived respectively.

$$\begin{aligned} V &= g\sum_i m_i\, p_{i[y]} && \text{(where the Y-axis is the up-axis and } [y] \\ &= g\sum_i m_i\,(P + Rd_i)_{[y]} && \text{projects the y-coordinate of the vector)} \\ &= gMP_{[y]} + gM(RC)_{[y]} && \textbf{(5.6)} \end{aligned}$$

(where C is the centre of mass related to the body frame and *g* is the universal constant of gravity for Earth = 9.80665ms^{-2})

$$\begin{aligned} T &= \tfrac{1}{2}\sum_i m_i \left\| \dot{p}_i \right\|^2 \\ &= \tfrac{1}{2}\sum_i m_i \left\| \dot{P} + \dot{R} d_i \right\|^2 && \text{(by Equation 5.3)} \\ &= \tfrac{1}{2}\sum_i m_i \left(\left\| \dot{P} \right\|^2 + 2\dot{P}^T \dot{R} d_i + \left\| \dot{R} d_i \right\|^2 \right) \\ &= \tfrac{1}{2} M \left\| \dot{P} \right\|^2 + M\dot{P}^T \dot{R} C + \tfrac{1}{2} w^T I w && \textbf{(5.7)} \end{aligned}$$

(where ω is the angular velocity of the body and *I* is the inertia tensor of the rigid body related to the body's local frame)

In the derivations of potential and kinetic energy, it is important to note that an arbitrary point on the body has been taken to represent the fixed local frame and therefore the centre of mass vector, *C*, is a vector from this fixed point. If the centre of mass were taken as the fixed point on the body, *C* would be identically zero, therefore Equation 5.6 and Equation 5.7 would simplify with the second and middle terms vanishing respectively. The equations that result from the *C* vector vanishing give rise to the more familiar equations published in classical mechanics texts however these terms have been left in because when dealing with connected rigid bodies it is more convenient to take the local body frame as the connection point, which is rarely the COM of the body.

The inertia tensor of Equation 5.7 is a 3x3 matrix that consists of moments of inertia with reference to the body frame and its axis of rotation. Its form is given mathematically in Equation 5.8.

$$I_{jk} = \sum_i m_i \left(d_i^2 d_{jk} - d_{i[j]} d_{i[k]} \right) \quad \text{(where } d_{jk} \text{ is the Kronecker delta)}$$

$$I = \sum_i m_i \begin{bmatrix} d_{i[y]}^2 + d_{i[z]}^2 & -d_{i[x]} d_{i[y]} & -d_{i[x]} d_{i[z]} \\ -d_{i[x]} d_{i[y]} & d_{i[x]}^2 + d_{i[z]}^2 & -d_{i[y]} d_{i[z]} \\ -d_{i[x]} d_{i[z]} & -d_{i[y]} d_{i[z]} & d_{i[x]}^2 + d_{i[y]}^2 \end{bmatrix} \qquad \textbf{(5.8)}$$

The orientation of the rigid body about the fixed body frame is represented using a 3-dimensional vector that specifies successive angular rotations to be applied to the body (recall that a right-to-left $R_zR_xR_y$ rotation order is being used). The components of the angular velocity, *w*, need to be formulated with respect to the three principal axes of the body. A brief derivation of the angular velocity is provided with the assistance of Figure 5.2 and Equation 5.9, where the amount of rotation about the X-, Y- and Z-axis is given by y, q and f respectively.

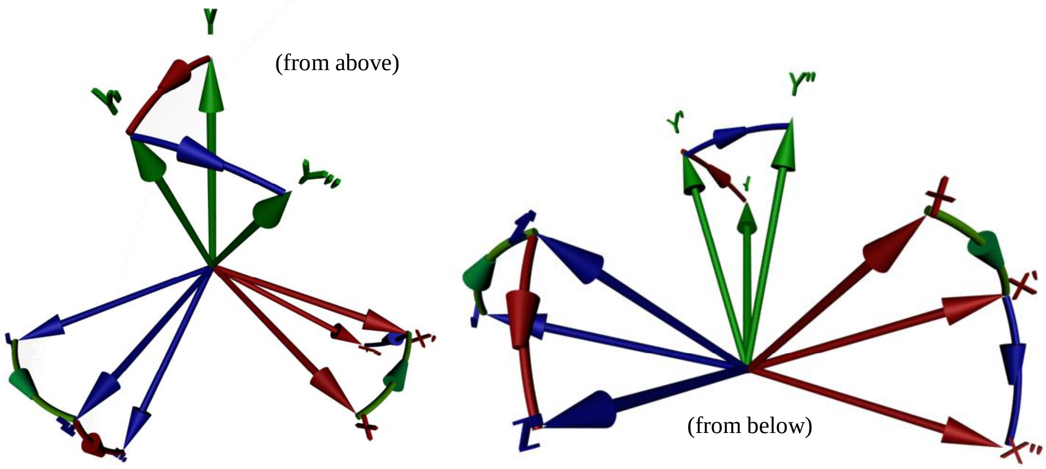


Figure 5.2: Deriving the angular velocity from the principal axes of rotation, where the coloured directional arcs represent the rotation about the axis with the corresponding colour

From Figure 5.2, a rotation through an angle of q is made about the *y*-axis (green axis), which brings the local axis into positions *x', y, z'*. Secondly, a local rotation about the *x*-axis (red axis) of y radians is applied which brings the local axes into positions *x', y', z''*. The final local rotation performed is about the *z*-axis (blue axis) of an angle of f, which subsequently brings the local axes into their required position *x'', y'', z''*.

If a small change in y is introduced but the other angles kept constant, from Figure 5.2, the rotation would be about the *x'*-axis and not the *x*-axis. This is due to the introduction of any constant rotation about the *y*-axis first. Similarly, for the independent angles, q, f, and keeping the other two constant, a rotation about the *y*-axis and *z''*-axis respectively would be achieved. Therefore, when all three angles are changing, the angular velocity is determined by summing the contributions of each angle in global space, which is derived in Equation 5.9.

$$w = w_f + w_y + w_q \qquad \textbf{(5.9)}$$

$$w_q = \begin{bmatrix} 0 & \frac{dq}{dt} & 0 & 0 \end{bmatrix}^T$$

$$w_y = R_q \begin{bmatrix} \frac{dy}{dt} & 0 & 0 & 0 \end{bmatrix}^T$$

$$w_f = R_y R_q \begin{bmatrix} 0 & 0 & \frac{df}{dt} & 0 \end{bmatrix}^T$$

The final component of Lagrange's equations of motion, as given in Equation 5.5, is the concept of generalised forces. These forces are the result of expressing external forces applied to the system in terms of generalised coordinates. This is achieved by summing the amount of work each force does in moving a particle with respect to the i^{th} generalised coordinate as Equation 5.10 illustrates.

$$Q_i = \sum_j F_j \frac{\partial p_j}{\partial q_i} \tag{5.10}$$

5.1.2 Chained Rigid Body Dynamics for Computer Characters

Broadly speaking, each jointed limb in the human body is represented as a mathematical rigid body, where it is assumed that the layers of skin and muscle over a limb have a negligible affect on the shape of the body. This simple definition is useful in minimising the representation of the endoskeleton because any bones that move together can be abstracted to a single one, or at least has very little inter-bone movement. In this research, the abstraction is taken slightly further by representing each hand as a single rigid body as opposed to going down to the level of fingers. The same reduction to the toes of the character's feet is performed. In total, this thesis represents a normal human character with 16 rigid bodies as illustrated in Figure 5.3, however the equations that are subsequently presented are general enough to encompass an arbitrary number of connected rigid bodies.

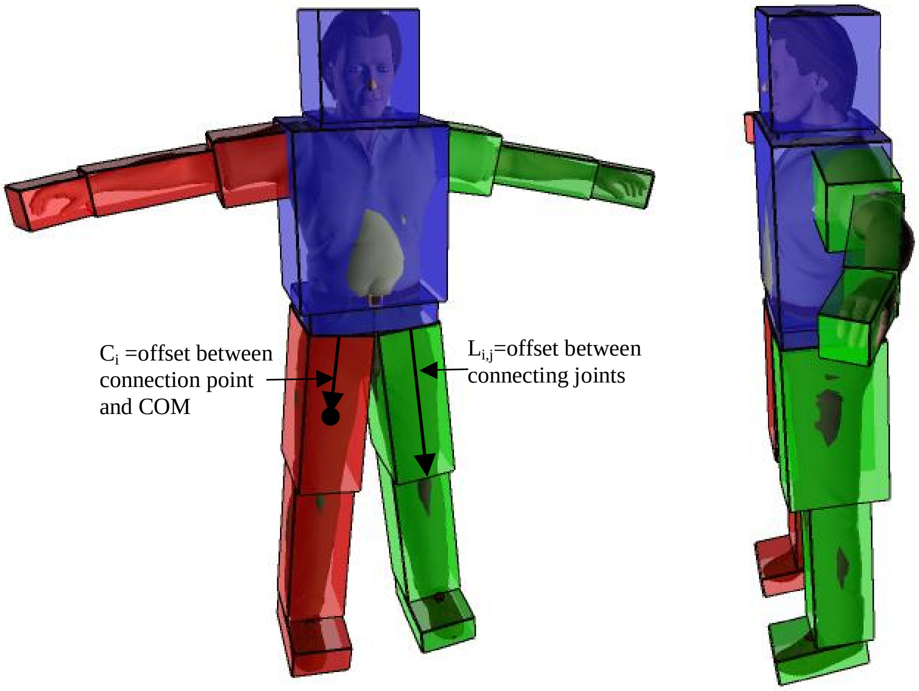

C_i =offset between connection point and COM
$L_{i,j}$=offset between connecting joints

Figure 5.3: A mapping between connected rigid bodies and a human character

Figure 5.3 additionally illustrates a variable naming convection that is adopted during the continued discussion in applying dynamics to connected rigid bodies. It is important to notice that when a particular variable is indexed, it is actually the link index within the chain that is being referred to as opposed to individual particles as in the preceding rigid body descriptions. The link indices are taken to start at 0 for the root and increase by one at each connected link within the chain. It is appreciated that a human body is not a simple chain of connected links but consists of branches. However, for the purposes of the extended equations, without loss of generality, a simple, non-branching chained system can be built up working back to the root node as each limb consists of only a single parent.

Each rigid body, will have a maximum of three rotation values about the X-, Y- and Z-axis, given by y, q and f respectively, depending on the part of the body that is being represented. For example, the upper leg would have three axis of rotation to replicate for the full movement allowed by the hip joint, whereas the lower leg would only have one axis of rotation that relates to the knee joint in the leg. It is more important to identify such redundancies in this case compared to that of modelling using inverse kinematics because the mathematical penalty is much higher.

In addition to the generalised coordinates used to define orientation, there are also three positional scalars that position the origin of the root body. In terms of the variable labelling that has been adopted in this chapter, the position of the *i*th link's origin, P_i, is given by Equation 5.11, where P_0 is the generalised coordinates for the root location, O_k is the local orientation/rotation matrix for the *k*th body and $L_{k,k+1}$ is the offset vector from the origin of the *k*th body to the origin of the *(k+1)*th body.

$$P_i = P_0 + \sum_{k=0}^{i-1}\left(\prod_{j=0}^{k} O_j\right)L_{k,k+1} \qquad \textbf{(5.11)}$$

where $O_j = O_{j,f} O_{j,y} O_{j,q}$

The position of each rigid body is an important component of Lagrange's equations of motion, as too is the orientation of the body with respect to the global frame, which is defined in Equation 5.12.

$$R_i = \prod\nolimits_{k=0}^{i} O_k \qquad \textbf{(5.12)}$$

Using Equations 5.11 and 5.12, the equations for potential and kinetic energy for an independent body, Equations 5.6 and 5.7, can be rewritten for the *i*th body within a chained hierarchy with Equations 5.13 and Equation 5.14.

$$V_i = gM_i\left(P_i + R_iC_i\right)_{[y]} \qquad \textbf{(5.13)}$$

$$T_i = \tfrac{1}{2}M_i\left\|\dot{P}_i\right\|^2 + M_i\dot{P}_i^{\,T}\dot{R}_iC_i + \tfrac{1}{2}w_i^{\,T}I_iw_i \qquad \textbf{(5.14)}$$

As illustrated in Figure 5.2 and Equation 5.9, the angular velocity, w_i, needs to be defined with respect to the principal axes of the rigid body. This requires that the angular velocities need to not only take into account the local angular velocities but also the angular velocities of the link to which it is attached. This is illustrated in the expanded solution for angular velocity given by Equation 5.15.

$$w_i = w_{i,i} = \sum_{k=0}^{i}(w_{i,k,f} + w_{i,k,y} + w_{i,k,q}) \qquad \textbf{(5.15)}$$

$$w_{i,k} = \begin{cases} w_{i,k,f} + w_{i,k,y} + w_{i,k,q} & if\ k = 0 \\ w_{i,k-1} + w_{i,k,f} + w_{i,k,y} + w_{i,k,q} & else \end{cases}$$

$$w_{i,k,f} = (O_{k,y}\, O_{k,q} \prod_{j=k+1}^{i} O_j)\begin{bmatrix} 0 & 0 & \frac{df}{dt} & 0 \end{bmatrix}^T$$

$$w_{i,k,y} = (O_{k,q} \prod_{j=k+1}^{i} O_j)\begin{bmatrix} \frac{dy}{dt} & 0 & 0 & 0 \end{bmatrix}^T$$

$$w_{i,k,q} = (\prod_{j=k+1}^{i} O_j)\begin{bmatrix} 0 & \frac{dq}{dt} & 0 & 0 \end{bmatrix}^T$$

The final Lagrangian of the system, as defined in Equation 5.4, is therefore obtained from the summation over all the rigid bodies of their kinetic and potential energies as defined by Equation 5.13 and 5.14, using Equation 5.11 through to Equation 5.15.

The final step in generating Lagrange's equations of motion is to take the partial derivates of the Lagrangian with respect to the generalised coordinates and time. The differentiation process is simply a case of bookwork therefore a complete derivation of Equation 5.5 is not presented. However, for future illustrative and discussion purposes, the expansion of Lagrange's equation of motion for a two link, rigid body system, differentiated with respect to the *y*-positional generalised coordinate is presented in Equation 5.16. The first body in Equation 5.16 has 6 degrees of freedom, 3 positional and 3 that define orientation, whereas the second link only has 3 orientation degrees of freedom.

$$L = T - V \qquad \text{(from 5.4)}$$

$$V = V_0 + V_1 = gM_0(P_0 + R_0C_0)_{[y]} + gM_1(P_1 + R_1C_1)_{[y]} \qquad \text{(from 5.13)}$$

$$T = T_0 + T_1 = \tfrac{1}{2}M_0\left\|\dot{P}_0\right\|^2 + M_0\dot{P}_0^{\,T}\dot{R}_0C_0 + \tfrac{1}{2}w_0^{\,T}I_0w_0 + \tfrac{1}{2}M_1\left\|\dot{P}_1\right\|^2 + M_1\dot{P}_1^{\,T}\dot{R}_1C_1 + \tfrac{1}{2}w_1^{\,T}I_1w_1 \qquad \text{(from 5.14)}$$

$$R_0 = O_0 = O_{0,f}\,O_{0,y}\,O_{0,q} \qquad \text{(from 5.12)}$$

$$R_1 = R_0O_1 = O_{0,f}\,O_{0,y}\,O_{0,q}\,O_{1,f}\,O_{1,y}\,O_{1,q}$$

$$P_0 = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$$

$$P_1 = P_0 + O_0 L_{0,1} \qquad \text{(from Equation 5.11)}$$

$$\frac{\partial V}{\partial y} = g(M_0 + M_1)$$

$$\frac{\partial T}{\partial y} = 0$$

$$\begin{aligned} \frac{d}{dt}\frac{\partial T}{\partial \dot{y}} = M_0 \ddot{y} + M_0 \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \ddot{R}_0 C_0 + \\ M_1 (\ddot{y} + (\ddot{O}_0 L_{0,1})_{[y]}) + M_1 \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \ddot{R}_1 C_1 \end{aligned} \qquad \textbf{(5.16)}$$

The Lagrange equation components of 5.16 have not been fully expanded down to the generalised coordinates. However, it can be seen from the expansion of the second differentials of the orientation matrices, R_i and O_i, will result in large and complex equations. It is therefore desirable at this stage to highlight some of the simplifying assumptions that this thesis makes to the rigid bodies in order to achieve this aim.

The first key point to make reiterates an issue concerning the removal of any redundancy within the rigid body chain. This is accomplished by eliminating any degrees of freedom from a connected rigid body that would result in impossible movement. To accomplish this, the limb DOF limits that were used during the inverse kinematics humanoid modification techniques (see section 2.14 are applied to each of the corresponding rigid bodies. This has the immediate effect of simplifying the global orientation matrices, R_i, which increases with complexity with every additional rigid body in the chain.

Another area that has been identified to reduce the complexity of Equation 5.16 lies in the construction of the rigid body offsets and centre of masses. Each of the orientation matrices are multiplied by a column vector that gives either the vector between connected rigid bodies or the body's COM. Therefore, if all bones are aligned along the y-axis, in the majority of cases, it follows that the connection offset becomes a single scalar along this axis which results in the x- and z-components of the vector vanishing. This means that only the calculation of the second column of the orientation matrix is required. In non-branching chains, this is the technique that has been adopted. However, in the case of rigid bodies with multiple children, it is clearly not possible to achieve this therefore the child bodies are aligned with as many primary axes as the shape of the body allows, i.e. the offset vectors are determined to have as many zeros elements as possible, which simplifies the expansion of Equation 5.16.

It would be desirable to coincide the local origin of a body with its COM thereby reducing this vector to zero. However, the local origin is already defined to correspond with the point of attachment in order to give a minimal representation of the joint orientation. Nevertheless, in the case of the root node, there is flexibility to aligning the local origin with the COM. In the remaining cases of attached rigid bodies, by aligning the bone along the y-axis for positional offset purposes, it follows that this axis should run through the centre of the rigid body. In doing this, and with the additional assumption that object is symmetrical along this axis, the COM vector can similarly be reduced to a single scalar along the y-axis and thus affords the same savings that are achieved by reducing the positional offset to a single scalar.

The assumption about the rigid body being symmetrical about the bone-length axis is acceptable because it is the endoskeleton that is largely of interested and the skin and tissue is built up around that, roughly centred round the skeleton. This view is supported by other anthropometry and mass distribution studies [Heym05, NBDL88, Vann95]. A further benefit of modelling the rigid body with these guidelines comes through the reduction of the inertia tensor of the body. With the body being symmetrical about an axis, the 3x3 inertia matrix reduces to a diagonal matrix, which becomes important when the Lagrangian is differentiated with respect to the orientation generalised coordinates.

5.2 Previous Usage of Dynamics to Modify Motions

Using dynamical descriptions, such as those reviewed in section 5.1, there are two different approaches to produce a motion. The first approach steps through time and solves the new state of the system based on the set of values that described the system at the previous step. The second approach is to solve the complete motion as a whole, which effectively becomes a problem of finding a suitable spatial path for the generalised coordinates, such that the dynamics are always satisfied. The former, per-step approach is the simplest form of describing a motion and is reviewed in section 5.2.1. However, it can be limited in comparison to the second technique, when considering time-based constraints, which is discussed in section 5.2.2.

5.2.1 Frame by Frame – Forward Dynamics

Per frame techniques use time differentials (gradient-based) and mathematical relationships between variables to determine a system state for the following time step, which is a predefined interval apart, to produce a physical animation [Barz88, Barz96, Hahn88]. These gradient-based techniques are commonly described as a forward (or time) integration process, using Euler integration [Boas83] or Runge-Kutta steps [Horn75] for example.

In order to start the iterative dynamics simulation, an initial starting state at $t=t_0$, has to be specified beforehand. However, at each time step, it is possible to introduce external forces, which influence the resulting motion. For example, it is possible to detect any collisions and hence react to them by introducing an appropriate force [Bara89, Bara91, Moor88]. Herein lies one of the major advantages of per-step techniques in that any “one-off” interactions can be determined as a secondary process that operates above the dynamics of the system and reacts accordingly before the next time step.

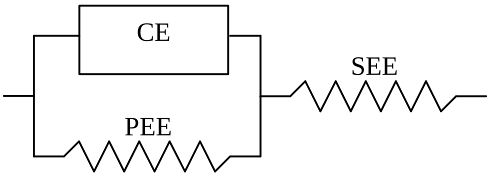
The simulation of human characters has successfully been demonstrated using a per-frame forward dynamics approach, most predominately by Hodgins [Hodg95, Hodg96, Hodg97]. Underlying much of this work is the concept of developing mathematical control structures that provides the interactions and muscle forces for specific motions. Where muscle forces and torques are defined, using proportional-derivative servos, whose form is given in Equation 5.17, where k_1 and k_2 are constants (called proportional and derivative gains respectively), and q_d and q are the desired and current joint angles respectively.

$$t = k_1(q_d - q) + k_2(\dot{q}_d - \dot{q}) \qquad \textbf{(5.17)}$$

The application of proportional-derivative servos has been successfully used to simulate human walking & running [Rail91, Hodg96], cycling & vaulting [Hodg95, Brog98] and diving [Woot95] as well as aiding the mapping of optical motion capture marker data to an articulated structure [Zord03]. The control structure and proportional-derivative combination has further been applied to modifying motion data with an aim to map it onto characters of different sizes [Hodg97, Poll00]. In this work, the basic control structure of a running character is uniformly scaled to the body size of the new characters and the proportional-derivative gain constants are “tuned” to match the build of the character. No additional external dynamics are included and the technique simply retargets the motion from one character movement to another similar movement without any interruptions to the motion.

In later work, which follows similar lines, Zordan & Hodgins [Zord99, Zord00, Zord02] propose a technique that includes the ability to track motion capture data and incorporate external dynamics. The concept behind this work is not so much a retargetting technique but more of environmental adaptation process wherein the dynamics generate the motion, guided by the motion capture data joint angles. Thus, environmental impacts or interactions are included while adding in the subtle details of real motion, something that pure simulations fail to capture. It is therefore possible to deviate completely from the original motion and hence an environmental adaptation process rather than a retargetting one. The techniques presented by Zordan & Hodgins for applying inter-body dynamics to existing motion make use of proportional-derivative servos for muscle forces, whereas Komura & Shinagawa [Komu97, Komu00, Komu01b] approaches the same problem using a Hill’s muscle model [Wint90].

While proportional-derivative servos give a good approximation to joint torques, the technique is somewhat basic in its concept of biomechanical muscle properties, whereas Hill’s model encapsulates musculotendon dynamics. Hill’s model, which is illustrated in Figure 5.4, simplifies a real muscle into 3 high-level entities: the contractile part (CE), which represents the muscle fibres, the Series Elastic Element (SEE) that represents the muscle tendon, and the Parallel Elastic Element (PEE), which simplifies the connective tissue around the muscle fibres and fibre bundles. The musculotendon force, F^{SE}, is limited by $F_{min}(a=0)$ £ F^{SE} £ $F_{max}(a=1)$, where natural fatigue and recovery can also be simulated by reducing and increasing the exertable force of the contractile part of the muscle. This is comparable to the resulting effects of real muscle as it tires.



The exertable force of the 3 components is given by:

$$F_{CE} = f(l_{CE}, v_{CE}, a)$$
$$F_{SEE} = f(l_{SEE})$$
$$F_{PEE} = f(l_{PEE})$$

and

$$F_{SEE} = F_{CE} + F_{SEE}$$

where l_x is the length of element x, v_{CE} is the contractile velocity and a is muscle activation level.

Figure 5.4: Hill's Muscle Model

Komura & Shinagawa use Hill's muscle model to attached muscle effects to existing motion data per-frame by solving a minimisation problem that attempts to mimic the original motion, but ensuring the muscle parameters can match the joint trajectories. Additionally, a zero momentum point construct is used to maintain the character's balance, which is the standard technique used in incorporating such behaviour. Compared to the work of Zordan & Hodgins, the technique demonstrated by Komura & Shinagawa provides a more intuitive approach for adjusting existing motion based on muscular builds, as the muscle forces are modelled in a more flexible and controllable manner; this is in comparison to the more abstract proportional-derivative servos. However this does come at the cost of increased computational demands.

Using the Hill's muscle mode, Komura *et al.* [Komu04] present a similar technique to Zordan & Hodgins [Zord02] a technique that allows motion data to be adapted in reaction to impulse forces. Whereas Zordan & Hodgins utilised a pure dynamics solution, Komura *et al.* focus on maintaining a good postural centre of mass therefore producing a stable gait. When an impulse is introduced into their gait motions, an analytical formulation determines how to reposition the centre of mass to maintain balance, with respect to the body's angular momentum. This is similar to the work presented by Oshita & Makinouchi [Oshi01] for adjusting static postures in response to impulse forces and that of Skin *et al.* [Shin03] for plausibly adjusting existing motions to ensure a good centre of mass is maintained, but without consideration to extra external forces. In order to determine the required reaction forces, Komura *et al.*'s control process relies on low angular velocities, which are present in natural walking, but less applicable to faster moving motions. Furthermore, because the inbuilt dynamics focus on purely maintaining balance, the technique is limited to these types of interactions. So unlike the work demonstrated by Zordan & Hodgins, the motion stays more faithful to the original, but the cost of this is to reduce generality.

A more recent technique by Zordan *et al.* [Zord05] demonstrates a technique that allows motion capture data to be interrupted by a physical force, such as an impact, or environmental interactions and react appropriately. This is achieved by fully simulating the resulting behaviour for an interval after the interaction using forward dynamics integration. After the simulated interval, the character's posture is compared to a large repository of motion data to find a motion that best suits the continued trajectories, thus demonstrating a hybrid solution to dynamic simulations. A consequence of the technique presented by Zordan *et al.* is the need for a large amount of motion capture data from which a suitable continuing motion can be found once the simulated interval expires. If the motion data to which the simulation is to transition towards is not suitably close then it is highly possible that visual artefacts are introduced, not least in breaking the physics that the dynamic simulation preserves at the point of impact.

In each of the forward, per-frame dynamic techniques that have been considered, two conclusions can be drawn. The first is that for each different kind of motion a control structure needs to be derived and, subsequently, only motions belonging to that category can be simulated. This problem is partly addressed by Faloutsos *et al.* [Falo01] who propose a framework for composing physics-based controllers, which allow different control structures that simulate different motions to be invoked under the appropriate conditions, thus simulating a wider scope of motions. The second, and potentially more serious issue lies in the use of proportional-derivative servos. Equation 5.17 consists of two gain coefficients that need to be determined for each joint angle, however this is not a straightforward process and determining the correct values tends to be more of an art than a science. Furthermore, care needs to be taken when determining the overall control structure (including the proportional-derivative servos) in order to reduce any instability effects that might result in the simulated cyclic motion.

In addition to the primary motion of the human characters, dynamics have additionally been included to animate secondary motion. For example, water splashing as the character steps into a puddle and a natural trampoline response as an aerobics character bounces about on it [Hodg96, OBri00]. The addition of secondary dynamics provides a visually richer scene as opposed to just focusing on the characters in the scene.

Per-frame forward dynamic integration techniques are comparatively quick since the differentials, or control structures, can usually be analytically defined off-line and hence quickly evaluated at runtime. Therefore, when it comes to determining the update vector for the next step, the differentials are readily available. Even in the case where the differentials are non-trivial and difficult to determine, which can occur in complex simulations, it is only an instantaneous point in time that is of concern. This reduces the complexity of the whole motion down into smaller, more manageable parts and subsequently, per-frame, the simulation is easier and quicker to compute.

In part, it is a consequence of the dynamics problem being simplified in a pre-frame manner, that allows the techniques to be employed in many real-time applications such as computer game architectures [Kokk04] and physics engines such as Havok [Havo05], and the Open Dynamics Engine [ODE05]. Indeed, in many applications it makes sense to use the forward integration process where all that is needed is to simulate the natural, consequential motions of objects within the scene. For example, if a character pushes over a table with objects on, the objects would be expected to fall off and hit the ground in a dynamically natural way – how the objects actually fall and their position and orientation as it lands is of no concern so long as it looks natural.

However, in many situations it is desirable to have control over the natural outcome of a dynamics simulation. For example, if a dice were thrown, it is physically plausible that when it comes to rest any one of the sides will be upward facing. But instead of this indeterminism, what is actually required is that the dice comes to rest with a particular face showing. This poses a problem for per-frame techniques, as does any similar problem of introducing a time-based target constraint. To be able to define a time-based constraint using forward dynamics, a guess at what the initial values should be in order to achieve the goal in a trial-and-error manner would be needed. A technique similar to this can be seen in the commercial character animation package Endorphin [Endo05]. However, it is still difficult to defining the initial values because a slight change can result in very different dynamical behaviour. A review of the work in this area is presented in the following subsection.

5.2.1.1 Forward Dynamics Time-based Constraints

Several researchers have incorporated time-based constraints into forward dynamics using a basic generate-and-trial procedure, such as Chenney & Forsth [Chen00] who use a Markov chain Monte Carlo algorithm to sample multiple generated animations and choose a plausible solution. However such approaches by themselves are very costly and inefficient. Genetic algorithms have also been posed as a solution to reducing the costs that are introduced by the generate-and-trial method [Ngo93, Park97, Tang95]. As opposed to accurately modelling the environmental dynamics, genetic algorithms instead simulate this behaviour using a combination of control methods and genetic parameters that are evolved at each generation. Genetic algorithms have two key advantages over their gradient-based counterparts. First, they do not suffer from gradient discontinuity problems, in cases such as collisions. Second, they are very quick at spawning a pool of possible motions at each step due to their lack of any mathematical representation of the system dynamics. However, genetic algorithms require carefully considered test criteria to determine which motions are allowed to proceed to the next generation, which can in complex cases be either very difficult to specify or be equally as complex as gradient-based techniques. Furthermore, there is no all-purpose physics-based genetic algorithm that will generically work in all cases. Therefore each application of such techniques requires a degree of oversight in its function to achieve the desired goal. Despite their speed in generating possible next step motions, the time it takes to nurture a solution to reach a desired goal can be quite considerable compared to the speed that they can be evaluated at.

An alternative approach presented by Grzeszczuk *et al.* [Grze98] utilises a neural network to represent the mathematical gradient functions at each step. The neural network is taught off-line through exposure to mathematical forward dynamics simulations, which when put into practice takes as input the present state of the world objects and produces their next time state. Effectively, Grzeszczuk *et al.* replaces the per-frame update process with a neural network as opposed to mathematical gradient-based formulations. Once the network has been taught, the process of applying it to objects is a much quicker procedure, similar to that of genetic algorithms but without the need of spawning many different possible solutions.

Grzeszczuk *et al.* extend their representation of the forward dynamics problem by including a back-propagation aspect to their algorithm, which allows them to incorporate time-based constraints. This starts with a forward dynamics solution, using the neural network that does not consider the constraints. Subsequently, the constraints at a given time frame are driven towards zero using an optimisation process that adjusts the original system configuration at that point in time. The adjustment of any parameters in the middle of the motion will have a consequential effect on the motion thereafter, which can be recalculated using forward dynamics. The motion before this point in time is adjusted using the back-propagation algorithm that allows the neural network to work in reverse; i.e. given a set of output parameters, it can determine a suitable input configuration.

Popovic *et al.* [Popo00a] borrow a similar concept to Grzeszczuk *et al.* in their approach to providing interactive user control over forward dynamics simulations, but in a more traditional gradient-based approach. Starting from a suitably close forward dynamics derived solution, time-based constraints are added into the solution through a minimisation process. Based on the constraints, the original motion is adjusted backwards in time such that the change in configuration state is minimised.

The problem however with both the approaches presented by Grzeszczuk *et al.* and Popovic *et al.* are that they only work reasonably well when the original motion is close to the intended one. Any large-scale changes to the original motion will cause the solution to destabilise due to the inversion processes going outside the locally linear-approximations of the gradients required in Popovic *et al.*'s optimisation process or Grzeszczuk *et al.*'s backward-propagation algorithm.

Another technique that allows time-based constraints to be added to per-frame forward dynamics is by Brotman & Netravali [Brot88]. In this work, object trajectories are simulated between user-specified key frames in a kind of dynamics interpolation technique (as opposed to a straightforward linear interpolation). Popovic *et al.* [Popo03] explore a similar technique where as opposed to simply using key frames, the user completely sketches out a suitable path that an object should take, which may not be true to dynamics. The sketch is subsequently used as a guide to a per-frame dynamics solver that attempts to minimise the distance between the actual object position and the sketched one while adhering to the mechanics of the system.

The inclusion of complex time-based constraints (as opposed to trial-and-error approaches) have only been applied to simple animations and not to complex articulated structures such as humanoid characters. This is because the stability of these algorithms is questionable with simple objects and the inclusion of articulated structures would only prove still harder to control An alternative technique for handling with time-based constraints is to deal with the whole motion at once, as opposed to on a per frame basis, which is explored in the next section.

5.2.2 Dynamics Simulations Taking the Whole Motion at Once

The alternative approach to per frame dynamic simulations is to determine the problem for a complete period of time, thereby solving a spatial problem through time. However, this can result in an infinite number of possible solutions from which to choose. Therefore a way of guiding the technique towards a desirable motion is needed, which is generally achieved through the use of minimisation criteria. For example, the criteria may be to pick a result that expends the least amount of energy or maybe that moves the slowest. This results in a spatial-time minimisation problem of an objective function, *R(S)*, subject to a set of dynamically derived motion equation constraints, *C(S)*, or more specifically a nonlinear constrained optimisation problem, as illustrated in Equation 5.18, where *S* is the set of system variables. The first occurrence of this style of formulating dynamics in computer animation comes from Witkin & Kass [Witk88]. The work has left an affect in this field since many approaches to solving motions as a whole have some relation to this work, the one presented in this thesis included.

$$\begin{aligned}&\textit{minimise } R(S)\\&\textit{subject to } C(S)=0\end{aligned}\qquad\textbf{(5.18)}$$

The rest of this section reviews previous work using this approach. The next section details the mathematical process of solving this formulation.

Witkin & Kass use the optimisation formulation of Equation 5.18 to simulate a dynamically realistic jumping Luxo lamp, whose springs give it the forces necessary to jump. The constraints are defined using discrete functions and thus finite differencing equations that model Newton's second law of motion, Equation 5.1, and Lagrangian mechanics, Equation 5.5, which completely define the range space of motions. A minimal energy optimisation function is employed to guide the solution to a motion that attempts to reduce the amount of work the lamp's springs need to do during the movement. In order to guide the simulated motion, time-based constraints are included, which state an object has to be at a certain position or orientation at a specific point in time. Using all of this data, the technique solves Equation 5.18 over a complete time range to produce motion control parameters.

This procedure highlights the comparative ease with time-based constraints are included in contrast to per frame based techniques. By specifying a constraint and combining it with the physical constraints, a naturally smooth motion can be generated to meet the constraint targets. This behaviour is one of the major advantages of solving the motion as a whole segment. Since it is possible to look ahead, it is also possible to determine what and when forces need to be adding in order to reach a specific posture. Unfortunately, by virtue of the fact a complete motion is being solved at once, the simulation is now intrinsically an off-line process, unlike some of the per-frame approaches that can be made to run in real-time.

Steps have been taken to combat this inherent deficiency of the spacetime constraints approach including approximating the non-linear physical constraints as linear ones [Fang03] and by introducing spacetime windows [Cohe92]. The spacetime window approach allows the whole motion to be split up into smaller time interval pieces with a view to making it more interactive for users to create motions. The visual result of performing such segmentation is that the motion appears more erratic, which in some cases can actually improve the movement. The example Cohen gives while presenting spacetime windows is that of a cat and mouse chase, where as opposed to the cat directly going to a point in time it knows the mouse will be, it chases the mouse around in a more naturally, unpredictable motion. The work presented by Cohen additionally moves away from the use of discrete control variables and implements uniform cubic B-spline curves. This allows continuous first and second differentials to be obtained as opposed to the finite differencing approach of Witkin & Kass.

A further claim that is made of the windowing technique is its ability to refine the solution for a subset of degree of freedoms or a region of time. This is not followed up by Cohen and because of the highly dependent nature of the control variables, it is unlikely that one parameter could change without having an effect on other variables. Subsequently, the encoded physical constraints of the system would have to be broken. A similar argument applies to that of the time period solution recalculation, however perhaps less so because new constraints can be included to ensure C^2 continuity of all the control parameters going in and out of the window. Although, if anything significant happened in that time window, it would be unlikely that the end values with the continuing motion would match up without breaking the physics of the system. Consequently, the rest of the motion from that point onwards would still need to be recalculated using the new values.

Liu *et al.* [Liu94b] also concentrate on a certain temporal period to add further refinement. In their work, the curve representing the control parameters is refined using hierarchical wavelets to be able to exhibit more detail, unlike the uniform version of Cohen's. Subsequently, more detail can be added to time intervals that require it, while maintaining course granularity for other regions, thus striking a balance between the number of control parameters and the resulting complexity of the system. This however mixes the control parameter representations in that sometimes uniform B-splines are used while at other times wavelets are used. The extra complexity this introduces is offset against the argument that it is possible to go back up the wavelet detail levels to less refined curves and use these to help reduce the complexity if, say, only a broad optimisation of a temporal region is required while the lower-level detail is kept. However, due to the tight coupling of more complex articulations, it is not overly practical to refine motions at higher levels than at the lowest granularity and still expect both continuity and realistic looking results.

The techniques presented by Witkin & Kass, Cohen and Liu *at al* all model the constraints using rigid body mechanics for relatively simple structures, where more complex structures are the focus of the work presented by Popovic & Witkin [Popo99]. Here, a slightly different approach is taken in that they use an existing motion to guide the solution. Furthermore, Popovic & Witkin abstract the complex articulated humanoid structure to a more simplified version, where for example the complete arm is represented as a single entity. The dynamics that is included in this approach is also less demanding since they only consider Newton's second law of motion with respect to point masses that represent each abstract limb entity. Consequently, similar to the work of Liu & Cohen [Liu94a], they remove the complexity of modelling angular velocities from the constraints, which is in contrast to the full rigid body dynamics of the earlier spacetime techniques.

The objective function component of the spacetime technique is extended by Popovic & Witkin to become a dual function, which takes into account the original motion. The minimisation function therefore attempts to minimise the distance to the original motion and simultaneously smoothes the muscle forces, which are represented using a formulation similar to proportional-derivative servos.

Once the spacetime optimisation process generates a suitable motion for the simplified and abstracted hierarchy in Popovic and Witkin's work, the complete structure is rebuilt using the motion capture data, thereby restoring the original complexity in the character. This technique demonstrates good results for fast moving, coarse motions where large energy dynamics are involved because in general the rigid bodies lost in abstraction play less of an important role in the motion. However, because the abstraction process assumes that the inter-joint angles will largely remain unadjusted (and hence no dynamic alterations are performed on these parts), the results can be less feasible for less energetic and more detailed motions. For example, this technique would not map well to processing a hand reaching a target in space because the whole arm is abstracted to a single entity. Consequently, the technique only produces a semi-physically realistic result by grossly adjusting the motion but ignoring the finer detail of it, a risk also shared by the wavelet approach of Liu *et al.* [Liu94b].

Safonova *et al.* [Safo04] present an alterative approach to reducing the complexity of the spacetime constraints technique. In this work, a collection of motion captures performing similar movements are sampled using a principal component analysis (PCA) technique to allow a low dimensional representation of the motion to be produced, similar to that used my Grochow *et al.* [Groc04] for stylised inverse kinematics. Using the lower dimensional representation of the articulated character, user-defined constraints are constructed that guide the simulation towards a desirable goal using momentum-based dynamics and joint torques to encompass the dynamics of the environment. Using this representation for the dynamic constraints, the optimisation function becomes one that combines a minimisation of the joint angle torque and joint angle smoothness. Similar to the work presented by Popovic & Witkin, when the spacetime process determines a solution for the lower-dimensionality space, the complete motion is rebuilt using the PCA that was used to first simplify it. This process reintroduces stylisation effects of the original character that was sampled. However it also introduces artefacts such as foot sliding or slightly incorrect environmental interactivity. To combat this, Safonova *et al.* make use of an inverse kinematics algorithm to reassert these conditions. However, this may in turn break the dynamics of the environment, as too could the process of rebuilding the full character description from the low dimensional version using the PCA, a trade-off that is always apparent when abstracting or reducing the complexity of the original articulated structure.

A further modification to the spacetime optimisation process for simulation dynamics has been demonstrated by Liu & Popovic [Liu02] and later built upon by Abe *et al.* [Abe04]. The basic concept behind these works is to reduce the complexity of representing the full dynamics of rigid bodies to momentum-based formulations, as opposed to the representation of the body itself as with the previously discussed techniques. The work reduces unconstrained motion, such as falling through the air, as a simple trajectory of the centre of mass, whereas for a constrained body, analytical models are used to define the momentum, based on empirical biomechanical studies. The optimisation function used to complete the spacetime optimisation process combines minimal mass displacement and DOF velocity as well as a measure to ensure a good centre of mass to simulate balance control. This complements the momentum-based modelling of the characters as a way of reducing the overall movement required to simulation a motion, which indirectly is a measure of energy. Consequently, despite the lack of modelling muscular forces, such energy measures are implicitly minimised to produce natural looking motions. A similar modelling technique is taken by Sulejmanpasic &

Popovic [Sule05], where instead of minimising the degree of DOF changes, they choose to favour motions that are similar to an example motion, but applied to ballistic portions of the motion only.

The simplification of the rigid body dynamics to momentum-based formulations can however lose some of the realism and in many situations is only suitable for ballistic motions. For example, when a character is falling thought the air, Liu & Popovic effectively treat it as a rag-doll in that all linear and angular momentum is preserved and only trace out the trajectory of the centre of mass. While this may be desirable for some situations, it does fail to capture any muscle forces that a conscious human body would exert while falling, such as trying to reach out for something to prevent it from tumbling. Furthermore, because the constraint momentum aspect is analytically derived, it only approximates the dynamics of the environment as opposed to completely adhering to it. These potential shortcomings of the technique are partly addressed by Abe *et al.* who sample existing motion data to produce suitable momentum-based curves, which help guide the solution towards a more natural looking motion.

The main advantage of using momentum-based optimisation is a reduction in the computational cost required for full rigid body dynamic representation, but this is offset against the range of motions that can be generated. Abe *et al.* demonstrate how to spawn off a family of different motions based on a single piece of motion capture data and then interpolate between these to get interactive results. However, the range of motions that can be generated must lie close to the momentum models used, whether these are analytically derived or through motion data. Despite the very favourable results demonstrated, as there is no concept of muscle forces, momentum-based techniques do not lend themselves well to simulating low-energy motions that require finer attention to detail. This is a consequence of the fact that any non-passive objects that are being simulated only have the forces that were present in the sampled motion and are unable to plausibly deviate from this much while still being true to the dynamics of the environment.

Liu *et al.* [Liu05] continue with the idea of extracting parameters from a motion dataset with a view to parameterising the style of a character, which can subsequently be used to simulate a new movement. In this work, a complete mathematical representation of full rigid body dynamics is used, where the whole body is considered, without simplification. Using an example motion, an initial parameter vector is determined, which includes components of physical style, such as muscle elasticity and muscle activation preferences that allow different muscles to be favoured more than others. The style learning process is achieved by finding a set of motion parameters to the rigid body dynamics that minimises the distance between the original motion and the generated motion. The stylisation parameters that are learnt at this stage are subsequently projected back into a spacetime optimisation process that determines a new motion based on foot plant constraints and the dynamics of the environment, while attempting to minimise energy functions.

The process illustrated by Liu *et al.* demonstrates the most comprehensive mathematical representation of articulated dynamics by neither simplifying the hierarchical structure nor reducing the rigid body dynamics. Since this approach has been taken, unlike the abstracted or simplified approaches taken in the past, this work illustrates very realistic, continuous results. Furthermore, their ability to capture the style of the character and map it onto a new motion shows great promise, but conversely, this comes at the very high computational costs incurred while both learning the style and generating a new one.

Other research has been proposed to capture the style of characters including the works of Urtasun *et al.* [Urta04] and Hsu *et al.* [Hsu05], however neither of these approaches take account of dynamics. In the first of these approaches, similar to the work of Safonova *et al.*, a principal component analysis is performed on an existing motion capture dataset. Motions are subsequently modelled as linear combinations of the principal components, where the combination of these components is adjusted to produce different effects within the resulting motion. For example, given a walking motion, new walking motions can be generated with different speeds and stride lengths. The limit of this technique lies in the simplistic process of linearly blending the principal components in that different movements cannot be generated that exhibit the style shown by Liu *et al.*. Therefore, if a walking motion is sampled, the technique is only able to produce other, similar walking motions that are only technically different as opposed to a turning motion. Although, because of the simple linear blending process of this technique, new motions can be simulated in real-time, which is something that is not possible in the more complex system presented by Liu *et al*.

In contrast, Hsu *et al.* approach the problem of stylisation by building a translation model that maps an input motion to an output motion, for example a walking movement to a limping one. The translation model is built using an iterative motion warping technique that attempts to produce a set of parameters that brings the input motion as close to the output motion as possible. The generated parameters are subsequently used by a linear time-invariant model to map any motion that is similar to the original sampled input one to a motion similar to the original output one but in the style of the new input motion. Similar to the approach used by Urtasun *et al.*, Hsu *et al.* demonstrate a real-time technique (although the translation model is determined off-line) of simulating a new motion that exhibits the same style as a sample one. However, since this method does not parameterise the style of the character, it relies on the definition of suitable translation models, which is not always possible. This consequently limits the flexibility of the generated motions because there is no control over the position of body parts unlike the work demonstrated by Liu *et al*.

Two further uses of the spacetime constraints technique in the area of character animation are worth mentioning, albeit not using dynamics. In the first, Rose *et al.* [Rose96] uses the technique to generate smooth transitions between existing motion clips by setting up a minimal joint change over time as the optimisation function and specifying the spatial connectivity between the two motions as the system constraints at each end of the motion. The result is a method that generates smooth and realistic transition between similar motions.

In the second, Gleicher [Glei98a] utilities the spacetime paradigm to solve the retargetting problem between characters of different sizes but identical hierarchical structures. The concept behind this work is to specify motion constraints, such as planting feet appropriately to the floor or grasping an object in space, and then by minimising the distance between the original motion and the generated motion, the spacetime optimisation produces a motion that allows smooth transitions into and out of these constraints.

In the remainder of this chapter, a dynamics-based optimisation process is described that closely resembles the complexity and completeness only before shown by Liu *et al.* [Liu05]. However, as opposed to capturing the style of an actor from their motion data, the technique that is described in this thesis demonstrates a process by which the motion of one actor can be successfully reconfigured to another using the target actor's biomechanics, not needing to sample motion data from the target actor. Furthermore, because the technique is based on a general encoding of dynamics, the method can be used to introduce additive movements. The mathematics behind the dynamics encoded used to achieve this process are discussed in the continuation of this chapter, where the results are presented in Chapter 6.

5.3 The Theoretical Aspect of Constructing the Whole Dynamics Representation

This section further elaborates upon the theoretical constructs of the dynamics representation that is used in this thesis to reconfigure motions, using the rigid body mechanic definitions of section 5.1. The system that is described is based on the non-linear optimisation problem of Equation 5.18, where the constraints, C, are primarily defined by Lagrange's equations of motion, which are given by Equation 5.5.

The equality constraints of the non-linear optimisation process of Equation 5.18 are modelled as equality conditions equalling zero, without loss of generality. However, inequality constraints have been omitted from this general definition because they need to be treated differently to their equality counterparts. Once a strategy for solving Equation 5.18 is outlined in section 5.31, it is extended in section 5.3.2 to include inequality constraints as they become invaluable when dealing with environmental interactions.

The expansion and representation of the Lagrangian generalised variables are subsequently discussed in section 5.3.3. This covers their time-based representation, which when combined with the non-linear optimisation process, concludes the theoretical aspect of the dynamics-based optimisation system used in this thesis to reconfigure motion capture data.

5.3.1 Solving the Non-Linear Optimisation Problem

The process of finding an optimal solution of a multi-variant system without regard to the constraints is first defined. Thereafter, the constraints will be reintroduced to construct a complete process for solving Equation 5.18. The algorithm outlined in this section presents the standard approach to solving the non-linear optimisation algorithm [Gill81]. However, the final set of equations that are used in this thesis are formulated slightly different to their normal guise and are thus presented here.

Equation 5.18 is solved using an iterative approach from a set of starting values to a local minimum of *R(S)*. Therefore, given the current state of generalised coordinates at iteration *k*, x_k, *R(S)* can be expanded using the Taylor-series to formulate a suitable step, *p*, toward the minimum of the system. This is given in Equation 5.19, where *R(S)* is expanded to the third term. It should also be noted that when referring to differentials of functions in this discussions of solving Equation 5.18, it is the partial derivates with respect to the generalised coordinates, *S*, that are being referred to and not the time derivates as is the usual case.

$$R(x_k + p) = R(x_k) + \dot{R}(x_k)^T p + \tfrac{1}{2} p^T \ddot{R}(x_k) p \qquad \textbf{(5.19)}$$

If the *p*-vector is taken to be the current location to a new minimum, it follows that the second and third terms of Equation 5.19 must be minimised to find a solution in *p*. Equation 5.19 can be rewritten to find the minimum of the quadratic function, in terms of *p*, given in Equation 5.20.

$$\Phi(p) = \dot{R}(x_k)^T p + \tfrac{1}{2} p^T \ddot{R}(x_k) p \qquad \textbf{(5.20)}$$

Through the Taylor-series expansion of Equation 5.20, a solution to the quadratic subproblem can be derived as given in Equation 5.21, where α is a scalar step length and *b* is the step direction.

$$\Phi(p + ab) = \Phi(p) + ab^T \left(\dot{R}(x_k) + \ddot{R}(x_k) p \right) + \tfrac{1}{2} a^2 b^T \ddot{R}(x_k) b \qquad \textbf{(5.21)}$$

A point that is a local minimum of the function *F(p)* must also be defined as a stationary point otherwise it can be shown neighbouring points are strictly less than the given point and hence it would not be a local minimum. Therefore, to achieve a local minimum of *F(p)*, the gradient vector must vanish, which corresponds to the second term of Equation 5.21. Consequently the solution of Equation 5.20, is given by the linear system of Equation 5.22, which gives rise to Newton’s method for solving minimisation problems without regard to any constraints.

$$\dot{R}(x_k) + \ddot{R}(x_k) p = 0 \quad \Rightarrow \quad \ddot{R}(x_k) p = -\dot{R}(x_k) \qquad \textbf{(5.22)}$$

Focusing now on the constraints, assuming that a current state, x_k, is defined and there is a possible move to an optimal solution, *x**, through a step *q*, the constraint relationship of Equation 5.23 can be defined.

$$C(x^*) = C(x_k + q) = 0 \qquad \textbf{(5.23)}$$

A linear approximation to 5.23 can be formulated by expanding it using the Taylor-series, about x_k, which gives Equation 5.24.

$$\begin{aligned}&C(x_k+q)=C(x_k)+\dot{C}(x_k)q=0\\ &\Rightarrow C(x_k)+\dot{C}(x_k)q=0 \quad \Rightarrow \quad \dot{C}(x_k)q=-C(x_k)\end{aligned} \tag{5.24}$$

The initial minimisation formulation of Equation 5.18 can be reformulated, which links the result of the objective quadratic subproblem of Equation 5.20 to that of the constraints, Equation 5.24, in terms of linear systems. The minimisation problem can thus be rewritten as Equation 5.25.

$$\begin{aligned}&\textit{minimise} && \dot{R}(x_k)^T p+\tfrac{1}{2}p^T\ddot{R}(x_k)p\\ &\textit{subject to} && \dot{C}(x_k)p=-C(x_k)\end{aligned} \tag{5.25}$$

Formulations, such as Equation 5.25, that have an explicit quadratic subproblem, F, are termed sequential quadratic problems and hence methods for solving nonlinear constraints that incur such a problem are termed sequential quadratic programs (SQP).

The solution of Equation 5.25, and hence the original problem, can be computed directly. By taking the solution to Equation 5.25 at iteration k as p_k, the solution vector can be rewritten as a combination of vectors whose space spans the range space and null space of the differentiated constraints matrix. This is illustrated in Equation 5.26 where Y_k is a matrix whose columns span the range space of C^T and the matrix Z_k defines the null space of C.

$$p_k=Y_k p_y+Z_k p_z \tag{5.26}$$

The range space vector, p_y, is determined through the constraints as given by Equation 5.27.

$$\begin{aligned}&\dot{C}(x_k)p_k=\dot{C}(x_k)\left(Y_k p_y+Z_k p_z\right)=\dot{C}(x_k)Y_k p_y+\dot{C}(x_k)Z_k p_z\\ &=\dot{C}(x_k)Y_k p_y=-C(x_k) \qquad \text{since} \qquad \dot{C}(x_k)Z_k=0\end{aligned} \tag{5.27}$$

The final step is therefore to find a solution to the null space vector, p_z. This is determined by minimising the quadratic objective function of Equation 5.25, using the linear system defined in Equation 5.22. The mathematical process is outlined in Equation 5.28.

$$\begin{aligned}&\ddot{R}(x_k)p_k=-\dot{R}(x_k)\\ &\Rightarrow \ddot{R}(x_k)\left(Y_k p_y+Z_k p_z\right)=-\dot{R}(x_k)\\ &\Rightarrow \ddot{R}(x_k)Y_k p_y+\ddot{R}(x_k)Z_k p_z=-\dot{R}(x_k)\\ &\Rightarrow \ddot{R}(x_k)Z_k p_z=-(\dot{R}(x_k)+\ddot{R}(x_k)Y_k p_y)\end{aligned} \tag{5.28}$$

Equation 5.27 and Equation 5.28 subsequently determine a step vector that enforces a system of equality constraints while at the same time minimises an objective function. Equation 5.27 enforces the constraints without consideration to the objective function. Equation 5.28 computes a hyperplane step vector to minimise the objective function, the result of which cannot affect the constraints as the vector must lie in the null space of the constraints. Hence Equation 5.29 holds for any vector p_z.

$$\dot{C}(x_k)Z_k p_z = 0 \tag{5.29}$$

The solution to the complete problem is consequentially the iterative calculation of the quadratic programming subproblem until the algorithm fails to produce reduced results in the optimisation function without breaking the constraints.

In practice, the theoretical equations for solving the SQP tend to favour reducing the minimisation function at the expense of breaking constraints. This is associated with a degree of ill-conditioning which leads to difficulty in determining the range and null space vectors of the system. If these vectors are not exact then, when the solution is determined for the objective function, it is possible to produce a vector that does not correlate with the desired behaviour of Equation 5.29 and hence break the constraints. Therefore, although the constraints are reaffirmed at the next iteration, the final result of a given iteration can always break the constraints.

The equations for solving the SQP are therefore formulated slightly differently so that this problem can be compensated for. The first step is to express the update vector slightly differently using only two vectors, given by Equation 5.30, where r_k is the solution of the objective function without regard to the constraints. This first step is given in Equation 5.31.

$$p_k = q_k + r_k \tag{5.30}$$

$$\ddot{R}(x_k)r_k = -\dot{R}(x_k) \tag{5.31}$$

The second step is to solve the linear approximation of the constraints, where the result from the objective function is projected into the null space of the differentiated constraint matrix, which is given by Equation 5.32.

$$\begin{aligned} &\dot{C}(x_k)\left(q_k + r_k\right) = -C(x_k) \\ &\Rightarrow \dot{C}(x_k)q_k = -\left(C(x_k) + \dot{C}(x_k)r_k\right) \end{aligned} \tag{5.32}$$

By reversing the order that the calculations are performed, the result is biased towards the constraints. If for example the optimisation vector, $r_{k,}$ violates any constraints, which it undoubtedly will, then they can be reaffirmed in the q_k vector. It is a direct implementation of Equation 5.30

through to Equation 5.32 that is used to solve the mechanical dynamic system over a complete period of time in this thesis.

5.3.2 Inequality Constraints

In the previous discussion of the SQP technique, the constraints have been defined as equality-based. However, it is useful to include inequality formulations. To this end, this section reviews two techniques of incorporating inequality constraints in to the SQP method; lax variables and active set. Without loss of generality the inequality equations are assumed to be of the form illustrated in Equation 5.33.

$$c_i(S) \geq 0 \tag{5.33}$$

The principle of lax variables is to rewrite the inequality constraint in the form of a new equality constraint. This is accomplished for each inequality constraint through the introduction of a new generalised coordinate, g_i, which is used in a replacement equality constraint as illustrated in Equation 5.34.

$$\begin{aligned} c_i(S) &= g_i^2 \\ c_i(S) - g_i^2 &= 0 \end{aligned} \tag{5.34}$$

The inequality of Equation 5.33 must therefore always hold if the equality constraint of 5.34 is sustained because it would be impossible for the square of the lax variable, g_i, to be negative in a real numbers environment. It is then the responsibility of the system to determine an appropriate value for the lax variables that correlates to how semi-positive the constraint is.

An alternative technique to lax variables is the use of active constraint sets, where for the k^{th} iteration of the SQP technique, a set of constraints to be involved in the optimisation process is constructed from all possible system constraints. The set of constraints used for the k^{th} iteration is termed the active constraint set. The active constraint set consists of binding constraints for that particular iteration, while leaving out any non-binding constraints, where a binding constraint is one that restricts the movement of the step in some way. All equality constraints are classified as binding because they always remove a degree of freedom from within the system by relating one or more generalised coordinates. This is illustrated in Figure 5.5a.

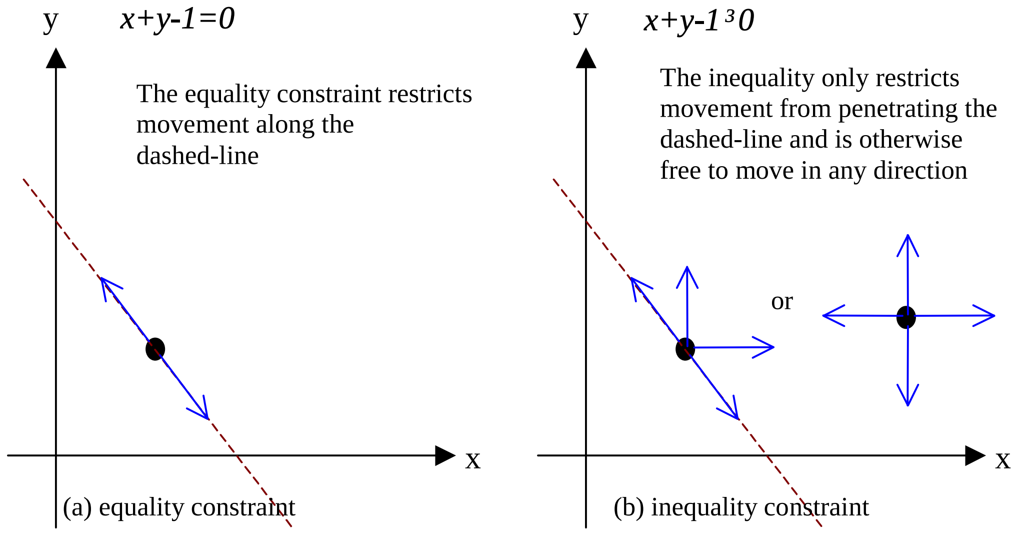

y
x+y-1=0
The equality constraint restricts movement along the dashed-line
x
(a) equality constraint
y
x+y-1³0
The inequality only restricts movement from penetrating the dashed-line and is otherwise free to move in any direction
or
x
(b) inequality constraint

Figure 5.5: Restricted movement for (a) equality constraints and (b) inequality constraints, where the blue arrows demonstrate the possible range of movement from a given spatial position

In the case of inequality constraints, the constraint may only restrict the space of step vectors for a particular configuration if the current state resides on the boundary of the constraint. The boundary of an inequality constraint is defined as being a state that results in the inequality equalling its test value, i.e. being equal to zero. Any other state for which the constraint is greater than its test value would not be classified as a boundary state, which is diagrammatically illustrated in Figure 5.5b.

As it is desirable to restrict the system from penetrating a boundary prescribed by an inequality constraint, any inequality constraints that lie on their boundary become binding and as such are included in the active constraint set. Furthermore, since Equation 5.33 must be identically zero if the constraint is on its boundary, the constraint is modified such that it is equal to zero and hence the active constrain set remains exclusively for equality constraints. The consequence of this high-level process is that if the system tries to penetrate the boundary, the constraint will resist the change, however once the system state has left the boundary, the constraint becomes non-binding and extra freedom is return to system.

The two techniques presented for handling inequality constraints both have their drawbacks. In the case of the lax variables, each inequality constraint introduces a new generalised coordinate into the system and hence the differentials of the objective and constraint functions are increased in both size and complexity. A further practical problem of the lax variable solution is that because of the square nature and loss of sign, the values tends to favour increasing over time, which means the constrained generalised coordinates must also do so to meet the constraint, a behaviour which is undesirable. This can be compensated for through the addition of a suitable objective function. However, this is both a difficult metric to balance and further adds complexity to the system.

In contrast, since the method of active constraints works in a completely different way to that of lax variables, the same problems are not apparent. However, there are issues associated with switching on and off constraints per iteration. It is conceivable that given the active set of constraints, a step vector is calculated that moves in a direction towards a boundary of a non-binding constraint.

This is not a problem provided that the magnitude of the step is small enough to avoid penetration. However, if this is not the case, the inequality constraint will be violated.

The solution in preventing the system from violating inequality constraints using the active set technique lies in the analysis of the problem in that regardless of the direction the step vector is taking, as long as the step magnitude is small enough, the system can at worse border the inequality constraint. Therefore a step length scalar, a, is introduced into the update step of Equation 5.30, which is determined using the relationship given in Equation 5.35, for all inequality constraints. An upper bound on a is imposed so that the magnitude of the original step vector is not exceeded as this would normally result in the system diverging.

$$\forall c_i : c_i(S_k + a_k p_k) \geq 0 \quad \text{where} \quad 0 < a_k \leq 1 \tag{5.35}$$

An additional problem that the active set technique can suffer from is that of stability. Since constraints can be switched on or off per iteration, the constraint differential matrix varies which can have a noticeable effect on the solution of the linear system given in Equation 5.32. Therefore, from iteration to iteration, there may be different optimum solutions, which result from the change in the number of restricted degrees of freedom. Consequently, the optimisation process could get caught between solutions as the inequality constraints are turned on and off. The potential instability depends on the neighbourhood of the optimal solution. For example, if in the neighbourhood of the optimal solution, the inequality constraints are consistent, it is unlikely that any instability problems will arise. However, this property cannot always be depended on and so the ability to manually set whether an inequality constraint is binding or not it reserved within the implementation of this thesis.

Between the two techniques of incorporating inequality constraints, because of the reduced costs in terms of complexity, the use of active constraints is opted for in this thesis. Furthermore, in practice the active set method provides more stable results when compared like for like to the lax variable approach.

5.3.3 Representation of the Generalised Coordinates

Thus far a mathematical description of a rigid body system has been given, but little attention has been paid to the form of the generalised coordinates. As the equations of section 5.1 has been stated, it would be assumed that each coordinate is represented as a single scalar value. However, this would give rise to a time-static system whereas what is required is a time-dynamic model that allows the motion of characters to be represented. Each generalised coordinate needs to be represented as a time-varying function. This can be achieved through the use of either discrete or continuous representations.

The use of discrete functions requires the differentials, needed by Lagrange’s equations, to be determined via finite differencing. This causes problems. For accurate differentials, finite differencing requires small time intervals between each sample, which in itself is not an issue, but if a complete motion were to be considered, this will lead to a large number of variables per coordinate. A further drawback of using discrete functions is their accuracy as the technique only approximates the gradients around a specific point and is not necessarily guaranteed to give correct results. Added to this the high computational costs that would be incurred in determining the differentials of the complex formulations of Equation 5.16 for example, the use of discrete functions would be an impractical choice.

Therefore continuous functions are used to representing the generalised coordinates. In order to avoid the practice of finite differencing, the representation needs to be at least continuous to the second differential (C^2 continuity). There are many different types of C^2 representations although non-uniform piecewise cubic B-spline curves are chosen in this thesis. This choice of continuous representation comes from the well-defined construction of the curve segments, which exhibit only local control over the curve, and the relatively simple differentials that result.

5.3.3.1 Non-uniform piecewise cubic B-Splines

In this section a brief overview of the construction for the non-uniform piecewise cubic B-spline is presented. This review only iterates the mathematical constructs from standard spline and geometric modelling sources (e.g. [Bart87, Mort85]). However, it is presented here because the formulations are later addressed in section 5.4 during a discussion of the practical problems of implementing the theoretical constructs.

The piecewise curve is constructed from multiple segments that are each defined as parameterised cubic functions. The parameterisation of the complete curve makes use of a knot vector that defines the joints in the curve between each segment and the construction of the cubic segments, where the traversal between two neighbouring knots in the sequence has the effect of running along one of the cubic segments. The knot sequence is defined as an increasing sequence of numbers as illustrated in Equation 5.36.

$$[u_0, u_1, \mathbf{K}, u_i, \mathbf{K}, u_{last}] \text{ where } u_0 < u_1 < \mathbf{K} < u_i < \mathbf{K} < u_{last} \qquad \textbf{(5.36)}$$

It should be noted that, with the exception of C^0 continuous curves, the start of the piecewise curve does not start at $u=u_0$ and likewise does not end at $u=u_{last}$. Also, in the definition of the knot vector, Equation 5.36, an always increasing knot sequence is imposed (as opposed to multiple knot values) but for generic B-Splines this does not have to be the case.

In addition to the knot spacing of Equation 5.36, the curve’s control points, V_i , are introduced, which define the range of the B-spline curve over the domain of u. As is the case for B-splines, the curve only approximates the control points by sweeping out a path that approaches the points as opposed to interpolating them.

Each of the cubic segments, Q_i, that make up the piecewise cubic curve are defined in terms of the control points and a set of basis functions, B_i. This relationship is defined in Equation 5.37. The complete piecewise cubic B-spline is subsequently the summation of each segment, as given by Equation 5.38.

$$Q_i(u) = V_i B_{i,-3}(u) + V_{i+1} B_{i+1,-2}(u) + V_{i+2} B_{i+2,-1}(u) + V_{i+3} B_{i+3,-0}(u) \qquad \textbf{(5.37)}$$

$$Q(u) = \sum_i Q_i(u) \qquad \textbf{(5.38)}$$

The basis functions are cubic functions defined with respect to the knot spacing and are non-zero over a consecutive sequence of only 5 knots. The basis functions are defined using one-sided basis functions, which leads to the recursive formula given in Equation 5.39, where k is the order of the associated B-spline.

$$B_{i,k}(u) = (-1)^k (u_{i+k} - u_i)[u_i(k):t](u-t)_+^{k-1} \qquad \textbf{(5.39)}$$

Through the evaluation and expansion of Equation 5.39 for 4th order basis functions, we obtain the basis functions required for our representation of generalised coordinates, which is illustrated in Equation 5.40(a, b, c, d). The basis equations of 5.40 and 5.37 are related by dropping the order index from the set of 5.40 formulations.

$$B_{i,4,-0}(u) = \frac{(u-u_i)^3}{(u_{i+1}-u_i)(u_{i+2}-u_i)(u_{i+3}-u_i)} \qquad u_i \text{ £ } u < u_{i+1} \qquad \textbf{(5.40a)}$$

$$\begin{aligned} B_{i,4,-1}(u) = {} & \frac{(u-u_i)^2(u_{i+2}-u)}{(u_{i+3}-u_i)(u_{i+2}-t_i)(u_{i+2}-u_{i+1})} + \\ & \frac{(u-u_i)(u_{i+3}-u)(u-u_{i+1})}{(u_{i+3}-u_i)(u_{i+3}-u_{i+1})(u_{i+2}-u_{i+1})} + \\ & \frac{(u_{i+4}-u)(u-u_{i+1})^2}{(u_{i+4}-u_{i+1})(u_{i+3}-u_{i+1})(u_{i+2}-u_{i+1})} \end{aligned} \qquad u_{i+1} \text{ £ } u < u_{i+2} \qquad \textbf{(5.40b)}$$

$$\begin{aligned} B_{i,4,-2}(u) = {} & \frac{(u-u_i)(u_{i+3}-u)^2}{(u_{i+3}-u_i)(u_{i+3}-u_{i+1})(u_{i+3}-u_{i+2})} + \\ & \frac{(u_{i+4}-u)(u-u_{i+1})(u_{i+3}-u)}{(u_{i+4}-u_{i+1})(u_{i+3}-u_{i+1})(u_{i+3}-u_{i+2})} + \\ & \frac{(u_{i+4}-u)^2(u-u_{i+2})}{(u_{i+4}-u_{i+1})(u_{i+4}-u_{i+2})(u_{i+3}-u_{i+2})} \end{aligned} \qquad u_{i+2} \text{ £ } u < u_{i+3} \qquad \textbf{(5.40c)}$$

$$B_{i,4,-3}(u) = \frac{(u_{i+4}-u)^3}{(u_{i+4}-u_{i+1})(u_{i+4}-u_{i+2})(u_{i+4}-u_{i+3})} \qquad u_{i+3} \text{ £ } u < u_{i+4} \qquad \textbf{(5.40d)}$$

From the equations of 5.40 each basis function spans 4 sequential knot intervals, whose left support is given by the u_i^{th} knot. This is illustrated in Figure 5.6. Taking the basis functions as the building blocks of the piecewise cubic B-spline of Equation 5.38, to define a curve using *c* curve segments, *c+3* control points and *c+3* basis functions will be needed. With *c+3* basis functions, *c+7* knots are required, while the span of the complete curve is between $u=u_3$ and $u=u_{3+c}$, which is shown in Figure 5.6.

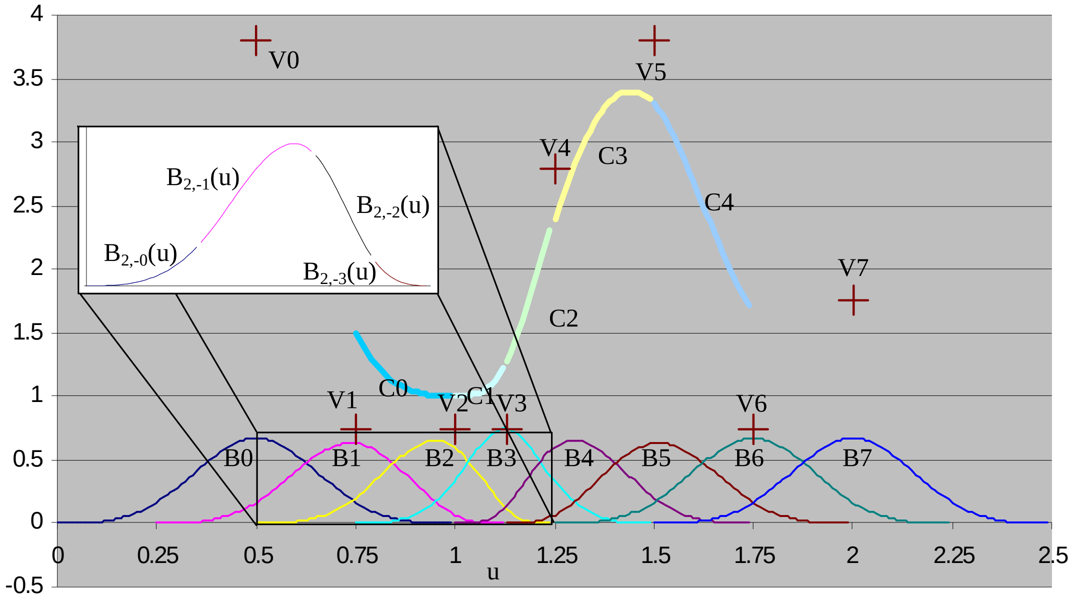


Figure 5.6: Cubic basis functions and the resulting piecewise cubic B-spline over the knot sequence, *[0, 0.25, 0.5, 0.75, 1.0, 1.125, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5]*

Thus, each generalised coordinate in the dynamics system can be modelled as a non-uniform piecewise cubic B-spline that passes through time parameterised on *u*. A linear time-based mapping to *u* is chosen because it is desirable to keep the knot values positive however the curve starts at u_3. Therefore it is not possible to directly map a value of $t=t_0=0$ to this knot, while still avoiding knot multiplicities. Hence a simple linear mapping where u_3 is added to the desired time to give the *u* parameterisation value, as given by Equation 5.41. The knot spacing is accordingly determined such that $t=t_{end}$ is equivalent to $u=u_{3+c}+u_3$.

$$u = t + u_3 \tag{5.41}$$

The main purpose for representing the generalised coordinates as continuous functions resides with the ability to numerically differentiate them. At the level of the generalised coordinates, the, differentials required by the dynamics system are those with respect to time. This is achieved by performing differentiation by substitution on the basis functions, as illustrated by Equation 5.42.

$$\begin{aligned} \dot{Q}_i(u) &= V_i\dot{B}_{i,-3}(u) + V_{i+1}\dot{B}_{i+1,-2}(u) + V_{i+2}\dot{B}_{i+2,-1}(u) + V_{i+3}\dot{B}_{i+3,-0}(u) \\ \dot{B}_i(u) &= \frac{d\left(B_i(u)\right)}{dt} = \frac{d\left(B_i(u)\right)}{du}\frac{du}{dt} \end{aligned} \tag{5.42}$$

Since there is a linear mapping between *u* and *t*, as defined in Equation 5.41, the second term on the right-hand side of Equation 5.42 resolves to unity. Therefore, the curve differentials can be determined by taking the first and second differentials of the basis functions themselves with respect to *u*.

It now becomes important to elaborate on the earlier constraint that prohibited knot multiplicities, i.e. each knot value has to be strictly larger than the previous one. In taking the second differential of the basis functions, the outcome is a linear equation that involves only two knots within the domain of the basis function. If however, these knots were identical then the result would be a vacuous second differential. If more than two knots were identical then this could cause the first differential, or even the original curve, to become vacuous, thereby preventing a successfully evaluation of the dynamic equations at such points in time. Hence the knot value multiplicity constraint.

Through the time and basis function parameterisation mapping, for the same period of time, if knots were placed closer together, more knot values would result, which gives greater curve segments and hence a higher level of frequency control when compared to a larger spacing for the same time frame. Therefore, ideally, a sparse knot vector, with respect to time, would be used in areas that have little or no change in the generalised coordinate, but have more dense knots in time intervals where the converse is true. The formulations presented in Equations 5.40 are sufficient to cope with this requirement of non-uniform spacing, however the control points and their spatial locations need to be considered. If it were known exactly at which points in time more control over the curve was needed, when it was initially constructed, future knot spacing refinement could be avoided. Rarely is this the case. The more common situation is that a knot spacing is defined and the control points are placed in space to characterise the path of the generalised coordinate through time, and it is later that finer control over a certain part of the curve is desirable.

Curve refinement subdivides an existing curve segment into two new curves that span the same time interval as the original. Given the earlier definitions that link the number of control points and knot values to the number of curves, it follows that the introduction of a new control point and knot value within the existing sequence of values is required. The location of the new knot is calculated using the value of *u* given a time *t* at which to subdivide the curve, using Equation 5.41. The basis functions can thus be recalculated following the set of Equations in 5.40.

A problem occurs when considering the location of the new control point and what value it should take such that the two new segments exactly replicate the original single segment. Since each control point is involved in up to 4 different curve segments, not only will the new control point need to be positioned, but also the surrounding control points require relocating. The Oslo algorithm [Joe97, Meye91] can be used to do this.

The implementation of the Oslo algorithm provides the ability to refine the resolution of generalised coordinates after the initial knot spacing is defined. Therefore, if a faithful solution to a dynamical system was found, but slightly more control over a certain time frame was needed, the complete curve can be refined whilst maintaining values for generalised coordinates that matched the original solution. Therefore the eliminates the need of manually reasserting the original solution with the new control points before continuing in finding a dynamically correct simulation. This behaviour becomes important when dealing with unpredictable fast changes in acceleration due to impulse forces such as contact so that the forces can be modelled more accurately using a finer knot spacing.

5.4 Building the Dynamics Optimisation-Based Character Modification Process

Over the previous few sections, the mathematical constructs required to model the dynamics of a connected rigid body have been outlined, a suitable representation for the generalised coordinates has been examined and a mathematical formulation for solving a constrained minimisation problem has been considered. In this section, these concepts are brought together to give a complete description of the dynamics-based character modification process. However, the process of collating the theoretical aspects and mapping them to a practical framework introduces some fundamental issues, which are described in this section. In addition to presenting the problem, original solutions are given to solve them, which are successfully utilised in the final system.

This section starts by considering the mapping between the generalised coordinates described in section 5.1 for defining Lagrange’s equations of motion with the free system variables of the optimisation-based algorithm described in section 5.3.1. The problem of ill-resolutioned equations, which results from the generalised coordinate representation of section 5.3.3, is highlighted and a novel solution to the problem presented. Thereafter, this section discusses the problems of dealing with impulse and discrete occurrences within the dynamics representation and since the representation of the generalised coordinates is continuous, an innovative approach of dealing with them is presented.

This section concludes by extending the work on discrete forces to encompass friction and finally the incarnation of muscle forces is presented.

5.4.1 Defining the Generalised Coordinates

During the discussions of the minimisation process in section 5.3.1, the defining variables, *S*, were referred to as the generalised coordinates of our system, however this is not strictly accurate. The set of variables *S*, are in fact free variables of the system in that they are not constant over the complete motion. This description correlates with the concept of the generalised coordinates, however, having described them as non-uniform piecewise cubic B-spline curves, generalised coordinates are not our lowest granularity. At the lowest level of granularity, the generalised coordinates are constructed in terms of a knot sequence and a set of control points. The knot sequence is considered constant over the range of the motion because, although there is the ability to refine the curve using the Oslo algorithm, it is a parameter that the user controls, not the minimisation process.

That leaves only the control points that the optimisation algorithm can adjust in order to produce a solution and it is these variables that go into making the set of free variables, *S*. Consequently, for each generalised coordinate, a set of equations are constructed using the piecewise cubic B-Spline representation of section 5.3.3.1, defined on a specified knot sequence and its own collection of control points.

Having refined the free variable membership, the partial differential equations of the optimisation process with respect to this set are elaborated on. Only the generalised coordinates themselves are involved in relationships with the control points and hence it is this level that shall be considered (i.e. from the generalised coordinates down to the free variables). The differentials of any equations above this level that makes use of the generalised coordinates can be derived through differential rules and the following base relationships.

Recalling the equation for a curve segment, as given by Equation 5.37, each product term can be seen to be between a single control point and a basis function. As the basis functions are derived through consideration to the knot sequence only, they remain constant with respect to the control point. Hence, the differential of a single curve segment, with respect to a given control point, is the associated basis function itself. This relationship also holds for first (Equation 5.42) and second time-derivatives of a generalised coordinate.

However in a piecewise cubic curve, the control point can be involved in up to 4 curve segments, as illustrated in Figure 5.6. Therefore the complete partial differential of a generalised coordinate with respect to a given control point is the sum of all basis functions involved in a product relationship with it, over the complete curve. The partial differential of any curve segment with respect to a control point that does not belong to it is zero.

5.4.2 Dealing with the Parameterisation

Despite the refinement of the free variable collection, Lagrange’s equation of motion constraints do not map into the constraints suitable for the optimisation process. The remaining variable that has yet to be dealt with is time; the basis functions are still defined over knot ranges and hence time. Despite time being non-constant over the complete motion, it cannot be added to the free variables set because it is a parameterisation variable and hence introduces a dependency within the generalised coordinates, which should be independent. Furthermore, there is a strict construct on time in that it starts and ends at a given point and has a constant gradient function, i.e., time moves with a constant rate, so in fact it is not truly a free variable in terms of those that have previously been discussed.

To eliminate the time variable, the equations are sampled at specified points in time. Given a set of Lagrange’s equations, each equation is evaluated against a set of predefined times which consequently results in a system of equations in terms of only constants and the free variables, *S*. These equations can now be presented to the optimisation process as constraints.

In the implementation of the optimisation process, sampling rate is based on the number of curves, where there are *n* samples per curve. The *n* samples are taken such that there is an even spacing along a complete uniform curve sequence, illustrated in Equation 5.43 for the j^{th} sample along the i^{th} curve segment, based on a knot spacing *u* (from Equation 5.41, $t_0=u_3$). Additionally there is a sample point at the end of the complete time range.

$$t_{i,j} = u_{4+i} * (j/n) + u_{3+i} * (1 - j/n) - u_3 \tag{5.43}$$

The result of performing the sampling is to replace each Lagrange equation with *cn+1* sampled equations, where *c* is the number of curve segments. It is therefore important to use as few samples per curve as is possible. In practice, the sample rates per curve segment can give good results using either 2 or 3 samples. Although a single sample can produce some good results, the technique is not as reliable as with more samples.

The other contributing factor that can lead to a large number of sampled equations is the number of curves used to represent a generalised coordinate and hence the knot space. As a general principle, it is useful to use as coarse a knot spacing as is practical so the system is not over defined. Initially, defining a uniform, coarse knot spacing does not hinder the results that can be produced using this technique because if a situation occurs where further granularity at a specified point in time is desired, for example to model a high-frequency occurrence, then, using the Oslo algorithm, there remains the ability to refine the knot spacing. Since the Oslo algorithm returns a new set of control point values and knot spacing that exactly replicates the unrefined curve, the optimisation process proceed from where it left off, without going back and starting from the beginning. The issue of knot refinement, and hence non-uniform knot spacing, does conversely introduce a slight inconsistency in the sampling technique presented above.

The sampling definition of Equation 5.43 equally samples a uniform curve and hence pays equal importance at each regular time interval, i.e. no one part of the curve is given higher precedence than another. However, once the curve is refined in a non-uniform manner, the sampling function no longer results in equidistant sampling. Therefore, more precedence is given to certain time periods because of the increased number of constraints in that region. Admittedly, having refined a section of curve, this should be reflected in the solution, but not at the expense of ignoring the rest of the motion.

However, it is still desirable to sampling over the individual curve segments as opposed to using a time-based sampling to ensure that a region of control points is never skipped. One counter to this problem would be to uniformly subdivide the whole curve, however this is extremely impractical because refinement is only required in a certain area. The solution that it used to maintaining a correlation between the importance of a constraint and the time interval it spans is to introduce a scaling factor that is applied to each sampled equation. The scaling factor for the i^{th} sample, b_i, is given in Equation 5.44, where t_i and t_{i+1} are the sample times for the current and next sample respectively, and *m* is the maximum time distance over all samples. In the case of the end-point sample, the time distance between samples in Equation 5.44, is taken as that between the end-point time and the previous sample time.

$$b_i = (t_{i+1} - t_i)/m \tag{5.44}$$

The scaling factor works by recognising that in reality it is unlikely that a state will ever be reached where all constraints are identically zero; the best that can be hoped for is that they are continually push towards a reduced error. Therefore, by introducing the scale factor, the residual affect that the constraint has on the outcome of the step vector can be dampened. It is through this principle that a balance is made between the non-uniform sampling over time while ensuring that the equations embody all the control points.

In comparison to the complete mathematical representation of the dynamics system, the inclusion of the scaling values results in negligible expense since they are constant with respect to the knot spacing and therefore need only be calculated once whenever the curve is refined.

5.4.3 Generalised Coordinate Refinement

As previously discussed, it is desirable to start out with as few curve segments as possible in the B-spline representation of system variables, and refine this at a later stage during the optimisation process. During the refinement process, it may only be one generalised coordinate that needs to be refined. However, from the mechanical formulations of Equations 5.13 and 5.14, it is likely that the generalised coordinate is involved in a complex mathematical relationship with others. Consequently, if a specific coordinate is to receive its extra granularity, this must also be given to the other coordinates. Furthermore, it is not only enough to give the extra refinement to coordinates related directly through an equation, because inevitably, any such coordinates will themselves be involved in mathematical relationships with other coordinates and hence need the extra refinement too. To this end, when a curve refinement is performed, each of the generalised coordinate curves at that point in time is also refined. This has the further advantage of eliminating the need to maintain many knot sequences and hence different basis functions for each coordinate.

However, at this point a fundamental problem occurs within the system of constraints as they have been defined. To study this problem, the formulation of the constraint equations needs to be examined. From section 5.1, the formulation for the Lagrange's equation of motion (Equation 5.5) is expanded for the *y*-generalised coordinate of a single bodied system, to give Equation 5.45 (this is a simplified version of Equation 5.16 for a single body). Equation 5.45 is expressed using the simplifying assumptions for the COM and body offset alignment discussed in section 5.1.2. For additional clarity, the number of rotational degrees of freedom for the body has been reduced such that it can only rotate about its x-axis, y_0, as well as dropping the parameterisation value, *t*, from the generalised coordinates (all notational symbols are taken from section 5.1). Generalised forces are also ignored for purposes of clarity.

$$M_0\ddot{y} + M_0\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}\ddot{R}_0 C_0 + gM_0 = 0$$
$$\Rightarrow M_0\ddot{y} + M_0 C_{0[y]}\left(-\sin(y_0)\ddot{y}_0 - \dot{y}_0^2\cos(y_0)\right) + gM_0 = 0 \qquad \textbf{(5.45)}$$

Equation 5.45 gives a relationship between two generalised coordinates, y and y, where all other parameters can be considered constant, including the trigonometric functions for a specific free variable state. What therefore remains are the parameterised curves of the second differentials of y and y and the first differential of y, which are all that can be used to balance the equation. The curves of these differentials are graphed in Figure 5.7 over an arbitrary time period. The constants of Equation 5.45 only serve to scale the curves and not alter the underlying resolution. Consequently, for the purposes of the continued discussion the constants will be ignored, which is reflected in Figure 5.7 by presenting each of the curves on a similar scale. Furthermore, the ensuing discussion will assume that each of the curves are independent, whereas in this example this is not true for the two curves involving the differentials of y. However, considering them as dependant, introduces an added complexity which we subsequently be addressed.

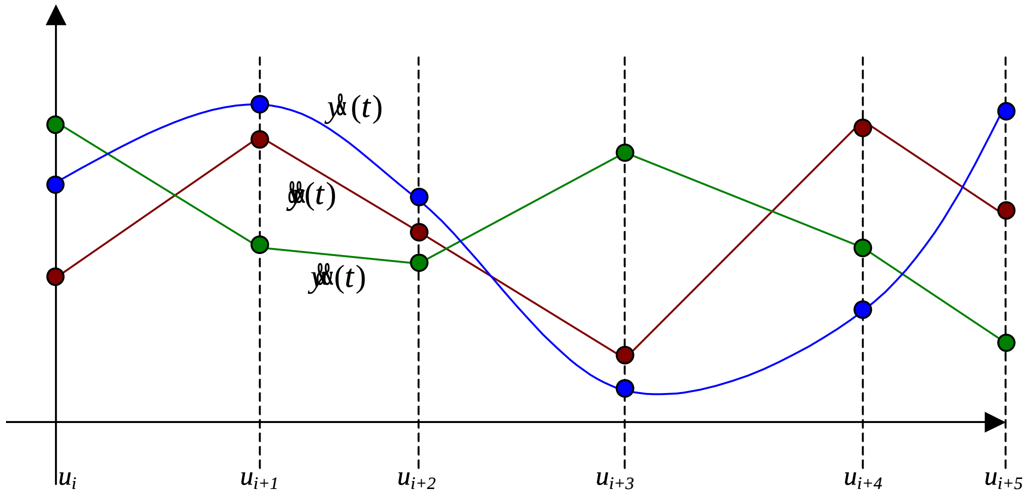


Figure 5.7: Derivatives of generalised coordinates over time

From Figure 5.7, the second differential curves are represented as piecewise linear "curves", where as the first derivative has a piecewise quadratic nature. The realisation of the problem comes through the recognition that for any given point in time, a scaled combination of these curves must cancel each other out for Equation 5.45 to hold. This is achievable if each point in time is considered independently. However, the problem is being considered over a period of time and so a combination of these curves must be found that match over time and not just at a specific point. If the curves cannot be matched over time, errors will occur in the constraint functions.

The combination of the two piecewise linear curves can always come together to cancel each other out. However, the quadratic curve can never be cancelled out using the linear curves, which means that Equation 5.45 is "ill-resolutioned".

The cause of this ill-resolutioning lies in the way the curves represents the generalised coordinates in that the knot spacing is kept the same across each of the generalised coordinates. This was to ensure that all other curves involved in a relationship during a refinement of a generalised coordinate also had the ability to exhibit the higher frequency within the time region. However, by fixing the resolution of the curves at this level, the quadratic curve cannot be further refined to better approximate the sharp turns inherent in piecewise linear curves.

Assuming for the moment that the same knot sequence over all coordinates is not enforced, this would provide the extra flexibility required to equate the components of Equation 5.45. However, for an exact representation of the linear curve, knot multiplicities would need to be introduced into the quadratic curve representation, which is prohibited in section 5.3.3.1 to avoid vacuous differentials. The additional problem with using knot multiplicities at this point lies in the relationship between the parameterisation and time. If knot multiplicities were to be introduced, this could result in curve segments spanning no time and hence become vacuous too.

Using independent knot vectors, it is conceivable that a very close approximation of the linear curve can be achieved using a highly refined knot spacing about the turning points of the joints of the piecewise linear curve. The overhead of maintaining a knot sequence for each generalised coordinate would therefore seem to be justified in terms of a mathematical point of view. A problem arises at the practical level: where should the higher degree curves be refined such that we can approximate the lower resolution ones? As a naïve solution, a few extra knots could be inserted in the higher resolution curves around the joints of the lower resolution ones. However, all of the original curves were of the cubic form in the first place, none having a higher resolution than another, so this easy solution does not map to the problem.

The problem that has been uncovered is due to mixing the differentials of the original curves and even assuming that it were possible to find a practical technique that would accurately determine how the generalised coordinates should be independently refined so that the formulations of Equation 5.45 could balance, there are still further fundamental issues. The original argument presented for keeping a knot vector the same for each generalised coordinate was to support the refinement of a particular coordinate by allowing other coordinates to mutually cooperate in having a higher frequency about a specific time period. The same argument can be applied in this instance in that if the refinement in a quadratic curve were achieved to model a linear one, other coordinates equated with it must also exhibit the extra refinement so that those sets of equations can continue to balance. Since complex relationships are being dealt with, this would inevitably loop back onto itself and result in the need to further refine the curve that was the focus of the original refinement. Hence, this would result in a paradoxical situation that continually refined the generalised coordinates in a circular manner.

The final aspect that is considered concerning the use of independent knot representation is two-fold. From Equation 5.45, there are two different resolutioned curves originated from the same generalised coordinate through its first and second time differential. Therefore considering these two curves, the only way they can equate is for them both to be identically zero because any refinement would prove futile; the refinement of the quadratic curve to meet the linear curve would naturally result in the linear curve being refined. This leads to the second conclusion that in order to resolve a given equation, similar resolution curves need to be equated. From a mathematical point of view, this is perfectly acceptable, however in a practical sense, this imposes a restriction on what the free variables should be, i.e. the linear terms would equate with each other as too would the quadratic terms, but there would be little overlap between the linear and quadratic terms. The restriction is not as strong as a regular system constraint, but it is enough to remove “half a degree of freedom” from the system. This conclusion is equally valid in the case of a similar knot sequence over all generalised coordinates.

It is therefore reasonable to conclude that any potential rewards gained using independent knot sequences are severely offset by the practical implications. Therefore, during the practical implementation, a knot sequence that can be refined, but is the same for each generalised coordinate is used, using an original solution to the ill-resolutioning based on the need to balance the differential terms.

Over a period of time, there is potential for a residual error in a constraint, which can be accounted for by the ill-resolutioning of the curves. Consequently, Equation 5.45 can be reformulated into Equation 5.46, where e_I is our residual error term for the i^{th} constraint.

$$M_0\ddot{y}_0 + M_0 C_{0[y]}\left(-\sin(y_0)\ddot{y}_0 - \dot{y}_0^2\cos(y_0)\right) + gM_0 + e_i = 0 \qquad \textbf{(5.46)}$$

Consequently, the error term can be used to introduce the extra granularity needed in order to balance the curve resolutions. In Equation 5.45, the only term that could not be balanced was the quadratic term from the first time differential of y. It would initially seem logical to model the residual error term as a piecewise quadratic B-spline so that it may cancel out the lone first differential term. However, this would effectively signify that the other two linear curves must correlate, which is not necessarily a desirable outcome.

To avoid this problem, the residual error should be represented as 3 terms that span the 0th, 1st and 2nd differentials of the piecewise cubic B-spline curves. However, in a practical sense, the addition of three extra error terms is computationally costly and can be reduced to only the 0th and 2nd differential terms, with negligible effect on the results (this has been empirically determined by exercising the system with and dynamics-based modification technique with and without the 1st differential). The error term for the i^{th} constraint is thus defined in Equation 5.47.

$$e_i = e_{i,4} + e_{i,2} \qquad \textbf{(5.47)}$$

The 0th, $e_{i,4}$, and 2nd, $e_{i,2}$, differential residual terms are represented as a piecewise cubic and linear B-Splines respectively over the same knot spacing as the other generalised coordinates. However, the 2nd differential curves span the knot range u_2 to u_{last-2}. A piecewise linear B-spline curve is used to correlate with the second differential terms as opposed to a second differential piecewise cubic B-spline because it is computationally more efficient. Furthermore, if $e_{i,2}$ were represented as a cubic curve, it is plausible that ever increasing control point values would result over time, which in terms of ill-conditioning is not a desirable consequence. The expansion of the piecewise linear B-spline curves is given in Appendix B.1.

In the context of the optimisation algorithm, the error curves will themselves be considered as generalised coordinates and as such the process can dynamically adjust them as required, although this does have consequences. The ideal dynamics solution is one in which the non-error generalised coordinates approximate each other as best as possible and only use the error generalised coordinates when this is not possible. However, the optimisation process is free to set the error curves to be arbitrary in size. To resolve this sub-problem and get back to the ideal solution, additional minimisation formulations are included into the objective function that attempt to minimise the square of the error terms, given by Equation 5.48, in effect driving them to zero as much as possible.

$$R_i = e_{i,4}^2 + e_{i,2}^2 \qquad \textbf{(5.48)}$$

Through the implementation of the error curves, a solution to absorb any residual errors within the constraints and hence balance the resolution of the constraints has been demonstrated. The importance of this is to stabilise the constraint equations and make it possible for the optimisation process to principally drive the constraints to zero.

5.4.4 Discrete Collisions and Impulses in a Continuous Domain

Throughout the discussions of the generalised coordinate representation, the need for curve refinement has been maintained. Such refinement would be used for quickly changing actions, such as a faster moving character or a karate kick. However, in this section the infinitely fast changing motions that belong to impulse and collision forces are addressed.

By their definition, impulse/collision forces happen at a specific point in time and not over a period and hence are discrete in their nature; herein lies the problem because the dynamics-based system described in this thesis is modelled in a continuous paradigm. Assuming that discrete events can be modelled continuously, through the process of refinement and with the aid of the constraint balancing equations of Equation 5.44, the optimisation system that has been defined would cope with such fast changes in motion. Admittedly, because the curves have been prevented from having knot multiplicities that would allow sharp changes in motion, the results would be approximations. This illustrates the importance of weighting the constraints dependent on the time period they span, as discussed in section 5.4.2 and given by Equation 5.44, because without them the local error of approximating the discrete change would dominate the set of constraint functions. This principle is illustrated in Figure 5.8, where a generalised coordinate that is subjected to a collision force over a range of motion is graphed.

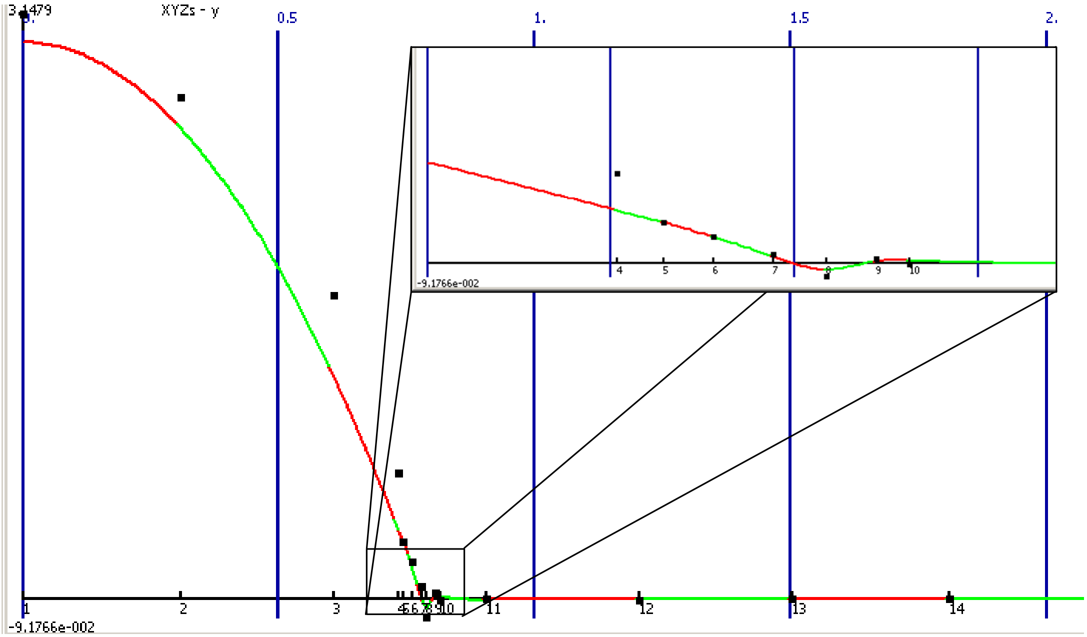


Figure 5.8: Approximated discontinuities using continuous piecewise cubic B-Spline curves. The red and green lines represent each odd and even curve segment respectively

Figure 5.8 demonstrates that at the broad level the generalised coordinate follows a smooth path until it encounters the collision. At this point, the coordinate appears to completely lose its velocity and stops, which would be impossible given the broad resolution of the curve. However, in the zoomed in portion of the generalised coordinate about the collision point, is can be seen that the curve has been refined. This allows the curve to change direction more quickly with respect to time, yet still being represented as a continuous function, thus giving the non-zoomed version the appearance of a discontinuity.

Having demonstrated that it is possible to simulate what appear to be time-based discontinuities within the optimisation process, while still maintaining C^2 continuous curves over the complete motion, the assumption of being able to model impulse forces is considered. In the continued discussion, a technique is presented which has been successfully utilised in the practical implementation.

The simple case of an object falling under gravity and hitting the ground is first considered. The interaction between the object and floor as they impact causes a force to be exerted back onto the object to prevent it penetrating the floor. As the solution for the motion is being computed over time, it is not possible to examining each frame and determining whether a collision is taking place and adjust the acceleration of the object accordingly. Instead, the reaction is defined as a generalised force that is defined over the complete time range. Consequently, this becomes part of the Lagrange equation of motion and hence a system constraint and from Equation 5.32, continuous first order differentiability is required. Additionally, constraints need to be placed on the force to specify when it can be non-zero.

Using discrete time, the force can be constrained using a test clause formulation as given in Equation 5.49, where F is the generalised force exerted by the floor on the object with a reaction force, R (the y-axis is taken to be the height axis). Without loss of generality, the reaction force is presented as an abstract value. The reaction force of Equation 5.49 has been further extended from the point of impact to encompass cases where the object is below the floor and hence only give two clauses. This is not an unreasonable classification because in reality it would be impossible for the object to be inside the solid floor having fallen from above it, and hence this measure is undefined.

$$F(t) = \begin{cases} 0 & \text{if } \; y(t) > floorlevel(x(t), z(t)) \\ R(t) & \text{otherwise} \end{cases} \tag{5.49}$$

When a collision occurs between the object and floor, the reaction force, *R*, will be very high so that it can instantaneously stop the object moving any further along the y-axis, which results in a large jump in magnitude of *F*, in an infinitesimal period of time. This can be partly approximated using a highly refined knot sequence at the point of impact, however the differential is still discontinuous as too is the original function given in Equation 5.49. The importance of not having discontinuities in the system is that they introduce differentials that tend to infinity at the point of change, which causes instability within the optimisation algorithm, thus diverging away from a solution.

The solution that is presented to resolve this issue is to smooth over any such discrete functions with a continuous function that closely approximates the original discrete function. This is illustrated in Figure 5.9, where Equation 5.49 (blue line) is overlaid with an appropriate continuous function (purple curve).

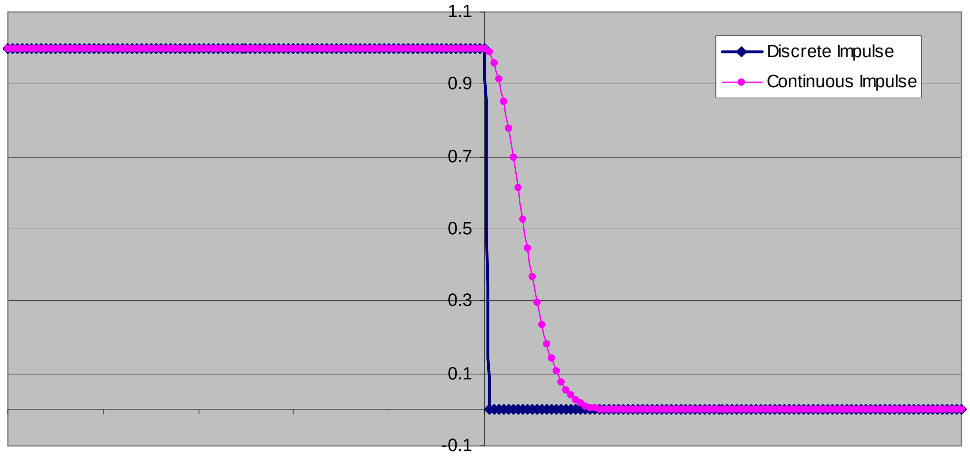


Figure 5.9: Smoothing over discontinuities

The smoothing function used in Figure 5.9 is based on an exponential function, given in Equation 5.50, where k is a scaling constant that can be used to adjust the span of the non-zero portion of the exponential equation and hence control the rate at which the impulse force comes in.

$$F'(t) = \begin{cases} R(t)e^{-\frac{d^2}{k}} & \text{if } \; y(t) > floorlevel(x(t), z(t)) \\ R(t) & \text{otherwise} \end{cases} \tag{5.50}$$

where $d = y(t) - floorlevel(x(t), z(t))$

The first term in Equation 5.50 provides the smoothed function increase on the right hand side of Figure 5.9, while the "otherwise" term provides the left hand side component. Without the test clause in Equation 5.50, the exponential function would symmetrically decrease on the left hand side as it does on the right hand side of Figure 5.9. Equation 5.50 is now completely continuous as too is its first differential. To prove this proposition, *F'(t)* is first shown to be continuous. Both clauses in Equation 5.50 are described with continuous functions and therefore both sides must be continuous in their own right. The part that needs to be shown continuous is the point at which they switch. The domain value at which this occurs is at *y(t)=floorlevel(x(t),z(t)).* At this point, *d* is identically equal to zero and consequentially the exponential component equals one. The result of *F'(t)* at this point for each clause is given by *R(t)* and hence the complete function must be continuous.

Equation 5.50 is now shown to be continuous in its first differential. When Equation 5.50 is differentiated it is only the partial derivatives with respect to the generalised coordinates that are of interested and not the time derivates. Therefore, differentiating Equation 5.50 with respect to anything other than *y* and *R*, will result in zero in both clauses. Equations 5.51a and 5.51b illustrate the result of differentiating Equation 5.50 with respect to *y* and *R* respectively.

$$\frac{F'(t)}{\partial y(t)} = \begin{cases} {}^{-2d}\!/\!_{k}\, R(t) e^{-\frac{d^2}{k}} & if\ \ y(t) > floorlevel(x(t), z(t)) \\ 0 & otherwise \end{cases} \qquad \textbf{(5.51a)}$$

$$\frac{F'(t)}{\partial R(t)} = \begin{cases} e^{-\frac{d^2}{k}} & if\ \ y(t) > floorlevel(x(t), z(t)) \\ 1 & otherwise \end{cases} \qquad \textbf{(5.51b)}$$

As the formulations of Equation 5.51 show, both clauses in the equations are continuous. Evaluating Equation 5.51a at *d=0* results in the first clause being identically zero, which corresponds to that of the second clause and hence proves continuity in this case. In Equation 5.51b, evaluating the first clause at *d=0* gives a result of 1, which again matches that of the second clause.

This proves that the original smoothing function of Equation 5.50 meets the continuity requirements and also provides a suitable approximation to discrete impulse forces. Furthermore, around the area of discontinuity, gradient information is present that allows the optimisation process to better formulate an appropriate step vector. This is due to the increased stability in such regions and that discrete events are being modelled over time, which suits the continuous nature of the generalised curves and hence gives less error in the constraints.

The formulation of Equation 5.50 extends the reaction force into the undefined space where the object exists inside the floor. The utilisation of this redundancy within the system is used with the foresight that the Lagrange's equations of motion are sampled over time to eliminate the parameterisation variable. Therefore, if the contact force of the floor is only turned on at the point of impact, i.e. *y(t)=floorlevel(x(t),z(t))*, there would be potential for jumping over that point completely. Furthermore, it is unlikely that the object can be pushed back to the floor level at a single point in time and hence require a few, highly refined curve segments to achieve this, as illustrated in Figure 5.8. Consequently, the force would have to exist when *y(t)* is strictly less than the floor level as well as equally it.

The above example only demonstrates a collision that is bounded below. The following discussion extends these definitions to support a collision that is bounded above and below. This is achieved by using exponential functions either each side of the spatial region to represent a smooth traversal towards the collision force from both ends, as illustrate in Figure 5.10. This is accomplished by adding in an extra test clause into Equation 5.50.

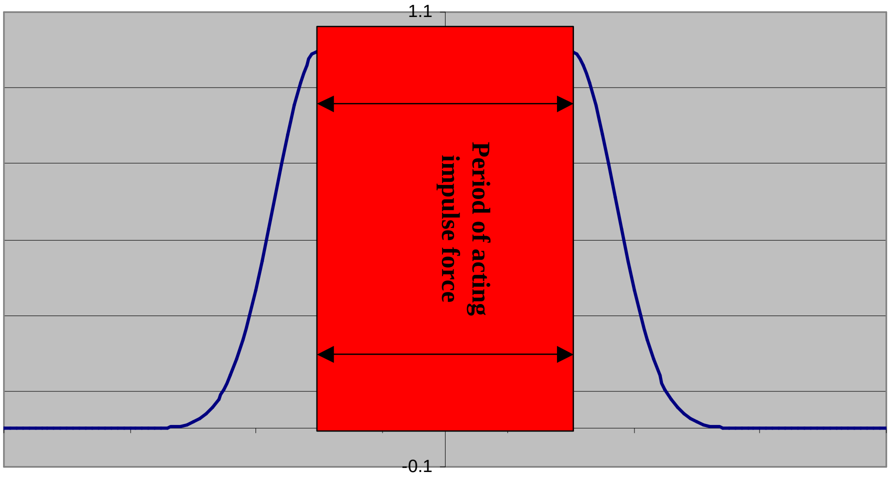


Figure 5.10: Continuous modelling of a discrete impulse, bounded above and below

To conclude the discussions on modelling discrete events in a continuous environment, the reaction force is considered. The average impulse force can be exactly determined based on the change in velocity of the object at the point of impact, *Dv*, and the time step to the next sample, *Dt*, as given in Equation 5.52.

$$R(t) = m\frac{\Delta v}{\Delta t} \qquad \textbf{(5.52)}$$

However an exact representation sets up a paradox within the dynamics representation. The change in acceleration of the object is proportional to *R(t).* Therefore, using Equation 5.1 the change in acceleration can be directly express through Equation 5.53.

$$R(t) = m\frac{d^2y(t)}{dt^2} \Rightarrow m\frac{\Delta v}{\Delta t} = m\frac{d^2y(t)}{dt^2} \Rightarrow -\frac{dy(t)}{dt} = d\Delta\frac{d^2y(t)}{dt^2} \qquad \textbf{(5.53)}$$

Equation 5.53 is a tightly coupled relationship between the velocity of the object and its acceleration. Therefore any change in the acceleration will naturally loop back and result in a change in velocity at that point in time, consequently changing the time of impact and thus setting up the paradox. Even if the velocity value were taken from the sample before and used in Equation 5.53, since the curve is C^2 continuous, the resulting local influence in the region would still have an effect on acceleration.

A mathematical representation of the reaction force, similar to Equation 5.52, would therefore prove impractical. Subsequently, as opposed to directly expressing the magnitude of the reaction force, it is encoded as a new generalised coordinate/free variables, which the optimisation process can determine the values. However, a minimisation function is included for each reaction force within the objective function so that over-elastic collisions do not result.

5.4.5 Friction Forces

In addition to handling discrete impulse forces within the dynamics representation, a model of friction is also included. Friction is a difficult force to model, even in a per-frame setting [Bara91]. Consequently the friction model presented here is a simplified view of what really happens. However, as the results demonstrate, it is a model that works well in the dynamics representation of this thesis.

A frictional force is either termed static or kinetic depending on whether the body is at rest or moving, respectively. Static friction cancels out externally exerted force on a body and hence the object remains in rest. When the exerted force is large enough, the static friction is overcome and the object starts to move which subsequently incurs kinetic friction, which is constant with respect to the two contacting surfaces and gradient. This is illustrated in Figure 5.11, where exerted force is graphed against the reaction force of friction. Figure 5.11 also shows that once a force exceeds the static friction, the reaction force of the kinetic friction is slightly lower than the peak of the static friction, which is why objects are always easier to keep moving rather than to start it off.

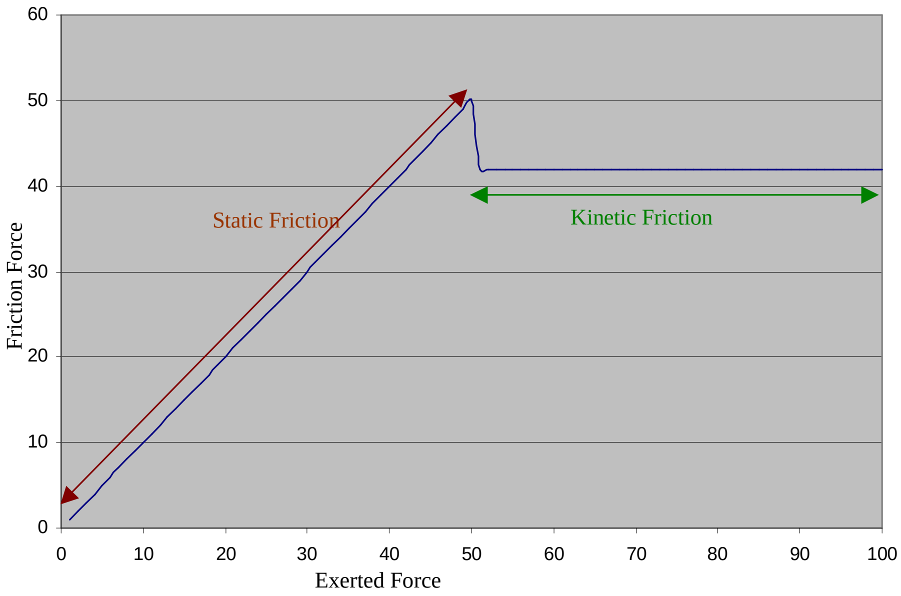


Figure 5.11: Transition between static and kinetic friction

Kinetic friction is defined using a relationship between the normal force of the object and a coefficient of friction factor that defines the roughness between the two surfaces. This relationship is given in Equation 5.54, where *N* is the normal reaction force, *F* is the kinetic friction force and *m* specifies the coefficient of friction between the two surfaces.

$$F = mN \tag{5.54}$$

Some common coefficients of friction are given in Appendix B.2 for reference when animating a human character. The source for the friction that is of interest in this thesis mainly comes from the feet interacting with the floor and hence for much of the implementation the rubber on concrete coefficient, a value of 0.8, is used.

The manner in which the dynamics system is represented means it is somewhat adverse to instantaneous changes. Therefore in the friction model the discrete change between static and kinetic friction is removed so that static friction has a maximum value determined by the kinetic friction. Consequently, the frictional force is modelled using an if-else clause given by Equation 5.55, where *v* is the velocity of the body, **m** is its mass and *A* is the lateral acceleration on it.

$$F(t) = \begin{cases} -\dfrac{v}{\|v\|} mN & if\ v > 0 \\ -\min(mA, \dfrac{v}{\|v\|} mN) & otherwise \end{cases} \tag{5.55}$$

Since the frictional force exists as a generalised force that is always present in Lagrange's equations of motion, to remove the problem of the system exerting a frictional force when two objects are not in contact, Equation 5.55 is multiplying by the contact exponential equation from Equation 5.50. Consequently, the resulting frictional force can only be recognised if contact is being made.

The technique proposed in this section to represent friction differs from the similar dynamics work of Liu *et al.* [2005]. In their work they formulate the Coulomb's static friction model as a function of predefined positional constraints, which represents a foot plant. Therefore the model presented here is more generic and works in conjunction with other subsystem of the dynamics formulation such as the automatically determined contact model.

5.4.6 Muscle Forces

To conclude the discussions on generalised forces, the muscle constructs that are used to drive the active changes in joint angles of the character are consider. The muscle model used is given in Equation 5.56, where *q* is the current joint angle and S_k and S_p are muscle gains and rest states respectively. The generalised force that subsequently feeds back into the Lagrange's equations of motion of Equation 5.5 is given in Equation 5.57.

$$F_M = S_k (q - S_p) \tag{5.56}$$

$$Q_i = \sum_j F_{m,j} \frac{\partial q_j}{\partial q_i} \tag{5.57}$$

The generalised force of Equation 5.57 has been previously used to approximate the behaviour of real muscles [Liu05, Witk88], where each interconnecting rotational joint in the rigid body representation of the character has a generalised muscle force of this form. The muscle gains and rest state variables are variable over an animation to approximate the behaviour of actual muscles and therefore are represented within the system as free variables and hence they have their own piecewise cubic B-Spline curve. With the exception of imposing inequality limits on the muscle gains rest state variables, the generalised muscle's free variables are left open to be determined by the optimisation process. The imposed inequality limits on the muscle gains establishes the maximum exertable force by the generalised force, whereas the rest state restrictions mimic those of the rotational joint that the muscle is for.

5.5 Practical Considerations

In this section, a few underlying practical considerations making up the dynamics modification program, named *Skeleton* are discussed. The initial focus will be to consider the representation of the mathematical constructs using a subsystem that is specifically developed for this application. However, it provides a general technique of representing and modifying mathematical equations and as such it is also rolled-out in the inverse skinning application of section 2.3. The system is called Polynomial Algebraic Manipulation & Polynomial Expression Representation System, or PAMPERS for short.

Thereafter, a description on the formulation of the constraint and minimisation functions of the optimisation process is given. Appendix C gives a brief description on several smaller, practical considerations such as matrix inversion, how the iterative optimisation-based solver is initiated and terminated to produce a suitable simulated motion, and the optimisation of the curve refinement process.

5.5.1 PAMPERS: Polynomial Algebraic Manipulation & Polynomial Expression Representation System

The mathematical equations that are used to represent the environmental and character dynamics are large and complex, especially as the number of DOFs increases. Consequently, it is desirable to have a flexible yet efficient manner of representing the formulations. There are various numerical/symbolic packages available, such as Maple[11], however the system presented in this thesis utilise a system called PAMPERS, which has been constructed to better suit the demands of this work, for example to define customised constructs more easily, such as contact and friction models and to extract common sub-expressions across multiple equations.

Since the aim of the dynamics system is to produce a method for dealing with generic articulations and dynamical constructs, it is not suitable to hardcode equations because the model would only allow instances of that specific structure, i.e. if a system to encapsulate 3 links were hardcoded, this would only allow models of exactly 3 linked systems and nothing else. Therefore, it is desirable to symbolically represent the mathematical constructs so that they can be constructed at runtime to exactly model the specific scenario that is being considering. The consequence of generating the equations at runtime is that they are unlikely to drop-out nicely optimised for later evaluation or manipulation. Therefore, to facilitate the efficient optimisation of the runtime-generated equations, two concepts are introduced – simplification and sub-expression representation.

Equation simplification takes an initial equation and attempts to reduce the number of operations required to evaluate the expression. Existing programs such as Maple do this process extremely well and it is conceded that the level of simplification in PAMPERS is very much at an infancy stage in comparison. This does however provide future scope for better optimising this technique.

Sub-expression representation is similar to simplification in that the original equation is reduced down to as few operations as possible. However, this is achieved by recognising similar sub-expressions and representing these as auxiliary equations that are evaluated first. For example, Equation 5.58 can be simplified to Equation 5.59[12].

$$y = (x+3)^2 + \frac{x(x+3)^2}{2h} \tag{5.58}$$

$$y = \frac{(x+3)^2(2h+x)}{2h} \tag{5.59}$$

However, Equation 5.58 can equally be represented through the use of sub-expression representation as illustrated in Equation 5.60, where a common sub-expression is extracted as an auxiliary equation, which is evaluated only once.

[11] Maple 7, Maplesoft, www.maplesoft.com

[12] This simplification is obtained using the inbuilt Maple 7 simplification command

$$y = z + \frac{xz}{2h} \quad \text{where} \quad z = (x+3)^2 \qquad \textbf{(5.60)}$$

By considering the number of operations required to evaluate the equations, it takes 8 steps for Equation 5.58, 7 steps for Equation 5.59 and 6 steps for Equation 5.60. What has been gained in this example is fewer steps for the sub-expression equation compared to both the original and simplified versions. Furthermore, computational time has been saved in performing a symbolic optimisation process to obtain the simplified version by keeping the exact same format of the original equation and substituting the auxiliary equation into the appropriate place.

As the equations increase in complexity in the dynamic representations, the process of sub-expression representation becomes a very attractive technique to reducing the overall complexity of the system. This process is something that Cohen [Cohe92] discusses was lacking from his spacetime constraints implementation and would have served a useful optimisation process. Liu *et al.* [Liu05] mention their use of sub-expression representation in a spacetime constraints formulation but stop at describing the degree to which this is implemented.

Maple can produce sub-expression representation in the form of C code however like all other symbolic manipulation applications, it only considers a single equation when it does this. In contrast, PAMPERS performs sub-expression extraction over multiple equations.

The sub-expressions (or BaseCons as they are termed in PAMPERS) are maintained inline with respect to the equations. They only need to be fully evaluated on their first occurrence within the set of equations after the free variables of the system have changed. The mechanism used to achieve this is to assign each BaseCon a byte and bitmask that identify a unique bit per BaseCon. When this bit is set to 0, the BaseCon needs to evaluate its sub-expression, which is stored for future use. The BaseCon's bit is set to 1 and the calculated result returned. If when it comes to evaluating the BaseCon and the bit is set to 1, it returns the result it last computed and stored. The only time that the cached BaseCon value may differ from the actual sub-expression evaluation is when the free variables of the system are updated at the end of each iteration. At this point, a linked-list array of bytes, which are the original bytes passed into the BaseCons, are reset back to 0, thereby telling each BaseCon that it needs to be recomputed when it is next evaluated as opposed to using the cached value.

The advantage of using such a procedure to encapsulate the BaseCons is the ability not to have to worry about what order the sub-expressions need to be evaluated in. Therefore, BaseCons can nest inside BaseCons up to an arbitrary level and complexity, which can always be evaluated in the correct order. A further gain in using this technique is that BaseCons are only evaluated if they are needed. For example, to determine if the calculated step size in the optimisation process needs to be reduced, only the BaseCons involved in constraint functions are recalculated as opposed to all BaseCons.

BaseCons are determined semi-automatically and there is still scope for further improvements with regards to detecting common sub-expressions. The primary source of BaseCons come from the application of differentiation and integration within PAMPERS because in many cases, when taking partial derivatives of complex equations, there will often be constant components with respect to the differentiating variable, although non-constant with respect to the system's free variables. At the lower granularity of the equations, each evaluated parameterisation of the piecewise cubic B-Spline curves is represented as BaseCons, as too are all trigonometric functions, which greatly reduces the execution time. A quantitative measure of the performance increase using BaseCons was given using the earlier inverse skinning algorithm of section 2.3. By allowing PAMPERS to use trigonometric BaseCons we achieve a reduction in computational time of approximately 44%[13] over preventing it from using them.

In addition to reducing the computational load of evaluating equations via common global (and local) sub-expressions, PAMPERS can also define custom mathematical operations or entities, such as contact forces. New, single operands are created that model the 2- or 3-way if-else clauses defined in section 5.1 for contact and friction forces, including their differentials. This allows more complex expressions to be treated as if they were simple operands, which consequently fits neatly into the complete system representation, also providing easier manipulation.

Beside the key features outline, PAMPERS has also been programmed with the ability to integrate & differentiate functions with respect to arbitrary constructs and perform substitutions. Therefore it is possible to start off with the abstract rigid body representations and then substitute the generalised coordinate placeholders with the actual parameterise piecewise cubic B-Spline curves. As an interesting aside, one further bonus PAMPERS gives over Maple is that when it comes to differentiating the Lagrangian equations of motion with respect to partial time-based derivatives (i.e. the *dx(t), dy(t), dz(t),* etc) Maple is unable to directly handle this and first requires a substitution between this symbolic representation to eliminate the time factor (i.e. *x(t)->x*). Once the substitution has been done, the differentiation can be achieved and the time-based factors put back in with a second substitution that reverses the original. This is presumably because differentiation can only be done with respect to a basic symbol as opposed to a compound one within Maple. However with PAMPERS it is possible to differentiate with respect to compound expressions and therefore eradicate the extra substitution steps; it is important to maintain the time-based parameterisation of the equations, because they need to differentiate with respect to time as well as the free variables.

PAMPERS provides a good basis from which to build the dynamics optimisation algorithm as it allows the rigid body equations to be easily built at runtime which can be automatically differentiated and integrated down to the constraint and minimisation functions. However, since PAMPERS is only a means to an end as opposed to the central focus of this work, there is still a great deal of scope to improve it. For example, it does not have the full simplification power of established packages such as Maple and the sub-expression extraction could do with more work to identify further common sub-expressions. Nevertheless, PAMPERS does serve its purpose and allows general equations to be constructed while reducing costs in terms of both storage requirements and evaluation time through the techniques described above. Its generality has been demonstrated by its enrolment in both the inverse skinning technique and the dynamics application, *Skeleton*, which is demonstrated in Chapter 6.

[13] The decrease in computational time is derived using the same marker data for a 367 frame motion

5.5.2 Incarnating the Mathematical Constructs: The Constraint and Minimisation Functions

The dynamics-based algorithm (called *Skeleton*) used to perform the motion modifications that are demonstrated in the Chapter 6 are somewhat complex in nature (and rather large), however they all follow a basic underlying structure, which is summarise here. During the results of Chapter 6, any modifications to this basic process are further elaborate on.

At the top-level, the system consists of two large sets of equations in terms of constraints and minimisation functions. When *Skeleton* solves the optimisation process of Equation 5.18 using the mathematical formulations of Equation 5.31 and Equation 5.32, for a single step, all active constraints are processed in the same manner, which is similarly true for any minimisation function. However, for conceptual ease, the complete set of constraints is segmented into three different classifications, which have some bearing on how they are derived. These are termed core, user and task constraints. User constraints are defined by the user to impose additional restrictions on the resulting motion, usually in the form of time-based, and are very dependent on the application of the algorithm. Therefore any user constraints used in the generation of the motions are highlighted during the results section of Chapter 6. The other two constraint concepts are discussed in the following subsection, which is followed by a review of the minimisation function.

5.5.2.1 Core and Task Constraints

The core constraints are constructed from the rigid dynamics representation of the character using Lagrange’s equations of motion (Equation 5.5), which is expressed using open chained kinetic (Equation 5.13) and potential (Equation 5.14) rigid body energies and system generalised forces (Equation 5.10). The Lagrange’s equations of motion (section 5.1) are constructed and represented using PAMPERS. The parameterisations of the Lagrangian equations are instantiated using a system of piecewise-cubic B-Spline curves (Equation 5.38) whose knot spacing is initially set at approximately 0.2 second intervals. To remove the time-based parameterisation of the piecewise-cubic B-Spline, two samples are taken per curve segment. This provides the ability to accurately control the first and second differentials of the system’s free variables, i.e. velocity and acceleration.

Task constraints consist of formulations that impose restrictions on the kinematics of the system as opposed to representing any new dynamical behaviour. These types of constraints can be further segmented into two smaller conceptual components. The first imposes joint restrictions while the second imposes auxiliary kinematics constraints, including guarded constraints. The addition of joint limits complements the core constraints in that between the two it is impossible to configure the character in an impossible posture, whether this is dynamically unrealistic or physically implausible. The joint restrictions are represented by upper and lower bound inequality constraints using an active set implementation (section 5.3.2), where the joint limits are giving in Figure 2.14. Furthermore, defined as part of the joint restriction task constraints, the contact force models are complemented by specifying minimal spatial constraints, for example to state that objects should exist above the floor level.

Guard constraints are similar to inequality constraints in that they can be either active or not, however unlike the regular inequality constraints used for the joint limits, the test for activity is not actually the invoked constraint. For example, recall from the earlier discussions on inequality constraints from section 5.3.2 an inequality constraint can be represented using a formulation similar to Equation 5.33 (which is repeated here).

$$c_i(S) \geq 0 \qquad \text{(from Equation 5.33)}$$

At any given step in the iterative optimisation process, if Equation 5.33 is evaluated such that $c_i(S)$ is greater than zero, then the constraint is not active. However, if the constraint evaluation yields a result equal or less then zero, the constraint would be active and into the constraint set would go the equality version of 5.33, i.e. $c_i(S)=0$. In the guarded constraints paradigm, there is still a test equation similar to Equation 5.33, which is called the guard, however there is also a completely separate constraint equation whose active state is determined by whether the associated guard equation is less than or equal to zero. This establishes in a cause and effect type of constraint.

Guard constraints form an important role within *Skeleton*, which is illustrated by considering the object-floor contact. When interactions occur, from Newton's 3 Law of Motion, every action has a reaction. Section 5.4 discussed the plausibility of representing such reactions as free variables, however it is still desirable to impose some constraints on them. It would follow that if an object is not in contact with the floor then there cannot be a reaction force. This type of constraint cannot be encapsulated by a standard inequality constraint and hence requires a guarded constraint that models the cause and effect (or in this case, no cause and no effect). In addition to the lack of reaction force exerted by the floor during a period of no interaction, an upper limit is also placed on the reaction force, derived from Equation 5.52, when contact has been make. This prevents the floor from ejecting the object from its surface, although a reaction force is also included in the objective function to alleviate this potential problem.

5.5.2.2 Minimisation Function

Unless exactly described, the simulation of motion is a very open-ended task, especially when considering complex scenes such as articulated characters, because of the high number of DOFs that need to be described. Therefore some kind of control structure needs to be used that guides the optimisation process to a desirable result. This is the role of the minimisation function.

Control processes have been well recognised in the past as a means for controlling the per-frame dynamic character animations [Brog98, Hodg95, Hodg96, Rail91, Woot95] and indeed, Chapter 4 presented several control processes used during the inverse kinematics motion modification techniques. However, there is no control process that performs every kind of modification tasks. For example, one possible measure of performance is that of energy conservation [Witk88], which works fine for simple objects, such as rocket ships, but is less suited for character animation without additional support. Therefore, a more defined structure within the minimisation function is necessary.

The control processes that are enforced to produce the results in Chapter 6 see the minimisation function become a three-fold process. The first is to minimise the muscle forces exerted by the character, which takes the form of minimising the squared energy exerted be each muscle, given in Equation 5.56, integrated over time. To compensate for the shortcomings of energy minimisation, the second prong of the minimisation function consists of reducing the squared DOF distance to an example (or base) motion. By doing so, a control structure is imposed that drives the character along a similar path to the base motion thereby enforcing the simulation to behave in a similar manner. As a third component of the minimisation function, error and reaction force minimisation is included, as previously described in section 5.4. These terms are essential to the formulation of the constraint terms and hence operate at a much more fundamental level than the other two components and as such they are largely ignored and unseen at the higher-levels. Hence the minimisation process is conceptually considered as consisting of only two parts – muscle and example motion minimisation.

The control process in *Skeleton* requires an existing motion to be present to guide the solution so in one sense the approach is not going to be a general method for modifying any motion to any other, although that can be said of all the motion editing techniques that have been considered in this thesis. It is however more general than the works of Liu & Popovic [Liu02] and Abe *et al.* [Abe04] as they abstract parts of the body away and consequently use more basic point-mass as opposed to rigid body dynamics. It is also slightly more general than the technique presented by Sulejmanpasic & Popovic [Sule05], who only considered parts of the motion where the body was in flight and not in contact with the floor. In terms of completeness and generality, the work that is presented in this thesis is on a par with that demonstrated by Liu *et al.* [Liu05], however diverges in significant key areas, which have already been and continue to be explored during the results of Chapter 6. A couple of conceptual differences have already been noted, including the manner contact and friction forces are handled.

5.6 Summary

The culmination of this chapter results in a “rigid body dynamics-based optimisation process for modifying motion capture through consideration of a complete time span at once” called *Skeleton*. The optimisation component of Skeleton is based on a sequential quadratic programming algorithm (section 5.3), whose constraints are primarily the encoding of the connected rigid body dynamics that represent a humanoid character (section 5.1). The objective function of the optimisation process is formulated from motion capture data and energy minimisation (section 5.5). The optimisation process simulates a complete time span of motion based on the piecewise cubic B-Spline representation used (section 5.3).

However, the amassing of these components into one coherent, working system brings with it some fundamental conflicts, such as ill-resolutioned equations and modelling discrete occurrences in a continuous domain. These issues were discussed in section 5.4, using the mathematical equations from section 5.1 and section 5.3, and novel remedies presented.

Using the extra practical considerations outlined in section 5.5, which pertain to the higher-level operation of *Skeleton*, the results of applying the mathematical systems, described in this chapter, to modify existing motion capture data is presented in Chapter 6.

Chapter 6:
Dynamics-Based Motion Capture Modification

This chapter presents the results of applying the dynamics-based optimisation process, described in Chapter 5, to modify existing motion capture data. The use of dynamics to modify existing motions can be broken down into two separate processes. The first process is to construct the equations that will be used to represent the dynamics of the character and environment. The hierarchical structure for the character this is taken directly from the example motion data file. This first process includes ensuring that only valid DOFs are represented in the collection of free variables, thus reducing the overall system complexity (see Figure 2.14 for a description of the valid DOFs and their ranges). The second step is the iterative evaluation of these constructs to produce a valid motion that satisfies the constraints in the most optimal way according to the objective function. All of the times relating to these two stages in this chapter are based on running the process on a Pentium 4, 1.4GHz processor with 768MB of RAM.

The first section of this chapter, section 6.1, discusses the application of the dynamics-based process to retarget motions to characters with different sized limbs. Technically, retargetting is purely a kinematics-based problem and hence there is no need for a dynamics-based approach. However, the additional benefit of including the dynamic formulations during retargetting is to ensure that any modifications are still physically plausible and not just kinematically correct. This can eliminate some of the previously needed high-level control processes of other approaches. As section 6.1 demonstrates, the use of the dynamics-based representation in this thesis provides a novel technique for automatically cleaning up foot sliding because of its lack of high-level control processes, relying purely on the dynamics. In addition to demonstrating the retargetting process, section 6.1 discusses some of the runtime considerations associated with the dynamics-based process. The attributes that are addressed in this section are subsequently assumed during the further examples presented in this chapter.

In section 6.2, the motion modifications that are considered maintain the limb lengths between the example motion and simulated motion but change other biomechanical properties to produce individualised movements. Due to the need of biomechanical information to perform such modifications, a purposely captured dataset of 4 different actors performing similar motions has been recorded for use in this thesis. This data was described in Chapter 3. Furthermore, in section 6.2 a correlation between the weighted inverse kinematics from Chapter 4 and the biomechanical effect of changing muscle masses is discussed. This correlation provides support for the cheaper IK-based character reconfiguration process beyond the evaluation performed in Chapter 4.

A complete reconfiguration of motion capture data is demonstrated in section 6.3 where the whole biomechanical information of a target actor is used (i.e. limb lengths, masses and muscle strengths are included). This section demonstrates how the dynamics-based modification process can transfer the motion of one actor to a completely different actor using only the target actor's biomechanical information. This is a novel contribution to the area of character motion modification and it is evaluated using the motion capture dataset from Chapter 3. The reconfiguration process is novel because what little work that has been done in this area before has relied on sampling an existing piece of motion capture data from the target actor before new motions can be generated in their style [Liu05]; this work only needs the biomechanical information of the actor and not a sample motion from them. The evaluation of the dynamics-based reconfiguration process in this thesis is also novel because in previous works no one has actually compared their simulated motions against multiple motions of the real target actors.

The final set of reconfigured motions that are presented in section 6.4 look at how the dynamics process can be harnessed to simulate movements that portray injuries in motions that previously had no injuries. These motions are generated by exaggerating the biomechanical properties that are discussed in section 6.2 and partly evaluated against an imaginary limping motion from the actors motion capture dataset of Chapter 3.

This chapter is concluded with a summary of the dynamics-based reconfiguration process in section 6.5 and a discussion in section 6.6 on the overall dynamics process. The discussion includes areas of future work that result from the findings and evaluations of the process presented in this chapter.

6.1 Retargetting Character Motions

In this section two retargetting examples are considered. The example motions are the same as those used during the demonstration of half-Jacobian IK retargetting process of Chapter 4 – walking and catching a football. In addition to the new limb lengths, the dynamics-based retargetting technique requires biomechanical information for the new character. This information is based on the measurements taken from a real actor of similar measurements, the details of which are given in Appendix D.1, which also include the limb lengths. A visual representation of the limb sizes and hence a reference for mass, are given as cylinders (as opposed to ellipsoids) where the circumferences proportionally represent the sizes of the actor the motion is being mapped to. The difference in visual appearance is illustrated in Figure 6.1 between the biomechanical dimensions of the target walking character given in Appendix D.1 (the character on the left in Figure 6.1) and the original actor (the character on the right in Figure 6.1), where the taller character has much slimmer limbs than the shorter one, as illustrated by the comparative cylindrical circumferences.

In order to achieve only retargeted motions (as opposed to individualised too), the muscle forces are treated as passive generalised forces, which do not partake in the objective function. However, the muscle forces continue to be present in the core constraints (i.e. in Lagrangian equations of motion).

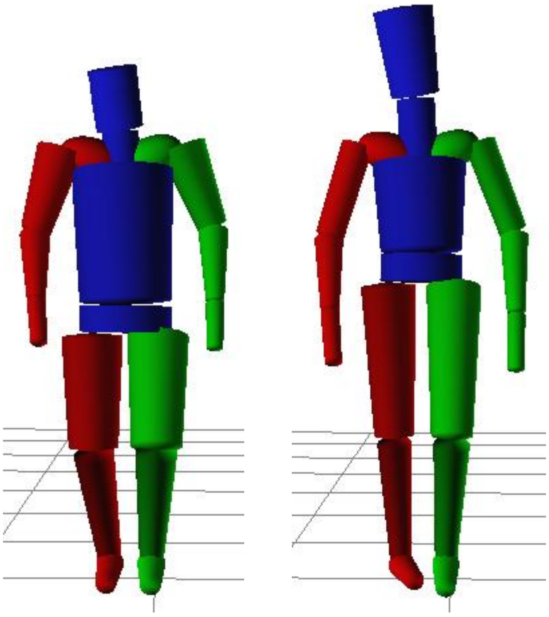


Figure 6.1: Biomechanically different characters represented using appropriately sized cylinders to indicate the limb dimensions and hence their mass

6.1.1 Walking in a Winter-Wonder Land

In this example the walking motion of a character is retargeted, which effectively becomes a process of foot skate cleanup since the upper body does not interact with the environment. The original character motion is illustrated in Figure 6.2.

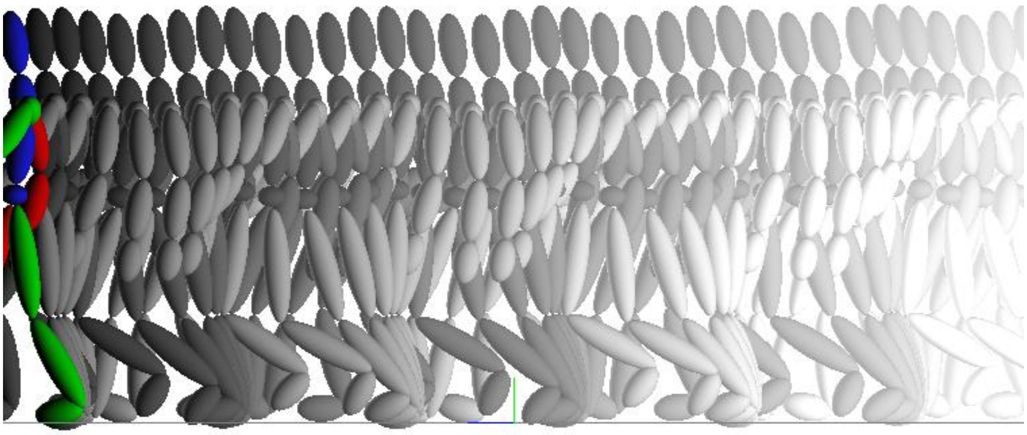


Figure 6.2: Original walking base motion

Before the results of dynamically retargetting the walking character are presented in section 6.1.1.4 a few aspects relating to the process are first addressed in the following subsections. The first of these looks at the control processes required for determining when end-effector locations are favoured over joint trajectories. This is followed in subsection 6.1.1.2 with a discussion on the desirable consequences of using refined knot sequences in certain portions of the motion to better capture rapid occurring events, such as foot impacts. The final subsection, section 6.1.1.3, before the results are given outlines the need for occasional user input into the optimisation process to guide the system to a more desirable motion when a local minima is reached.

6.1.1.1 Determining Joint Trajectories vs. End-Effector Locations – The Control Processes

To maintain natural looking postures when foot plants are not present, the optimisation function consists of minimising the difference between the joint angles of the example and resulting motions. In the case of foot contact with the ground, the plants and contact points are automatically determined and maintained by the dynamics formulations through the collision and friction forces within the core constraints (see section 5.4).

The process of foot skate cleanup is therefore completely encapsulated within the standard application of the dynamics-based technique presented in Chapter 5 – no other higher-level control processes are required. This is contrast to all previous inverse kinematics approaches, including the one presented in Chapter 4, where there is no need for additional control processes that determine what should happen when a foot hits the ground. In effect, the dynamics-based technique has an inherent technique that can automatically determine some importance factors (i.e. when end-effectors locations are preferred over maintaining joint space configurations) as the constraints have higher priority over the minimisation function and will always be solved, even at the potential increase in the minimisation function. This contributes towards making this dynamics-based retargetting process novel in terms of foot skate cleanup, which will be further discussed and compared to other techniques during the dynamics retargetting summary of section 6.1.3.

6.1.1.2 Initiating & Refining the Dynamics Optimisation Algorithm

Figure 6.3a illustrates the upper left leg's local Z-axis rotation from the set of starting values used to initiate the iterative algorithm (see section 5.5.3). By comparing the new motion DOF track to the original one in Figure 6.3a, it can be seen that the default resolution of the B-Splines is not sufficient to accurately model the original DOF trajectory. The left foot contact pattern, which is given in Figure 6.3b, serves to highlight that the local minimum regions of the new DOF curve that are failing to capture the detail of the original DOF curve correlate with foot plants. It is therefore beneficial to refine the curve in these regions to better model the contact regions, which is illustrated in Figure 6.3c by the spacing of the black dots. Figure 6.3c additionally shows a second application of the non-constrained optimisation process to provide better initial values using the refined knot sequence.

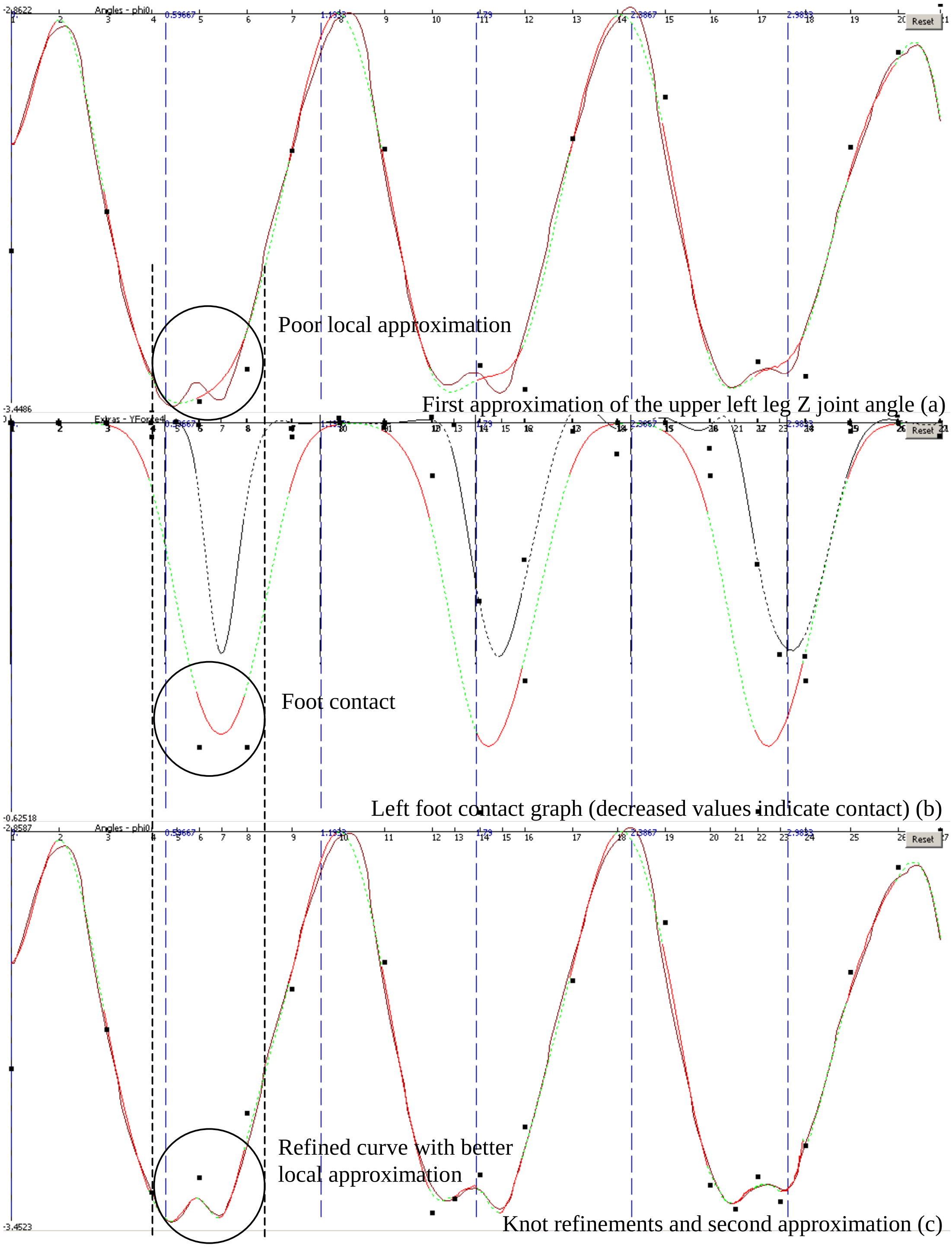


Figure 6.3: Initial value approximation of the example motion's upper left leg Z joint angle. (a) Approximation using an initial uniform knot spacing of 179ms. (b) The left foot contact curve. (c) The non-uniform, refined initial value approximation where the refined contact curve is given in (b) as the black overlaid curve. The solid dark red curve gives the joint path of the example motion and the interchanging light red and green curve paths the piecewise cubic B-Spline curve of the approximated trajectories in (a) & (c)

The determination of curve refinement requires user input to specify where the refinement should take place. This can usually be judged by viewing the first approximation of the initial values (and a little experience of using the optimisation process). For example, since the poorly approximated regions in Figure 6.3a relate to foot plants and due to the sampling process of

Lagrange's equations of motion, it is desirable to have larger sampling frequencies in these regions to more accurately capture the contact behaviour, i.e. the points at which the force becomes active and inactive. This is highlighted in Figure 6.3b where, as well as the uniform contact curve, the refined contact curve is overlaid, which is given by the black line. From the refined curve it can be seen that the contact points span a much narrower period of time, which is due to the better temporal resolution of the curve in these regions.

If the curve were not refined, as illustrated in Figure 6.3, slightly unstable constraint equations could arise around the point of foot contact. Due to the relatively high resolution of the curve it is difficult to capture a representation that correctly has contact and friction on when it should be and the subsequent result can still exhibit foot sliding and pivoting as illustrated in Figure 6.4. Any foot sliding is however only slight in comparison to the scaled version.

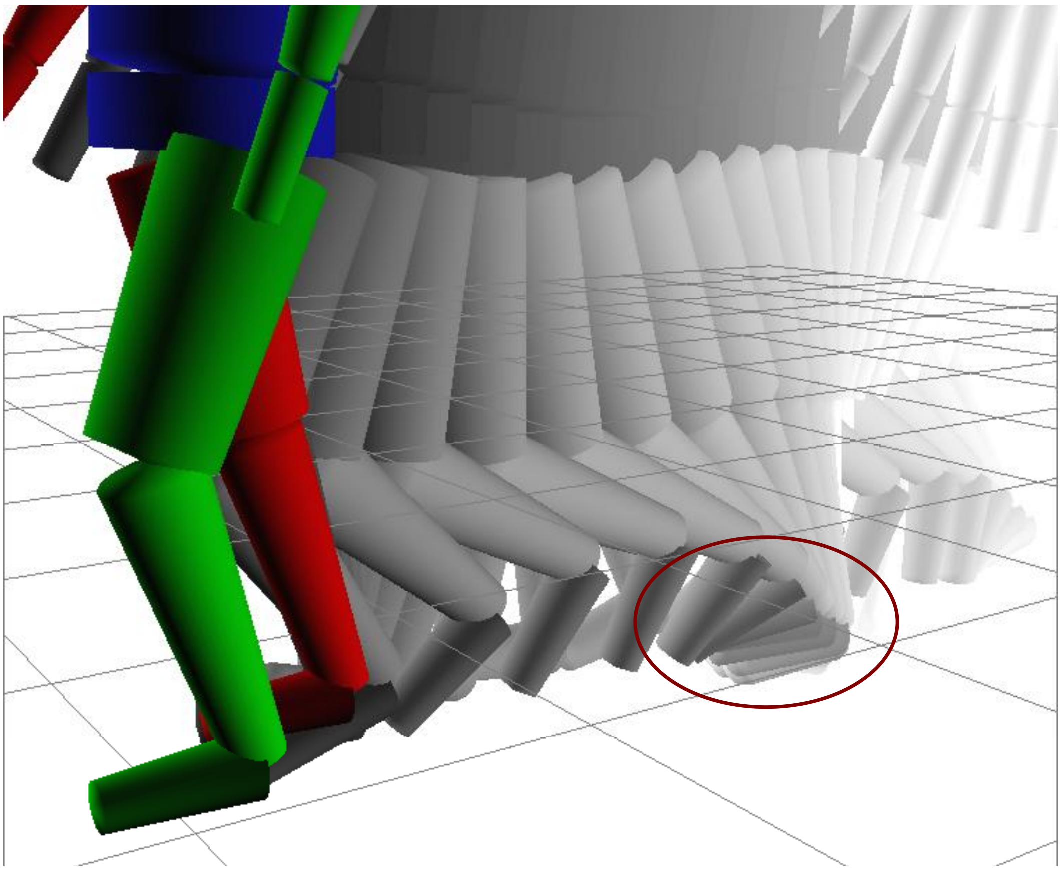


Figure 6.4: Retargetting with a low-resolution uniform piecewise cubic B-Spline curve results in an unstable mathematical representation and hence visual artefacts where in this case the heel is able to pivot on the spot

Figure 6.4 (and more noticeably in the accompanying animation file) illustrates that because of the lack of curve resolution, in the temporal region of a foot plant, the toes (which are highlighted in Figure 6.4) are planted however the heel friction has not been suitably recognised and hence is allowed to pivot around on the spot. By increasing the knot resolution of the DOFs in regions of contact, this problem can be eliminated because the contact force magnitudes can be better defined at specific points in time, as demonstrated in Figure 6.3b. The resulting motion using the refined knot sequence of Figure 6.3 is illustrated in section 6.1.1.4.

6.1.1.3 Animator Guidance for the Optimisation Process

Using the initial values, the optimisation process iteratively solves the system with respect to all constraints enabled and the objective function. In this retargetting example, the process took a total of 20 iterations. However, during the iterative calculations user intervention was required once in this example to guide the solution to a preferable configuration. The cause of this was due to the solver jumping out of a local minimum and subsequently converging towards a different, less suitable, local minimum. Figure 6.5 illustrates the problem encountered where during an iteration, the solver has introduced a rotation of the left leg by approximately 50 degrees. However, because the foot makes contact with the floor, the friction forces prevent it from rotating back to the more natural posture of the minimisation function.

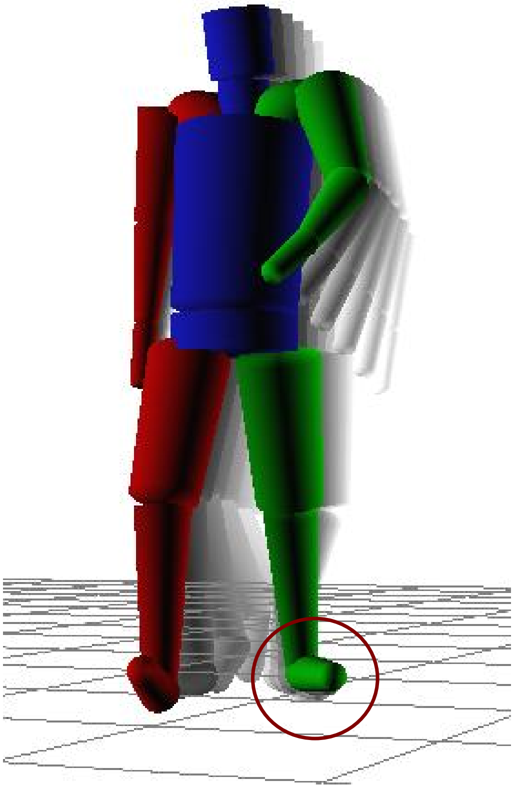


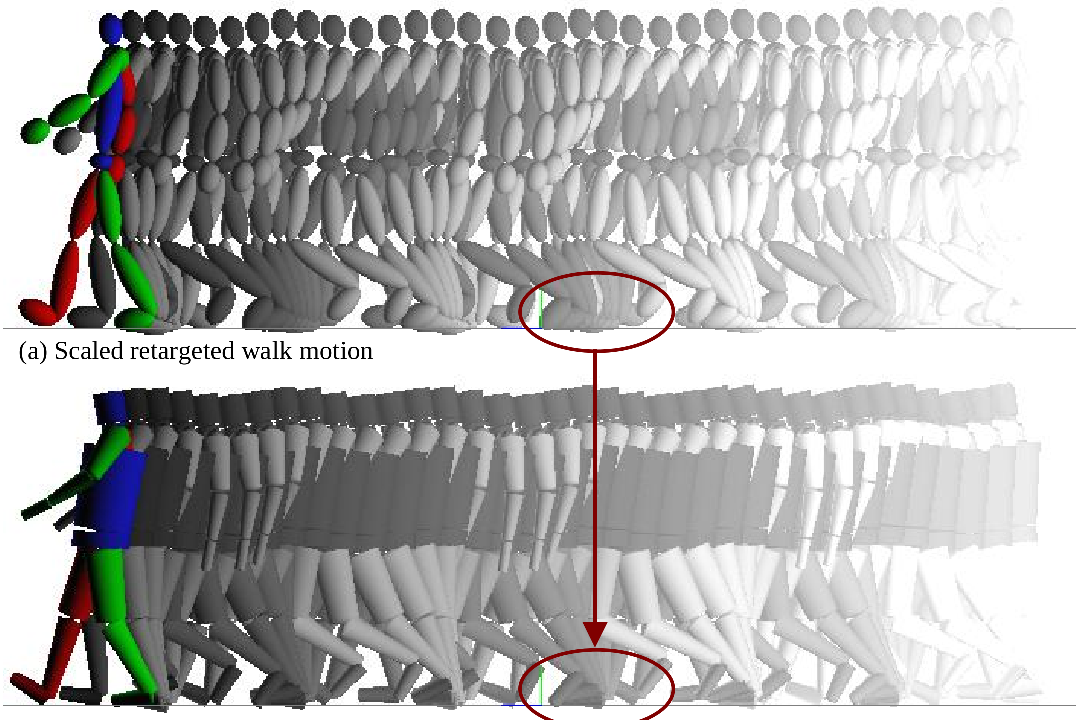
Figure 6.5: Ill-posturing of the character’s left leg which results in the visual appearance of the foot pointing sideways due to the solver jumping between local minima and then being trapped by the friction model

Applying more rigid joint angle restrictions on the offending joint can reduce the specific problem that is illustrated in Figure 6.5. However, the task constraints, which specify the joint limit inequalities, are left at the physical limits and it remains to be the animator’s task to provide additional guidance because in certain circumstances the posture illustrated in Figure 6.5 might be desirable.

The problem illustrated in Figure 6.5 of the optimisation process moving between local minimums can be partly traced to equation instability. Within a small change of system state, constraints can be turned on or off, which modifies the dynamics description of the system and hence can cause the solver to move between the different states. To alleviate this problem and to ensure a stable system state in the neighbourhood of a valid solution, the animator can adjust the joint trajectory, refine the DOF curves and set regions of DOF trajectories as system constraints. The combination of these three approaches has been all the user input required to generate and stabilise (if needed) the dynamics-based simulated motions used in this thesis.

6.1.1.4 Dynamics-Based Retargeted Walk Visual Results

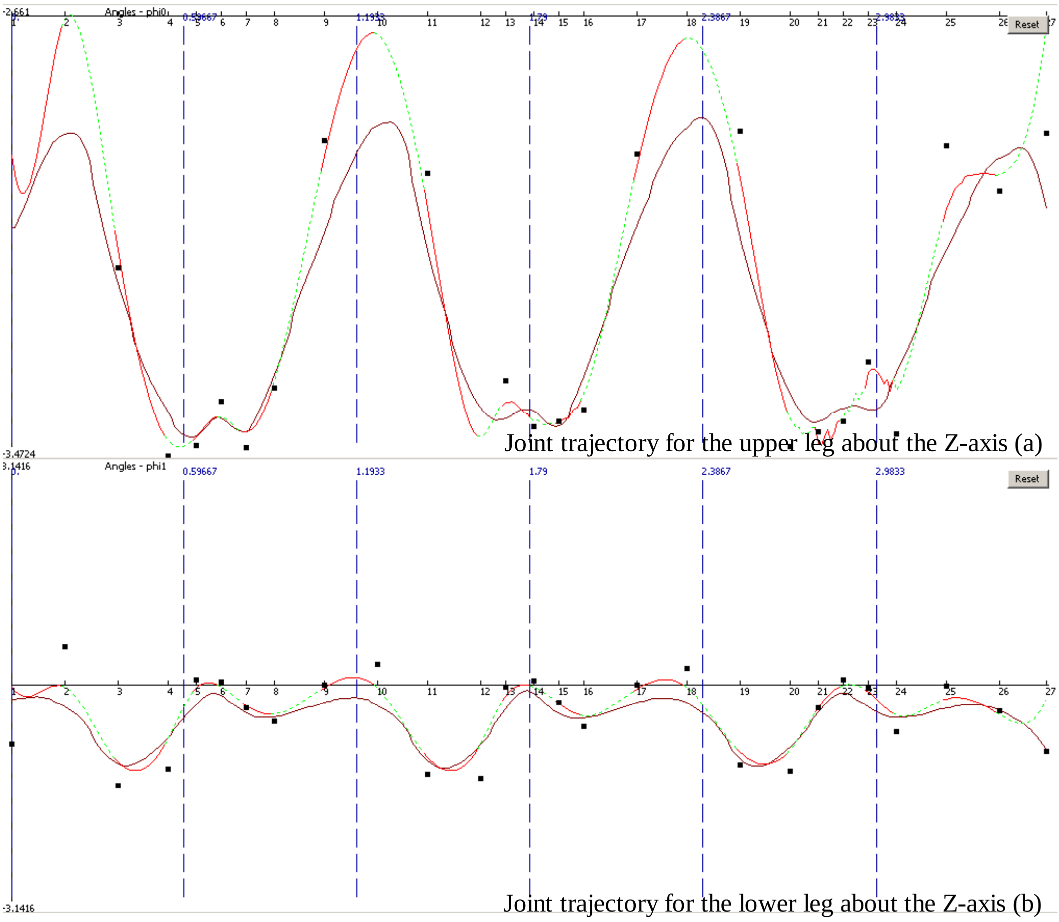
The final result of the dynamic retargetting algorithm is illustrated in Figure 6.6b, where the scaled character is provided in Figure 6.6a to illustrate the degree of foot skating present.


(a) Scaled retargeted walk motion

(b) Dynamically retargeted walk motion

Figure 6.6: Dynamic retargetting of a gait motion

From Figure 6.6b, it can be seen that the spatial foot locations in the dynamically retargeted motion are much closer together when they come in contact with the ground. This translates into good foot planting and hence a successfully retargeted/foot skate clean character. Furthermore, the dynamically retargeted motion of Figure 6.6 exhibits joint smoothness both in leading in and out of the foot plant. This can be seen in Figure 6.7, where the Z-axis joint trajectories of the left leg are graphed over time for the generated motion, which are overlaid upon the joint trajectories of the same DOFs of the example motion.



Joint trajectory for the upper leg about the Z-axis (a)

Joint trajectory for the lower leg about the Z-axis (b)

Figure 6.7: The trajectory of the upper left leg Z-axis joint angle of the retargeted characters of Figure 6.6. The solid dark red curve gives the joint path of the example motion and the interchanging light red and green curve paths the piecewise cubic B-Spline curve of the generated motion

Figure 6.7 further highlights an artefact of the dynamic motion adaptation technique that is apparent in several of the results presented in this thesis. The joint angles graphed in Figure 6.7 show the general path of the new motion closely resembles that of the original trajectory, albeit the dynamically retargeted character has a much larger range of motion to accommodate the change in size of the target character. However, in the first half of the leftmost curve region, the retargeted and original trajectories diverge. The same can also be seen in the second half of the rightmost curve segment of the retargeted joint trajectory in Figure 6.7.

The reason for the divergence seen in Figure 6.7 can be accounted for by considering the mathematical constructs of the system constraints. As section 5.3.3.1 describes, within the middle of the complete curve, each control point is included in 4 consecutive curve segments. However, at the beginning and end of the complete curve, each control point becomes included in progressively fewer curve segments as illustrated in Figure 5.6 and the accompanying formulations of Equation 5.37 and Equation 5.38. Therefore, whereas within the middle of the complete curve, where each control point has well defined pre- and post- states due to neighbouring curve segments, the first and last control points have undefined pre- and post- states respectively. The under-defined nature of the control points gives rise to the artefact seen in Figure 6.7, which is despite instantiating constraint equations at the very start and end points of the time range. However, the resulting motion does not generally portray his artefact because the solver equally adjusts all DOFs in order to minimise the constraint error.

If this artefact were to become noticeable the simulated DOF curves could be cropped between the second half of the first curve segment and the first half of the last segment. The subsequent loss of motion can be reduced by introducing new knot refinements at the start and end points of the curve since it is only half the curve segments of the first and last curves that is lost.

6.1.1.5 Dynamics-Based Retargeted Walk Timing Results

The example motion file is approximately 4 seconds, where the timings for the different stages of solving the retargetting motion are given in Table 6.1. The times given in Table 6.1 is pure computational time regardless of whether it is in response to user interactions or the automatic processes and it does not include any computer idle time taken while the artist is considering the current state. Furthermore, the time to build the initial equations in Table 6.1 includes the muscle optimisation formulations of the minimisation function, which are subsequently turn off for this example.

Retargetting Stage	Time taken to perform operation (in seconds)
Creation of dynamics equations	500.0
Initial value calculations	80.0
Each curve refinement	10.0
Minimum iterative step	20.1
Maximum iterative step	32.1
Average iterative step	24.2
Total number of main iterations	20 iterations
Total time to solve the retargeted motion	1064.0 seconds (~17 minutes)

Table 6.1: Execution time to generate the retargetting of a walking motion

The reason for the difference in the minimum and maximum iteration time given in Table 6.1 is twofold. The first factor is the number of active task constraints, which changes the size of the constraint vector and Jacobian matrix. The second factor is due to the iterative nature of the SVD algorithm used to perform inversion of the constraints Jacobian matrix (Equation 5.31) and the minimisation Hessian matrix (Equation 5.32) – the sub-iterations of the SVD algorithm are capped at 30 cycles per inversion.

When initialising the starting values, all constraints are turned off and therefore only the SVD of the Hessian matrix is required. Furthermore, during the main iteration stage, the optimisation process is performed on a subset of the character’s DOFs, which are those of the lower body because at no time will upper body deviate from the initial values as muscle minimisation is turned off. This effectively introduces a spacetime window [Cohe92].

As Table 6.1 serves to highlight, the dynamics solver is very computationally demanding in terms of time especially in the overhead associated with creating the system of dynamics equations. However, notwithstanding the very complex nature of the system’s constraint and minimisation functions and their associated differentials, the time it takes to evaluate the equations down to their scalar values is small compared to the time it takes to SVD the differential matrices. From the empirical results, on average only 9%[14] of the time it took to perform the SVD process was needed to evaluate the mathematical formulations down to scalar values.

6.1.2 Catching a Football

The second application of the retargetting process considers catching a football, as given by the IK retargetting example of section 4.6. This involves specifying the hand locations in order to meet the incoming target and ensuring the feet do not unrealistically float over the ground.

6.1.2.1 Determining Joint Trajectories vs. End-Effector Locations – The Control Processes

The initial time of contact between the hands and the football is known from the original motion. Therefore to retarget the hands to a new location, new user, time-based equality constraints are added into the dynamics optimisation process that specify the desired hand locations at time of contact. As seen in the previous retargetting example, the constraints override the joint space configuration of the motion data and hence the desired hand locations are present in the simulated motion. The formulations in Equation 6.1 outlines the user-specified equality constraints used to reposition the hands, where *{L|R}Hand{X|Y|Z}* are place-holder names that represent the end-effector positions, given by the expansion of Equation 5.11. The central ball catch location is given by $[x, y, z, 1]^T$, which makes contact at $t=1.0$. Constant offsets are applied to the central ball location to account for the size of the ball.

$$\begin{aligned}
&LHandX(t)-(x-1.42)=0\\
&LHandY(t)-y=0\\
&LHandZ(t)-(z+4.0)=0\\
&RHandX(t)-(x+1.42)=0\\
&RHandY(t)-y=0\\
&RHandZ(t)-(z+4.0)=0
\end{aligned} \qquad \textbf{(6.1)}$$

Due to the manner in which the dynamics representation is specified, no additional control processes are needed to assist in retargetting the hands; the formulates of Equation 6.1 is all the input need to retarget both the feet and hands using the dynamics-based technique. This is in direct comparison with the inverse kinematics version from section 4.6, where there was there a control strategy for the feet and one to smoothly reposition the hand end-effectors to the new catch location. With the dynamics solver, smoothing in and out between the catch point and the joint space

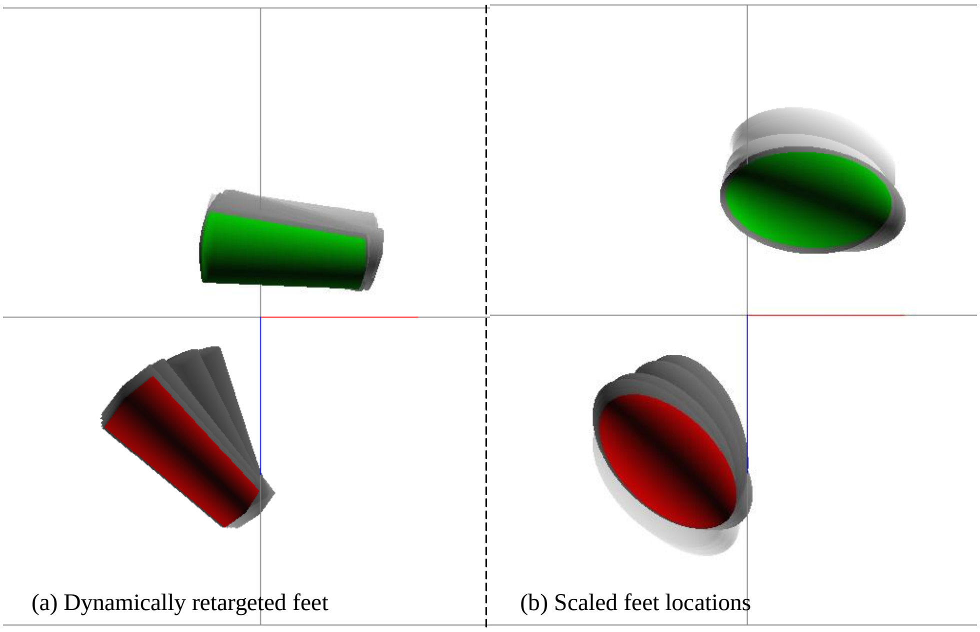
[14] This value has been extracted by averaging the time it takes to perform the SVD process compared to the time it took to evaluate the equations for the same iteration

configuration is not an issue because the trajectories are modelled using smooth piecewise cubic B-Spline curves.

The knot spacing used to represent the motion curves in this example is set uniformly at 0.142 seconds, which is not subsequently refined. The reason for the decreased knot spacing in this example is due to the high frequency with which the character “bounces” about on its feet within a small space of time. When attempted with a knot spacing of 0.2 seconds, the equation instability discussed during the walk retargetting example proves too significant while still giving good results for the feet, even though the hands retarget well at the higher resolution.

6.1.2.2 Dynamics-Based Retargeted Catch Visual Results for the Feet

The dynamically retargeted feet of the catch motion are illustrated in Figure 6.8, where the green and red objects illustrate the left and right feet respectively. The motion shadow of the dynamically retargeted feet is more clustered than the naively scaled feet. The motion shadow remaining in Figure 6.8a accounts for the foot pivoting about either the heel or the toes. This is an improvement on the scaled version of Figure 6.8b where what is seen is pure foot sliding as opposed to pivoting (which is further illustrated in the accompanying animations).



(a) Dynamically retargeted feet

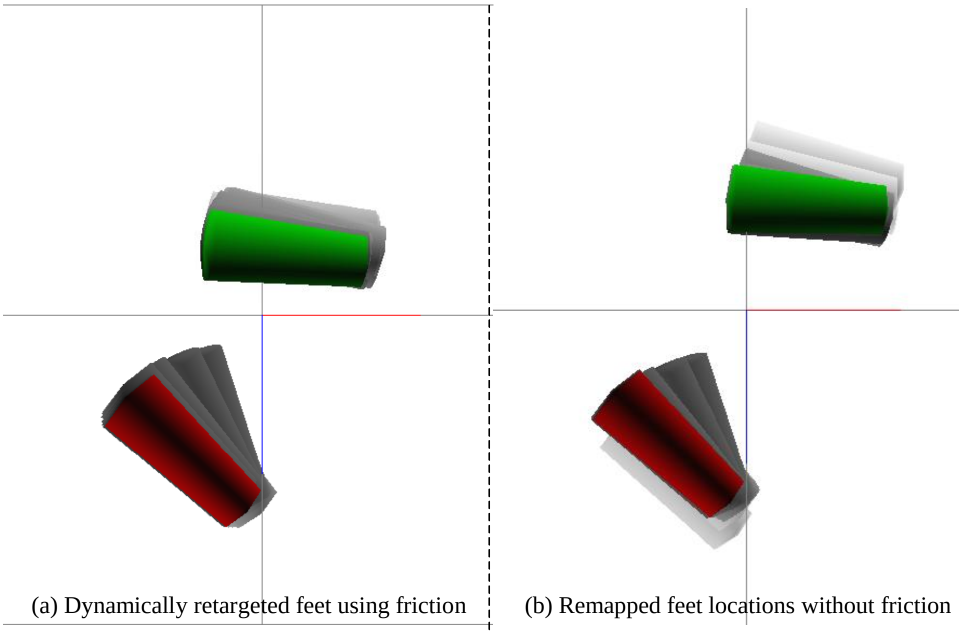
(b) Scaled feet locations

Figure 6.8: Foot retargetting of the football catch motion. (a) Dynamically retargeted, (b) simple scaling retargetting

Figure 6.8 demonstrates how the automatic contact and friction models of the dynamics representation successfully plant the simulated character’s feet to the ground in a realistic manner. However, in the original motion there is a small amount of intentional foot sliding over the ground, which is desirable to keep, yet the dynamics retargetting process has removed. This demonstrates that sometimes the dynamics representation can lose some of the subtle detail of the original motion.

This problem stems from the low forces acting on the feet, which are not sufficient to overcome the friction. However, the coefficient of friction cannot be reduced because this would introduce foot sliding for the whole motion. The reason why the original motion is able to slide over the ground is because the foot is very slightly lifted from the ground. However, when scaled, the motion loses this minute detail, which is subsequently interpreted by the dynamics optimisation process as foot plants.

The subtlety of the feet sliding over the ground can be reintroduced by including additional user constraints that position the foot locations back to the positions of the original character using similar constructs as given by Equation 6.1 for the hand locations. The friction models additionally need to be turned off within the dynamics representation to avoid conflicting constraints. The application of remapping the feet back to the original motion is illustrated in Figure 6.9, where some foot sliding has been reintroduced, but in a more realistic manner than the scaled version of Figure 6.8 as the foot locations are based on those of the real actor.


(a) Dynamically retargeted feet using friction

(b) Remapped feet locations without friction

Figure 6.9: Dynamically retargetting foot plants; (a) using friction, (b) without friction but with additional user constraints to remap the end-effector locations to the original motion

Figure 6.9b shows that the left foot has two stance positions. The solid green foot illustrates one of these whereas the other can be seen above this position as continuous grey shading. From the animation of this motion, the left foot can be seen to slide over the ground between these positions. However, in the friction enabled version this is prevented and hence the foot location is more concentrated as illustrated in Figure 6.9a.

This example has shown that there are two different approaches that can be taken to maintain foot plants. However, the friction version is much more general than remapping the original end-effector locations because, unless the two characters are roughly the same size the, mapping end-effector locations will result in unnatural looking postures. Some of the very subtle movements are lost although the resulting motion is physically correct and what’s more, it looks realistic because any undesirable foot floating has been eliminated.

6.1.2.3 Dynamics-Based Retargeted Catch Visual Results for the Hands

The hand locations are retargeted simultaneously with the feet within the dynamics-based technique as opposed to separately in the case of IK retargetting. Figure 6.10 demonstrates a range of different height values for Equation 6.1 that are used to specify user constraints for the catch height, *y*. The *x* and *z* components of Equation 6.1 are kept constant at the values given by the example motion at the catch point.

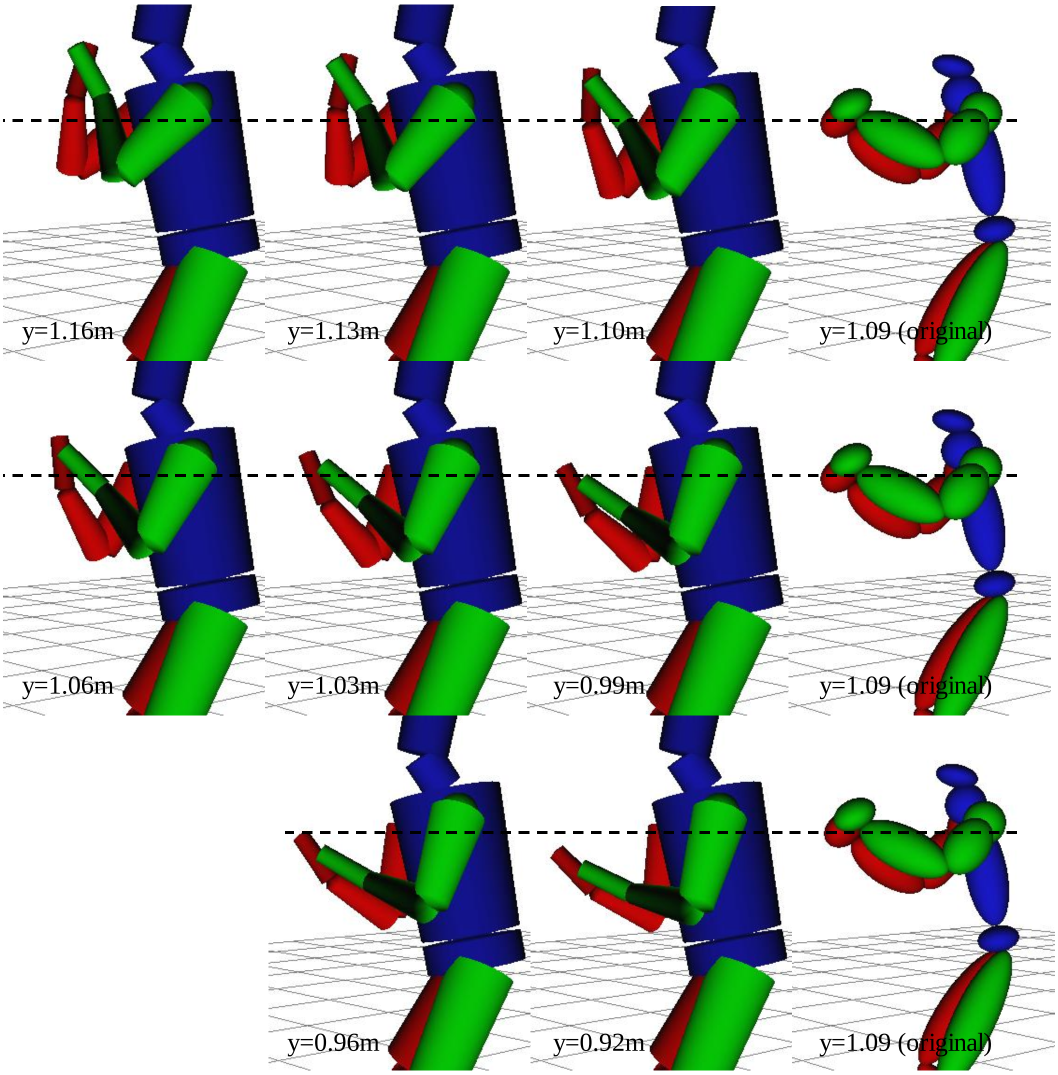


Figure 6.10: Dynamically retargeted hands to meet different target locations using Equation 6.1 as user constraints

Figure 6.10 shows how the dynamic retargetting solver can override the joint space minimisation of the example motion in favour of the preferred hand location, which are represented as user constraints. Furthermore, because of the underlying smooth curve representation of the generalised coordinates, this transition is achieved smoothly. Consequently, this example serves to demonstrate the ease with which the solver responds to simple user input, such as kinematic constraints, during the generation of the motion as well as the dynamics of the system. This is in comparison to the inverse kinematics technique where a whole control process is needed to change the location of the arms. In this example, other than addition of the hand location constraints, no other adjustments to the standard dynamics solver are needed.

6.1.2.4 Dynamics-Based Retargeted Catch Timing Results

Table 6.2 presents the times of the whole retargetting process for the friction-enabled foot planting version. The example motion is just over 4 seconds in length.

Retargetting Stage	Time taken to perform operation (in seconds)
Creation of dynamics equations	510.0
Initial value calculations	20.0
Minimum iterative step	45.2
Maximum iterative step	70.1
Average iterative step	60.4
Total number of main iterations	25 iterations
Total time to solve the retargeted motion	2040.0 seconds (34 minutes)

Table 6.2: Execution time to generate the retargetting of a catch motion

The number of iterations given in Table 6.2 is largely required to stabilise the contact forces for the feet since the hand locations were solved after approximately 6 iterations to an error of less than 0.5cm. Consequently, after 6 iterations, if the optimisation process were windowed for the lower body only, similar timings for the remaining iterations to those given in Table 6.1 could be achieved.

Similar to the retargeted walk timings of Table 6.1, the timings for the retargeted catch motion in Table 6.2 demonstrate very heavy computational costs, which per-iteration, most is consumed during the SVD process of the Jacobian and Hessian matrices. However, the high computational costs come with the benefit of ensuring the retargeted motion is realistically plausible, which is an issue that is further addressed in Chapter 7 where the IK and dynamics algorithms are compared.

6.1.3 Dynamics-Based Retargetting Summary

Gleicher [Glei98a] was the first to use the spacetime constraints technique to retarget characters, however in this work only kinematic correctness is addressed and not dynamical constructs. Works that have considered dynamics-based character adaptation [Popo99, Popo00b, Sofa04] have simplified their dynamics representation, which has consequently meant the incursion of additional control processes to produce a solution. For example, this has included the enforcing of spatial constraints through the application of an inverse kinematics solver over the dynamically simulated motion [Sofa04]. The shortcomings of these dynamics-based works in the area of retargetting is attributed to the abstraction of the dynamic models to point mass mechanics or more gross mechanical representations that only map well to large ballistic motions where detail is sacrificed [Popo00b]. This is in contrast to the complete rigid body descriptions used in this thesis. Due to the extra dynamical detail over previous works, more precise and accurate models of the character can be constructed and animated without the need for extra control processes and hence can be applied to non-ballistic and less energetic motions.

Liu *et al* [Liu05] present a very similar dynamical representation and solver to that used in this thesis and although they do not directly address retargetting, it is conceivable that their approach could achieve similar results. However, based on the description of their environment constraint forces, similar to Gleicher's work [Glei98a], positional constraints still need to be defined to drive these forces. In contrast, the dynamics representation of this thesis automatically determines these forces without the need of an additional control process and hence it can automatically define foot plants for example. Essentially, the retargetting constructs required are woven into the very fabric of the spacetime dynamic mathematical representation used in this thesis. This makes the dynamics-based retargetting algorithm different from other implementations and it is because of these constructs that the process of retargetting is such a simplistic one, i.e. no extra control processes are needed.

As the retargetting times of Table 6.1 and Table 6.2 demonstrate, the dynamics modification process is not real-time. However, since the whole motion is considered at once, it would not have been possible to apply the technique on the fly. To offset the large computational time costs required to retarget characters, there are benefits of the dynamics-based technique. One advantage lies in the inherent smoothness between frames, which can cause problems for per-frame-based techniques. Furthermore, as the hand retargetting example demonstrated, time-based constraints can be easily included without any backtracking or easing in and out control routines, which is another problem for per-frame techniques. A further advantage is the lack of control processes, other than the original motion, to guide the dynamics solver to a suitable, non-foot sliding motion. In comparison to IK-based retargetting techniques, such as that of Chapter 4, the main advantage that comes from using the retargetting process described in this chapter is that the resulting motion is guaranteed to be physically plausible.

6.2 Biomechanical Character Individualisation

In this section, the dynamical constructs are applied to example motions to individualising them, which demonstrates the variability that can be induced into the character with respect to the dynamics. Unlike the retargetting examples of section 6.1, the characters muscle forces are included in the minimisation function and hence placed in an active state. The minimisation function therefore consists of two competing purposes: one to maintain joint angle configuration between the generated motion and the example motion and one to minimise the energy exerted by the muscles. To balance the two minimisation components, an empirically obtained value of approximately 0.0001 is used to reduce the muscle force contributions such that is has similar magnitude to the joint angle minimisation. The magnitude reduction of the muscle forces additionally helps to prevent ill-conditioning when the hyperplane estimates are used in the optimisation process to calculate the projected estimates using the Jacobian matrix of Equation 5.32.

As this section looks at the effect of changing the dynamics of the simulated characters, to remove the variability of limb lengths this metric of the character is kept constant over all modifications in this section (foot sliding is however still prevented in this section because of its encoding into the dynamical representation of the system). The complete combination of changing the dynamical and kinematic properties of the character is demonstrated in section 6.3 where motions are mapped between the real actors.

In this section, unless otherwise stated, the base motion that is used to perform the modification comes from actor C. The following dynamical/biomechanical modifications are considered in this section:

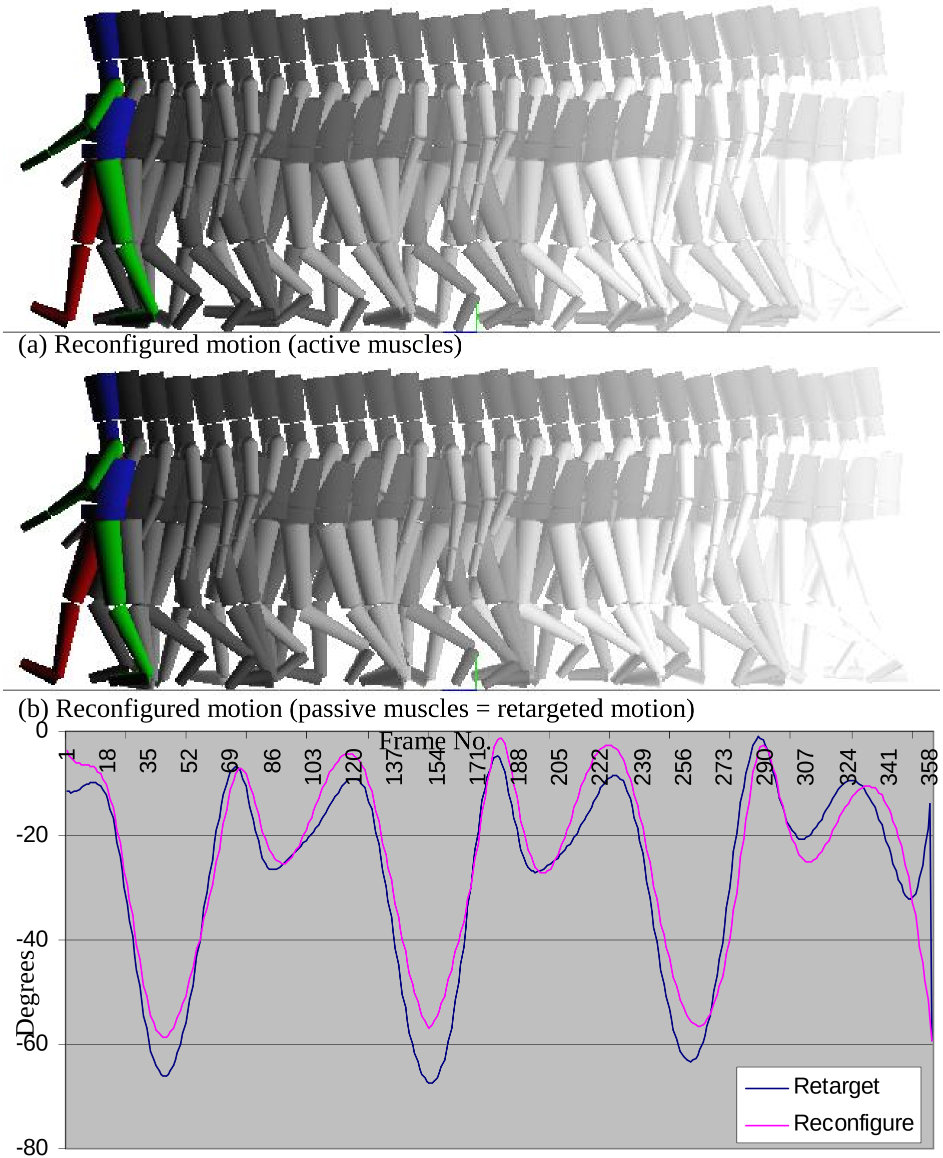
- Using active muscles (this gives the control simulated motion where all dynamical properties are left unchanged from the standard algorithm),
- Inter-muscle weighting ratios
- Muscle Gain Limiting
- Changing the biometric masses

Up to and including changing the inter-muscle weighting rations, parallels with the inverse kinematics approach of Chapter 4 can be drawn despite the large dissimilarity between the different techniques. However, the process of limiting the muscle gains and changing the biometric masses of the limbs goes beyond any direct comparisons with the IK technique because they are physically based. Although section 6.2.4 demonstrates there is a close correlation between the weighted inverse kinematics technique and changing the biometric masses of the character. This lends support to the weighted inverse kinematics reconfiguration process demonstrated and evaluated in Chapter 4.

6.2.1 Using Active Muscles

To provide a control motion to compare the individualisation process, a motion whose foundations are given by the walking gait of actor C is generated, where the muscles are placed in an active state. The control motion is generated using identical biometric data to actor C, as given in Appendix D.2. The gait of the individualised control motion is given in Figure 6.11, which is compared with the motion generated from the standard dynamics-based retargetting process, i.e. where the muscles are in a passive state.

Figure 6.11 shows that there is a slight difference between the retargeted and reconfigured motion, which is best viewed in the knee joint trajectory plot. However, the visual effect of the change demonstrated in Figure 6.11c is virtually indistinguishable, which can be seen in Figure 6.11a and Figure 4.9b (and the accompanying animation file).



(a) Reconfigured motion (active muscles)

(b) Reconfigured motion (passive muscles = retargeted motion)

(c) Comparison between the knee angles of motions (a) and (b)

Figure 6.11: Control motion generated from the gait movement of actor C. (a) Illustrates the reconfigured motion (with active muscles), where (b) demonstrates applying the same process to the same skeleton but with passive muscles. (c) Compares the left knee joint trajectories of the two motions

Whereas in the case of the retargetting examples the optimisation process generally converges towards a solution in relatively few iterations, the reconfiguration process takes considerably longer. This is highlighted in Table 6.3 for the reconfigured motion of Figure 6.11 (which consists of 358 frames, about 4 seconds). The need for so many iterations is attributed to the many non-linear equations that the muscle forces are involved in and hence the updates happen very slowly in order to maintain a stable set of constraints while simultaneously attempting to reduce the optimisation function. Due to the very slow progress made at each step, it is difficult to determine whether the results are demonstrating any meaningful changes. Therefore, if the stopping conditions defined in section 5.5.3.3 are not met, the global iteration count is capped at 1000 cycles. From empirical results, going beyond this limit does not generally yield any noticeable difference in the results.

Retargetting Stage	Time taken to perform operation (in seconds)
Initialisation of dynamics equations	520.0
Initial value calculations	82.0
Each curve refinement	10.0
Minimum iterative step	21.3
Maximum iterative step	37.5
Average iterative step	25.1
Total number of main iterations	1000 iterations
Total time to solve the retargeted motion	25,702.0 seconds (~7 hours)

Table 6.3: Computation execution time to generate the retargetting of a walking motion

Although the reconfiguration process requires many more iterations than the retargetting process, the per-iteration times are comparable. This is demonstrated by comparing the retargetting times of Table 6.1 and reconfiguration times of Table 6.3, where both applications use the same example motion. Therefore the inclusion of the muscle minimisation optimisation function to convert the dynamics optimisation process from a retargetting algorithm to a reconfiguration technique does not have any detrimental effect in terms of per-iteration time.

Given that the time to solve the dynamics-based optimisation process tends to be in the 4-7 hour computational time range (where Table 6.3 demonstrates the higher end of the computational time taken of the simulated motions), the timings of the subsequent reconfiguration are omitted for both clarity and the obvious reason that, compared to the inverse kinematics techniques, there is little purpose in performing timing comparison. Therefore, the continued examples focus on the actual simulated motion rather than how long it takes to generate.

6.2.2 Inter-muscle weighting ratios

The inter-muscle weighting ratio is defined by the relationship between the muscles that drive the motion of the character's individual rigid body nodes. Changing the inter-muscle weighting ratios translates into the optimisation algorithm preferring to minimise one joint's muscle over its neighbours and hence reduce its overall movement. For example, similar to the individualisation process employed for the inverse kinematics algorithm (see section 4.8), if more weighting is given to the character's upper leg muscles than the lower ones, the upper leg will appear more stiff in its movement than the lower leg.

Figure 6.12[15] illustrates the potential of modifying the motion by adjusting the inter-muscle weightings. The muscle weighting ratio used to produce the motion of Figure 6.12 is 3:1:1 for the femur, tibia and foot respectively.

[15] The playback speed of the accompanying animation file for Figure 6.12 has been reduced by a factor of 2 to better illustrate the difference in the leg motion

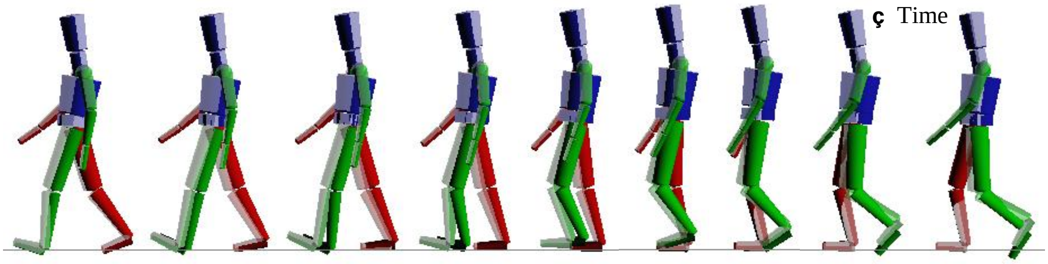
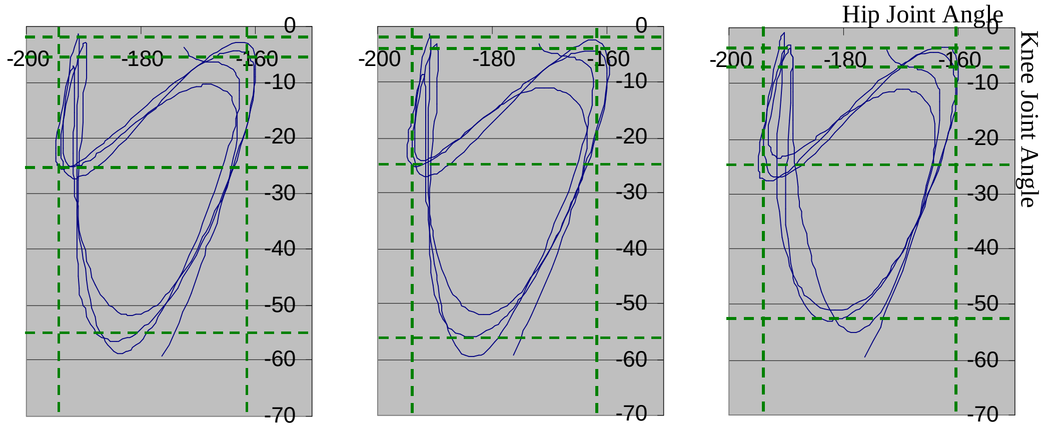


Figure 6.12: Reconfigured of the base motion using an inter-muscle weighting ratio of 3:1:1 for the femur, tibia and foot respectively. The ghosted motion represents the reconfigured control motion of Figure 6.11, whereas the opaque motion gives the newly reconfigured movement

Figure 6.12 illustrates how the weighted muscle reconfigured character's upper leg limb (opaque motion) consistently lags behind that of the reconfigured control motion, whose motion is signified by the ghosted character. This is the behaviour that was expected given that the upper leg's weighting ratio is increased comparison to the rest of the leg. To further illustrate the difference in the joint configurations, Figure 6.13 gives the gait signatures of the motions in Figure 6.12. Figure 6.13 additionally includes the reconfigured gait signature of the motion that results from a leg muscle weighting ratio of 1:3:1 for the femur, tibia and foot respectively.

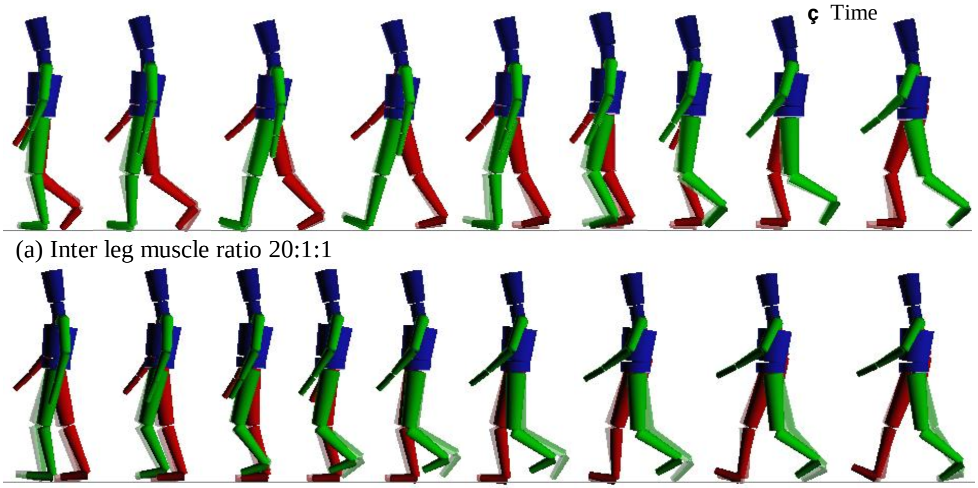


(a) 1:1:1 leg muscle ratio (b) 3:1:1 leg muscle ratio (c) 1:3:1 leg muscle ratio

Figure 6.13: Gait signatures of three reconfigured characters using different leg muscle weightings where the ratios relate to the femur, tibia and foot respectively

In the case a higher weighting contribution to the femur in Figure 6.13b, the tibia joint angle stretches out slightly further to compensate. This is most noticeable in the bottom range of the knee angle, which is slighter lower than that of the reconfigured control motion given in Figure 6.13a. When the weighing is changed to give the tibia more contribution towards the minimisation function in Figure 6.13c, the range of the knee joint is much more reduce compared to the both the evenly weighted and femur biased reconfigurations.

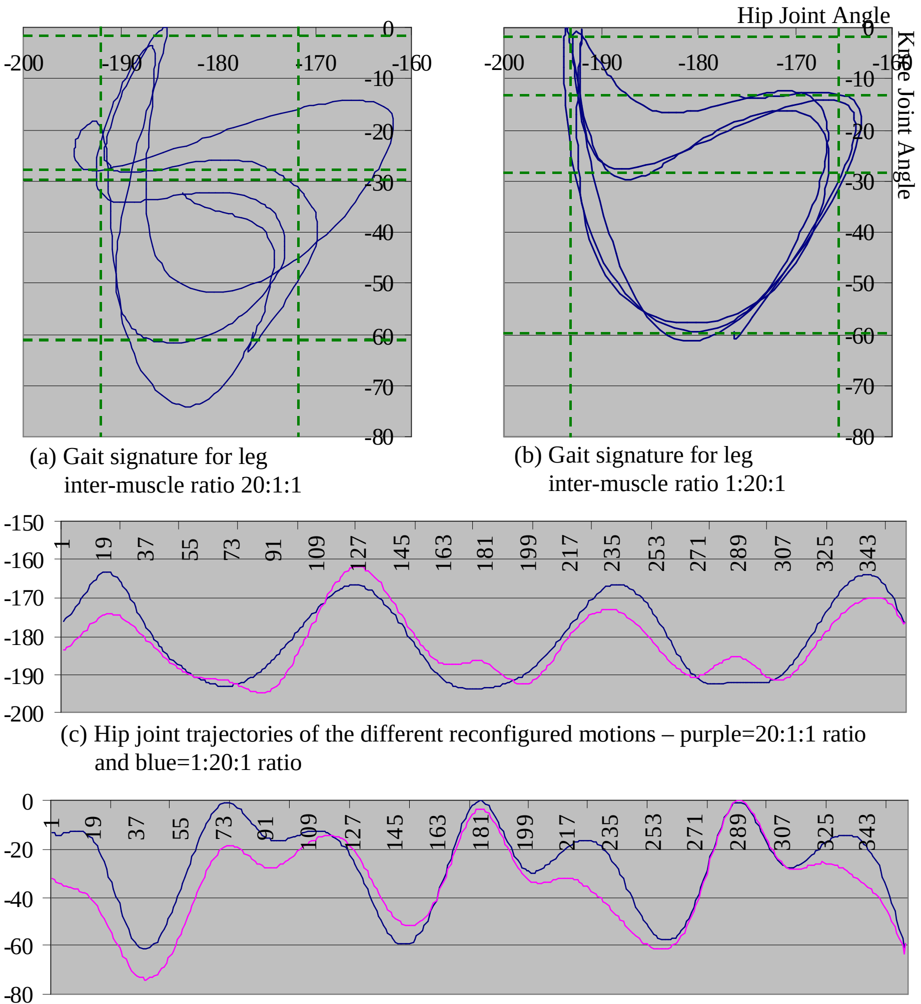
The subtle adjustments that are introduced by changing the inter-muscle weightings are comparable to those demonstrated during the weighted inverse kinematics individualisation process of section 4.8. As with the inverse kinematics case, small variances in the inter-muscle ratios relate to small and subtle changes in the resulting reconfigured motion, whereas large variances tend to over exaggerate the outcome, which is demonstrated in Figure 6.14. The gait signatures of the motions demonstrated in Figure 6.14 are given in Figure 6.15 as well as the joint trajectories of the femur and tibia for comparison.



(a) Inter leg muscle ratio 20:1:1

(b) Inter leg muscle ratio 1:20:1

Figure 6.14: Large variances in the inter-muscle weightings using a ratio of (a) 20:1:1 and (b) 1:20:1 for the femur, tibia and foot respectively



(a) Gait signature for leg inter-muscle ratio 20:1:1

(b) Gait signature for leg inter-muscle ratio 1:20:1

(c) Hip joint trajectories of the different reconfigured motions – purple=20:1:1 ratio and blue=1:20:1 ratio

(d) Knee joint trajectories of the different reconfigured motions; purple=20:1:1 ratio and blue=1:20:1 ratio

Figure 6.15: Gait signatures of large variance inter-muscle weightings using a ratio of (a) 20:1:1 and (b) 1:20:1 for the femur, tibia and foot respectively. The plots of the joint trajectories of the leg joints of the two reconfigured motions are given in (c) for the hip joint and (d) for the knee joint

The resulting effect of disproportionately exaggerating the inter-muscle ratios is to force a more pronounced change in the ankle joint to compensate for the change. This can be seen in the illustrations of Figure 6.14, and more clearly in the accompanying animation files, by the newly-reconfigured character being more inclined to quickly lift its heel from the floor in order to compensate for the reduced upper leg motion. This is in contrast to the low variance weightings used in Figure 6.12 where the foot orientation closely resembles that of the reconfigured control motion.

The effect of the larger increase of the inter-muscle ratios can be clearly seen in the joint trajectories of Figure 6.15, where the femur-biased reconfiguration has a greater tendency to reduce its overall range of motion, as illustrated by the purple joint trajectory. This is a feature additionally supported by the corresponding gait signature of Figure 6.15a. One characteristic to notice in the gait signature of Figure 6.15a is the visually erratic behaviour of the pattern, where the range of motion for the hip and knee joints change with different stride cycles, which can also be seen in the trajectory curves of Figure 6.15. This is due to there being consequential effects between stride cycles.

The characteristic erratic nature of the gait signature for high inter-muscle weighting ratios is also apparent, but perhaps not so much, in the gait signature and joint trajectories of the motions that bias the tibia of Figure 6.15 (blue joint trajectory). The irregular nature of the stride cycles in Figure 6.15 are, although appearing visually consistent, portraying a more peculiar gait, which could be synonymous with injuries, as was seen with the weighted inverse kinematics algorithm.

In each of the low variance ratio gait signatures, it can be seen that the overlapping pattern in the top left region, where the curve goes up and back down around onto itself has been preserved. However, this region in the high variance inter-muscle ratios has been stretched out in Figure 6.15b to reduce the overlapping portion. This is due to the very different joint trajectory paths that the hip and knee joints take because the ankle is being more excessively used and thus producing a very different motion compared to the example one.

A consistent feature has been the relatively immutable when changing the inter-muscle weighting ratio have been the foot plant locations. The cause of this is that the initial values used to start the optimisation process provide an estimated guess as to the contact and hence the friction forces. Although these values do change over each iteration to sharpen foot plant locations, there is little incentive to deviate too much from them because the solution is in a region of local minima. Adjusting the minimisation function does not prove to be enough to temporarily override the constraints in order to allow the contact and friction forces to be reinitialised at alternative locations and hence move away from the current local minima. However, the foot plants can be adjusted without manually overriding their positions by adjusting the muscle forces, which is discussed in the following section.

6.2.3 Muscle Gain Limiting

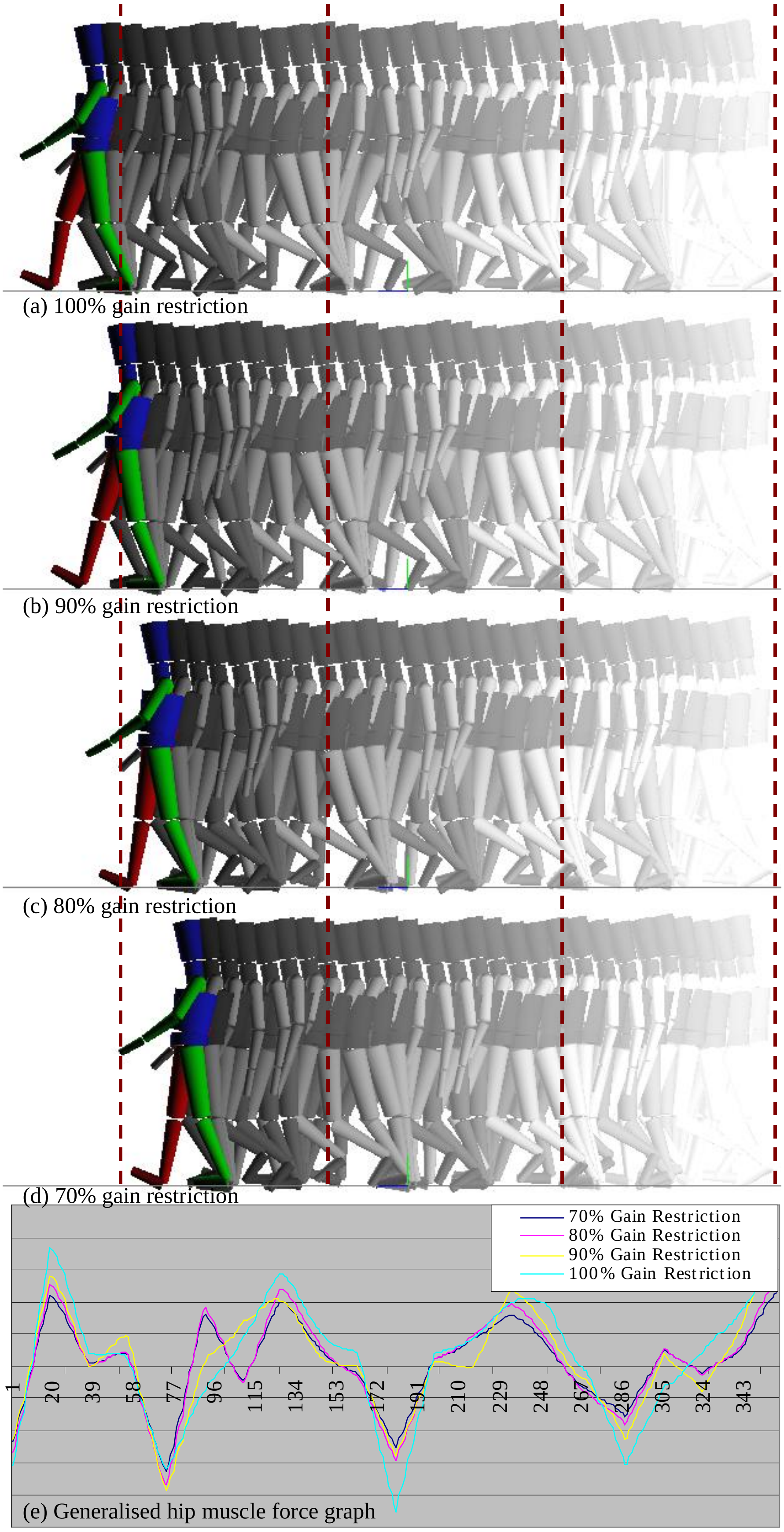
The effect of limiting muscle gains is to reduce the maximum amount of force exerted by the muscles thereby producing a slower, lazier moving character. This is different to the objective function attempting to minimise the global energy because a physical inequality constraint is used to limit the maximum force at any point in time as opposed to globally. Furthermore, because the muscle gain limiting is achieved using constraints, the system is made to reduce the amount of injectable energy into the motion (recall that constraints are favoured at the potential cost of the objective function). The consequence of this is to force a reduction in the stride length because the character cannot throw its legs as far forward on take off and hence a change in foot plant locations.

Figure 6.16 illustrates the result of placing restrictions on the muscle gains that limit the femur strength to 90%, 80% and 70% of the maximum exerted force that was seen during the reconfiguration of the control motion of Figure 6.11. An even inter-muscle weighting ratio is maintained. From the graph of Figure 6.16e it should be commented that because the muscle gains are only being capped, the generalised muscle forces do not directly take an equally large reduction in their magnitude. This is due to the rest state variable partly contributing to the loss of gain, but not completely since it is limited itself (see Equation 5.56). Nevertheless, Figure 6.16e demonstrates that there is a corresponding overall decrease between the maximum magnitudes of the generalised forces and the muscle gain restrictions.

The gait examples in Figure 6.16 demonstrated the property of reducing the footstep length, which decreases as the maximum gain value is reduced. Unfortunately, starting from the same base motion, attempting to cap the muscle gains at 60% or lower results in a system that becomes unstable, failing to converge on a suitable solution. This is because there is too large a difference between the initial values and the final result. Indeed, at each progressive decrease of the muscle gain limits, the optimisation process found it increasingly difficult to converge towards a desirable solution without the aid of some user intervention to relocate any system free variables that inflated the system state (a process that could have been somewhat automated).

The main cause of the unsteadiness during the generation of the motions in Figure 6.16 is linked to the fact that, even though the process is trying to solve the whole motion at once, it first needs to find an established state for earlier foot plants before the later ones. This is due to the hip location being dragged backwards because of the shorter stride lengths, which consequently alters both the locations and timings of subsequent foot plants – any adjustments performed in the past have an effect on the future. However, this problem only becomes an issue if the starting motion is significantly different to the closest possible simulated one based on the system constraints.

A fundamental similarity between the reconfigured motions of Figure 6.16 is that each of the characters takes the same number of strides during the time span, which is a characteristic that was also apparent in the inter-muscle weighting ratio adjustments. This is a consequence of the optimisation process trying to maintain similar joint trajectories to the base motion, which dictates the number of strides through its defined joint trajectories. Therefore, at an elementary level the simulated motion will always closely mimic that of the example motion. This highlights the need for a good example motion that resembles the desired motion from which to start the reconfiguration process. For example, it would be difficult to convert a running motion into a hopping one because fundamentally they are very different types of motions.


(a) 100% gain restriction

(b) 90% gain restriction

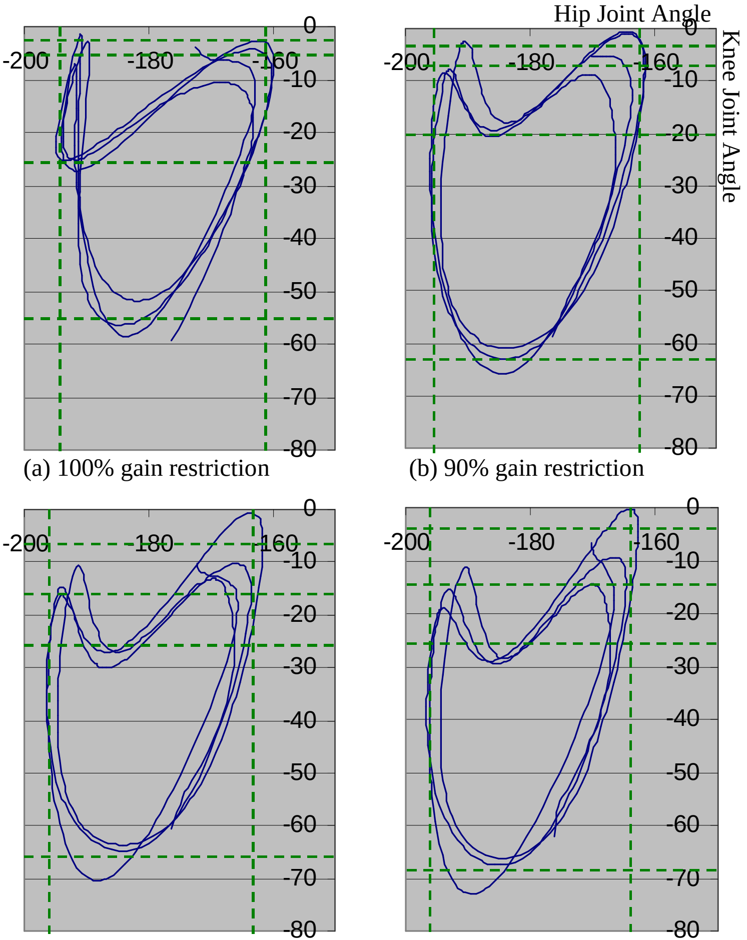
(c) 80% gain restriction

(d) 70% gain restriction

(e) Generalised hip muscle force graph

Figure 6.16: The effect of applying muscle gain restrictions using inequality constraints to bound the gain by (a) 100%, (b) 90%, (c) 80% and (d) 70% of the reconfigured control motion's maximum muscle gains. The resulting generalised hip muscle force graphs of the 4 motions are given in (e)

To complement the visually effect of limiting the muscle gains in the reduce stride length, Figure 6.17 plots the gait signatures of the different gain-restricted motions.



(a) 100% gain restriction

(b) 90% gain restriction

(c) 80% gain restriction

(d) 70% gain restriction

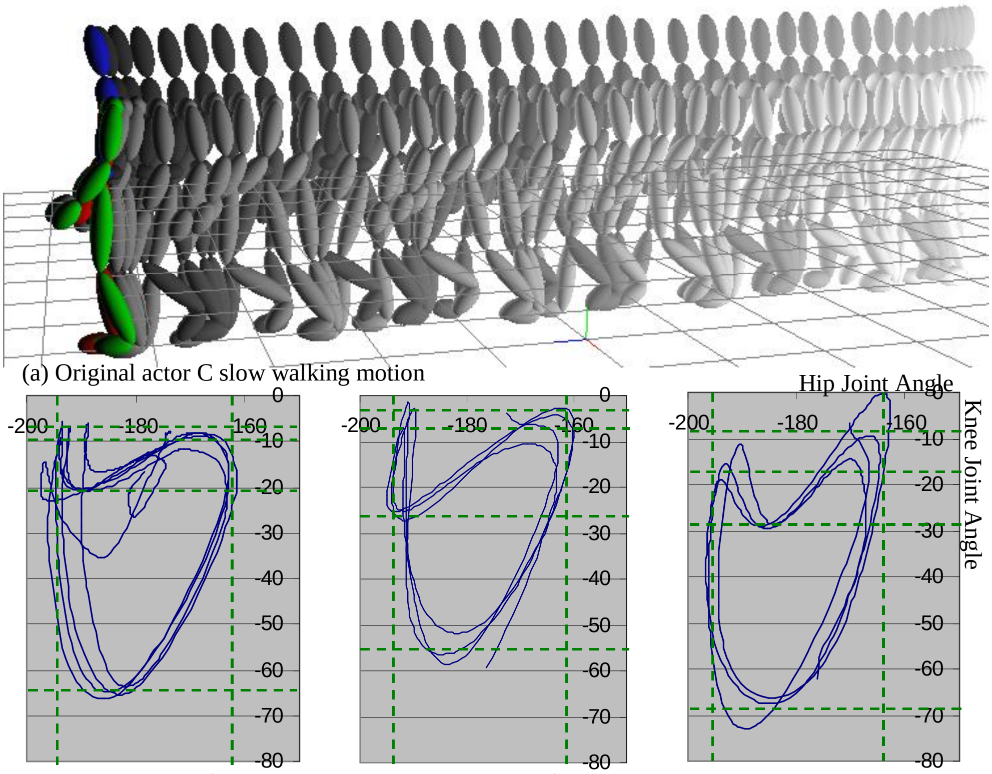
Figure 6.17: Gait signatures of (a) 100%, (b) 90%, (c) 80% and (d) 70% muscle gain restricted reconfigured motions of Figure 6.16

The gait signatures in Figure 6.17 of the muscle gain restricted motions show an increasing trend to reduce the overall hip angle range as the limb becomes less energetic. This is a feature that was absent from the previous inter-muscle ratio adaptations of Figure 6.13, where the hip angle maintained a fixed range due to the footsteps remaining unchanged. Furthermore, each of the muscle gain restricted motions tends towards a more straighten leg posture at certain parts of the motion. This can be seen as the knee joint approaches an angle of zero degrees near the top right region of the gait signatures. This is due to the optimisation process attempting to meet the original foot plants, which are subsequently restricted by the femur muscle gain restrictions and hence the knee attempts to compensate by bending the knee forward towards the plant. This, however, is restricted by the joint limits, which pushes the knee joint back within its assigned range. Consequently, the only solution the optimisation process can take is to move the foot backwards. However, the virtually straight knee joint remains present because the optimisation process only moves the hip joint back enough for the straight leg to make contact with the floor in a physically correct manner.

The other noticeable change in the gait signatures of Figure 6.17 is the shape around the top left region of the angle relationship. In the original motion, the knee angle has much more variation over a small range of hip angle, whereas the gain restricted motions have a more spread out hip joint angle for the same knee joint range. A reason for this is that the joint trajectories are still attempting to align themselves with the original, however because the foot plant locations have changed, the original trajectories have to temporarily make way for end-effector correctness. In between the new footsteps, the trajectories are subsequently allowed to approximate the original motion, which is what we are seeing in the gait signatures of Figure 6.17. This cause is further compounded by a constant stride time in the simulated motion, which is dictated by the example motion for the same reason as a constant number of foot strides is determined by the base motion. Therefore the solver has to maintain the timings in between and during foot plants and hence needs to modify the motion curves to compensate for the rigid timing, which is being achieved by stretching out the overlapping region. A possible extension to the work presented in this thesis to resolve this problem, a time warping factor could be introduced so that the dynamics optimisation process time sample does not necessarily need to correspond to the time equivalent base motion frame.

6.2.3.1 Evaluating the "Lazier" Walk Motions Produced by Restricting Muscle Gain Limits

To evaluate the effectiveness of reducing the muscle gains to produce a slower, lazier walking style, the limited simulated motion is compared with the slow walking movement recorded from the same actor whose normal gait motion is used to produce the gain-restricted motions. The recorded slow walking motion and gait signature of actor C is illustrated in Figure 6.18.



(a) Original actor C slow walking motion

(b) gait signature of slow walking motion

(c) gait signature of normal walking pace

(d) muscle gain restricted walking motion

Figure 6.18: (a) Actual motion of actor C walking slowly with its corresponding (b) gait signature. The gait signatures for the normal walk and 70% gain restricted motion of actor C are repeated from Figure 6.17 in (c) and (d) respectively for comparison

From initial inspection of the actual slow walking motion and the muscle gain restricted gait signatures of Figure 6.18 their patterns do not appear to be very similar. However, there are important features present in both. The first attribute similar between the gait signatures of (a) and (c) from Figure 6.18 is the increased knee joint angle range over the original motion (b). Simultaneously, the second characteristic of importance is that both the slow walking gait motion and the gain restricted one have a slightly smaller hip angle range compared to the normal motion, which is indicative of smaller steps. Based on the gait signatures of Figure 6.18, it would appear that actor C's slow walking motion consists of using the knee joint slightly more compared to the hip joint as a more relaxed style of walk, which is exactly what is being simulated by employing muscle gain restrictions.

Aside from the stretching out effect in the top left region of the gait signatures, which has previously been discussed, another difference between the gait signatures of Figure 6.18 lies in the knee angle range along the top of the gait signatures. Compared to the slow walking motion, it can be seen that the other two motions have a larger range in the region between the first and the third horizontal local minimum/maximum lines (where the first line is the top dash, green line). For the normal walking motion, the comparative large range is due to the more energetic motion as the foot is being lifted higher per walk cycle. However, the muscle gain restricted motion does not exhibit a reduction in this quantity because the muscle gain restrictions are only placed on the hip joint to prevent the leg being thrown forward as much. Therefore, they are mimicking the joint space configuration of the base motion as opposed to taking on a lazier appearance.

The gross changes between the normal walk to the lazier style demonstrated for actor C in Figure 6.18 are similarly favourable when applied to the other 3 actors in the dataset, which is illustrated in Figure 6.19. The gaits simulated for Figure 6.19 have all been simulated with an upper leg muscle gain restriction of 70%.

The results of Figure 6.19 show that the desirable change of reducing the overall hip joint range of movement has been introduced. The change can be seen by the horizontally more squashed curve pattern, which is visible in both the gain restricted reconfigured motion and the slow walking motion compared to the normal walk of the actor. The gait signatures of actors A and D show a similar result to the gain restricted reconfiguration of actor C, where the sharp definition of the hip-knee relationship in the top left region of the pattern is lost. However, the reconfigured motion for actor B from Figure 6.19 has preserve this feature. Actor B's normal walking motion is also much slower than the motions of any of the other actors, which means that stride time is much larger for actor B then it is for the other actors. This supports the reason that the loss of sharpness is partly attributed to the lack of time-warping from the base motion because the time-warping aspect has less of an effect on actor B because of his already slow walking pace compared to the other actors.

Figure 6.19 further illustrates the previously noticed effect that the slow walking motion has a reduced knee angle range along the top region of the gait signature compared to the original walking motion. Although, similar to the result demonstrated for actor C, the muscle gain restricted motions more closely correlate to the original walking motion in this region than the slow walking motion. This is because of the optimisation process's desire to minimise the distance to the original knee joint in this region and the lack of extra time to execute a lazier style of walking in between foot strides.

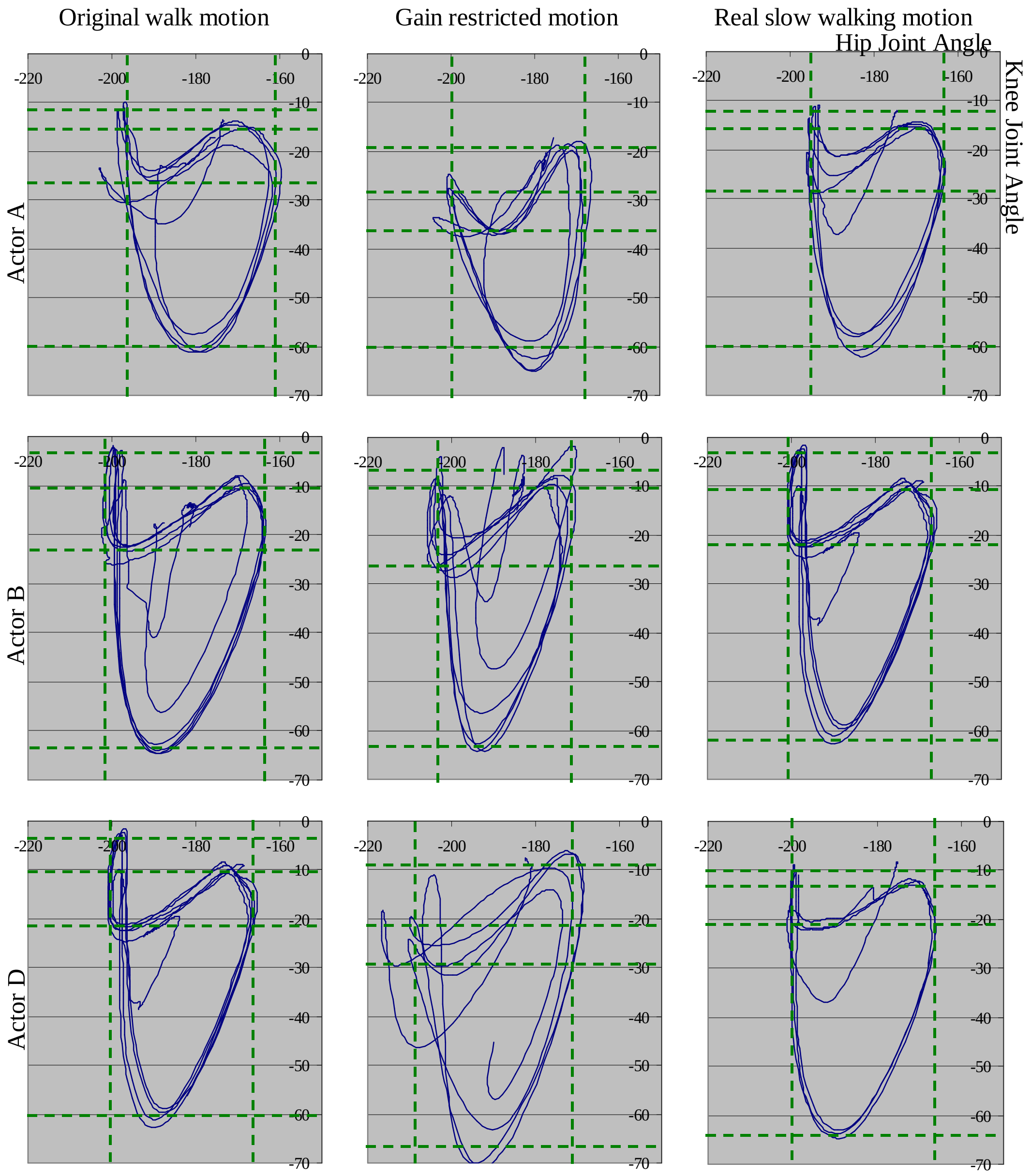


Figure 6.19: 70% muscle gain restricted gaits of 3 different actors compared to their real slow walking motions

In addition to the reduced hip joint angle, Figure 6.19 shows that the lower range of the knee angle has been reduced in both the simulated and slow walking motions compared to the normal walk. This follows the same pattern that was recognised in Figure 6.18 for actor C. The reconfigured gait signature for actor D in Figure 6.19 is overly elongated, although this can be adjusted by increasing the reduction limit, which has been shown in this section to have a directly proportional influence.

From the analysis of the muscle gain restricted motions of the 4 different actors, it has been demonstrated that it is possible to affect the desirable gross joint angle range changes that are apparent between the normal and slow walking motions of the actors. One feature that has eluded the optimisation process has been the reduce knee joint range that runs along the top portion of the gain signatures. However, the motions used to evaluate the process have not truly been like for like because of the reduced speeds involved in the real slow walking motions and not the simulated ones. This highlights the need for a time-warping feature that would make the existing dynamics solution less dependent on the fundamental timings of the example motion. Although an alternative solution within the current framework would be to apply different gain restrictions on the knee joint, however, without independent timing, this could cause conflict within the constraints by placing too many limits on the rigid body dynamics.

Although a degree of evidence can be provided that suggests restricting the muscle gains result in a natural and similar motion to that of a real actor walking slowly/lazily, this cannot be conclusively proven because of both the lack of data available and the time-dependent nature of the dynamics optimisation process. However, the visual results do look very plausible and realistic which can be seen in the accompanying animations of Figure 6.16 and Figure 6.18.

6.2.4 Changing the Biometrics Masses

In this subsection the potential of the dynamics representation is demonstrated through the more intuitive manner of changing only the biomechanical mass information of the character. The important result that is demonstrated in this section is that there is a correlation between changing the biometric masses of the character's limbs and changing the inter-muscles weighting ratios. Using the result of this section in conjunction with the correlation between the inter-muscle weighting ratios and the weighted inverse kinematics results of section 4.8, the relationship becomes very significant. Through chaining the two correlations, what it implies is that weighted inverse kinematics is reliably able to approximate changing the biomechanical masses of a character of a much more complex dynamics-based motion modification approach. This supports the practical application of weighted inverse kinematics and further complements the evaluation of the technique that was performed in Chapter 4. The change in only biomechanical masses of the character additionally sets the scene for later reconfiguring one motion to another using only the target character's biomechanical information (which includes limb length differences) in section 6.3.

Three different gait motions are simulated from the same base motion using the identical limb lengths to the original motion but using the masses outlined in Table 6.4. The limb weights are taken directly from the real actors given in Appendix D.2. All other parameters of the dynamics representation are maintained at their unbiased, standard state, i.e. even inter-muscle weighting ratios are used with no muscle gain restrictions. The gait signatures of the simulated motions using the biomechanical masses of Table 6.4 are illustrated in Figure 6.20.

Node Name	Actor C (Control Weights)	Actor A	Actor B	Actor D
Femur	6.95 kgm/s^2	6.76 kgm/s^2	5.12 kgm/s^2	5.66 kgm/s^2
Tibia	3.02 kgm/s^2	3.40 kgm/s^2	4.30 kgm/s^2	3.48 kgm/s^2
Foot	1.14 kgm/s^2	0.96 kgm/s^2	1.04 kgm/s^2	1.01 kgm/s^2
Relative Ratio	10:10:10	11:14:10	10:20:12	10:14:10

Table 6.4: Limb weights used to dynamically affect the physical appearance of a character's motion. The relative ratio indicates the relationship between the node masses with respect to the control weights

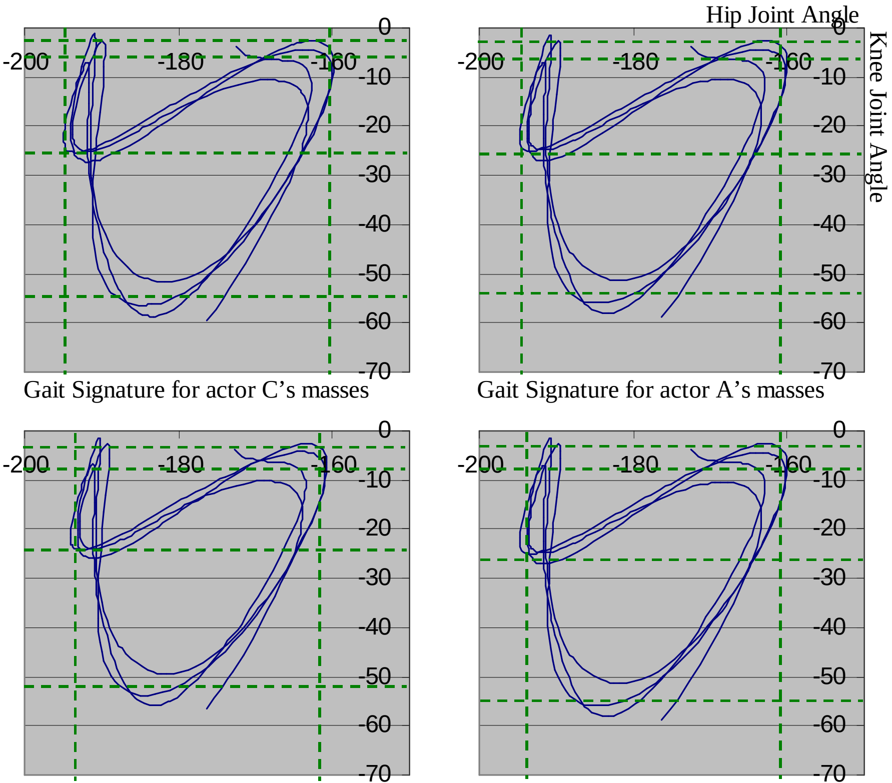


Figure 6.20: Gait modifications on actor C's walking motion using the limb weight biomechanical mass information from Table 6.4

The gait signatures in Figure 6.20 between limb masses of actors A, C and D are very similar. This is because their mass distribution along the leg is virtually identical, despite the actors having different inter-leg masses, which is highlighted by the relative ratios in Table 6.4. However, the masses of actor C give rise to a slightly reduced knee joint range in its gait signature of Figure 6.20 because of the marginally larger difference between the mass distributions along the leg from actor C.

The most visually different gait signature to that of the control signature of actor C is illustrated by applying the mass information belonging to actor B. In this gait signature there is a more noticeable decreasing in the range of motion for the knee joint compared to that of actors A and D and the control motion of actor C.

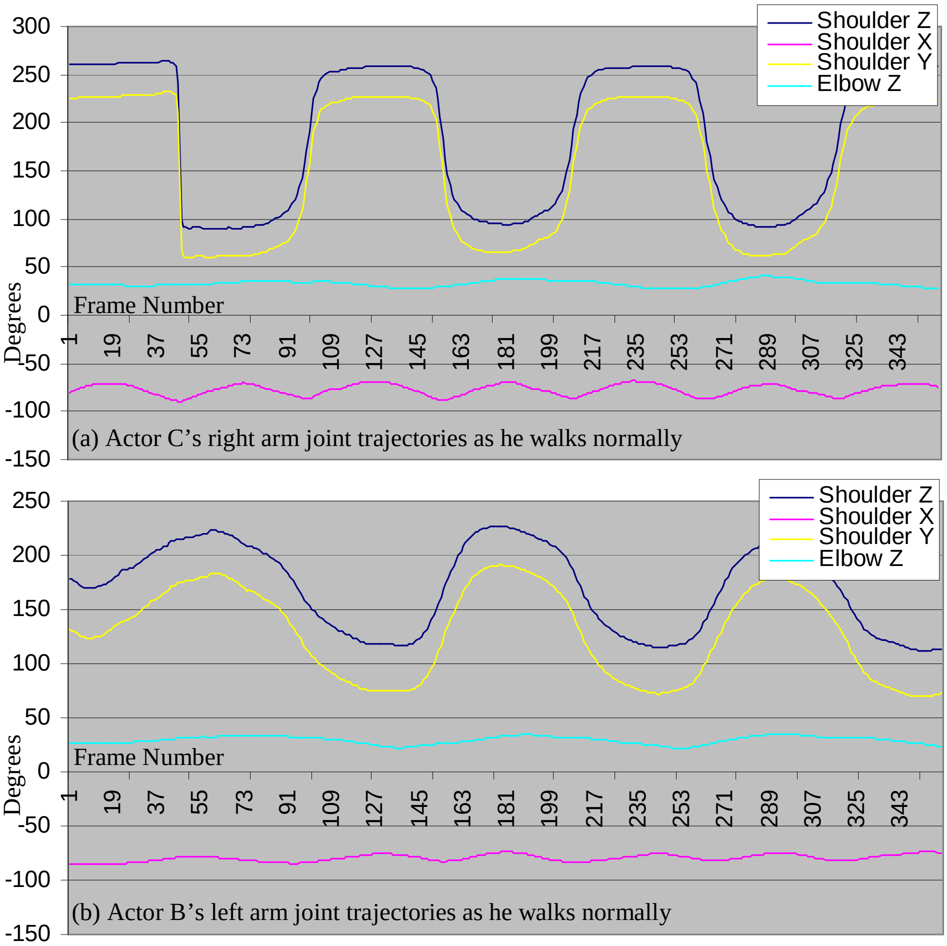
The reason for the influencing effect induced by changing only the limb masses is due to a fundamental parameter change within the dynamics representation. Recalling Equation 5.16 (Lagrange's equations of motion), the rigid body masses are taken into consideration when calculating both the bodies linear and angular velocity. Therefore, it follows that given varying limb masses, different amounts of energy are required to preserve the same motion as these two factors are directly related as illustrated by Equation 5.16. However, the dynamics optimisation algorithm is not attempting to exactly preserve the original motion, but, in part, minimise the resulting muscle forces required. Therefore, when minimising energy equally over the connected rigid body, in regions that proportionally require more energy than others, the resulting trajectories will be dampened compared to the lighter nodes which now require relatively less energy to move. This is exactly what can be seen in Figure 6.20 when comparing across the motions.

The results that are demonstrated in Figure 6.20 are very similar to that those using inter-muscle ratio weightings. This is highlighted when comparing the relative ratios in Table 6.4 with the inter-muscle weighting ratios of Figure 6.13 and the corresponding gait signatures. The close correlation between the different modification techniques, when considered, is only to be expected because the two changes amount to a very similar mathematical system change. However, this result is extremely significant because it supports the concept that dampened joint trajectories in a kinematics chain produce a similar effect to dynamically modelling rigid body mechanics using mass and moments of inertia. This is exactly the property that weighted inverse kinematics (section 4.7) exploits to individualise motions.

Although a close correlation between changing biomechanical masses and the effect produced by manipulating the inter-muscle ratios has been shown, the inter-muscle ratios offer the further ability to add extra value into the biomechanically correct motions. This is especially true in the case of overly exaggerated motions. Consequently, inter-muscle ratios remain a useful feature from which to generate motions, which will be further discussed in connection to simulating motions that depict injuries in section 6.4.

6.2.5 The Upper Body

The effect of changing dynamics and biomechanical properties has thus far focused on their influence over the lower body. As this section will demonstrate, the richness of these modifications does not completely map to the upper body, primarily because of the lack of external contact and friction forces with the ground. The limited variation in the results for the upper body are further compounded because, unlike the legs where both the hip and knee joints noticeably contribute to the motion, the shoulder and elbow joint relationship during walking is much more asymmetrical. Figure 6.21a shows that the elbow joint hardly deviates from a rest state during motion, which plots the DOF trajectories for the right arm of actor C as he walks normally. This is further demonstrated by the arm joint trajectories of actor B, which are illustrated in Figure 6.21b, and the other two actors that were captured.



(a) Actor C’s right arm joint trajectories as he walks normally

(b) Actor B’s left arm joint trajectories as he walks normally

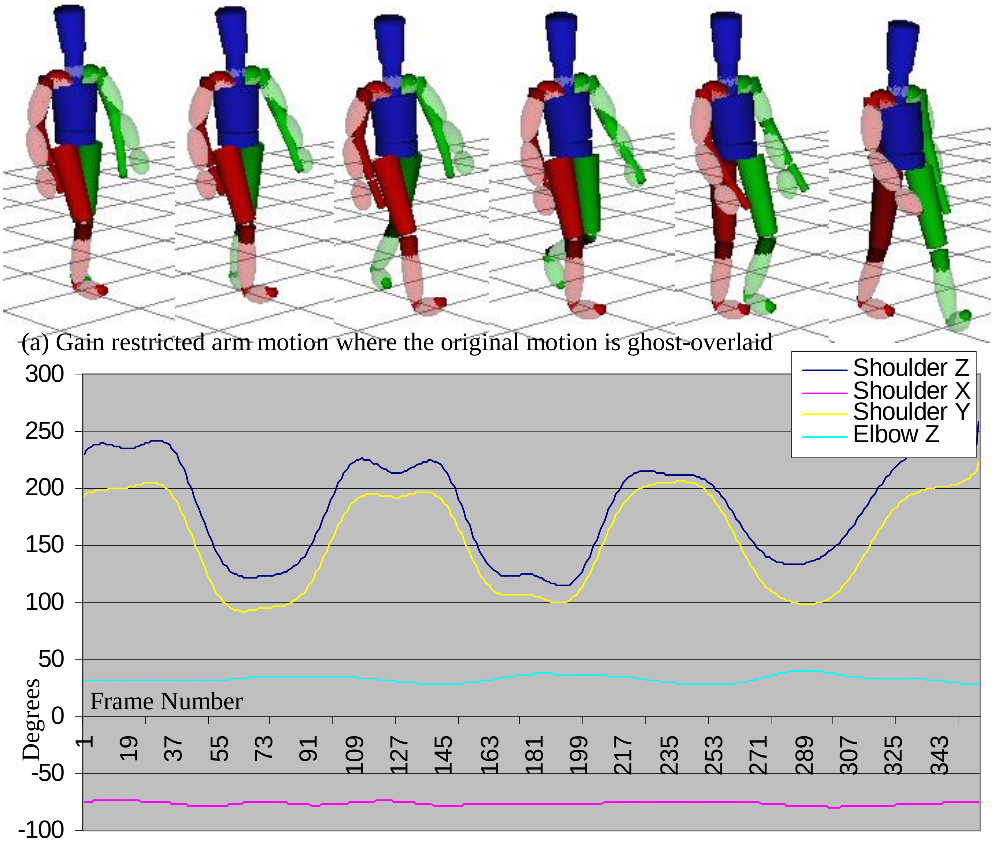
Figure 6.21: Arm DOF trajectories for actor (a) C & (b) B as they walking normally

6.2.5.1 Upper Body Inter-Muscle Weighting Ratios

When inter-muscle ratios are introduced into the minimisation function for the arms in the gait motions, there is negligible perceived difference between varying ratios. This is because of the small degree of motion and hence very little muscle energy attributed to the lower arm for the optimisation process to have an influence on. However, in motions where there is more deviation in the elbow joint, similar results to those demonstrated for the legs would be expected. This is because the effect of changing the inter-muscle ratios is to mathematically dampen joint trajectories. Therefore, if the elbow joint were involved in a more energetic motion there would be more force to differently dampen the effects of the shoulder and elbow joints as was seen in the case of the legs. Therefore, although the example motions in the dataset of Chapter 3 prohibit the demonstration of this property, it is still completely conceivable that the desirable consequence would be obtained in that the appearance of how the character moves its arms can be controlled at a fine level of detail, i.e. whether it favours its upper or lower arm depending on its structure/mass.

6.2.5.2 Upper Body Muscle Gain Restrictions

The effect of applying muscle gain restrictions on the upper body is to reduce the amplitude of the arm swing, which is analogous to that of reducing the foot stride length for the lower body. This occurs because the muscle forces peak at the bottom point of the swinging motion in order to project the arm back up against gravity, which is therefore the portion of the motion that is being restricted. Figure 6.22 illustrates this process by applying an 80% muscle force limit on the shoulder compared to the maximum shoulder exerted force of the control motion.



(a) Gain restricted arm motion where the original motion is ghost-overlaid

(b) Right arm joint trajectories of the gain restricted motion from (a)

Figure 6.22: Muscle gain restrictions applied to the arms of actor C’s normal walking motion (a) illustrates the first stride cycle of the motion where the original motion is ghosted over the solid, reconfigured character and (b) gives the trajectory plot for the right arm over the complete motion

Comparing the joint trajectories of Figure 6.21a and Figure 6.22b show that not only has the overall amplitude of the arm swing been reduced, but also the sharpness with which the joint trajectories change has been smoothed over. This is a consequence of the reduced muscle gains that prohibit the large forces that are required at these points for the sudden changes in direction. However, the relatively large plateau regions at the height of the range of motion are still apparent in the reconfigured motion, which shows preservation towards the original swing style due to the minimisation function attempting to maintain similar joint space configurations between the original and simulated motion. This plateau region of actor C’s is compared to the much smoother sinusoidal motions of actor B’s arm motion, which is illustrated in Figure 6.21b.

6.2.5.3 Balancing the Components of the Optimisation Function

The balance between the muscle force and joint trajectory minimisation aspects of the optimisation function has a more influential outcome on the motion of the upper body than it did for the lower body. This is because of the reduced amount of contact and friction constraints that resulted in deeper local minima for the legs. Since the local minimums for the upper body are much more shallow, when the motion of the arms are considered, the initial values are less likely to fix in a solution via the constraints and hence the optimisation function plays a more important role in the final motion. The consequences of this are two-fold: the first is that the upper body portion of the equations are much more stable to larger changes than the lower part, and the second is that the change affected by the muscle gain restriction can also be approximated by changing the ratio between the two competing components of the optimisation function. However, adjusting the ratio between the muscle and joint trajectory minimisation components of the optimisation function are not as predictable or reliable as the muscle gain restrictions due to their placement in the optimisation process. As the difference between the inter-muscle ratios (which was purely minimisation function) and gain-restricted (which was constraint-based) motions for the legs illustrated, a more reliable method of effecting a change is to encode the difference in the constraint portion of the optimisation process as opposed the minimisation function. Furthermore, it is preferable to maintain the use of muscle gain restrictions over changing the optimisation function balance, as the former is more biomechanically correct than the latter and it also appeals more to the adjustment of the legs.

6.3 Mapping the Motion of one Actor to Another using Dynamics

In this section, the concepts discussed in section 6.1 for dynamically retargetting motion to different sized characters and the section 6.2.4 for effecting biomechanical mass changes are combined together to produce a completely reconfigured motion. This is achieved by taking the base motion for a particular actor from the dataset described in Chapter 3 and by setting the biomechanical properties of the other actors their simulated motions are generated. The biomechanical information used for each of the target actors is given in Appendix D.2. The simulated motions are compared against the real motion of the target actor to provide an evaluation of the dynamics-based reconfiguration technique.

The following motion conversions are discussed in this section:

§ Mapping the normal walking motion of actor C to actors A, B & D

§ Mapping the normal walking motion of actor B to actor C

§ Mapping the tight left turn motion of actor A to actors B, C & D

For each of the reconfigured motions, the inter-muscle weighting ratio is kept constantly even. Additionally, when transferring the motion to actors A, C and D no gain restriction is imposed because they share a similar level of energy in their motions. However, because actor B’s real motion is much less energetic than the other actors, a muscle gain restriction of 80% is imposed on the femur and humerus limbs.

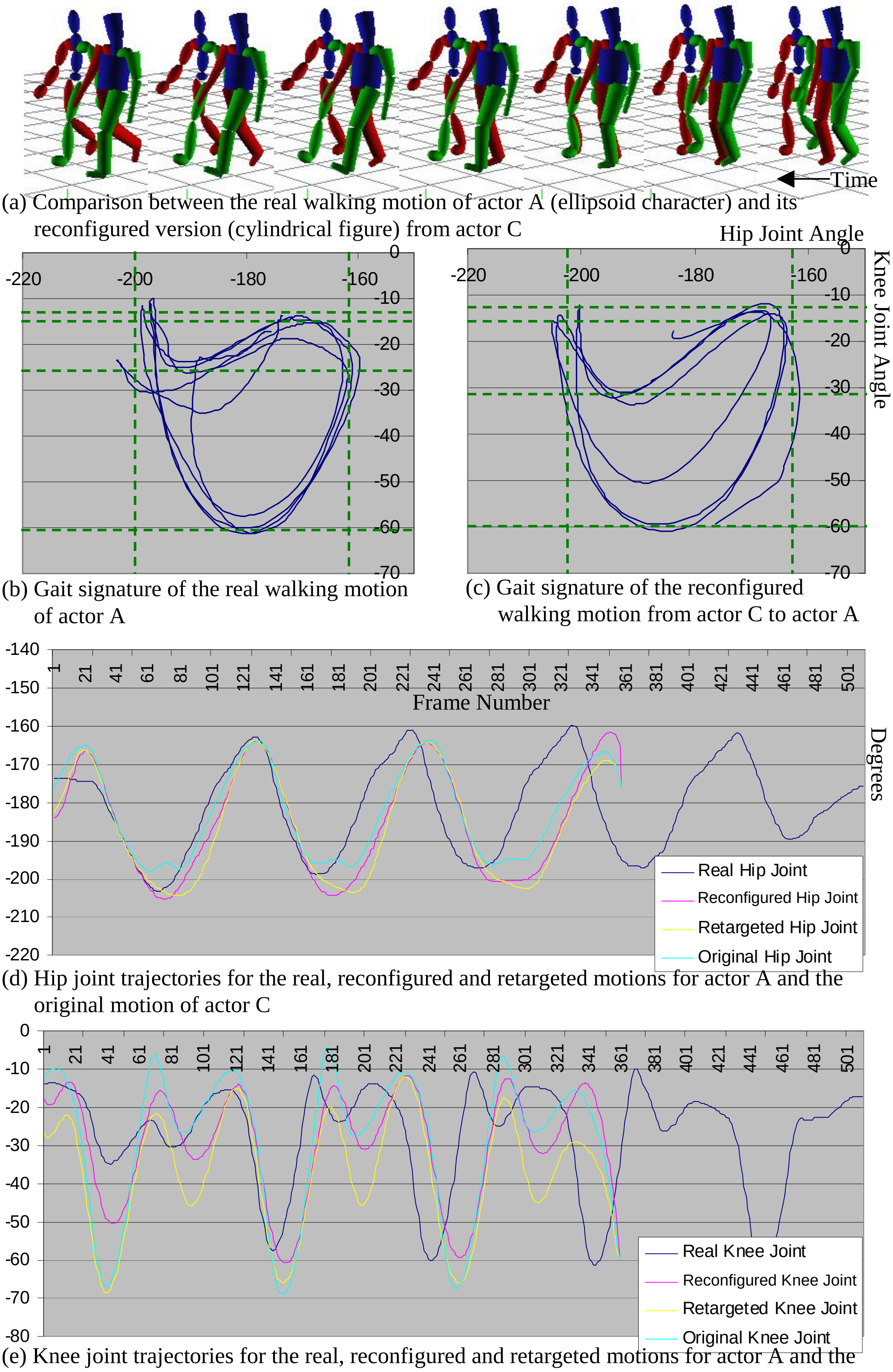
6.3.1 Mapping the Normal Walking Motion of Actor C to Actors A, B & D

The results obtained from mapping the normal walking motion of actor C to actors A, B and D are illustrated in Figures 6.23 through to 6.25. The joint trajectories of these figures include the following motions:

- The real walking motion of the target actor,
- The reconfigured/simulated walking motion for the target actor,
- The retargeted walking motion for the target actor (using the target actors limb lengths but with the muscles in a passive state as described in section 6.1),
- The original walking motion of actor C

Although the original base walking motion joint trajectories of actor C are included for completeness, for comparison purposes the retargeted motion is taken as the base level because this forms the most fundamental operation on the character to make it appear realistic in its environment by removing any foot sliding. Furthermore, in the joint trajectory curves of Figures 6.23 through to 6.25, the real joint paths do not align with the three other curve plots. This is because the speed of the retargeted and reconfigured motions closely resembles that of the example motion due to the lack of time warping within the dynamics solution. Therefore during the evaluation of the motion mapping process from one character to another, the periodicity of the curves are largely ignored and it is the detail within the strides that are considered as it is the more fundamental aspects between the real actor and the reconfigured motion that are of importance in the evaluations.

Figure 6.26 collates the gait signatures of Figures 6.23 through to 6.25 to highlight both the differences between the original motion of actor C and the reconfigured motions, and the similarities between the reconfigured and real motions of actors A, B and D.



(a) Comparison between the real walking motion of actor A (ellipsoid character) and its reconfigured version (cylindrical figure) from actor C

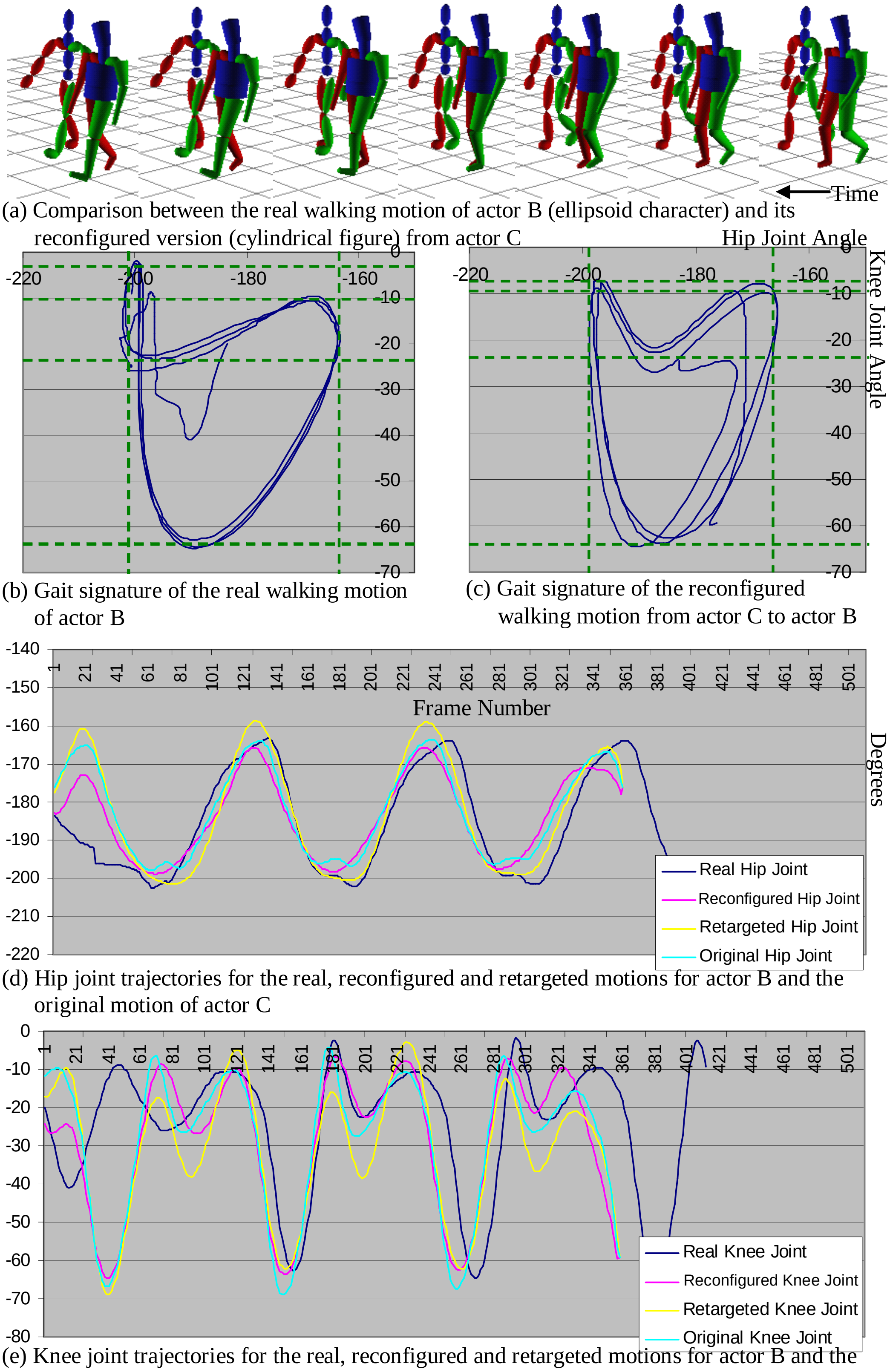
(b) Gait signature of the real walking motion of actor A

(c) Gait signature of the reconfigured walking motion from actor C to actor A

(d) Hip joint trajectories for the real, reconfigured and retargeted motions for actor A and the original motion of actor C

(e) Knee joint trajectories for the real, reconfigured and retargeted motions for actor A and the original motion of actor C

Figure 6.23: Actor C to Actor A: Dynamically-simulated mapping of the normal gait motion of actor C to actor A using biomechanical data to drive the modification to the new actor. (a) Illustrates the reconfigured motion compared to the real one for actor A, (b) gives the gait signature of the real walking motion of actor A, (c) gives the gait signature of reconfigured motion from actor C to actor A, and (d) & (e) graph the hip and knee joint trajectories respectively of the real, retargeted and simulated motion for actor A and the original base motion of actor C



(a) Comparison between the real walking motion of actor B (ellipsoid character) and its reconfigured version (cylindrical figure) from actor C

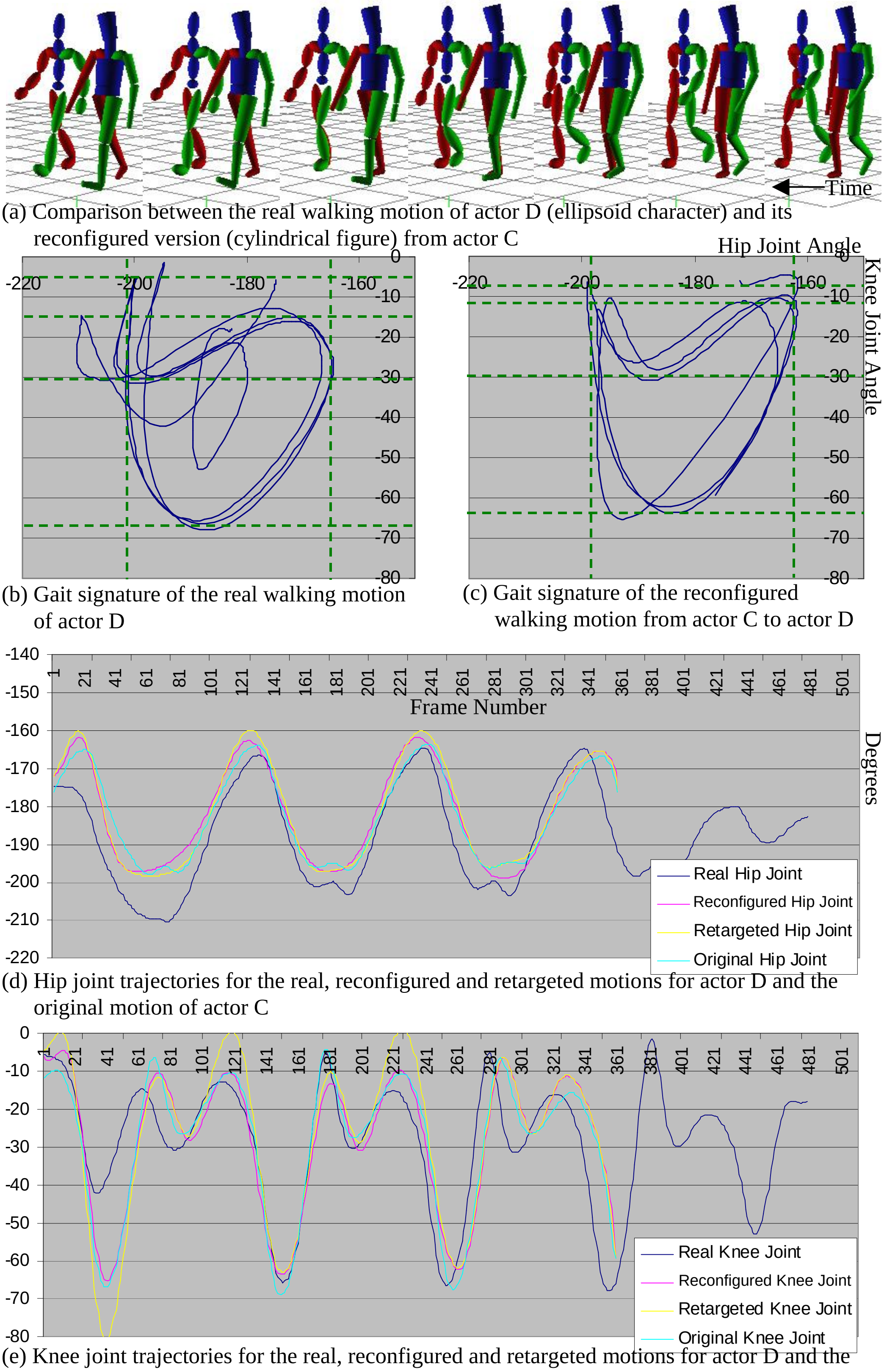
(b) Gait signature of the real walking motion of actor B

(c) Gait signature of the reconfigured walking motion from actor C to actor B

(d) Hip joint trajectories for the real, reconfigured and retargeted motions for actor B and the original motion of actor C

(e) Knee joint trajectories for the real, reconfigured and retargeted motions for actor B and the original motion of actor C

Figure 6.24: Actor C to Actor B: Dynamically-simulated mapping of the normal gait motion of actor C to actor B using biomechanical data to drive the modification to the new actor. (a) Illustrates the reconfigured motion compared to the real one for actor B, (b) gives the gait signature of the real walking motion of actor B, (c) gives the gait signature of reconfigured motion from actor C to actor B, and (d) & (e) graph the hip and knee joint trajectories respectively of the real, retargeted and simulated motion for actor B and the original base motion of actor C



(a) Comparison between the real walking motion of actor D (ellipsoid character) and its reconfigured version (cylindrical figure) from actor C

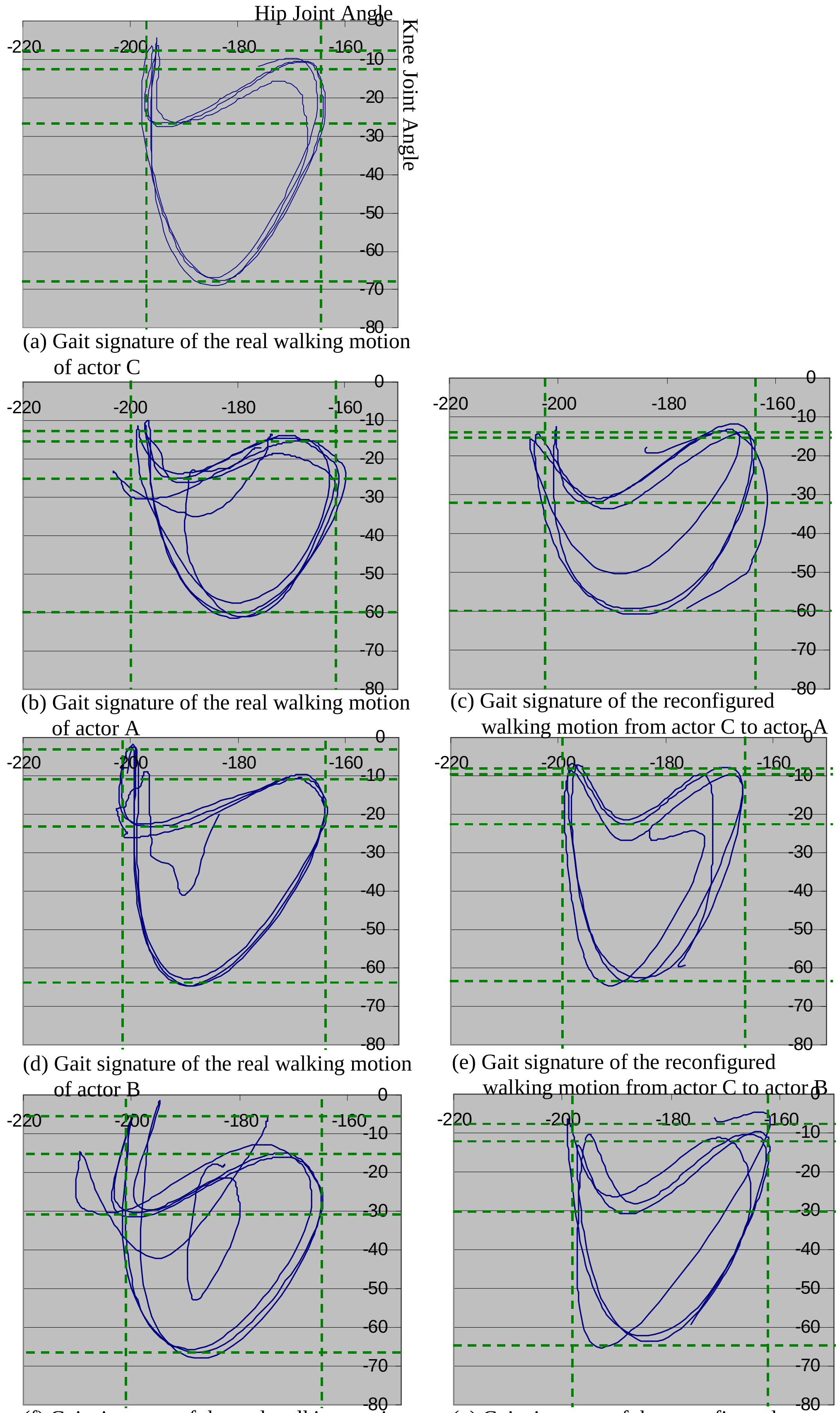
(b) Gait signature of the real walking motion of actor D

(c) Gait signature of the reconfigured walking motion from actor C to actor D

(d) Hip joint trajectories for the real, reconfigured and retargeted motions for actor D and the original motion of actor C

(e) Knee joint trajectories for the real, reconfigured and retargeted motions for actor D and the original motion of actor C

Figure 6.25: Actor C to Actor D: Dynamically-simulated mapping of the normal gait motion of actor C to actor D using biomechanical data to drive the modification to the new actor. (a) Illustrates the reconfigured motion compared to the real one for actor D, (b) gives the gait signature of the real walking motion of actor D, (c) gives the gait signature of reconfigured motion from actor C to actor D, and (d) & (e) graph the hip and knee joint trajectories respectively of the real, retargeted and simulated motion for actor D and the original base motion of actor C



(a) Gait signature of the real walking motion of actor C

(b) Gait signature of the real walking motion of actor A

(c) Gait signature of the reconfigured walking motion from actor C to actor A

(d) Gait signature of the real walking motion of actor B

(e) Gait signature of the reconfigured walking motion from actor C to actor B

(f) Gait signature of the real walking motion of actor D

(g) Gait signature of the reconfigured walking motion from actor C to actor D

Figure 6.26: Comparison of gait signatures for the reconfigured walking motion of (a) actor C to (c) actor A, (e) actor B, and (g) actor D. The original gait signatures of actor A, B and D are given in (b), (d) and (f) respectively

6.3.1.1 Reconfiguring Actor C to Actor D

Beyond the periodicity, the joint trajectories of actor D in Figure 6.25 show that the reconfigured motion curves better approximate the real actors compared to the retargeted version or original motion. This is especially noticeable in the large knee joint range apparent in the retargeted version, which is reduced in the reconfigured motion due to the relative mass differences in the lower leg. Interestingly, the original knee joint trajectory is closer to the real or reconfigured motion than the retargeted version. This is because the effect of eliminating the foot sliding has evenly distributed the change over the leg as opposed to bias its distribution according to the limb masses.

At the less negative range of the knee joint, it can be seen that the reconfigured motion better approximates a more constant bimodal curved shape, which corresponds to a foot plant, than either the original knee joint of actor C or the retargeted version. At the other end of the knee joint range (more negative), except for the first dip, each of the motions closely matches that of the real joint trajectory. Although the overall knee joint ranges are comparable between the real and reconfigured motions there is a slight offsetting of this range, which is due to the slightly different stride lengths used by the reconfigured and real motions.

In terms of the hip joint given in Figure 6.25d, there is little to choose between the original, retargeted and reconfigured trajectories. However, the top range of the curve (less negative) of the reconfigured motion is consistently closer to that of the real motion, as too it is bottom range (more negative), which is most noticeable on the final stride cycle. Similar to that of the knee joint, the overall hip joint range of motion is not quite aligned between the reconfigured and real motion, however the ranges are equivalent. Despite the slight phase shift of the joint range of motions, a good similarity between the hip and knee joint ranges of the real and reconfigured motions has been achieved, which can be seen by comparing the outlying minimum and maximum markers of the gait signatures in Figure 6.25b and Figure 6.25c.

On first appearances, the gait signatures of the real and reconfigured motions of Figure 6.25b and Figure 6.25c don’t appear to be very similar, primarily because of the lacking sharp point in the top left region of the path, which slightly doubles back on itself in the real motion but stretches out on the reconfigured motion. During section 6.2, this characteristic absence in the gait signature was indirectly attributed to the lack of time warping effect within the dynamics optimisation process in conjunction to the different stride lengths. More direct evidence for this can be seen by comparing across the gait signatures of the actors, which are presented in Figure 6.26. Figure 6.26 shows that each of the reconfigured motions stretches out this characteristic in the gait signature compared to the real motion. However, this is less significant in the case of actor A, whose sharpness is not as well defined as those of actors B and D. Ranking the degree of overlap, or sharpness seen in the real gait signatures results in the order A, D, B, starting with the least amount. This ranking is also the same ordering of the walking speeds from highest to lowest, which are given in Figure 6.27. Additionally, with decreasing overall body speed (ordering A, D, B), there is an increasingly noticeable perturbation at the bottom of the hip curve (most negative) as the foot is in contact with the ground for a longer period of time. This would directly account for the sharpness in the real gait signature, however it is a feature not being captured by the dynamics optimisation process because of the quick moving pace of actor C's base motion.

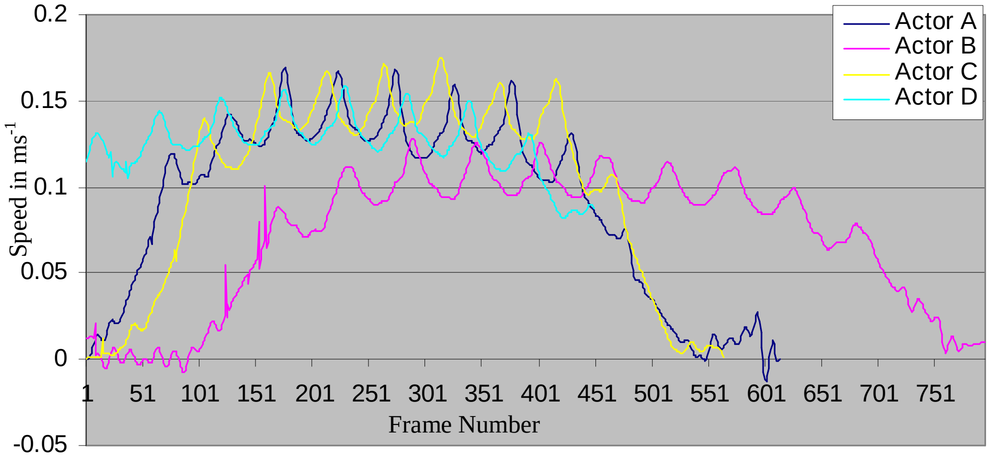


Figure 6.27: Speed of motion of the hips for actors A, B, C and D performing their real walking motions

Unfortunately, because the dynamics-based solution does not support time warping the example motion, this link cannot be explored any further. It is indeed questionable whether modifying the timing parameters will result in what appears to be very subtle differences between slow and fast walking motions; although a correlation can be demonstrated, the dataset used is very small and the relationship could just be coincidence where in fact these changes are unique to the manner in which the individual walks – something that biomechanics and dynamics alone cannot simulate. However, later in section 6.3.2, when a motion transfer is done between actor B to actor C, the sharpness of this characteristic is maintained to a high degree (actor B being the slowest walking motion). This further supports the claim that this gait signature feature is attributed to the speed of motion (including stride length) as opposed to purely a personal character trait.

Aside from the absence of the sharp kink in the gait signature of Figure 6.25c of the actor C reconfigured motion to actor D, the reconfigured pattern closely matches the dimensions of the real gait signature of actor D in Figure 6.25b. This includes the angle ranges and the degree to which the knee joint varies along the top part of the gait signature (between the first and third horizontal green-dash local minimum/maximum lines). The similarity between the top part of the gait signature correlates with the joint trajectories, where, unlike the retargeted knee joint, the reconfigured curve better mimics the bimodal pattern of the real knee trajectory in terms of its range (which is the temporal region that corresponds to the top portion of the gait signature).

From a mathematical point of view, the dynamics optimisation process has done a good job at bringing the reconfigured motion inline with the real motion, especially when compared to the base level retargeted motion. This is further complemented by the visual outcome of the reconfigured motion, which is also very promising, as can be seen in the accompanying animation file for Figure 6.25 that simultaneously plays back the real and reconfigured motions for actor D.

6.3.1.2 Reconfiguring Actor C to Actor A

The promising results demonstrated in Figure 6.25 are further complemented by the outcome of mapping actor C to actor A, which is given in Figure 6.23. In this example, the real motion of actor A has a slightly quick stride cycle than the original of actor C. However, Figure 6.27 also shows the overall speed of actors A and C to be similar which is due to actor A's shorter leg length compared to actor C. Therefore, the joint trajectories of the original, retargeted and reconfigured motions of Figure 6.23 appear much less aligned then those encountered for actor D in Figure 6.25. From the knee joint trajectories in Figure 6.23e, the original motion can be seen to have a much greater range than the real one, which extends beyond the real trajectory equally at both ends of the spectrum. The base level retargeted motion reduces the upper bounds of the curve (least negative), however maintains much the same lower end range (more negative). The reduction in the upper bounds of the retargeted motion actually overshoots the trajectory of the real motion as the range is reduced, which additionally transfers through to the hip joint trajectories. The real joint trajectory for the knee lies in between the original and retargeted version and it is this region that the reconfigured curve resides. Furthermore, with the exception of an increase dip between the bimodal feature of the real actor's knee joint trajectory, the reconfigured curve closely matches that of the real actor, taking into consideration the gait timing.

The very similar match between the real and reconfigured motion for actor A is further illustrated in the gait signatures of Figures 4.40b and 4.40c which, compared to the knee joint range of both actors B & D, demonstrates that individual characters have unique ranges of motion. This is a feature that the dynamics optimisation process has capture well. However, the reconfigured hip joint range is slightly more offset for actor A as it exhibits a slightly lower minimum value compared to the real motion. This is due to the difference in stride length between the reconfigured and real motion, which is slightly less for actor A than actor C. It is a consequence of the increased stride length of the reconfigured motion that a larger range of motion for the top portion of the knee joint in the gait signature is noticed. This correlates to the deeper valley region between the bimodal knee joint features of Figure 6.23e. As section 6.2.3 demonstrated, the stride length can be reduced by introducing slight gain restrictions on the character's femurs, which would influence the range of motion in this portion of the gait signature. However, this would also have the effect of reducing the overall speed of the walking motion, which, when taking into consideration the quicker stride time but smaller stride length, is just about right for the real actor's motion, as Figure 6.27 shows, i.e. the real motion speeds of actor A and C are very similar.

From both the joint trajectories and gait signatures of Figure 6.23 it can be seen that reconfigured motion is very similar to the real motion, especially compared to both the original and retargeted motions. This mathematical similarity for the reconfigured motion again translates well into a visual motion when compared to the real motion of actor A, which can be seen in the accompanying animation for Figure 6.23.

6.3.1.3 Reconfiguring Actor C to Actor B

The final motion of this set looks at the normal walking motion of actor C mapped to the biomechanics of actor B, which is presented in Figure 6.24. Recall also that this motion has an imposed muscle gain restriction on the reconfigured character to account for the much slower pace of the real actor's motion. This introduces different foot plant locations between the retargeted motion and reconfigured motion. This extra biomechanical modification to reconfigure the motion to actor B is justified by comparing the knee trajectories of Figure 6.24e. In Figure 6.24e, the retargeted joint track is very different to the original, however the application of restricting the leg muscle gains demonstrates an effective result in bringing its trajectory much closer to that of the real actor. The reconfigured motion also consists of a range of motion in both the knee and hip joints that most closely resembles that of the original actor, much closer than either the original or retargeted joint paths. The reconfigured motion additionally maintains a much more familiar range of motion to the real one in the bimodal region of the knee joint path, which is again much closer than either the original or retargeted version. This can be seen in both the knee trajectory curves and the gait signatures of Figure 6.24.

However, similar to the other two reconfigured motions, the gait signature in Figure 6.24c for actor B demonstrates a fundamental difference between the original and reconfigured motion in the top left region of the pattern. This is more apparent in this reconfigured motion than the other actors because the real gait signature of Figure 6.24b contains a very distinctive pattern that doubles back upon itself in this region. This is again attributed to the lack of time warping facility within the dynamics-based process and because the reconfigured motion is moving much more quickly than the real motion. Underneath this difference in pattern, the top left portion of the real gait signature of actor B shows that the actor tends to almost straighten his leg to the point approaching zero degrees, practically locking it out at each step. This feature corresponds to the left of the bimodal peaks in the knee trace of Figure 6.24e. However, in the reconfigured motion, although this limit is better approach than the original or the retargeted version, it is still approximately 5 degrees away from the real motion.

Despite there only being a few degrees between the reconfigured motion and the real motion of actor B, the way the real actor nearly locks out his legs during walking produces a very visually distinct gait that is not replicated in the reconfigured motion as well as with the previous examples. The actual motion of the real actor is in parts more akin to that produced using large variance inter-muscle ratios to restrict the movement of the knee until absolutely necessary (see section 6.2.2). This would indicate that it might be possible to adjust the ratios for this reconfiguration to exaggerate the influence of the limb masses. However, this is refrained from in order to demonstrate the process from purely a biomechanical point of view. Consequently, this is the first major indication that has been seen of how the biomechanics alone cannot necessarily reproduce the motion of a real actor and that there are some personal attributes that are not captured by a purely dynamic simulation.

The dissimilarities between the real motion of actor B and the biomechanically reconfigured motion in Figure 6.24 do not end at just the leg straightening effect. Although the joint trajectories and the remaining portion of the gait signatures appear to be good matches, especially when compared to the original and retargeted joint traces, there are other key aspects that do not show up in the graphs, which become very apparent when viewing the accompanying animation file. In terms of the upper body arm motion, the reconfiguration process desirably reduced the arm swing amplitude to bring it closer to that of the real motion, however the motion of the real actor holds his arms much further away from his body and depicted in the reconfigured motion. This is despite the technical fact that even though the reconfigured arms are swung beside the body much more closely, they do not impact with it. This suggests it is simply the style of the actor. To induce this effect in the reconfigured motion the initial values of these joints can be adjusted such that the arms are rotated away from the body a little more without affecting the final muscle forces too as their contributions are insignificant compared to the rest of the swinging arms (plus the rest state variable would absorb much of the change). However, the adjustment is again refrained from because it is not one that is done automatically by the system based on the example motion and the biomechanics of the new actor, which is what is being evaluated in this section.

When comparing the playback of the real motion compared to the reconfigured one of Figure 6.24 (which is given by the accompanying animation file) a similar effect to the arms is noticed in the legs, where they are also spaced further apart in the real motion than in the reconfigured gait. This is again despite the technical fact that they do not impact during walking when they are spaced as they are for the reconfigured motion. Again, this issue can be artificially fixed by increasing the amount of rotation in the upper leg away from the body in the initial value determinations. Therefore, there is scope to manoeuvre around such issues, however these are very much artist driven as opposed to purely from the biomechanics of the actor.

The reconfiguration of actor C to actor B in Figure 6.24 has illustrated that the joint trajectories in the axis of major movement can be appropriately reproduce. However, unlike the previous examples, this does not visually translate well to be a good matching motion for the real actor. This is because of the difference in the minor axis of rotation, which, to all intense and purposes, take on the same configuration as the example motion because there is very little muscle force to minimise in these cases. An artist tweaking the initial values can overcome the issues, however these are not necessarily based on biomechanical properties, but more stylisations, which is where the dynamics-based modification process starts to fail.

6.3.1.4 Mapping the motion of actor C

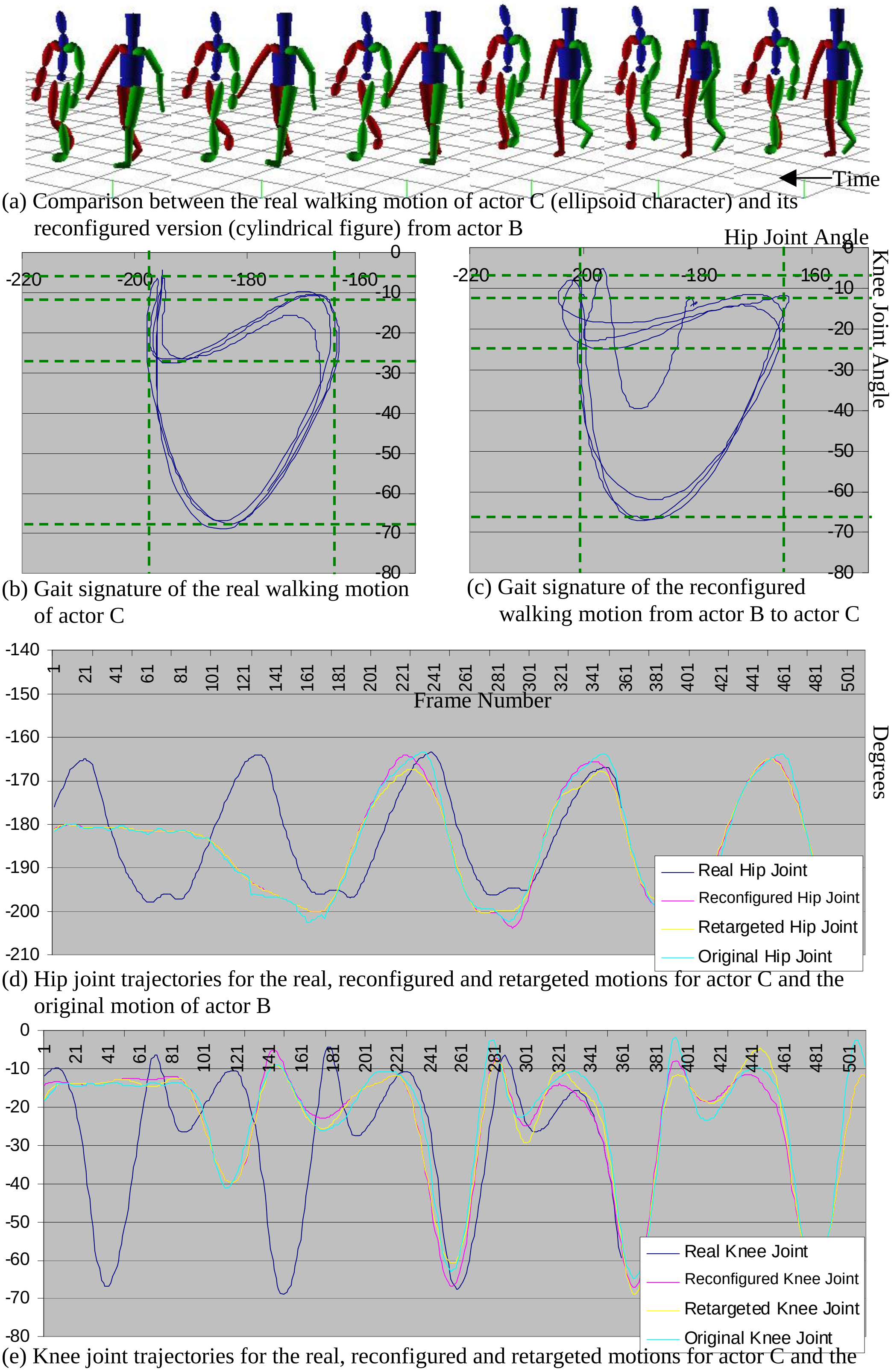
During the reconfigurations of mapping the motion of actor C to actors A, B and D, the target actors have always had an equally high or lower energetic appearance and the joint trajectories have been very similar to the real actor. This has generally resulted in plausible motions that the target actor could have made naturally. However, the optimisation process is effectively a dampening mechanism, not least because muscle forces are being minimised. Sometimes, this dampening has the effect of making other limbs move more, which has been desirable. However, the system is always biased towards reducing the energy in the system. This works fine if the starting motion is energetic, as in this section, but proves to be an issue if the example motion is less energetic than the target actor. For example, as opposed to going from actor C to actor B (more energetic to less energetic), which has been shown possible using the biomechanical adaptations of the dynamics solver, it were desirable to reconfigure the motion of actor B to actor C (less energetic to more energetic). This issue is discussed in the following section.

6.3.2 Mapping the Normal Walking Motion of Actor B to Actor C

Figure 6.28 illustrates the result of performing the reconfiguration of a normal walking motion from actor B to the biomechanical information of actor C, the reverse mapping of section 6.3.1 and Figure 6.24.

Figure 6.28 illustrates that both the joint trajectories and gait signatures of actor B's reconfigured walking motion to actor C show significant similarities compared to the real motion of actor C. That is however with the noticeable exception that the reconfigured hip joint has a slightly lower minimum negative value when compared to the real motion in Figure 6.28d. This is due to the different stride lengths between the example and real gait motions. The reconfigured motion also exhibits the little kinked region in the top left portion of the gait signature in Figure 6.28c, which was previously removed when considering the faster paced reconfigured motions of section 6.3.1. Indeed, the kinked region in this reconfigured motion loops back upon itself more than the real motion. This corresponds with the earlier discussions upon the overall speed of the motion having an effect because the reconfigured motion (which is based on the timings of actor B) is much slower than the real one, which can be seen from Figure 6.27. However, it is also plausible that the dynamics optimisation process has maintained this feature in the reconfiguration of Figure 6.28 more than it did for the reconfiguration processes from actor C (section 6.3.1), where the reconfigurations started with the motion of actor C, because of its predominance in the base motion. It is therefore still inconclusive as to whether the degree of loop back in this region of the gait signature is speed dependent or a style of the character.

The accompanying animation file for Figure 6.28 reinforces the similarity between the real and reconfigured motions, where the lower portion of the body does look very similar to that of the real actor, just moving more slowly. However, the motion of the upper body lets down the similarity achieved with the lower part because the real motion has a much greater swinging arm motion then that illustrated in the simulated motion. Although techniques have been discussed to adjust the motion of the arms, they have all given rise to a dampening effect compared to the base motion whereas what is needed here is to energise the motion.



(a) Comparison between the real walking motion of actor C (ellipsoid character) and its reconfigured version (cylindrical figure) from actor B

(b) Gait signature of the real walking motion of actor C

(c) Gait signature of the reconfigured walking motion from actor B to actor C

(d) Hip joint trajectories for the real, reconfigured and retargeted motions for actor C and the original motion of actor B

(e) Knee joint trajectories for the real, reconfigured and retargeted motions for actor C and the original motion of actor B

Figure 6.28: Actor B to Actor C: Dynamically-simulated mapping of the normal gait motion of actor B to actor C using biomechanical data to drive the modification to the new actor. (a) Illustrates the reconfigured motion compared to the real one for actor C, (b) gives the gait signature of the real walking motion of actor C, (c) gives the gait signature of reconfigured motion from actor B to actor C, and (d) & (e) graph the hip and knee joint trajectories respectively of the real, retargeted and simulated motion for actor C and the original base motion of actor B

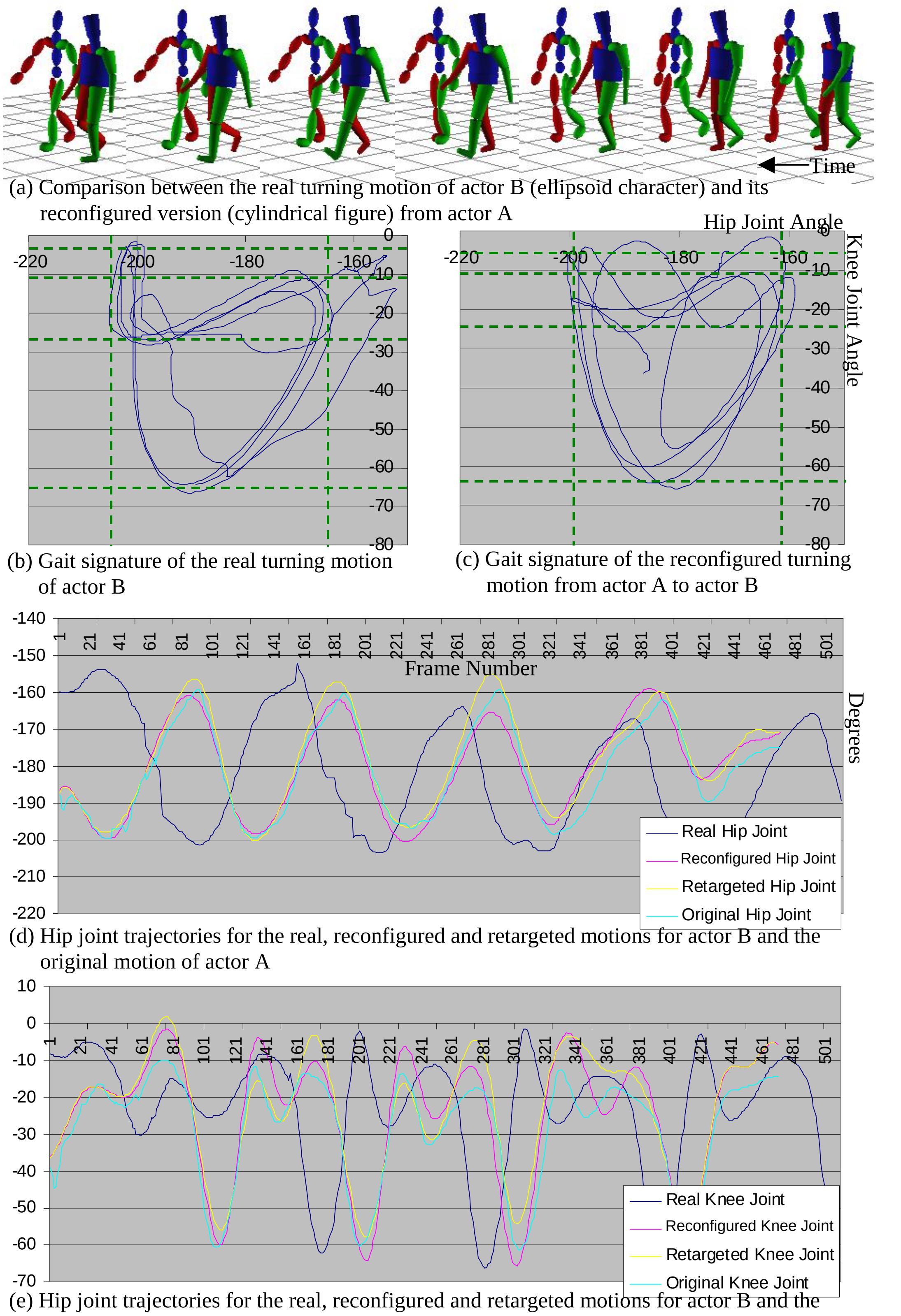
Even though a good lower body motion of the reconfigured gait of Figure 6.28 is achieved it is not indicative of a normal walking motion for actor C, but more of a relaxed version. Although the visual results and trajectory matches look good, the desirable aim has not been reached, i.e. from a normal motion of actor B, the target is a normal gait of actor C, and not a slower paced walking motion for actor C. This highlights a fundamental limitation of the current dynamics-based modification system of reconfiguring slower moving motions to actors who are more energetic in their gaits – effectively the system always requires a motion of higher or equal energy compared to that of the target actor as a base motion. This point is further discussed in section 6.5 during the general overview discussion of the reconfiguration process.

6.3.3 Mapping a Tight Left Turn Motion of Actor A to Actors B, C & D

The motion of actor A marking a sharp left turn (which is illustrated in Figure 3.1) provides the base motion for this section's reconfigured motions. The result of taking this motion and reconfiguring it to the biomechanics of actors B, C and D are illustrated in Figure 6.29 through to Figure 6.31 respectively.

The reconfiguration examples using the motion of actor A walking into a tight left turn demonstrate similarly plausible results as the reconfiguration of actor C's normal gait motion from section 6.3.1. This is best highlighted in both the accompanying animation files and the joint curve trajectory curves, where the reconfigured trajectories can be seen to much better follow the path of the real motion compared to both the original and retargeted traces. Due to the change in motion direction, and hence the irregular walking pattern, the gait signatures are somewhat more erratic than for the normal walking motion, so little conclusive information can be inferred by their comparison. They do still serve to show that a similar joint range for each of the reconfigured walks has been achieved compared to those of the real actor's. The slight exception to this however is the gait signature of Figure 6.29c, which corresponds to the reconfigured to actor B's biomechanical information.

In the reconfigured gait signature for actor B in Figure 6.29c, there is a slight angular offsetting within the hip joint, i.e. the mathematical range between the simulated and real motions of Figure 6.29 are equivalent, however the simulated hip joint starts at a less negative point on the scale. This is due to the relative hip and foot plant positions such that if the foot plants were moved slightly backwards while maintaining the hip location, the resulting hip range would accordingly adjust itself along the axis but still maintain the same overall range.



(a) Comparison between the real turning motion of actor B (ellipsoid character) and its reconfigured version (cylindrical figure) from actor A

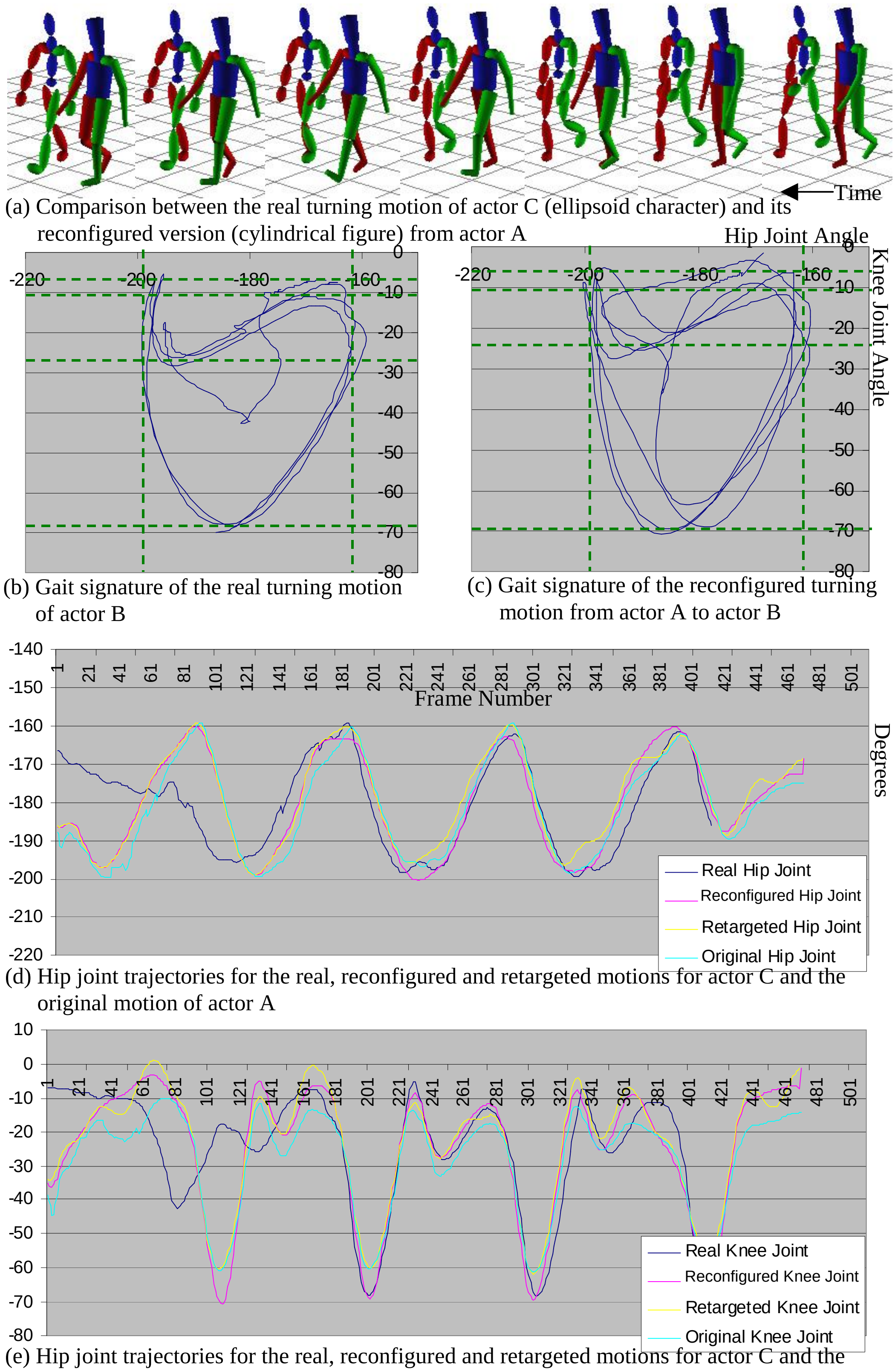
(b) Gait signature of the real turning motion of actor B

(c) Gait signature of the reconfigured turning motion from actor A to actor B

(d) Hip joint trajectories for the real, reconfigured and retargeted motions for actor B and the original motion of actor A

(e) Hip joint trajectories for the real, reconfigured and retargeted motions for actor B and the original motion of actor A

Figure 6.29: Actor A to Actor B: Dynamically-simulated mapping of a sharp left turn gait of actor A to actor B using biomechanical data to drive the modification to the new actor. (a) Illustrates the reconfigured motion compared to the real one for actor B, (b) gives the gait signature of the real walking motion of actor B, (c) gives the gait signature of reconfigured motion from actor A to actor B, and (d) & (e) graph the hip and knee joint trajectories respectively of the real, retargeted and simulated motion for actor B and the original base motion of actor A



(a) Comparison between the real turning motion of actor C (ellipsoid character) and its reconfigured version (cylindrical figure) from actor A

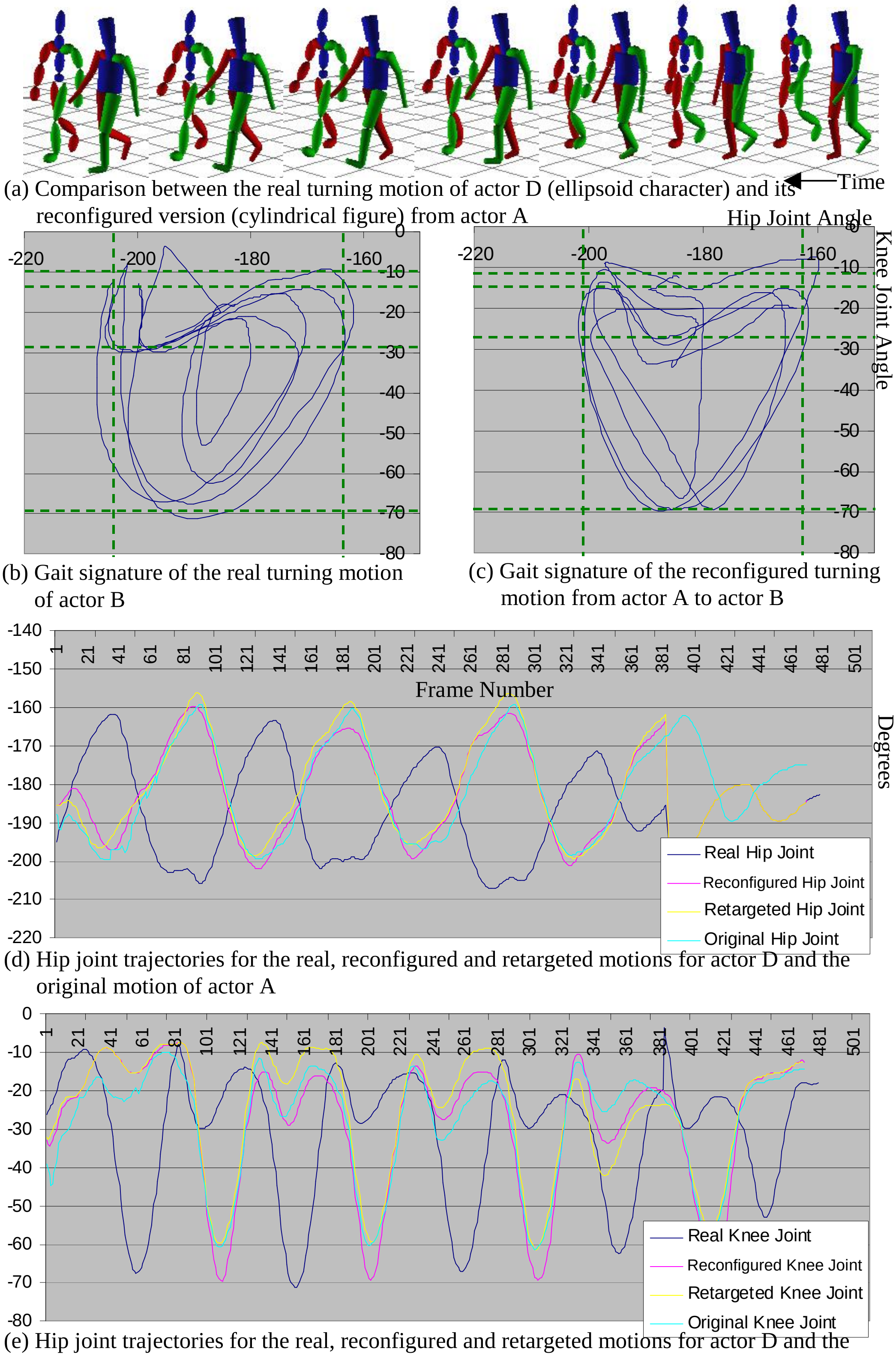
(b) Gait signature of the real turning motion of actor B

(c) Gait signature of the reconfigured turning motion from actor A to actor B

(d) Hip joint trajectories for the real, reconfigured and retargeted motions for actor C and the original motion of actor A

(e) Hip joint trajectories for the real, reconfigured and retargeted motions for actor C and the original motion of actor A

Figure 6.30: Actor A to Actor C: Dynamically-simulated mapping of a sharp left turn gait of actor A to actor C using biomechanical data to drive the modification to the new actor. (a) Illustrates the reconfigured motion compared to the real one for actor C, (b) gives the gait signature of the real walking motion of actor C, (c) gives the gait signature of reconfigured motion from actor A to actor C, and (d) & (e) graph the hip and knee joint trajectories respectively of the real, retargeted and simulated motion for actor C and the original base motion of actor A



(a) Comparison between the real turning motion of actor D (ellipsoid character) and its reconfigured version (cylindrical figure) from actor A

(b) Gait signature of the real turning motion of actor B

(c) Gait signature of the reconfigured turning motion from actor A to actor B

(d) Hip joint trajectories for the real, reconfigured and retargeted motions for actor D and the original motion of actor A

(e) Hip joint trajectories for the real, reconfigured and retargeted motions for actor D and the original motion of actor A

Figure 6.31: Actor A to Actor D: Dynamically-simulated mapping of a sharp left turn gait of actor A to actor D using biomechanical data to drive the modification to the new actor. (a) Illustrates the reconfigured motion compared to the real one for actor D, (b) gives the gait signature of the real walking motion of actor D, (c) gives the gait signature of reconfigured motion from actor A to actor D, and (d) & (e) graph the hip and knee joint trajectories respectively of the real, retargeted and simulated motion for actor D and the original base motion of actor A

What is being noticed between the reconfigured and real motions of Figure 6.29 is that the real actor prefers to plant his foot slightly further backward in relation to his hip position over the course of a stride than the simulated motion is doing. This issue relates to the fact that the example motion initially determines the hip and foot plant locations and, generally, unless a constraint forces the foot plant to move, those features from the example motion will persist through to the simulated one. This highlights another slight restriction in the dynamics modification technique. Nonetheless, to better recreate the motion of actor B, the foot plants can be moved backward slightly through the addition of user constraints. However, similar to the arm and leg spread adjustments, this would not really be a biomechanical adjustment, but more of an artist guided modification.

Aside from the comments on the slight differences and similarities between the real and reconfigured motions of Figure 6.29 through to Figure 6.31 that were made for the character mappings from actor C's normal gait, there are a couple of extra considerations brought forward from these reconfigurations. The first is that in the real motion of actor B (Figure 6.29), unlike each of the other actors, he turns left by pivoting on his right leg. This is in contrast to the other motions that perform this turn about their left. However, the simulated motion follows the base motion and turns on its left leg which consequently affects the joint trajectories in this region, thus making it unfair to directly compare the joint tracks between frames 225 and 300 of the reconfigured motion and frames 370 and 450 of the real motion. Ignoring the change in patterns around these regions, it can still be seen that the general cyclic stride of the motion reconfigured for actor B is close to that of the real actor's gait, and the turn itself still looks plausible for the real actor B. This can be seen in the accompanying animation file for Figure 6.29.

Figure 6.30, which demonstrates the reconfiguration of actor A's motion to that of actor C, again illustrates very plausible results, however it gives rise to the second point of interest that arises from this set of reconfigurations. During the optimisation process, the system of equations grew rather unstable near the solution, which is born out by the visual jerkiness about the foot plants of the reconfigured motion. This is despite receiving more curve refinement in the foot plant regions than the other reconfigured motions. Nevertheless, looking beyond the jerkiness in the animation file, the real and reconfigured motions still appear very similar, which is reinforced by considering the gait signatures and joint trajectories of Figure 6.30.

Notwithstanding these two issues that have arisen from the mapping process from actor A, each of the reconfigured motions are suitably different from the original motion to warrant it being a completely different style of gait in its own right. The joint trajectories and accompanying animation files takes this further by demonstrating that the reconfigured motions are in fact very similar to the real motion of the intended actor – much more similar than both the original and retargeted motions.

6.3.4 Actor Motion Mapping Summary

Throughout the examples of mapping motions between actors, the consistent trend has been for the joint trajectories of the reconfigured motion to better approach those of the real motion than either the retargeted or original movement. This process has been fully controlled by the biomechanics of the target actor. The similarity in joint trajectories have visually translated into a similar looking reconfigured motion to those of the real actor, especially when focusing on the lower portion of the body. This does imply that a biomechanical representation can, to a certain extent, be used to model the motion of a real actor, which also demonstrates the effectiveness of the dynamics optimisation process presented in this thesis. Although, as actor B most predominately showed, there is more to human motion than just the biomechanics of the character and some details cannot be automatically modelled. For example, the motions reconfigured using actor B's biomechanical information lacked the spread in the arms and legs that was shown in the real motions of actor B (Figure 6.24). This detail however can still be attained because the real actor's motion must still by physically correct but it requires the skill and eye of an artist to occasionally add that little extra.

Unfortunately, in some of the reconfigured motions the upper body movement has detracted from the effectiveness of applying the reconfiguration process to the legs. The reconfiguration of the legs has worked well because end-effector locations are fixed and hence the optimisation process distributes the changing joint angles during the motion based on the biomechanics of the actor. This is in contrast to the upper body, where there are no end-effector locations to maintain and hence the motion more closely aligns with the example motion presented to the optimisation process and the inter-joint relationships of the real actor are much harder simulate. This aspect of the optimisation process therefore needs further consideration, as does the case where our original motion is less energetic than the intended one; effectively the system needs to be injected with energy while still minimising it.

At a more fundamental level, the joint trajectories of the reconfigured motion are much closer to those of the real actor than those of the example base motion. Consequently, even if a perfect match for the real actor's motion is not obtained, completely new looking character motions have still been spawned from the base motion by merely changing the biomechanics of the reconfigured actor. This, in itself, is a notable goal because it achieves one of the fundamental aims of this work and is both controllable and intuitive, even more so than the novel individualisation/reconfiguration process of the weighted inverse kinematics approach from Chapter 4.

6.4 Dynamics-Based Injury Simulation

Although there is a very wide scope for performing dynamics-based motion modifications, this final dynamics-based modification section looks at how it can be used to successfully simulate injuries onto the example motion.

The process of simulating injuries is viewed as an asymmetrical application of attributes towards the different sides of the body; the injuries that are of interest only directly affect one side of the body. This is a view shared by other researches [Popo99, Liu05]. However, unlike these works, this section looks at how to simulate injuries by other means than just simply remove DOFs from the leg to restrict the movement. Since there is a wealth of potential modifications that could be applied to a character asymmetrically to produce abnormal motions that could depict injuries, it is those that have previously been discussed in this chapter that are considered in this section.

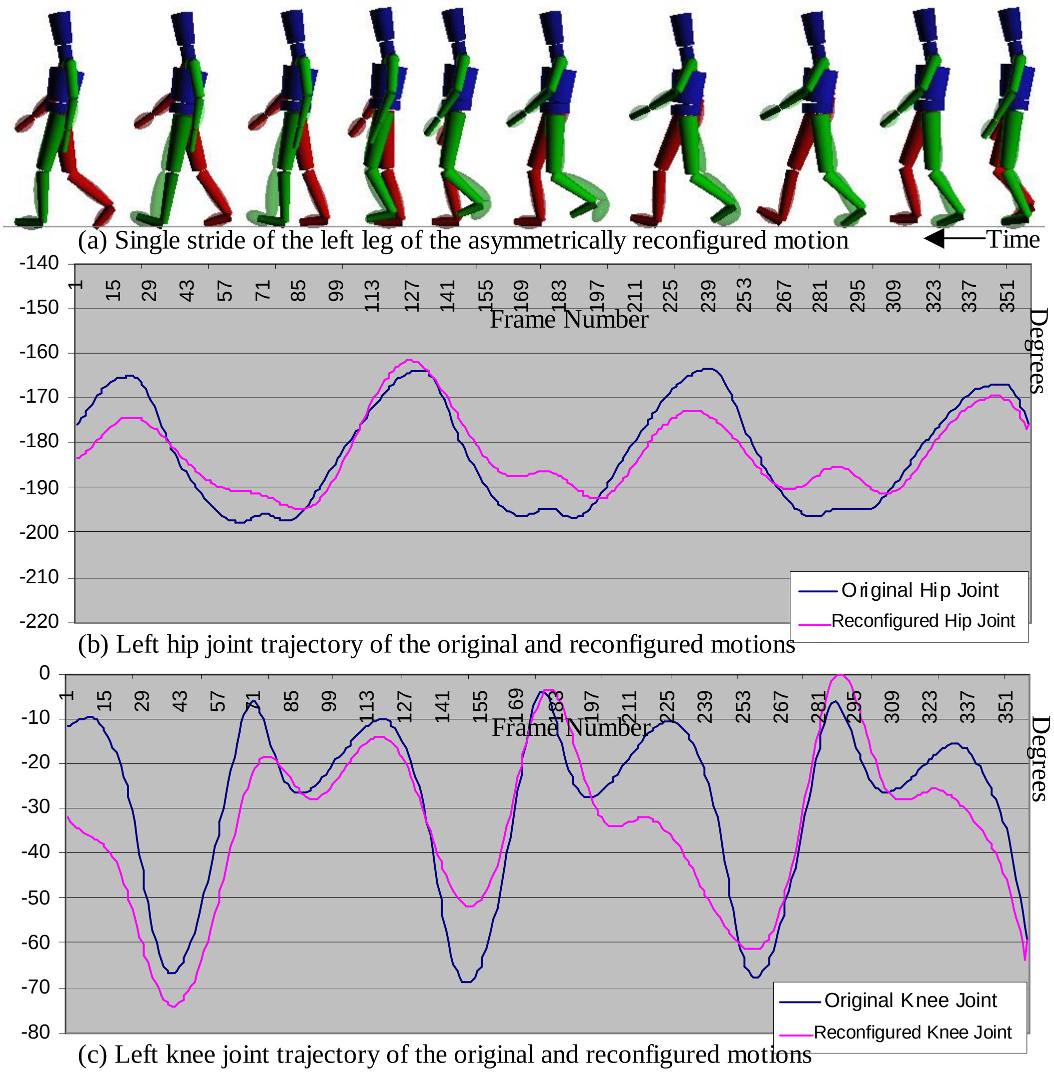
6.4.1 Injury Simulation via Asymmetrical Inter-Muscle Weighting Ratios

As opposed to applying identical inter-muscle weightings to both legs (see section 6.2.2), an inter-muscle weighting ratio with a large variance is applied to the left leg only, leaving the right leg with an evenly weighted ratio. The result of this modification is illustrated in Figure 6.32, where only the left leg is subjected to a ratio weighting of 20:1:1 for the femur, tibia and foot respectively.

The reconfigured motion portrayed in Figure 6.32a shows that the trajectory of the left leg through its flight path has been reliably adjusted in correlation with the inter-muscle adjustments that were seen in section 6.2.2. The outcome of the modification to the left leg is to significantly reduce the hip joint range, which has correspondingly adjusted the knee joint to compensate for the hip's repressed, but not prevented, range of motion. The change in knee motion is most predominately seen in its trajectory curve of Figure 6.32c where the second peak of the bimodal region of the original track is much reduced.

The changes in the hip and knee joints therefore fundamentally alter the flight curve of the left leg, which is illustrated in Figure 6.32a. However, the inter-muscle ratio for the right leg has been left unchanged with a ratio of 1:1:1. Therefore the right leg's flight corresponds with the more natural trajectory of the original motion, which Figure 6.32 shows to be very different from the reconfigured motion path of the left leg. The result of this is to produce a desirable asymmetrical motion that appears as if the left leg is walking more on its toes than the right leg. The effect of the motion walking on his left toes more than the right could be indicative of the motion having an upper leg injury as it is attempting to reduce the overall range of motion on this part of the body. This corresponds to the inter-muscle weightings that were applied to the left leg as it biases the minimisation function towards the femur and hence reduces its overall motion compared to the knee and foot joints.

While the use of the asymmetrical inter-muscle ratios illustrates a good step towards producing a motion that appears injured without actually removing DOFs from the motion, as in previous studies [Popo99, Liu05], it is not completely convincing. The reason for this is that the stride speed of the left leg remains the same as that of the right, whereas in reality it would be more usual to see a difference between them. However, as this chapter has demonstrated, the timings are reasonably fixed to the original motion due to a lack of time warping from the original motion. Although an overall speed adjustment can still be introduced into the left leg by imposing muscle gain restrictions. This reduces the stride length (see section 6.2.3) and hence indirectly changes the timings; recall that *speed=distance/time*. This adjustment is discussed in the following section.



(a) Single stride of the left leg of the asymmetrically reconfigured motion

(b) Left hip joint trajectory of the original and reconfigured motions

(c) Left knee joint trajectory of the original and reconfigured motions

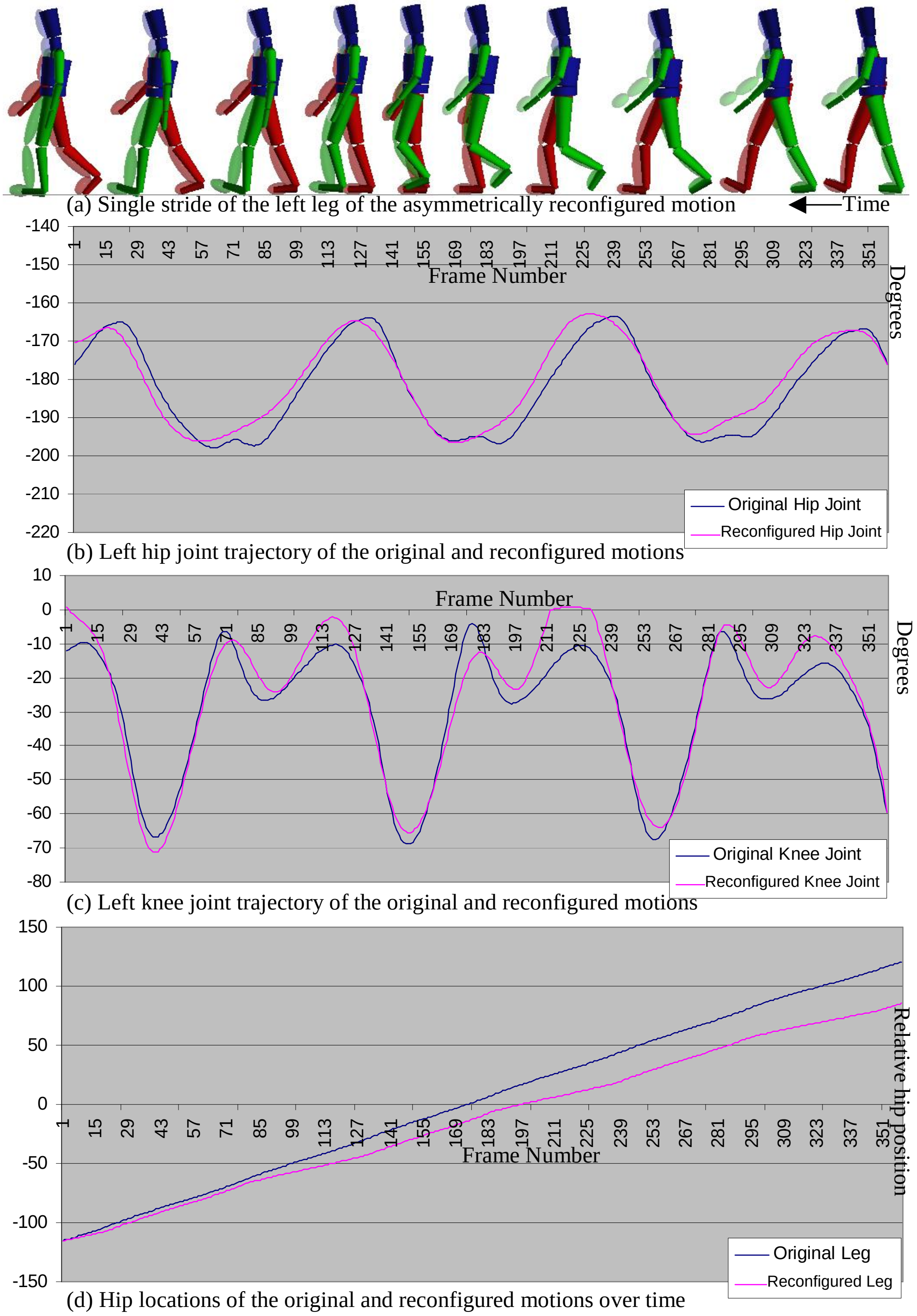
Figure 6.32: Injury simulation by applying different inter-muscle ratios between the two legs. The left leg is subjected to a ratio of 20:1:1 whereas the right is 1:1:1 for the femur, tibia and foot respectively. In (a), the original motion is depicted as the semi-transparent ellipsoid character

6.4.2 Injury Simulation via Asymmetrical Muscle Gain Restrictions

In this section, an asymmetrical muscle gain restriction is imposed on the base motion to portray an injured motion, where all other parameters remain unchanged to the standard reconfiguration process. Figure 6.33 illustrates the result of applying a gain restriction of 70% compared to the maximum exertable force of the base motion, to the left leg. The right leg is left unrestricted.

The character motion illustrated in Figure 6.33a demonstrates a successfully reduced stride length of the left leg, where by the end of the stride cycle the reconfigured character is lagging behind that of the original. However, the result of introducing the muscle gain restrictions has done little to change the pattern of the joint trajectories of either the hip of the knee.

Based on the joint trajectories of the hip and knee joints it would appear that the differences between the left and right legs are minimal. However, the hip location plot of Figure 6.33d shows that during the left leg foot strides, the hip curve has a reduced gradient than during the flight of the right leg. This shows that an overall reduction in the speed of motion has been achieved. The visual effect of this change is for the character to have shorter foot strides for the left leg, which is illustrated in Figure 6.33a.



(a) Single stride of the left leg of the asymmetrically reconfigured motion

(b) Left hip joint trajectory of the original and reconfigured motions

(c) Left knee joint trajectory of the original and reconfigured motions

(d) Hip locations of the original and reconfigured motions over time

Figure 6.33: Injury simulation by applying asymmetrical muscle gain restrictions; the right leg is left without bound, whereas the left leg is restricted to 70% of the reconfigured control motions maximum muscle gain. In (a), the original motion is depicted as the semi-transparent ellipsoid character, (b) and (c) give the joint trajectories of the hip and knee joints respectively and (d) gives a measure of the absolute position of the character’s hips over time

Although an indirect change in the timing of the character's left foot stride has been achieved, because the joint trajectories are similar between the original and reconfigured motions, the final motion simply appears as if the character is getting bogged down during the left leg strides and not really carrying any kind of injury. This is in contrast to more visually engaging injured motions where there is a distinguished changed between the knee and hip joint tracks of the original and reconfigured motions. To this aim, the effects of adjusting the inter-muscle weighting ratio and the muscle gain restrictions are combined to produce a hybrid limping motion that takes the desirable properties from the two modifications to produce a much better looking injured motion. This is discussed in the following section.

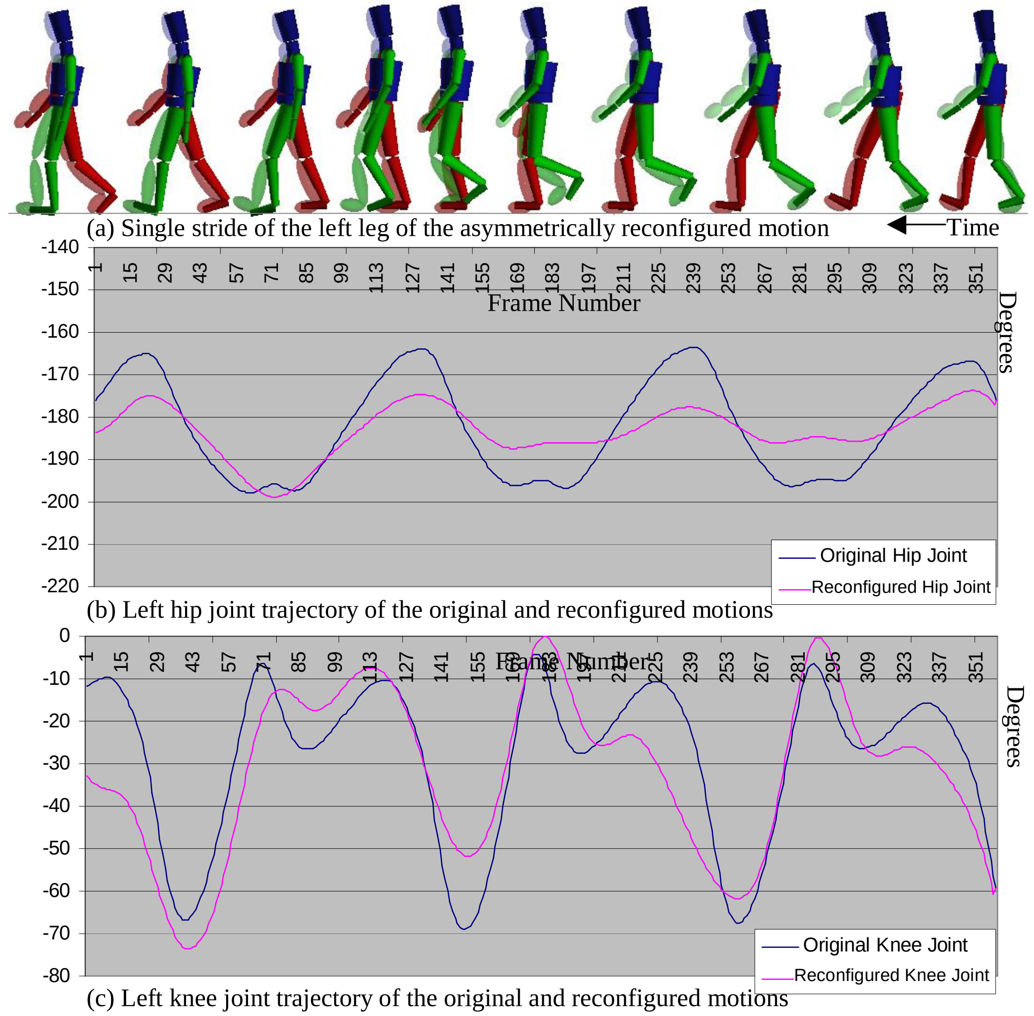
6.4.3 Injury Simulation via Asymmetrical Inter-Muscle Weighting Ratios and Muscle Gain Restrictions

To produce more visually appealing injured looking motions, both inter-muscle weighting ratios and muscle gain restrictions are applied to the base motion in an asymmetrical manner, which is illustrated in Figure 6.34. This motion is simulated using the inter-muscle ratio from Figure 6.32 (20:1:1) and the muscle gain restrictions used in Figure 6.33 (70% gain restriction) applied to the left leg only. The right leg remains subject to only the natural biomechanical consequences of the system as before.

The motion demonstrated in Figure 6.34 takes the reduced stride length from the muscle gain restricted component, which can be seen in Figure 6.34a, and the modified joint trajectories of the hip and knee joint, which can be seen in Figure 6.34b and Figure 6.34c respectively. What is most noticeable about the hip trajectory of Figure 6.34b is that the range of motion has been further reduced than the standalone inter-muscle ratio adjustment of Figure 6.32b. This is caused by the combination of the reduced stride length and the hip joint having a bias in the minimisation function. Therefore the optimisation function favours keeping the hip joint as steady as possible.

The gross changes that have been introduced in Figure 6.34 are supported by the joint trajectories of the same actor (actor C) walking with an imaginary left leg limp, which are presented in Figure 6.35.

The joint trajectories of the imaginary left leg limp in Figure 6.35 show the marked decrease in both the hip and knee joints compared to the right leg, which is the same that is seen in the reconfigured limping motion of Figure 6.34 (although only the left leg joint trajectories are shown in Figure 6.34, the shown original trajectory closely resembles that of the right leg of the reconfigured limping motion because there is little change to the original trajectory in this limb and the left and right leg flight paths are virtually identical – just phased shifted). Figure 6.35b does however also demonstrate a slight decrease in the overall range of motion of the right knee joint compared to the normal walking motion. This is due to the actor taking slightly shorter strides for the right leg as well as the left leg, but the left leg stride lengths are still much shorter.



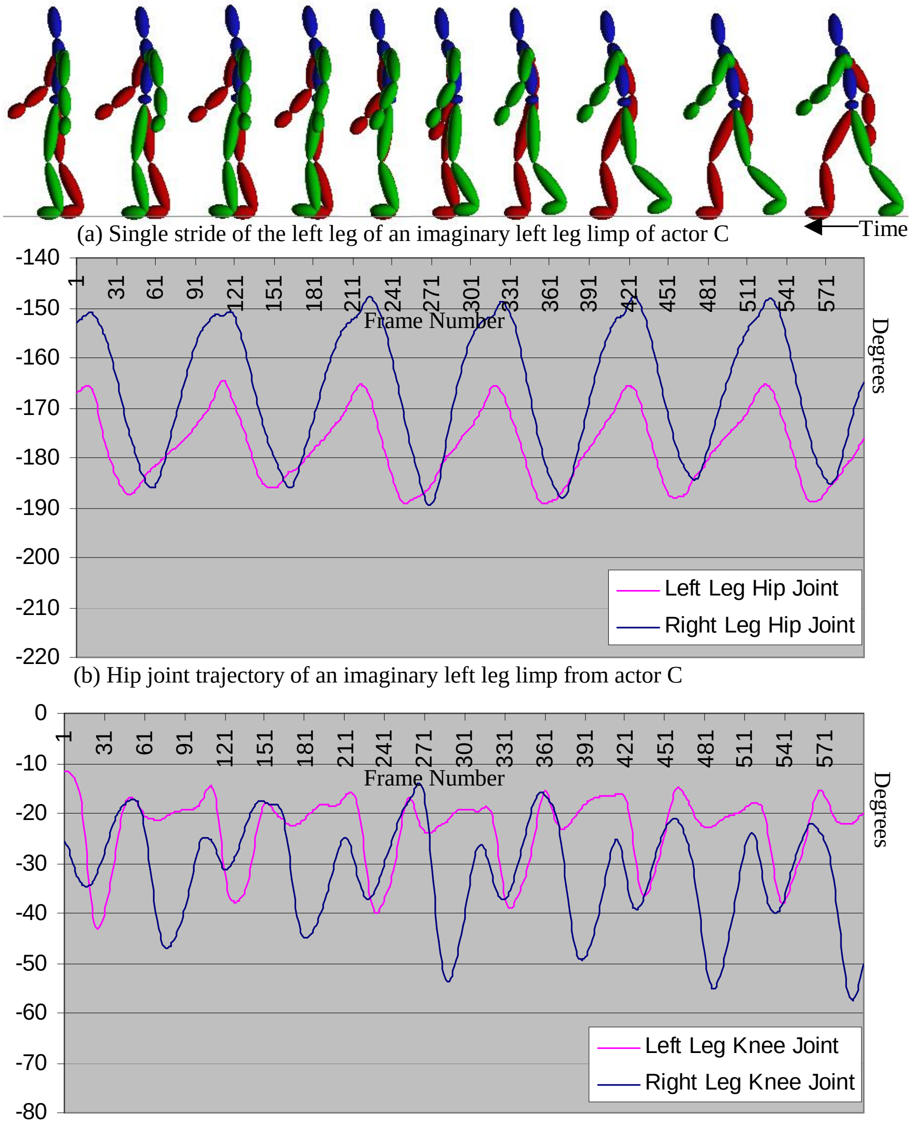
(a) Single stride of the left leg of the asymmetrically reconfigured motion

(b) Left hip joint trajectory of the original and reconfigured motions

(c) Left knee joint trajectory of the original and reconfigured motions

Figure 6.34: Injury simulation by applying asymmetrical muscle gain restrictions and inter-muscle ratios. The right leg is subjected to only the normal biomechanical modifications, where the left leg is restricted to 70% of the reconfigured control motion's maximum muscle gain and an inter-muscle weighting of 20:1:1 for the femur, tibia and foot respectively. In (a), the original motion is depicted as the semi-transparent ellipsoid character. (b) and (c) give the hip and knee joints respectively of the original and reconfigured motion

Beyond the similarity in trends between the left and right legs when comparing the imaginary limping motion and the reconfigured motion, the imaginary limping motions cannot be fully used as a basis for evaluation. As Figure 6.35 demonstrates, the imaginary limping motion is walking on the flat of his foot whereas the reconfigured injury is walking more on his toes. Furthermore, the degree to which the left leg is swung in front of the body is much less in the imagery limping motion of Figure 6.35 than it is for the reconfigured motion of Figure 6.34. Therefore, the two motions are not sufficiently similar to draw more subtle results from.



(a) Single stride of the left leg of an imaginary left leg limp of actor C

(b) Hip joint trajectory of an imaginary left leg limp from actor C

(c) Knee joint trajectory of an imaginary left leg limp from actor C

Figure 6.35: Imaginary left leg limping motion from actor C

6.4.4 Dynamics-Based Injury Simulation Summary

The combination of the two exaggerated dynamics-based modifications applied to only the left leg demonstrates a good and plausible looking injury. This has served to further demonstrate the potential of the dynamics-based system in that with very little effort, not only can motions be reconfigured to other actors, but dynamically plausible injuries can also be simulated onto the base motion of an uninjured movement.

Although only three different approaches to simulating injuries have been demonstrated within the dynamics optimisation-based system, because the dynamics representation is rich in parameters, there are many other types of modifications that could be used to simulate an injured motion. For example, joint limits could be capped or even remove DOFs completely, or the muscle gains could be dynamically reduced over time so as to produce a fatigued motion. There is a huge scope with which the base motion could be easily adjusted to portray different traits, where the subset that has been discussed in this section only contributes towards a small portion.

The exaggerated modifications presented in this section are really just guesses as to the biomechanical processes that are happening during the motion of an injured person. Therefore despite their visually good looking results, this thesis only posing what might result in an injured looking motion. To produce more realistic biomechanical injuries, it would be required to look in more depth as to what really happens in such cases. It would then be interesting to see if the true manner in which people carry themselves when injured could actually be encoded within our dynamics representation, which would form a subsequent area to study in itself.

6.5 Reconfiguration Discussion & Summary

During the presentation and evaluation of the dynamics optimisation process, which posses several mathematical novelties in its construct, the potential power of the approach for generating reconfigured motions has been exemplified. This includes not only the ability to introduce subtle, yet personalised effects for characters of the same dimensions, but also to take a base motion and biomechanically map it to an actor, whose mass and dimensions are completely different. These were demonstrated with plausible looking motions when compared to those of the real actor. Furthermore, by exaggerating various properties, it has been shown how it is possible to simulated injured looking motions from an uninjured base motion.

However, the evaluation of the technique has highlighted certain aspect of the dynamics optimisation process that need further work to better reproduce the real motions of different actors. The first, and probably most significant, of these areas would be to include the ability to time warp from the base motion. This is due to the optimisation process, as described, using the timings of the example motion, which is subsequently difficult to deviate from the fundamental pattern because part of the minimisation function is to effectively simulate the original movement. Through the use of muscle gain restrictions the timing have indirectly been influenced by reducing the overall speed of the motion, however a more direct influence is desirable.

A further fundamental issue that arose from the evaluation of the dynamics adaptation process is that the algorithm only reduces or equals the energy input by the example motion. Therefore, in the examples of section 6.3.1 where a more energetic motion is mapped to an equally or less energetic character, the process exhibited very good results when compared to the real motions. However, when attempting to map the motion of the less energetic actor B to a more energetic movement of actor C (see section 6.3.2), the directly desirable results were not obtained; the resulting motion looked like it could have really come from actor C, however it was at a slower walking pace than the actor would have naturally moved at. Therefore, a further area of future work would be to investigate how to inject energy into the dynamics system so that a positive affect on the simulation could be achieved rather than a dampening one. One possibility to explore would be to place minimum gain limits on the muscles, thereby forcing energy into the system. However, this would have to be carefully controlled so as not to create an unnatural motion since most motions are energy efficient and not wasteful. An alternative suggestion would be to adjust the muscle model to over-weigh the resulting contribution such that it exaggerates the resulting forces, but again care would be needed to make sure the system did not explode apart due to excessive energy input.

Following on from the issues discussed with the character mapping from actor B to actor C, when performing the reverse reconfiguration, although good approximates to the real actor's joint trajectories in the major axis were achieved, there was a noticeable difference in the visual motions. The real motions of actor B move with his arms and legs spread quite far apart from his body, which was not translated in the simulated motion. This is attributed to the subtleness of the real actor and hence could not be biomechanically created. It was further speculated that it would be possible for an artist to tweak the motion to reproduce the effect. This extension additionally provides a solution to the earlier problem that was highlighted in Figure 6.5, where due to the full range of motion allowed at the hip joint, during retargetting it was possible to get a motion that turned its leg in upon itself. However, if an artist was given the option to manually adjust the valid joint ranges of motion, this retargetting problem could be eliminated by restriction the amount that the leg is allowed to swing out to the side of its body. The same solution could also be used to specify the valid range of motion for a reconfigured movement to force the hip joint rotation away from the body more and hence. This would better result in the real motion that is aimed for in the reconfigured motions with actor B as the target actor.

Despite these limitations, the dynamics optimisation process still demonstrated a very reliable and consistent ability to map the motion from once actor to another by only adjusting the biomechanical information. The trend that was shown in this chapter was that the retargeted version was actually quite a poor representation of the new actor, even though the retargeted version is the least amount of modification that can be done in order to clean up the issue of foot skating. Going beyond the retargeted version, the evaluations provided evidence that the motion of the real actor is being much better approximated by the reconfigured version; many of the reconfigured motions looked as if the target actor could have actually performed the motion. However, there were some subtle differences that the biomechanical modifications alone could not capture, which are those of a more personal preference to the actor. It would therefore be interesting to build upon this work to explore ways in which more emotional behaviour can be factored into the biomechanically correct motions to see if movements closer still to the real actor can be obtained.

The final observation to conclude the reconfiguration process is a discovery that has little bearing on the dynamic simulations, but significant impact on the weighted inverse kinematics solution. In section 6.2.4, a correlation between changing limb masses directly and adjusting the inter-muscle weightings was demonstrated. For the higher relative limb mass, the dynamics model exhibited less motion in them by preferring to move lighter limbs more in order to reduce the global energy requirements. This directly corresponds with the inter-muscle ratio findings, which in turn correlates with the effects of changing the weighting matrix in the weighted inverse kinematics solution. Consequently, through this chain, the technique of reconfiguring motions using weighted inverse kinematics has a definite correlation with dynamically adjusting limb masses. This provides a degree of mathematical evidence in addition to the empirical evaluations performed in Chapter 4 to fundamentally support the technique's ability to approximate the dynamics of real motions without any dynamics-based knowledge.

6.6 Discussion

This chapter has presented the application of the dynamics-based optimisation process of Chapter 5 to adjust base motions for various purposes. The dynamics representation in itself has consisted of several original constructs for dealing with the lower level mathematical issues such as ill-resolutioned equations, where systems of equations could not equate to each other because of their underlying curvature. Furthermore, the implementation has proposed a way of converting a discrete or impulse occurrence into the twice-differential continuous time-domain without introducing knot multiplicities and hence vacuous differentials. This manifest itself through to the contact and friction forces that allowed the foot plants to be automatically and uniquely determined, which completely eliminates the need for any control processes for foot skate clean up in both retargeted and reconfigured motions. Thus, in the overall dynamics representation, a new approach for the computational encoding has been presented that can achieve a more general representation than previous works [Abe04, Liu02, Popo99, Sule05]. The dynamics model used in this thesis does not introduce new physical, real-world concepts, it is only that the representation differs by the level of complexity previously seen, with perhaps the exception of that demonstrated by Liu *et al* [Liu05]. The representation presented here extends certain aspects of the work by Liu *et al* in that in their work their foot plants need to be specified, whereas the technique presented in this thesis performs this process through the novel manner in which contact and frictional forces are dealt with. Furthermore, some of the fundamental mathematical issues associated with the manner in which the dynamics system is represent as a whole have been addressed.

However, the dynamics optimisation process has not been without its problems, not least the issue of stability. Although the earlier uses of the dynamics optimisation process in this chapter highlighted these issues and the steps that have been taken to resolve these problems were discussed, such issues were thereafter ignored to improve the clarity of the results presented. However, during the retargetting and reconfiguration of the motions, it was at times very difficult to get the optimisation process to stabilise on a solution. Steps taken to achieve this goal were to introduce extra knots along the piecewise cubic B-Spline and at times to manually reduce the step length taken by the optimisation process so as not to ill-condition the system and hence produce divergent results. To help improve the stability of the optimisation process, further work is needed to safeguard against such issues as opposed to the user needing to interrupt the process.

The lack of curve refinement in certain areas of the curve can indeed be easily configured to automatically refine itself, because many of the issues are related to areas where foot plants exist. However, along with a curve refinement process, it would additionally be desirable to have a curve unrefinement process. Therefore, if a region had been refined due to a potential foot plant, which is subsequently relocated, it would be advantageous to reduce redundant knot spacing so as to reduce the overall complexity of the representation.

The application of the dynamics optimisation process has shown that the approach taken to convert discrete impulses into continuous functions (see section 5.4) paid dividends when dealing with foot plants because of the systems ability to automatically detect and process them. However, despite an improved continuity and hence stability in the equations around the areas of impacts, this still contributes most towards overall system instability along with the turning on and off of inequality constraints. Although suitable and novel solutions to these problems were presented in Chapter 5, the whole dynamics optimisation process is by no means a perfect one and needs further work to reduce instability issues. A possible solution to the instability problem would be to graduate towards a solution by changing control parameters gradually towards the final endpoint, where at each graduation the system takes on a stable state before moving onto the next. For example, taking the restricted muscle gain modifications from section 6.2.3 it was found that going beyond a 70% reduction produced a state that was too different from the initial values for the optimisation process to converge towards. This was because the friction and contact forces had to be substantially moved to replant the footsteps to accommodate for the constraint-imposed changes. However, to achieve a reduction of 60% it is plausible that first a 70%, or even 80%, reduction state is simulated, which is stable. From this intermediate motion, the further reduction can be included to bring the final gain restrictions down to the desired 60% level. Thus the problem would be broken down into more manageable pieces that allows the optimisation process to move reliably to the final state by taking smaller steps, converging on a stable state before moving to the next step.

The way that the foot plants are handled in the dynamics-based representation of this thesis offered a novel and successful method of retargetting characters to new limb lengths with its lack of control processes. Thereafter, the effectiveness of the process to reconfigure the motion of one actor to another was demonstrated, building upon the work shown for character retargetting. Character reconfiguration contributes an original approach using the dynamics simulation process because in the past little work has been done in actually achieving such mappings. The closest work to that presented in this thesis in terms of both the complexity of the dynamics representation and results come from Liu *et al* [Liu05]. In this work they attempt to capture the style of an actor through the actor’s motion captured data and apply this style to different motions or recreate a completely different motion based on this parameterisation. In contrast, this thesis has presented algorithms that simulate the style of a target actor using only their biomechanical information; an example of their motion data is not first needed.

During the discussion on the reconfiguration process in section 6.3, several key points were mentioned that are worth reiterating here to conclude with. In summary, aside from those mentioned above, future developments towards the usefulness of the dynamics optimisation process include the implementation of a time warping feature so that the simulated motion is freer to deviate from the fundamental timings of the example motion. Furthermore, to aid the retargetting and reconfiguration process it would be useful to allow the artist to set their preferred joint range, which would aid the optimisation process by reducing the possible solution state space and also produce a motion that better fits with what the artist has envisaged. Additionally, the inherent issues of motion dampening open up an area of future work where appropriate methods need to be found that allow motions to be generated that are more energetic than the original motion to be generated.

The work that has been demonstrated through this chapter has shown promising results. However there is also a lot of scope within the dynamics optimisation process that has not been touched upon. For example, because a physically plausible environment is being modelled, it is possible to affect a change in the motion through environmental or additive components. Such changes might consist of adding a step (i.e. an obstacle that the motion walks onto and over) into the gait of the character or placing a weight in one hand to simulate the actor carrying something and hence dull the motion of the arm. The former of these suggestions has already been seen possible in the way that the locations of the hands were remapped during the football catch example from section 6.1.2. As future work, it would also be interesting to determine how different the simulated motion can actually get from the base motion, for example how easy would it be to go from a run to a hop motion? The aforementioned adjustment to the optimisation process of graduating towards a solution would assist in achieving this goal of mapping to a distinctly different motion from the base motion, say going from the run movement to a walk and then from the walk to hop. Furthermore, a time warping feature would be vital in such applications so that the minimisation function can distinctly deviate from the base motion.

Chapter 7:
Comparing IK and Dynamics for Motion Retargetting and Reconfiguration

Chapter 4 presented a real-time, inverse kinematics approach to modify character motions, whereas Chapter 5 and Chapter 6 demonstrated a novel dynamics-based technique that is inherently an offline process. In this chapter two areas of motion modification are used to compare the two algorithms, which have formed the primary focus of this thesis: motion retargetting and motion reconfiguration.

Character retargetting deals with one of the most basic character modifications by ensuring end-effector locations correctly interact with the environment, thus overriding the base motion's joint space configurations. Section 7.1 looks at this comparison. Motion reconfiguration, which is considered in section 7.2, extends retargetting so that not only does the motion correctly interact with the environment, but it is also adjusted in an attempt to capture the biomechanical style of the target actor. Reconfiguration thus adds extra richness to the retargeted motions by being able to spawn many different looking motions from the same base motion.

The comparison in this chapter examines whether the extra computational cost of the dynamics-based technique is justified compared to the motions of the IK techniques, and whether the IK technique can obtain movements as realistic as the dynamics-based version.

7.1 Retargetting

To compare the differences between inverse kinematics and dynamics-based retargetting, this section considers the following criteria:

§ Control routines
§ Joint angle change distributions
§ Visual continuity
§ Physical plausibility
§ Computational time

7.1.1 Control Routines

The inverse kinematics solution requires specific control algorithms to identify when end-effector location is preferred over maintaining joint space configuration. For example, to tackle the problem of foot skating, a control process is required to monitor the feet locations per frame, such as the one outlined in Chapter 4. This is in contrast to the dynamics version where, because contact and friction forces have been modelled within the dynamics representation, the system automatically picks up foot contacts and enforces friction, which prevents the feet from sliding.

Chapter 4 (IK) and Chapter 6 (dynamics-based) demonstrated retargetting of hand positions to meet an incoming football. There is a big difference in the number of control processes required to influence this change. In the case of the dynamics-based technique of Chapter 6, only 6 equality user constraints were added. For the IK version of Chapter 4, a whole new control process that allowed the specific point of catch to ease in and out of the posture, going between the new end-effector location and the original joint angle trajectories, was required.

As the examples for relocating the hands and feet demonstrate, the dynamics-based technique does not require additional control processes to achieve retargetting. This is because it can make use of its knowledge about the dynamic motion of rigid bodies and the environment to determine suitable motions based on the constraints. This is in contrast to the inverse kinematics-based technique which has no such knowledge and therefore the changes that need to take place have to be specifically encoded as additional control processes. Effectively, the control processes that IK makes use of simulate or approximate the desirable behaviour of the kinematically connected bodies, which are already defined within its dynamics counterpart.

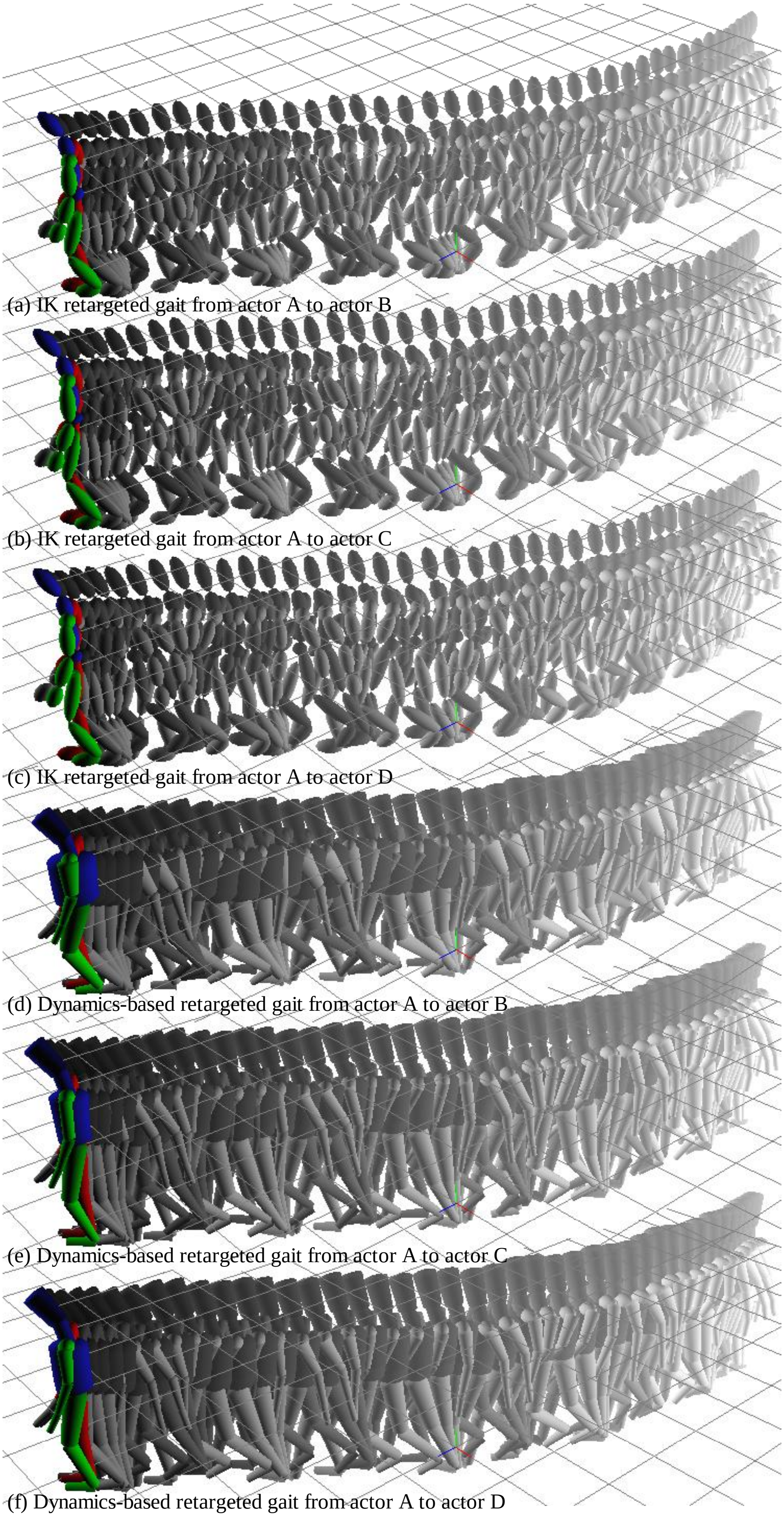
Therefore the dynamics-based technique is much better equipped to handle generic retargetting problems without the need of additional control processes, and hence provides a more complete package than inverse kinematics.

7.1.2 Joint Angle Change Distributions

When motions are only retargeted to new characters, i.e. no individualisation, it is important to maintain a balanced distribution between the joint angles that need to be changed to meet new end-effector locations. When an even distribution is not achieved during the updates, uncontrollable individualisation occurs, which could result in undesirably abnormal looking motions. It is therefore important that both IK and dynamics-based techniques meet this requirement.

To demonstrate that both techniques equally distribute the change in joint angles over the kinematics chain in a standard retargetting application, the walking right motion of actor A, from Chapter 3’s dataset, is retargeted to the limb lengths of actors B, C and D using both the IK and dynamics-based techniques. Figure 7.1 illustrates the retargeted motions, where the important aspect of the modified motions is maintaining the foot plants.

To demonstrate the equal distribution of the joint angles during retargetting the gait signatures of Figure 7.1’s motions are considered, which are given in Figure 7.2.



(a) IK retargeted gait from actor A to actor B

(b) IK retargeted gait from actor A to actor C

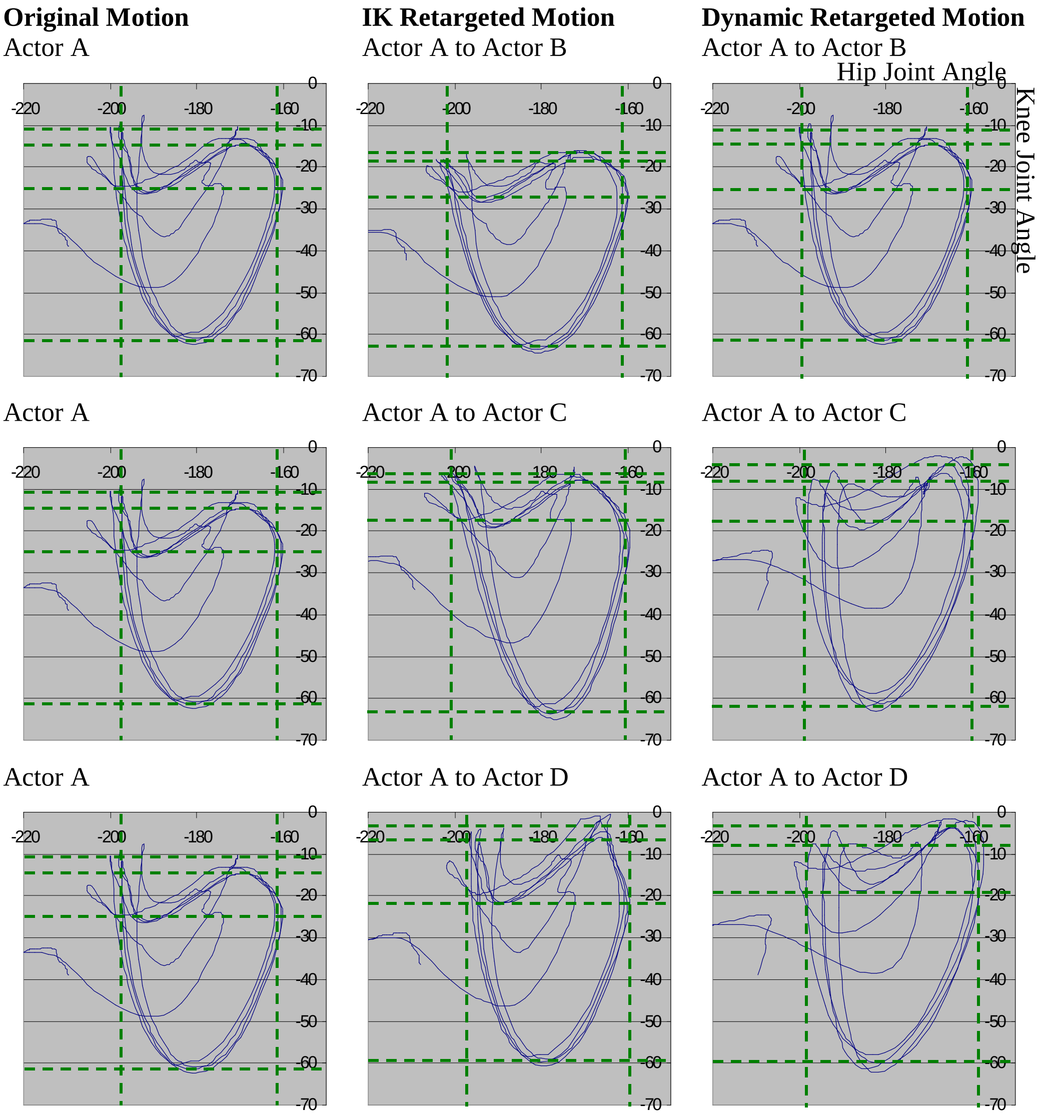
(c) IK retargeted gait from actor A to actor D

(d) Dynamics-based retargeted gait from actor A to actor B

(e) Dynamics-based retargeted gait from actor A to actor C

(f) Dynamics-based retargeted gait from actor A to actor D

Figure 7.1: Retargeted right turning gait of actor A using (a), (b) and (c) inverse kinematics and (d), (e) and (f) dynamics for the actor dimensions B, C, and D respectively



Original Motion
Actor A

IK Retargeted Motion
Actor A to Actor B

Dynamic Retargeted Motion
Actor A to Actor B

Actor A

Actor A to Actor C

Actor A to Actor C

Actor A

Actor A to Actor D

Actor A to Actor D

Figure 7.2: Gait signatures of the walking right motion of actor A retargeted to actors B, C and D using both the IK and dynamics-based techniques

The gait signatures presented in Figure 7.2 are all very similar in shape despite each of the actors having different sized limb lengths. This is in contrast to the variations in gait signatures between different actors, illustrated in Chapter 3 (the original dataset), and indeed the reconfigured motions demonstrated in Chapters 4 for weighted inverse kinematics and Chapter 6 for dynamics-based reconfigurations. The motion retargeted for actor B's limb lengths show very little change in the gait signatures because of him having the closest dimensional match for actor A. The retargeted motions for actor C and D do however have a more stretched out gait signature compared to actor A's due to their overall increase in leg length.

Although the gait signatures of actor C and D in Figure 7.2 are more stretched out compared to actor A's, the fundamental shape and relative proportions of the gait signatures remain similar. This is indicative of the retargetting process equally distributing the joint angle changes across the hierarchical structure. If the joint angles were not being equally distributed then a more distorted pattern would be expected in the retargeted gait signatures as opposed to the very similar patterns demonstrated in Figure 7.2.

A more visual consequence of the evenly distributed joint angle changes is that the retargetting techniques preserve the style of the original character, which can be seen by the similar gait signatures. This underlines the importance of being able to introduce variation when using the same motion for multiple characters such as using unevenly weighted inverse kinematics or dynamics-based biomechanical mass changes. However, purely in terms of retargetting, both the IK and dynamics-based techniques achieve the desirable goal of equally distributing the joint angle changes over the articulated hierarchy.

7.1.3 Visual Continuity

The inverse kinematics technique is performed at every frame and hence along the time domain it is discontinuous, whereas time is defined along a continuous domain in the case of the dynamics algorithm. The effect of this is to afford the IK algorithm with a much higher degree of granularity and hence discontinuous events, such as foot impacts, can be modelled exactly. Conversely, in the case of the dynamics version, discontinuous events can only be approximated using knot refinement to a suitably high frequency. However, if the knot spacing is not suitably refined enough there can be some visual disturbances as the joint angles attempt to keep the first differential of the foot location identically zero (i.e. the foot end-point does not move when planted). This is illustrated in Figure 7.3 where a single foot plant from Figure 7.1 is magnified. Figure 7.3 additionally includes the scaled motion using the same limb lengths as the IK and dynamics-based retargeted version for comparison.

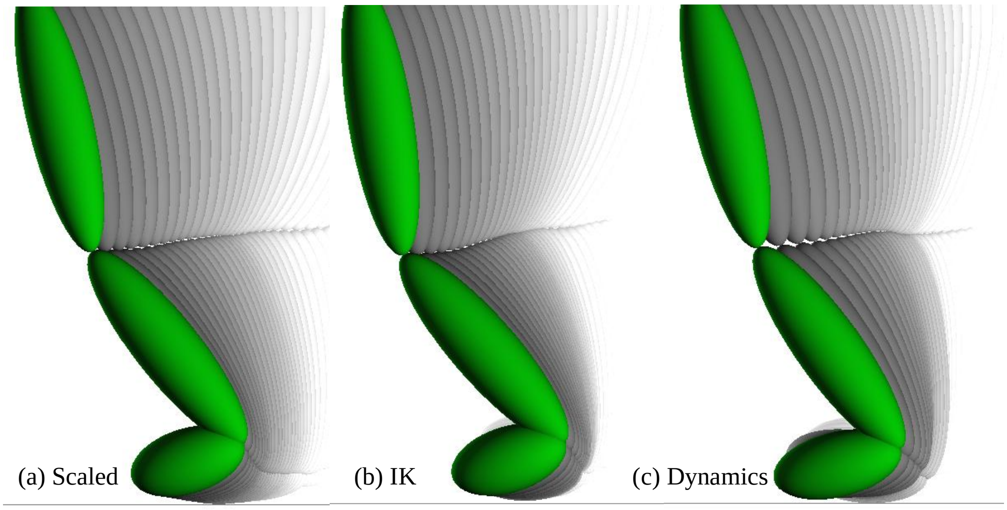


Figure 7.3: Comparison between a left foot plant for the (a) scaled, (b) IK and (c) dynamically retargeted walking motion of Figure 7.1. Each trailing greyscale posture is taken at 0.02 second intervals

Figure 7.3 shows that on close inspection the dynamically modified leg configuration takes a slightly larger amount of time to assert its contact force and hence a proper foot plant when compared to the IK version. This is illustrated by the trailing greyscale postures where the dynamics version has a slighter larger trail, given by the lightest grey shades. This region corresponds to approximately 0.1 seconds. Once the magnitude corresponding to the foot plant in the dynamics solver increases to its maximum state of 1.0, the foot can be seen to maintain a consistent position, similar to that of the IK version. The cause of the darker greyscale stills of the IK and dynamics solvers is because the motion of Figure 7.3 captures a complete foot plant from the heel coming in to make impact up to the heel leaving the ground and subsequently leaving only the toes in contact with the ground.

Despite the dynamics-based technique not immediately snapping to a foot plant, for example, the period over which contact forces occur is virtually indistinguishable when viewing the motion. However, in contrast, the discontinuous time domain of the IK technique does mean that snapping and non-smooth results can occur, which may visually manifest themselves. Such scenarios are impossible in the dynamics-based version because of its inherent twice-continuous time domain. To illustrate this, the joint trajectory curves of actor A's turning walking motion retargeted to actor C are given in Figure 7.4 (these correspond to the motions illustrated in Figure 7.1b and Figure 7.1e for the IK and dynamics version respectively). The motion curves of Figure 7.4 only graph the major axis of movement for the left femur, ignoring the Y-axis rotation as it is along the bone length axis, which contributes little to the main swing of the leg.

In each of the IK joint trajectories of Figure 7.4, the path appears jitterier in places, such as around frame 505 in Figure 7.4a and Figure 7.4b, whereas the dynamics version is shown as a smooth curve. Visually, the motions generated using the inverse kinematics technique do not portray any untoward discontinuities because the jitteriness of Figure 7.4 for the IK motion is at a very low resolution and there are no sudden peaks of changes in direction. There are no major changes in joint trajectories in Figure 7.4 because the control process that specifies the end-effector location to the IK solution consists of an easing in and out procedure that prevents a sudden snapping between preferring end-effector location or joint space configuration. This is further assisted by using the previous frame's joint space configuration to determine the joint angles in a particular frame and hence the IK solution only needs to change the joint angles by a small degree as opposed to completely recalculating them from a constant configuration. However, because there is no inter-frame cohesion for the inverse kinematics solution, it is still possible to suddenly get a very different joint space configuration between frames and hence produce visually poor results.

In terms of dealing with discontinuous events, the inverse kinematics technique offers a better solution, although the B-Spline curve of the dynamics version can be refined to locally approximate any discrete occurrence without any noticeable visual artefacts. However, the issue of the IK approach potentially selecting noticeably different joint space configurations between frames cannot be so easily overcome. Additional control processes can be layered on top of the IK technique to filter out any unwanted discontinuities [Whit69, Sul98, Lee99, Tak02, Tak05]. However the inclusion of a filter can results in a slight motion lag as frames within a given temporal window are assessed for similarity. Alternatively a filter may mean that an end-effector is not directly met at a specific frame as the joint space configuration is prevented from moving too far away from the configuration space of previous frames.

It is therefore generally more preferable to have continuity and because impulses and sudden changes of motion can be handled within the dynamics-based technique, it is this technique that will overall exhibit better visual continuity compared to inverse kinematics.

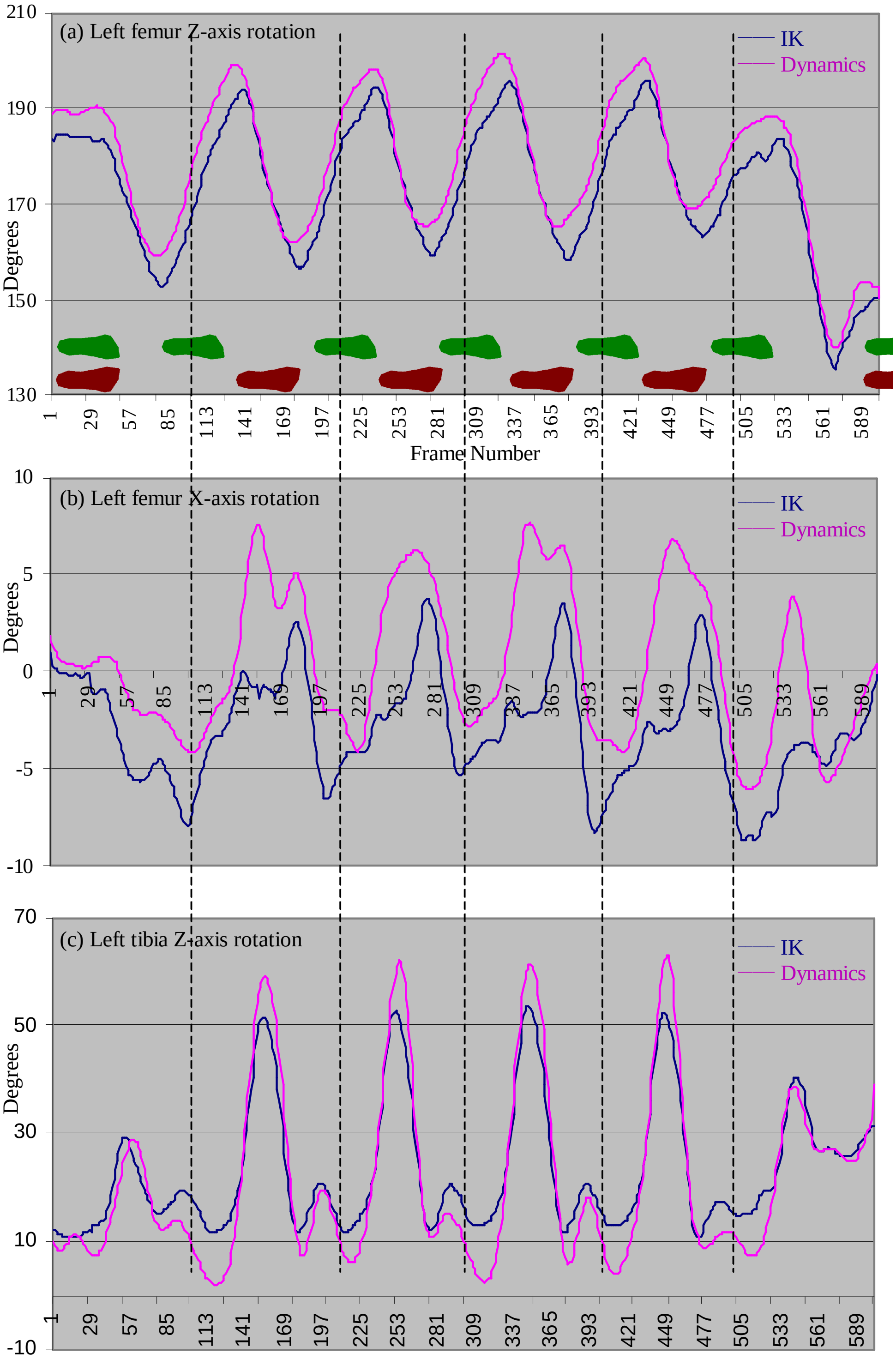


Figure 7.4: DOF comparison curves between the inverse kinematics and dynamically solved retargeted gait motion for the (a) left femur Z-axis, (b) left femur X-axis and (c) left tibia Z-axis, where foot plants are shown in (a)

7.1.4 Physical Plausibility

True physical plausibility can only be achieved using the dynamics-based version because the inverse kinematics technique has no concept of physical correctness. However, if the retargetting process only slightly deviates from the base motion then the resulting motion is unlikely to be noticeably physically implausible because the base motion is dynamically correct. Additionally, joint angle limits are imposed within both techniques so that only kinematically correct postures can occur. This further reduces the possible solution space and hence aids the inverse kinematics technique towards correct looking motions. All of the IK retargetting examples presented in this thesis have only affected a small change in the motion and hence the motion appears physically plausible. This can be seen in the retargeted turning motion of actor A in Figure7.1 and further supported by the similar joint trajectories in Figure 7.4 between the IK and dynamics-based retargetting algorithms.

Anything beyond small adjustments to the base motion can result in noticeable physical incorrectness using inverse kinematics. Therefore only the dynamics-based solution can guarantee physical correctness in comparison to the inverse kinematics technique.

7.1.5 Computational Time

The computational times of the motion modification techniques using inverse kinematics and dynamics have previously shown that inverse kinematics provides a much quicker solution. This is further iterated in Table 7.1 and Table 7.2 for the timings of actor A's retargeted turning motion to actors B, C and D for both IK and dynamics-based version respectively. The whole animation is 6 seconds in length, recorded at 100fps, and the dynamics-based retargeting timings given in Table 7.2 use a knot spacing of 0.15 second intervals.

IK Retargetting Stage	Measure of Operation		
	Actor B	Actor C	Actor D
Initialisation of IK system (seconds)	0.001	0.001	0.001
Average number of iterations per frame	8.404	9. 194	11.082
Overall frame rate	5157.0 fps	5157.0 fps	4776.92 fps
Total time to retargeted motion	0.12 seconds	0.12 seconds	0.13 seconds

Table 7.1: Execution time for retargetting actor A's walk right motion onto 3 different sized actors using the inverse kinematics-based algorithm

Dynamic Retargetting Stage	Time taken to perform operation (in seconds)		
	Actor B	Actor C	Actor D
Initialisation of dynamics equations	600.0	600.0	600.0
Initial value calculations	43.0	43.0	43.0
Minimum iterative step	96.0	97.0	97.0
Maximum iterative step	104.0	110.0	110.0
Average iterative step	100.0	102.0	102.0
Total number of main iterations	10 iterations	14 iteration	15 iterations
Total time to retargeted motion	1643.0 seconds	2071.0 seconds	2173.0 seconds

Table 7.2: Execution time to retarget actor A's walk right motion onto 3 different sized actors using the dynamics-based algorithm

The computational times for the inverse kinematics technique therefore show that it is real-time, whereas the dynamics-based version is not.

7.1.6 Retargetting Summary

The comparison between the inverse kinematics and dynamics-based retargetting algorithms has demonstrated that in terms of visual appearance, both are very similar; both techniques equally distribute the joint angle changes evenly over the kinematics chain. Nevertheless, the inverse kinematics approach is more prone to visual discontinuities and physical incorrectness because of its discrete time domain and lack of dynamics knowledge. Furthermore, the inverse kinematics technique is a less general retargetting algorithm because it requires specific control routines to guide the process. Therefore a specific collection of control processes can only achieve a specific type of retargetting. For example, in Chapter 4 a specific control process was implemented to position the hands of the character to catch a football, whereas the same control process could not be used unaltered to retarget the hands to pick up an object from a table. The dynamics-based retargetting technique thus has the ability to consistently produce more general and realistic retargeted motions with fewer visual artefacts than the inverse kinematics approach.

However, the dynamics-based technique runs at very high computational costs compared to the IK technique, which can operate in real-time. This issue can at times be the deciding factor as to whether an inverse kinematics or a dynamics-based retargetting solution is employed. For example, for any motions that need to be modified on the fly, such as in games, the IK technique has to be used. Fortunately, the comparisons in this chapter have shown that small deviations to the base motion do not result in any noticeable difference to the dynamics-based approach so the use of IK in real-time applications is visually feasible. In cases where real-time performance is not required, the dynamics-based technique has been shown to be the more complete package, primarily because of its ability to adhere to physical correctness.

7.2 Reconfiguration

Much of the criteria discussed in section 7.1 for character retargetting also apply to the comparison between inverse kinematics and dynamics-based reconfiguration with similar conclusions. Therefore this section focuses on the specific features that reconfiguration offers. The comparison between the IK and dynamics-based reconfiguration process will be judged on the following criteria:

- Introducing controllable uneven joint angle change distributions
- Full Reconfiguration: Mapping the motion of one actor to another

7.2.1 Introducing Controllable Uneven Joint Angle Change Distributions

Figure 7.2 illustrated that the standard retargetting algorithms for both techniques evenly distributed the joint angle changes over the kinematics chain when changing end-effector positions. This resulted in all of the motions looking very similar to the original gait of the base motion. However, when reusing motion data it is desirable to introduce subtle changes so that each character’s motion visually appears different, despite the fundamental motion remaining the same. This can be achieved using both the inverse kinematics and dynamics-based techniques described in this thesis, but in very different ways. Yet, as Chapter 6 demonstrated, there is a correlation between the two approaches.

Since the fundamental motion remains constant, the only change that can be induced into the motion is to modify the contributions of the joint angles during motion. This is achieved using a weighting vector within the inverse kinematics technique to dampen the motion of joints relative to each other. Weighted inverse kinematics does not prohibit the full range of motion and therefore end-effector locations can still be maintained, but with different joint trajectories depending on the weighting vector, which was demonstrated in Chapter 4.

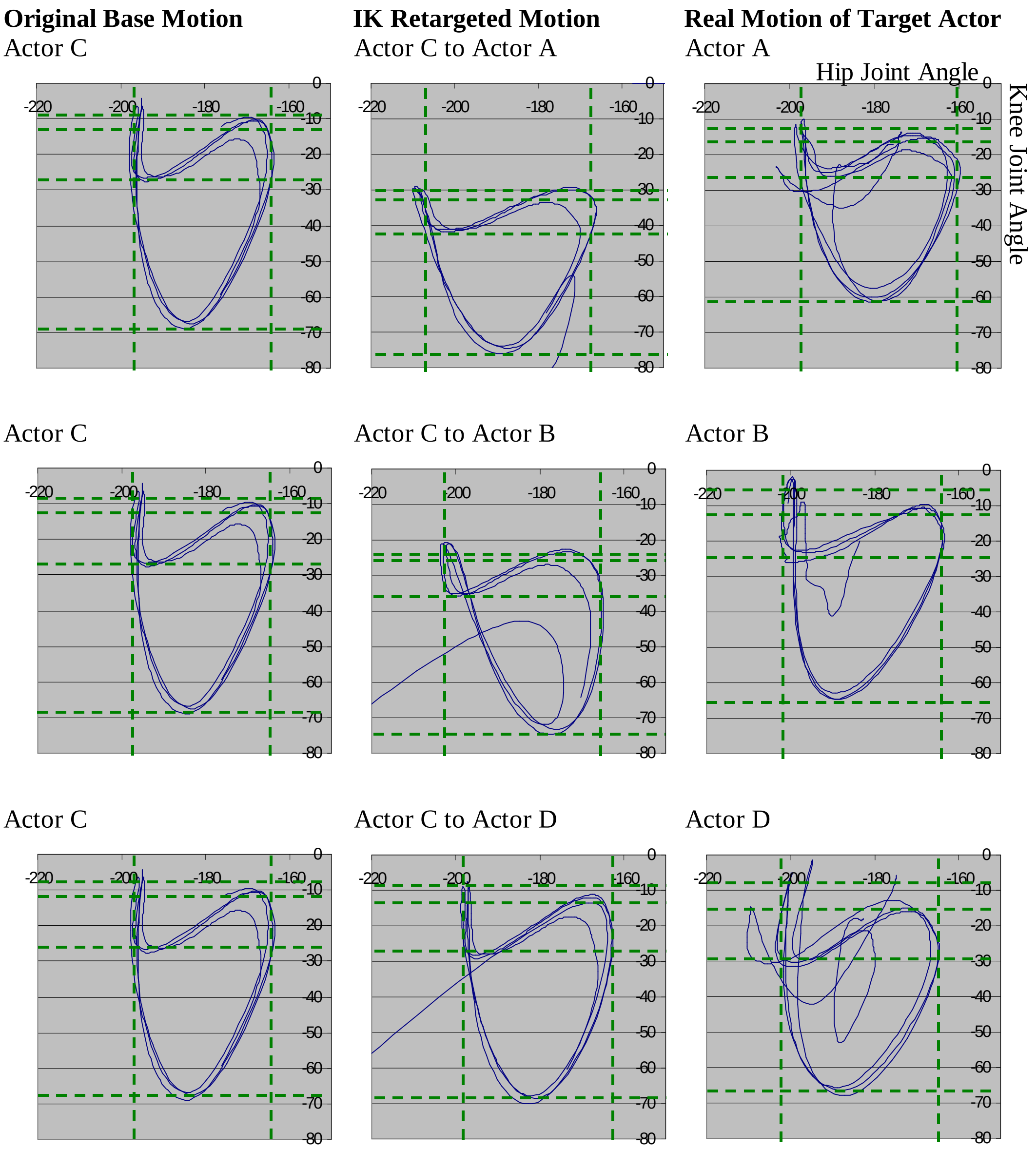
In contrast, there are two different approaches to controllable influence the joint angle distributions within the dynamics-based technique. The first is to modify the inter-muscle weighting ratios (see section 6.2.2). The second is to adjust the biomechanical masses of the limbs (see section 6.2.4). Both of these techniques effectively amount to the same mathematical change, which is to reduce the movement in the heavier limbs to reduce the overall energy expenditure. Hence the joint angle distributions can be reliably controlled to produce different looking movements with the same fundamental motion.

Both techniques therefore have the ability to influence an uneven joint angle change to spawn many different individualised motions for a given base motion, while still remaining faithful to the basic movement of the example motion. The two techniques have also shown how the uneven joint angle distribution can be exaggerated to simulate injuries (section 4.8 for weighted inverse kinematics and section 6.4 for dynamics-based), thus producing a wide variety of visually different motions from a single base motion.

7.2.2 Mapping the Motion of One Actor to Another

The ability to map the motion of a specific actor to the real motion of a completely different actor builds upon the discussions of the previous section. However, as opposed to producing an arbitrarily spawned individualised movement from a base motion, the biomechanics of the target actor are considered to guide the solution towards a motion that appears as if the real target actor has performed the movement. The biomechanical information provides a guide to an appropriate weighting vector for the inverse kinematics approach whereas the dynamics-based version directly interprets it.

The evaluation in section 4.9 demonstrated that the weighted inverse kinematics technique is able to approximate the gross features of the real motion by reproducing appropriate joint ranges of the target actor. However, weighted IK introduced a couple of artefacts in the generated motions that were noticeable in the gait signatures. The gait signatures are reproduced from section 4.9 in Figure 7.5 based on the evaluation of reconfiguring the walking motion of actor C to actors A, B and D.


Original Base Motion Actor C

IK Retargeted Motion Actor C to Actor A

Real Motion of Target Actor Actor A

Actor C

Actor C to Actor B

Actor B

Actor C

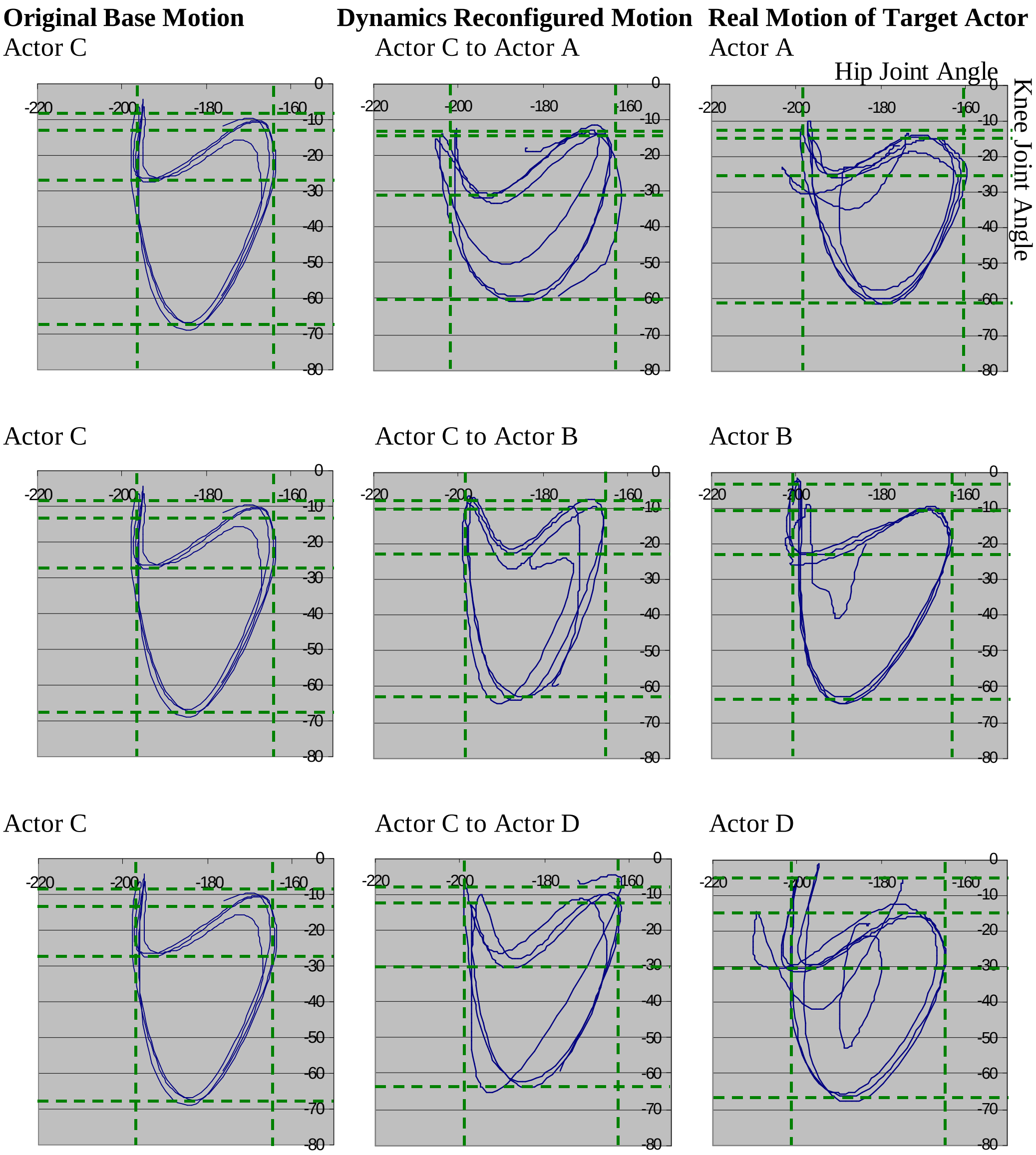
Actor C to Actor D

Actor D

Figure 7.5: Gait signatures of the walking motion of actor C reconfigured to actors B, C and D using weighted inverse kinematics

The first noticeable artefact introduced by weighted inverse kinematics is that of shifting the whole joint range of motion for the knee and hip joint. The second is to introduce a slight slant in gait signatures of the reconfigured motions. Both of these problems, which are illustrated in Figure 7.5, are due to the dampening influence that is always constant and present throughout the modification of the motion, i.e. the weighting vector is applied to the inverse kinematics solution at each frame with unchanging component values.

In contrast, the dynamics-based technique is better able to cope with the complex physical interaction that occurs during the motion of the legs because it has encoded knowledge of rigid body dynamics. The dynamics-based reconfiguration process demonstrated its effectiveness of mapping motions between different characters using only the biomechanics during the evaluation of the process in Chapter 6. This is highlighted in the similarity of the gait signatures, comparing the dynamics-based reconfigured motions and the real movements of the target actor. Figure 7.6 reproduces the gait signatures of section 6.3.1 for the dynamics-based reconfiguration of actor C's walking motion to actors A, B and D.



Original Base Motion **Dynamics Reconfigured Motion** **Real Motion of Target Actor**

Actor C — Actor C to Actor A — Actor A

Actor C — Actor C to Actor B — Actor B

Actor C — Actor C to Actor D — Actor D

Figure 7.6: Gait signatures of the walking motion of actor C dynamics-based reconfigured to actors B, C and D

The gait signatures of Figure 7.6 demonstrate a much closer correlation between the reconfigured and real motions of the target actors using dynamics than that demonstrated in Figure 7.5 using weighted inverse kinematics. The comparison between the gait signatures of Figure 7.5 and Figure 7.6 are further exemplified in the resulting motions, where the weighted inverse kinematics reconfigured motions, although good, show a reduced amount of resemblance towards the real motion compared to the dynamics-based version.

Therefore in terms of being able to map motion from one actor to another based purely on the biomechanics of the target actor, the dynamics-based reconfiguration process achieves this to a much higher degree than the motions reconfigured using weighted inverse kinematics.

7.2.3 Reconfiguration Summary

Both weighted inverse kinematics and dynamics-based techniques have the ability to reliably spawn multiple motions from a single base motion whose appearance is subtly different, thus achieving individualisation. However, the weighted inverse kinematics technique is not able to faithfully replicate the motion of a real actor as well as the dynamics-based version. The weighted IK version does however produce visually approximate motions for the target actor in a faction of the time it takes to produce a more realistic conversion using dynamics. Therefore the reconfiguration of motions using weighted inverse kinematics is not without its potential in real-time applications such as games. Furthermore, the cost of fully reconfiguring a motion using weighted inverse kinematics comes at virtually no extra computational expense compared to the retargetting version.

Therefore similar to the conclusions drawn about retargetting, if real-time performance is not required then the dynamics-based reconfiguration process offers the best visual results. Conversely, in cases where real-time solutions are needed, the weighted inverse kinematics technique provides an adequate substitute.

Chapter 8:
Conclusions

This thesis has presented novel contributions in the areas of motion capture data processing, inverse kinematics and dynamics-based motion capture modification. The primary motion capture modification process that has been the focus of this work using both inverse kinematics and dynamics is to provide the ability to completely reconfigure the motion of one actor to another using only the target actor's biomechanical information. The reconfiguration process, which both solves the retargetting problem and introduces biomechanical-based changes, can also be used to spawn off different looking motions from a single base motion to the extent of portraying injuries.

Optical motion capture provides a technique for recording the motion of a real performer by capturing the locations of optical markers that are placed on the human actor. However, the raw positional marker locations need to be transformed into a suitable animated hierarchical structure. Chapter 2 presented Inverse Skinning as a novel technique for performing the conversion from the optical marker locations to an animated articulated structure. The inverse skinning algorithm handles some erroneous marker data by checking relative marker locality, which distinguishes any markers that are significantly out of place compared to the rest of the marker cloud and hence are ignored. Inverse skinning demonstrated successful results when converting the motions described in Chapter 3's motion capture dataset, however there is still scope for extending this work. The main direction to continue the technique would be to introduce more robust control routines to handle different kinds of erroneous marker data as opposed to the gross marker positional errors that are currently handled. An example of this would be to handle the data when the majority of marks relating a specific limb are lost.

Using the inverse skinning algorithm to generate an animated hierarchical structure, the majority of this thesis looks at techniques to modify the articulation to produce desirable variability when it is attached to a virtual skin. The first of these techniques is based on inverse kinematics.

The discussion of Jacobian-based inverse kinematics in Chapter 4 presented an analytical and empirical comparison between the half- and full-Jacobian techniques, the full-Jacobian providing positional and orientation end-effector correctness to the half-Jacobian's positional only constraint. Proving that the full-Jacobian requires more than twice the computational resources as its half-Jacobian counterpart, the novel introduction of dynamic constraints facilitates the use of the cheaper half-Jacobian in a traditional full-Jacobian problem domain. The joint constraints restrict the final solution space, which subsequently binds the solver to orientate the end-effector within a given region. This effectively provides the tools to transform some more complex problem of orientation and positional correctness, which the full-Jacobian tackles, into the simpler positional-only problem, which can be solved using the quicker half-Jacobian.

Chapter 4 validated the conversion process by presenting a real-time, IK-driven walking character application that could manoeuvre about over uneven terrain. This application demonstrated how half-Jacobian inverse kinematics could be used to position the character's foot while also maintaining a forward-facing direction. Furthermore, Chapter 4 showed how the half-Jacobian could be used in the field of character retargetting. Both applications were therefore able to benefit from the reduced computational cost afforded by the half-Jacobian while still supporting an orientation-based aspect within the problem.

Building on the work demonstrated by the half-Jacobian solution, but also applicable to the full-Jacobian techniques, a new technique called Weighted Inverse Kinematics was introduced in Chapter 4. Weighted inverse kinematics introduces a weighting vector into the Jacobian-based IK algorithm that allows the rate of relative change of joint angles within an articulation to be reliably controlled as the end-effector is positioned. Therefore joints can appear stiffer in comparison to others within the IK chain and, importantly, the full range of motion of a joint is not restricted by the technique. Additionally the inclusion of the weighting vector comes at negligible extra cost compared to the standard Jacobian-based technique and hence also runs in real-time. The introduction of weighted inverse kinematics in this thesis is a mathematical extension to the standard IK algorithm and hence can be applied to a variety of different applications. However, this thesis makes use of weighted inverse kinematics to individualise character motions.

Weighted inverse kinematics character individualisation makes use of the controllable rate of change of limbs to potentially spawn an infinite number of subtly different motions from the same control source. It has been shown that the control source can be either a procedural model or motion capture data. The use of high variances in the component elements of weighting vectors generates very distinct individualised motions that take on the visual appearance of an injury, whereas low variance weighting vectors produce different, but more normal motions. The theoretical principle of dampening the motion of specific joints is given further support in Chapter 6 where a correlation between the true effect of changing biomechanical limb masses and the dampening of joint ranges is discussed.

The potential of using weighted inverse kinematics to accurately map the motion of one actor to a completely different character was evaluated in Chapter 4. This evaluation demonstrated the technique's ability to grossly approximate the joint ranges of motion of the target actor, which visually translated into plausible looking motions for the target actor. However, the evaluation also highlighted that the technique was limited by its lack of dynamics-based knowledge, thereby being unable to accurately reproduce the complex biomechanical motions of the real actors. Despite the limited success of the weighted inverse kinematics process to faithfully map the motion of one actor to another, it is still conceivable that better results using the technique are obtainable, especially in consideration to the correlation with dynamically changing biomechanical limb masses. Therefore to further develop this work, the evaluation of the technique suggested it might be possible to get more accurate results if the weighting vector were dynamically changed over time to correspond with the different phases of the motion. Although this direction requires more detailed analysis of real human motion, if successful it would provide a computationally cheap method of accurately producing realistic motions for a specific actor using the motion of a completely different actor without resorting to a dynamics-based solution.

The weighted inverse kinematics solution using the half-Jacobian therefore presents a tool for producing varied motions that are computationally cheap and hence applicable to real-time applications such as games[16]. However, without any knowledge of dynamics its applications are somewhat limited when accurate, physically plausible motions are required. To address this issue, this thesis introduced a novel motion capture modification technique that make use of a complete rigid body dynamics representation of the articulated character, thereby complementing the inverse kinematics technique.

The dynamics-based process presented in this thesis is inherently an offline process since it uses an optimisation process over the whole duration of motion to produce a solution. The representation used to model the dynamics includes the novel encoding of discrete events, such as contact and hence frictional forces, in a continuous time-based domain, which are described in Chapter 5. The inclusion of the contact and friction forces subsequently provides the ability to automatically determine foot plants when applied to character motions. Aside from encoding discrete occurrences within a continuous domain, the theoretical dynamics representation poses several unique problems when they are implemented. These problems, such as ill-resolutioned equations, are addressed in Chapter 5 with working solutions. Although these solutions consistently work, there are still instability issues within the dynamics-based optimisation process, which provides the foundations for potential future work. Any improvement in equation stability will have the effect of making the dynamics-based modification process more usable and hence more reliably converge towards a desirable motion. Based on the usage of the dynamics-based optimisation process, the areas that most need attention include addressing the issue of switching inequality equations between an active and inactive state and a refinement of the contact forces.

The implementation of the dynamics-based optimisation system provides an algorithm to modify motion capture data, which result in physically plausible motions. The dynamics representation is used in Chapter 6 to present a collection of novel methods of producing individualised motions from motion capture data. The main contribution of this chapter is however the ability to reconfigure the motion of one actor to a completely different one using only the target actor's biomechanics. This is similar to the process attempted using weighted inverse kinematics but the biomechanics are directly interpreted to provide mass, dimensionality, momentum, inertia and force parameters to the dynamics representation. The process of dynamically reconfiguring motions to different actors was demonstrated to be more successful than the weighted inverse kinematics version. During the evaluation of the process, which used the real motions and biomechanical information from 4 different actors, the dynamics-based reconfigured motions were demonstrated to be much closer to the real movements of the target actor than both the original base motion used and the dynamics-based retargeted motion. The visual results of the dynamics reconfiguration process also, on the whole, produced motions that appeared as if the target actor could have actually performed them.

[16] A non-disclosure agreement has recently been signed with Codemasters to discuss the potential of this work in the computer games industry.

The evaluation of the dynamics-based reconfigured motions did however highlighted subtle differences to the real motions. Many of these issues have been identified and discussed as areas of potential future work in Chapter 6, such as the inclusion of a time warping component to allow the process to deviate more from the example motion. Aside from the purely dynamics-based areas of future work, the evaluation process demonstrated that real actors add in extra, personal individualisations that biomechanics alone are unable to reproduce. Therefore a different direction of future work would be to look at including more personal and emotional control into the simulated motion, which would reside above the pure dynamics of the character.

Through exaggerating the biomechanical properties of a character, it was further shown how the dynamics-based technique could produce injured looking motions. However despite the partial evaluation of the approach by comparing it against the imaginary limping motion of a real actor, much more work can still be done in this area. Given that the representation is physically plausible, it should be possible to simulate physically realistic injuries, however this will require a more detailed analysis of the biomechanical processes that occur when an injury is sustained to invoke the correct parameters.

The dynamics-based modification applications in this thesis have primarily focused on individualising motions to new characters, however there is still a wide scope of other potential modifications that the technique should be equally well suited to. For example, the system should be able to appropriately react when adding more external changes into the base motion such as carrying something or being struck by an external object.

Chapter 7 compared the dynamics-based and the weighted inverse kinematics techniques in the area of retargetting and reconfiguration, showing that the dynamics-based version produced more realistic results over its counterpart. Therefore, time-based performance notwithstanding, the dynamics solutions currently present the better tools to perform such operations. However, the inverse kinematics techniques are able to generate results in a fraction of a time while still producing believable motions, albeit not as realistic as the dynamics-based motions. Therefore, the novel inverse kinematics techniques presented in this chapter have suitable realism to be considered for real-time applications to add extra richness to character motions. It is possible that with refinement the weighted inverse kinematics technique can still better approximate the motions of real actors without the need of complex and computationally demanding dynamics knowledge.

The techniques presented in this thesis have presented novel steps forwards in mapping the motion of one actor to another without sampling the motion of the target actor. The motion data of a different actor has however been a vital component in these techniques. Without the example motion the generated motions would look somewhat mechanical because their mathematically defined systems would not be able to produce the subtleties within the motion given by the motion capture data. However, the process of dynamically modifying motion capture data can still only go so far in producing realistic looking motions. Past the biomechanical correctness of a movement, every person changes their motion dependant on their feelings – a great deal about a person’s mood can be deduced from their posture. Therefore, beyond reconfiguring motion capture data to new actors will be the inclusion of more emotional depth. This will allow the animator to not only make their motions look plausible for a given character, but also to convey a subconscious meaning via their movement, all without the feet sliding over the ground.

Appendix A: Bibliography

Abe04 Y. Abe, C. K. Liu, Z. Popovic, “*Momentum-based Parameterisation of Dynamics Character Motion*”, SIIGRAPH/Eurographics Symposium on Computer Animation, pp.173-182, 2004

Aydi99 Y. Aydin, M. Nakajima, “*Realistic Articulated Character Positioning and Balance Control in Interactive Environments*”, Computer Animation, pp.160-168, May 1999

Bala91 C. A. Balafoutis, R. V. Patel, “*Dynamic Analysis of Robotic Manipulators: A Cartesian Tensor Approach*”, Kluwer Academic Publishers, 1991

Bara89 D. Baraff, “*Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies*”, Computer Graphics, Vol. 23, No. 3, pp.223-232, July 1989

Bara91 D. Baraff, “*Coping with Friction for Non-Penetrating Rigid Body Simulation*”, Computer Graphics, Vol. 25, No. 4, pp.31-40, August 1991

Bart87 R. H. Bartels, J. C. Beatty, B. A. Barsky, “*An Introduction to Splines for use in Computer Graphics and Geometric Modelling*”, Morgan Kaufmann Publishers, Inc. 1987

Barz88 R. Barzel, A. H. Barr, “*A Modelling System Based On Dynamic Constraints*”, Computer Graphics, Vol. 22, No. 4, August 1988

Barz96 R. Barzel, J. F. Hughes, D. N. Wood, “*Plausible Motion Simulation for Computer Graphics”, Computer Animation and Simulation* ’96, pp.183-197, 1996

Bind98 R. Bindiganavale, N. Badler, “*Motion Abstraction and Mapping with Spatial Constraints*”, Modelling and Motion Capture Technology for Virtual Environments, International Workshop, CAPTECH ’98, pp.70-82, November 1998

Boas83 M. L. Boas, “*Mathematical Methods in the Physical Sciences – 2nd Edition*”, Addison-Wesley, 1983

Bode97 B. Bodenheimer, C. Rose, S. Rosenthal, J. Pella, “*The process of motion capture: Dealing with the data*”, Computer Animation and Simulation '97, pp.3-18, 1997

Boul92 R. Boulic, D. Thalmann, “*Combined Direct and Inverse Kinematic Control for Articulated Figure Motion Editing*”, Computer Graphics Forum, Vol. 2, No. 4, 1992

Boul96 R. Boulic, R. Mas, D. Thalmann, “*A robust approach for the control of the center of mass with inverse kinetics*”, Computer & Graphics, Vol. 20. No. 5, pp.693-701, October 1996

Brog98 D. C. Brogan, R. A. Metoyer, J. K. Hodgins, “*Dynamically Simulated Characters in Virtual Environments*”, Computer Graphics and Applications, Vol. 18, No. 5, pp.58-69, September 1988

Brot88 L. S. Brotman, A. N. Netravali, “*Motion Interpolation by Optimal Control*”, Computer Graphics, Vol. 22, No. 5, pp.309-315, August 1988

Brud95 A. Bruderlin, L. Williams, “*Motion Signal Processing*”, SIGGRAPH ’95, pp.97-104, 1995

Burt83 P. Burt, E. Adelson, “*A Multiresolution Spline With Application to Image Merging*”, Transactions on Graphics, pp.217-236, October 1983

Carl97 D. A. Carlson, J. Hodgins, “*Simulating Level of Detail for Real-time Animation*”, Graphics Interface ’97, pp.1-8, 1997

Chen00 S. Chenney, D. A. Forsyth, “*Sampling Plausible Solutions to Multi-Body Constraint Problems*”, SIGGRAPH 2000, July 2000

Chia91 L. S. P. Chaicchio, S. Chiaverini, B. Siciliano, "*Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators With Task Space Augmentation and Task Priority Strategy*", International Journal of Robotics Research, Vol. 10, No. 4, 1991

Chin96 K.W. Chin, “*Closed-form and generalized inverse kinematic solutions for animating the human articulated structure*”, Bachelor's Thesis in Computer Science, Curtin University of Technology, 1996

Choi99 K. Choi, S. Park, H. Ko, "*Processing Motion Capture Data to Achieve Positional Accuracy*", Graphical Models and Image Processing Vol. 61, pp.260-273, 1999

Choi00 K. J. Choi, H.S. Ko, "*On-line Motion Retargetting*", The Journal of Visualization and Computer Animation, Vol. 11, pp.223-235, 2000

Chun99 S. Chung, J. K. Hahn, "*Animation of Human Walking in Virtual Environments*", Computer Animation, pp.4-15, May 1999

Cohe92 M. F. Cohen, "*Interactive Spacetime Control for Animation*", Computer Graphics, Vol.26, No. 2, pp.293-302, July 1992

Crai55 J. J. Craig, "*Introduction to Robotics: Mechanics and Control*", Addison-Wesley, 1955

Demo86 R. Demori, D. Probst, "*Handbook of Pattern Recognition and Image Processing*", Academic Press, 1986

Dens97 D. Densly, P. Willis, "*Emotional Posturing: A Method Towards Achieving Emotional Figure Animation*", Computer Animation, 1997

Eber01 D. H. Eberly, "*3D Game Engine Design*", Morgan Kaufmann, 2001

Endo05 Endorphin 2, *http://www.naturalmotion.com*

Espi85 B. Espiau, R. Boulic, "*Collision Avoidance for Redundant Robots with Proximity Sensors*", 3rd International Symposium of Robotic Research, October 1985

Falo01 P. Faloursos, M. van de Panne, D. Terzopoulos, "*Composable Controllers for Physics-Based Character Animation*", 28th Annual Conference on Computer Graphics and Interactive Techniques, pp.251-260, 2001

Fang03 A. C. Fang, N. S. Pollard, "*Efficient Synthesis of Physically Valid Human Motion*", Transactions on Graphics, Vol. 22, No. 3, July 2003

Fedo03 M. Fedor, "*Application of Inverse Kinematics for Skeleton Manipulation in Real-time*", International Conference on Computer Graphics and Interactive Techniques, pp.203-212, 2003

Ge05 C. Ge, Y. Chen, C. Yang, "*Motion Retargeting for the Hand Gesture*", Winter School of Computer Graphics (WSCG), 2005

Gill81 P. E. Gill, W. Murray, M. Wright, "*Practical Optimization*", Academic Press, 1981

Gill99 M. F. P. Gillies, N. A. Dodgson, "*Psychologically-based Walking in a Cluttered Environment*", Eurographics '99 – Short Papers & Demos, 1999.

Gira85 M. Girard, A. A. Maciejewski, "*Computational Modelling for the Computer Animation of Legged Figures*", SIGGRAPH 85 Computer Graphics, Vol. 19, No. 3, July 1985

Glei97 M. Gleicher, "*Motion Editing with Spacetime Constraints*", Proceedings of Symposium on Interactive 3D Graphics, 1997

Glei98a M. Gleicher, "*Retargetting Motion to New Characters*", Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pp.33-42, 1998

Glei98b M. Gleicher, P. Litwinowicz, "*Constraint-based Motion Adaptation*", The Journal of Visualization and Computer Animation, Vol. 9, No. 3, pp.66-94, 1998

Glei00 M. Gleicher, "*Comparative Analysis of Constraint-Based Motion Editing Method*", 2000 Workshop on Human Modelling and Animation, Seoul Korea, June 2000

Glei01 M. Gleicher, "*Motion Path Editing*", ACM Symposium on Interactive 3D Graphics, pp.195-202, 2001

Gold80 H. Goldstein, "*Classical Mechanics – 2nd Edition*", Addison-Wesley, 1980

Gosw98 A. Goswami, "*A New Gait Parameterisation Technique by Means of Cyclogram Moments: Application of Human Slope Walking*", Gait & Posture, Vol. 8, No. 1, pp.15-36, August 1998

Gran95 J. P. Granieri, J. Crabtree, N. Badler, "*Production and Playback of Human Figure Motion for Visual Simulation*", ACM Transitions on Modelling Computer Simulation, Vol. 5, No.3, pp.222-241, July 1995

Gree04 R. D. Green, L. Guan, "*Quantifying and Recognising Human Movement Patterns from Monocular Video Images – Part 1: A New Framework for Modelling Human Motion*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, No. 2, pp.179-190, Feb 2004

Groc04 K. Grochow, S. L. Martin, A. Hertzmann, Z. Popovic, "*Style-Based Inverse Kinematics*", ACM Transactions on Graphics, SIGGRAPH 04, Vol. 23, No. 3, pp.522-531, August 2004

Grze98 R. Grzeszczuk, D. Terzopoulos, G. Hinton, "*NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models*", Computer Graphics, Proceedings of SIGGRAPH 98, pp.9-20, July 1998

Hahn88 J. K. Hahn, "*Realistic Animation of Rigid Bodies*", Computer Graphics, Vol.22, No.4, pp.299-308, August 1988

Havo05 Havok Physics Engine, *http://www.havok.com*

Heym05 S. B. Heymsfield, T. G. Lohman, Z. Wang, S. B. Going, "*Human Body Composition – 2nd Edition*", Champaign, IL: Human Kinematics, 2005

Hodg95 J. K. Hodgins, W. L. Wooten, D. C. Brogan, J. G. O'Brien, "*Animating Human Athletics*", Computer Graphics and Interactive Techniques, pp.71-78, 1995

Hodg96 J. K. Hodgins, "*Three-Dimensional Human Running*", IEEE Conference on Robotics and Automation, 1996

Hodg97 J. K. Hodgins, N. S. Pollard, "*Adapting Simulated Behaviours for New Characters*", Computer Graphics and Interactive Techniques, pp.153-162, 1997

Horn75 R. Hornbeck, "*Numerical Methods*", Quantum Publishers Incorporated, New York, 1975

Hsu05 E. Hsu, K. Pulli, J. Popovic, "*Style Translation for Human Motion*", SIGGRAPH 2005, 2005

Joe87 B. Joe, "*Discrete Beta-Splines*", Computer Graphics, Vol. 21, No. 4, pp.137-144, July 1987

Kalr98 J. Kalra, N. Magnenat-Thalman, L. Moccozet, G. Sannier, A. Aubel, D. Thalmann, "*Real-time Animation of Realistic Virtual Humans*", IEEE Computer Graphics and Applications, p 42-56, September/October 1998

Kibb96 T. W. B. Kibble, "*Classical Mechanics – 4th Edition*", Harlow: Longman, 1996

Kokk04 E. Kokkevis, "*Practical Physics for Articulated Characters*", Game Developers Conference, 2004

Komu97 T. Komura, Y. Shinagawa, T. L. Kunii, "*A Muscle-based Feed-forward Controller of the Human Body*", Eurographics '97, Vol. 16, No. 3, 1997

Komu00 T. Komura, Y. Shinagawa, T. L. Kunii, "*Creating and Retargetting Motions by the Musculoskeletal Human Body Model*", The Visual Computer, Vol. 16, No. 5, pp.254-270, 2000

Komu01a T. Komura, Y. Shinagawa, T. L. Kunii, "*An Inverse Kinematics Method based on Muscle Dynamics*", Proceedings of Computer Graphics International 2001, pp. 15-22, July 2001

Komu01b T. Komura, Y. Shinagawa, "*Attaching Physiological Effects to Motion-Captured Data*" Proceedings of Graphics Interface 2001, pp.27-36, 2001

Komu04 T. Komura, H. Leung, J. Kuffner, "*Animating Reactive Motions for Biped Locomotion*", Virtual Reality Software and Technology, 2004

Kore82 J. U. Korein, N. I. Badler, "*Techniques for Generating the Goal-Directed Motion of Articulated Structures*", IEEE Computer Graphics and Applications, Vol.2, No.9, pp.71-81, 1982

Kova02 L. Kovar, M. Gleicher, J. Schreiner, “*Footskate Cleanup for Motion Capture Editing*”, ACM SIGGRAPH Symposium on Computer Animation, July 2002

Lasz96 J. Laszlo, M. Van de Panne, E. Fiume, “*Limit Cycle Control and its Application to the Animation of Balance and Walking*”, Computer Graphics, Vol. 30, pp.155-162, 1996

Lawr04 N. D. Lawrence, “*Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data*”, Proceedings of Neural Information Processing Systems 2004

Lee99 J. Lee, S. Y. Shin, “*A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures*”, SIIGRAPH 99, pp.39-48, 1999

Liu94a Z. Liu, M. F. Cohen, “*Decomposition of Linked Figure Motion: Diving*”, 5th Eurographics Workshop on Animation and Simulation, 1994

Liu94b Z. Liu, S. J. Gortler, M. F. Cohen, “*Hierarchical Spacetime Control*”, Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pp.35-42, 1994

Liu02 C. K. Liu, Z. Popovic, “*Synthesis of Complex Dynamic Character Motion from Simple Animations*”, Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, pp.408-416, 2002

Liu05 C. K. Liu, A. Hertzmann, Z. Popovic, “*Learning Physics-Based Motion Style with Nonlinear Inverse Optimisation*”, SIGGRAPH 2005, August 2005

Maes96 G. Maestri, “Digital Character Animation”, New Riders Publishing, 1996

Madh98 N. Madhavapeddy, S. Ferguson, "*Specialised Constraints for an Inverse Kinematics Animation System Applied to Articulated Figures*", Eurographics UK, 1998

McKe90 M. McKenna, D. Zeltzer, “*Dynamics Simulation of Autonomous Legged Locomotion*”, Computer Graphics, Vol. 24, No. 4, pp.29-38, August 1990

McMa84 T. A. McMahon, “*Mechanics of Locomotion*”, International Journal of Robotics Research, Vol. 3, No. 2, pp.4-28, 1984

Mena00 A. Menache, “*Understanding Motion Capture for Computer Animation and Video Games*”, Academic Press, 2000

Mere01 M. Meredith, S. Maddock, “*Motion Capture File Formats Explained*”, Department of Computer Science Technical Report CS-01-11, The University of Sheffield, UK, 2001

Mere04a M. Meredith, S. Maddock, “*Using a Half-Jacobian for Real-Time Inverse Kinematics*”, Computer Graphics, Artificial Intelligence, Design and Education, pp.81-88, 2004

Mere04b M. Meredith, S. Maddock, “*Individualised Character Motion Using Weighted Real-Time Inverse Kinematics*”, GameOn 2004, pp.57-64, 2004

Mere04c M. Meredith, S. Maddock, “*Adapting Motion Capture using weighted Real-Time Inverse Kinematics*”, Graphics Design Technology Workshop, pp.120-129, 2004

Mere05 M. Meredith, S. Maddock, “*Adapting Motion Capture using weighted Real-Time Inverse Kinematics*”, ACM Computers in Entertainment, Vol. 3, No. 1, 2005

Meye91 A. Meyer, “*A Linear Time Oslo Algorithm*”, Transactions on Graphics, Vol.10, No.3, pp.312-318, July 1991

Mizu01 M. Mizuguchi, J. Buchanan, T. Calvert, “*Data Driven Motion Transitions For Interactive Games*”, Eurographics ’01 – Short Presentation, 2001

Moch80 S. Mochon, T. A. McMahon, “*A Ballistic Model for Walking*”, Journal of Biomechanics, Vol. 13, pp.49-57, 1980

Monz00 J. S. Monzani, P. Baerlocher, R. Boulic, D. Thalmann, “*Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargetting*”, Eurographics 2000, Vol.19, No.3, 2000

Moor88 M. Moore, J. Wilhelms, “*Collision Detection and Response for Computer Animation*”, Computer Graphics, Vol. 22, No. 4, pp.289-298, August 1988

Mort85 M. E. Mortenson, “*Geometric Modelling*”, John Wiley and Sons, New York, 1985

Muka05 T. Mukai, S. Kuriyama, "*Geostatistical Motion Interpolation*", Proceedings of SIGGRAPH 2005, August 2005

Mult99 F. Multon, L. France, M. Cani-Gascuel, G. Debunne, "*Computer Animation of Human Walking: A Survey*", The Journal of Visualization and Computer Animation, Vol. 10, pp.39-54, 1999

Muyb55 E. Muybridge, "*The Human Figure in Motion*", Dover Publications, 1955

Muyb84 E. Muybridge, "*The Male and Female Figure in Motion: 60 Classic Photographic Sequences*", Dover Publications, 1984

NBDL88 "*Anthropometry and Mass Distribution for Human Analogues – Volume 1: Military Male Aviators*", Naval Biodynamics Laboratory, NBDL-87R003, March 1988

Ngo93 J. T. Ngo, J. Marks, "*Spacetime Constraints Revisited*", Computer Graphics, Vol. 27, pp.343-350, August 1993

OBri00 J. F. O'Brien, V. B. Zordan, J. K. Hodgins, "*Combining Active and Passive Simulations for Secondary Motion*", Computer Graphics and Applications, Vol. 20, No. 4, pp.86-96, July 2000

ODE05 Open Dynamics Engine, *http://ode.org*

Odge85 J. Odgen, E. Adelson, J. Bergeb, P. Burt, "*Pyramid-Based Computer Graphics*", RCA Engineer, Vol. 30, pp.4-15, 1985

Oshi01 M. Oshita, A. Makinouchi, "*A Dynamic Motion Control Technique for Human-line Articulated Figures*", Eurographics 2001, Vol. 20, No. 3, pp. 192-202, 2001

Pann96 M. Van de Panne, "*Parameterized Gait Synthesis*", IEEE Computer Graphics & Applications, pp.40-49, March 1996

Park97 J. K. Park, Y. M. Kang, S. S. Kim, H. G. Cho, "*Expressive Character Animation with Energy Constraints*", Compugraphics '97, pp.260-268, 1997

Paul88 R. P. Paul, B. Shimano, G. E. Mayer, "*Kinematic Control Equations for Simple Manipulators*", IEEE Transactions on System, Man & Cybernetics, Vol. 11, No. 6, 1988

Phil91 C. B. Phillips, N. I. Badler, "*Interactive Behaviours for Bipedal Articulated Figures*", SIGGRAPH 91 Computer Graphics, Vol. 24, No. 4, July 1991

Poll99 N. S. Pollard, "*Simple Machines for Scaling Human Motion*", Eurographics Workshop on Animation and Simulation, 1999

Poll00 N. S. Pollard, F. Behmaram-Mosavat, "*Force-Based Motion Editing for Location Tasks*", Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 1, pp.663-669, April 2000

Popo99 Z. Popovic, A. Witkin, "*Physically Based Motion Transformation*", SIGGRAPH '99, pp.11-20, August 1999

Popo00a J. Popovic, S. M. Seitz, M. Erdmann, Z. Popovic, A. Witkin, "*Interactive Manipulation of Rigid Body Simulations*", SIGGRAPH 2000, pp.209-218, July 2000

Popo00b Z. Popovic, "*Controlling Physics in Realistic Character Animation*", Communications of the ACM, Vol. 43, No. 7, pp.50-59, July 2000

Popo03 J. Popovic, S. M. Seitz, M. Erdmann, "*Motion Sketching for Control of Rigid-Body Simulations*", Transactions on Graphics, Vol. 22, No. 4, pp.1034-1054, October 2003

Pres92 W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, "*Numerical Recipes in C: The Art of Scientific Computing*", Cambridge University Press, 1992

Raib91 M. H. Railbert, J. K. Hodgins, "*Animation of Dynamics Legged Locomotion*", Computer Graphics, Vol. 25, No. 4, pp. 349-358, July 1991

Rose96 C. Rose, B. Guenter, B. Bodenheimer, M. F. Cohen, "*Efficient Generation of Motion Transitions Using Spacetime Constraints*", Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp.147-154, 1996

Rose98 C. Rose, M. Cohen, B. Bodenheimer, "*Verbs and Adverbs: Multidimensional Motion Interpolation Using Radial Basis Functions*", IEEE Computer Graphics and Applications, Vol. 18, No. 5, pp.32-40, 1998

Rose01 C. F. Rose, P. J. Sloan, M. F. Cohen, "*Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation*", Eurographics 2001, Vol. 20, No. 3, 2001

Ryck77 J. P. Ryckaert, G. Ciccotti, H. J. C. Berendsen, "*Numerical Integration of the Cartesian Equations of Motions of a System with Constraints: Molecular Dynamics of n-Alkanes*", Journal of Computational Physics, Vol. 23 pp.327-341, 1977

Safo04 A. Safonova, J. K. Hodgins, N. S. Pollard, "*Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behaviour-Specific Spaces*", ACM Transactions on Graphics, Vol. 23. No. 3, pp.514-521, August 2004

Scia87 L. Sciavicco, B. Siciliano, "*A Dynamic Solution to the Inverse Kinematic Problem for Redundant Manipulators*", IEEE International Conference on Robotics & Automation, pp.1081-1087, 1987

Scia96 L. Sciavicco, B. Siciliano, "*Modelling and control of robot manipulators*", McGraw-Hill, 1996

Shin01 H. J. Shin, J. Lee, M. Gleicher, S. Y. Shin, "*Computer Puppetry: An Importance-Based Approach*", ACM Transactions On Graphics, Vol. 20, No. 2, pp.67 94, April 2001

Shin03 H. J. Shin, L. Kovar, M. Gleicher, "*Physical Touch-Up of Human Motions*", 11th Pacific Conference on Computer Graphics and Applications, pp.194-203, 2003

Sici99 B. Siciliano, "*Closed-loop inverse kinematics algorithm for constrained flexible manipulators under gravity*", Journal of Robotic Systems, Vol. 16, No. 6, May 1999

Sul98 C. W. Sul, S. K. Jung, K. Wohn, "*Synthesis of Human Motion Using Kalman Filter*", International Workshop on Modelling and Motion Capture Techniques for Virtual Environments, pp.100-112, 1998

Sule05 A. Sulejmanipsic, J. Popovic, "Adaptation of Performing Ballistic Motion", Transactions on Graphics, Vol. 24, No. 1, pp.165-179, 2005

Tak02 S. Tak, O. Y. Song, H. S. Ko, "*Spacetime Sweeping: An Interactive Dynamic Constraints Solver*", Computer Animation, pp.261-270, 2002

Tak05 S. Tak, H. Ko, "*A Physically-Based Motion Retargeting Filter*", ACM Transactions on Graphics, Vol. 24, No. 1, pp. 98-117, January 2005

Tang95 D. Tang, J. T. Ngo, "*N-Body Spacetime Constraints*", Journal of Visualisation and Computer Animation, Vol. 6, pp.143-154, 1995

Tang99 W. Tang, M. Cavazza, D. Mountain, R. Earnshaw, "*A Constrained Inverse Kinematics Technique for Real-time Motion Capture Animation*", The Visual Computer, Vol. 15, pp.413-425, 1999

Teva00 G. Tevatia, S. Schaal, "*Inverse Kinematics for Humanoid Robots*", International Conference on Robotics and Automation, April 2000

Tola96 D. Tolani, N. I. Badler, "*Real-time inverse kinematics of the human arm*", Presence, Vol. 5, No. 4, 1996

Tola00 D. Tolani, A. Goswami, N. I. Badler, "*Real-time Inverse Kinematics Techniques for Anthropomorphic Limbs*", Computer Graphics Vol. 62, No. 5, 2000

Unum91 M. Unuma, R. Takeuchi, "*Generation of Human Motion with Emotion*", Computer Animation '91, pp.77-88, 1991

Unum95 M. Unuma, K. Anjyo, R. Takeuchi, "*Fourier Principles for Emotion-Based Human Figure Animation*", SIGGRAPH '95, pp.91-96, 1995

Urta04 R. Urtasun, P. Glardon, R. Boulic, D. Thalmann, P. Fua, "*Style-Based Motion Synthesis*", Computer Graphics Forum, Vol. 23, No. 4, pp.799-812, 2004

Vann95 M. W. Vannier, K. M. Robinette, "*Three Dimensional Anthropometry*", Biomedical Visualisation '95, pp.1-8, October 1995

Wagg04 D. K. Wagg, M. S. Nixon, "*Automated Markerless Extraction of Walking People Using Deformable Contour Models*", Computer Animation and Virtual World", Vol. 15, pp.399-406, 2004

Watt92 A. Watt, M. Watt, "*Advanced animation and rendering techniques*", Addison-Wesley, 1992

Watt03 A. Watt, F. Policarpo, "3D Games: Advance Real-time Rendering and Animation", Addison-Wesley, 2003

Wang91 L. Wang, C. Chen, "*A Combined Optimisation Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators*", IEEE Transactions on Robotics & Applications, Vol. 7, No. 4, pp.489-499, 1991

Welm93 C. Welman, "*Inverse kinematics and geometric constraints for articulated figure manipulation*", Master of Science Thesis, School of Computing Science, Simon Fraser University, 1993

Whit69 D. E. Whitney, "*Resolved Motion Rate Control of Manipulators and Human Prostheses*", IEEE Transaction Man-Machine Systems, Vol. 10, 1969

Wile97 D. Wiley, J. Hahn, "*Interpolation Synthesis of Articulated Figure Motion*", IEEE Computer Graphics and Applications, Vol. 17, No. 6, pp.39-45, 1997

Wint90 J. M. Winters, S. L-Y, "*Multiple muscle systems*: *Biomechanics and Movement Organization*", New-York, 1990

Witk88 A. Witkin, M. Kass, "*Spacetime Constraints*", Computer Graphics, Vol. 22. No. 4, pp.159-168, August 1988

Witk95 A. Witkin, Z. Popovic, "*Motion Warping*", SIGGRAPH '95, pp. 6-11, August 1995

Woot95 W. L. Wooten, J. K. Hodgins, "*Animation of Human Diving*", Computer Graphics Forum, Vol. 15, No. 1, pp.3-13, 1995

Yoo03 J. H. Yoo, M. S. Nixon, "*Markerless Human Gait Analysis via Image Sequences*", International Society of Biomechanics XIXth Congress, 2003

Yu98 Q. Yu, D. Terzopoulos, "*Synthetic Motion Capture for Interactive Virtual Worlds*", Computer Animation, pp.2-10, June 1998

Zelt82 D. Zeltzer, "*Motion Control Techniques for Figure Animation*", IEEE Computer Graphics and Applications, Vol. 2, No. 9, pp.53-69, November 1982

Zhao94 J. Zhao, N. I. Badler, "*Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures*", ACM Transactions on Graphics, Vol. 13, No. 4, pp.313-336, 1994

Zhao04 J. Zhao, L. Li, "*Human Motion Reconstruction from Monocular Images Using Genetic Algorithms*", Computer Animation and Virtual Worlds, Vol. 15, pp.407-414, 2004

Zord99 V. B. Zordan, J. K. Hodgins, "*Tracking and Modifying Upper-Body Human Motion Data with Dynamic Simulation*", Computer Animation and Simulation '99, pp.13-22, 1999

Zord00 V. B. Zordan, "*Simulating upper-body intensive activities from human motion examples*", Animation Humans by Combining Simulation and Motion Capture, ACM SIGGRAPH 2000, Course Notes Vol. 33, pp. 41-49.

Zord02 V. B. Zordan, J. K. Hodgins, "*Motion Capture-Driven Simulations That Hit and React*", ACM SIGGRRAPH/Eurographics Symposium on Computer Animation, pp.89-96, 2002

Zord03 V. B. Zordan, N. C. Van Der Horst, "*Mapping Optical Motion Capture Data to Skeleton Motion Using a Physical Model*", ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp.245-250, 2003

Zord05 V. B. Zordan, A. Majkowska, B. Chiu, M. Fast, "*Dynamic Response for Motion Capture Animation*", Proceedings of SIGGRAPH 2005, August 2005

Appendix B:
Mathematical Constructs

B.1 Expansion of Piecewise Linear B-Spline Curves

Similar to the piecewise cubic B-spline curves, the linear counterparts are defined over a given knot sequence, whose definition is repeated from Equation 5.36.

$$\left[u_0, u_1, \mathbf{K}, u_i, \mathbf{K}, u_{last}\right] \text{ where } u_0 < u_1 < \mathbf{K} < u_i < \mathbf{K} < u_{last} \qquad \text{(from Equation 5.36)}$$

Each linear B-spline is defined by two basis functions and two consecutive control points as illustrated in Equation B.1a. The complete curve is thus given by the summation of each curve segment, which is expressed in Equation B.1b.

$$Q_i(u) = V_i B_{i,-1}(u) + V_{i+1} B_{i+1,-0}(u) \qquad \textbf{(B.1a)}$$

$$Q(u) = \sum_i Q_i(u) \qquad \textbf{(B.1b)}$$

The basis functions for the piecewise linear B-spline curves are derived using the one-sided basis functions that were used during the derivation of the cubic counterparts, whose definition is repeated from Equation 5.39.

$$B_{i,k}(u) = (-1)^k (u_{i+k} - u_i)[u_i(k) : t](u-t)_+^{k-1} \qquad \text{(from Equation 5.39)}$$

By expanding Equation 5.39 for *k=2,* the basis functions required for Equation B.1a can be determined. Equation B.2a and Equation B.2b give the expansion of Equation 5.39, where the basis equations of B.1a and B.1b are related by dropping the order index from the set of B.2 formulations.

$$\begin{aligned}
B_{i,2}(u) &= (-1)^2 (u_{i+2} - u_i)[u_i, u_{i+1}, u_{i+2} : t](u-t)_+^1 \\
&= (u_{i+2} - u_i)\left[\frac{[u_{i+1}, u_{i+2} : t](u-t)_+^1 - [u_i, u_{i+1} : t](u-t)_+^1}{(u_{i+2} - u_i)}\right] \\
&= \left[\frac{[u_{i+2} : t](u-t)_+^1 - [u_i : t](u-t)_+^1}{(u_{i+2} - u_{i+1})}\right] - \left[\frac{[u_{i+1} : t](u-t)_+^1 - [u_i : t](u-t)_+^1}{(u_{i+1} - u_i)}\right] \\
&= \left[\frac{(u - u_{i+2})_+^1 - (u - u_i)_+^1}{(u_{i+2} - u_{i+1})}\right] - \left[\frac{(u - u_{i+1})_+^1 - (u - u_i)_+^1}{(u_{i+1} - u_i)}\right]
\end{aligned}$$

$$B_{i,2,-0}(u) = \frac{u - u_i}{u_{i+1} - u_i} \qquad u_i \text{ £ } u < u_{i+1} \qquad \textbf{(B.2a)}$$

$$B_{i,2,-1}(u) = \frac{u_{i+1} - u}{u_{i+2} - u_{i+1}} + 1 \qquad u_{i+1} \text{ £ } u < u_{i+2} \qquad \textbf{(B.2b)}$$

It is important to note that because the basis functions only span 2 knot intervals, to keep the same number of curve segments, which are aligned with the cubic curves, it follows that there will be 4 less knots used than the cubic version. Consequently, to maintain the alignments, the first knot of the linear sequence is taken as that of the third knot of the cubic sequence. This subsequently means that the final knot in the linear sequence will be the last but 2 in the cubic version.

B.2 Common Coefficients of Friction

Material A	**Coefficient of Static Friction,** m_s	**Coefficient of Kinetic Friction,** m_k.
Steel on Steel	0.74	0.57
Steel on Aluminium	0.61	0.47
Steel on Copper	0.53	0.36
Rubber on Concrete	1.00	0.80
Rubber on Asphalt	1.00	0.80
Wood on Wood	0.40	0.20
Wood on Brick	0.60	0.40
Glass on Glass	0.94	0.40
Ice on Ice	0.10	0.03

Table B.1: Common coefficients of static, m_s, and kinetic m_k friction.

Appendix C:
Solving the Non-Linear Optimisation System

In this appendix, a brief overview of the smaller, miscellaneous details of the practical implementation is considered. This starts by introducing the technique employed to inverse the non-square matrices of the system, which is continued with a look at how the solver is initially seeded. This section concludes by a brief discussion on how the solver terminates the iterations with its optimal solution and hence the motion and a practical optimisation used when curve segments are refined.

C.1 Matrix Inversion

To calculate the solution of the objective and constraint formulation of Equation 5.31 and Equation 5.32, Singular Value Decomposition (SVD) from numerical recipes [Pres92] is used to decompose the non-square Hessian matrix (for the objective function) and Jacobian matrix (for the constraints), which result in the hyperplane and projected update vectors respectively. Equation C.1 demonstrates the principle of the decomposition process where *A* is the original matrix, *S* and *V* are square orthogonal matrices and *U* is a diagonal matrix consisting of the singular values.

$$\begin{aligned} A &= SUV^T \\ A^{-1} &= V(\frac{1}{U})S^T \end{aligned} \tag{C.1}$$

During the calculation of the reciprocal of the *U* matrix elements, if a dividing by zero occurs, the element is set to zero.

C.2 Starting Conditions

As with all constraint-based optimisation algorithms, it is advantageous to start off with a good set of initial values. To this aim, in addition to the main iterative steps that are taken to converge towards a solution, a few pre-iterative steps are performed that give a suitable starting point. The first iteration minimises the difference between the example motion DOFs and the free variables representing them. Due to the independent and positive definite nature of the Hessian matrix, the solution to this problem is determined in a single iterative step. However, this first iteration only gives good initial values for the joint angles and still leaves other free variables, such as muscle forces, with a potentially poor initial set of values.

To provide a better starting basis for other free variables, the joint space configuration free variables are locked in by setting temporary positional constraints for each of the control points. This system is subsequently solved over several iterations to obtain good starting values for the other free variables. At this point, the contact forces are turned off to avoid constraint contradiction and hence a poor solution. With a suitable set of initial values, which generally takes fewer than 5 iterations, the temporary joint space constraints are removed and the contact forces are enabled, ready for the application of the complete solver and the main iterations involved in the solver process.

C.3 Stopping Conditions

From a theoretical point of view, the iterative optimisation process would cease when all the constraints are driven to zero and any further reduction in the optimisation function would violate them. In practice, it is highly unlikely that the constraints will be identically zero. Hence, the strategy employed to determine when the solver reaches its most optimal solution is to compare the difference between the previous and current frame's average squared sum error over the number of active constraints. In the event of a decrease in error, another iteration is invoked but at the same time the current state error is compared with the best global minimum error the system has seen. If the error is smaller, the best-seen state is replaced with the current. If the error decrease over 10 consecutive iterations has been less then an average of 0.1% and similarly for the minimisation function, the iterations are automatically terminate as it is unlikely that a better solution will be found given the current state.

Conversely, in the case of an increase in error, the step length taken along the calculated update vector is decreased. This is achieved by normalising the update vector. In the very rare case where the magnitude of the update vector is already less then one, the original update vector is maintained since the updates are already very small in comparison to the scale of the system units.

When an increase in constraint errors is seen, the process is not immediately terminate because the increase could be due to the turning on of an active constraints. Therefore, the optimisation process is allowed to continue to explore its path, although if the constraint error continues to rise on 10 or more successive iterations, it is terminate with the option of going back to the most optimal solution found.

While the system is not iteratively looking for a solution (which the user can also stop as well as the aforementioned automatic strategy), the user is free to adjust the system state and continue the iteration process. Such changes included adding new user constraints, modifying the path of the generalised variables and refining the B-Spline knot vector and hence the frequency of the free variables to allow for more control in certain temporal locations. The refinement of the B-Splines, as described in section 5.3.3.1, does not require a complete determination of a new set of initial values as the Oslo algorithms returns a set of curves that are equivalent to the existing ones. However, the constraints and minimisation functions needs to be reproduced based on the new curve representation.

C.4 Curve Refinement

After a curve refinement process, the regeneration of the mathematical constructs can be achieved without reproducing the complete set by recognising that only the equations involving 8 consecutive curve segments needs to be updated. Figure 5.6 demonstrates this where it can be seen that each knot value is involved in at most 7 curve segments. The eighth curve segment comes from the extra curve segment that is introduced through the refinement process. Consequently, once the curve set has been refined, only the equations that are based on the 4 curve segments before and including the refinement point and the 4 curve segments immediately after, need to be regenerated. All other knot values remain constant and hence so do their corresponding basis functions and curve segments. For example, if a new curve segment were to be introduced at curve index 8, new formulations for equations involving any of the new curve segments 5, 6, 7, 8, 9, 10, 11, and 12 would need to be developed, where all other equations remain unchanged.

Through the use of this regeneration, the time it takes to refine curves can be reduced, which can be considerable if the whole set of equations were regenerated. Furthermore, the equation regeneration task is further reduced if curve segments near the ends of the piecewise cubic B-Spline are refined. This is because the upper and lower bounds of the curve segments that need to be regenerated are capped between the first curve and last, i.e. if the first curve we to be refined, only the equations involving segments 0, 1, 2, 3, and 4 would need to be regenerated.

Appendix D:
Biomechanical Information

D.1 Retargetting Biomechanical Information

Motion	Walking Character		Catch Character	
	Original	Retargetted	Original	Retargetted
Foot				
Length (m)	0.22	0.2103	0.20276	0.2103
Lower Circumference (m)		0.28		0.28
Upper Circumference (m)		0.2		0.2
Mass (kg)		1.02001		1.017228448
Lower Leg				
Length	0.4501	0.41434	0.41176	0.4142
Lower Circumference		0.4		0.4
Upper Circumference		0.24		0.24
Mass		3.613695809		3.602623623
Upper Leg				
Length	0.50444	0.36766	0.42074	0.37064
Lower Circumference		0.6		0.6
Upper Circumference		0.49		0.49
Mass		9.14220918		9.191176922
Hips				
Length	0.09072	0.0866	0.07592	0.0866
Lower Circumference		0.9		0.9
Upper Circumference		0.9		0.9
Mass		5.852519129		5.836559396
Chest				
Length	0.28986	0.43098	0.30364	0.43098
Lower Circumference		0.9		0.9
Upper Circumference		1		1
Mass		32.48217367		32.39359526
Upper Arm				
Length	0.31624	0.27966	0.17216	0.28174
Lower Circumference		0.46		0.46
Upper Circumference		0.3		0.3
Mass		3.419059931		3.435096468
Lower Arm				
Length	0.26984	0.2454	0.26916	0.24742
Lower Circumference		0.3		0.3
Upper Circumference		0.18		0.18
Mass		1.203903884		1.210503717
Hand				
Length	0.19	0.15546	0.08028	0.15594
Lower Circumference		0.18		0.18
Upper Circumference		0.16		0.16
Mass		0.375281421		0.375413598
Neck				
Length	0.2	0.113	0.11358	0.113
Lower Circumference		0.36		0.36

Upper Circumference		0.36		0.36
Mass		1.221865426		1.218533417
Head				
Length	0.28	0.15706	0.064	0.15706
Lower Circumference		0.46		0.46
Upper Circumference		0.48		0.48
Mass		2.895121331		2.887226378
Total Mass		80		80

Lower circumference refers to the circumference nearest the limbs "origin"

Table D.1: Biomechanical information used to demonstrate the retargetting approach of Chapter 4 and Chapter 6

D.2 Motion Captured Biomechanical Information

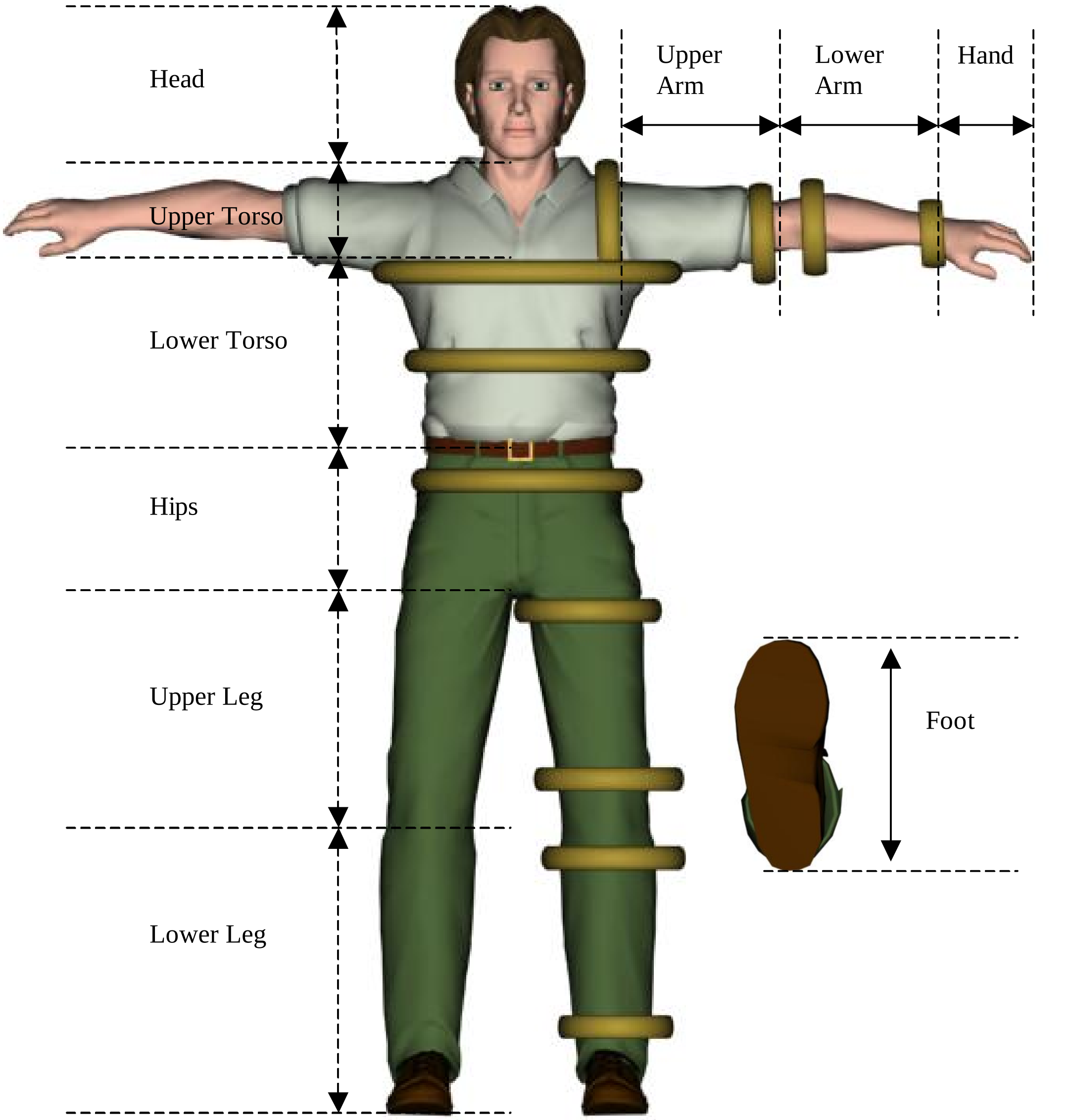
The biomechanical information gathered from the 4 actors we motion captured and used to evaluate our different motion adaptation processes are given in Table D.2. The correlation between the measurements in Table D.2 is illustrated in Figure D.1, where we treat the body as being symmetrical in terms of similar limbs having the same weight and lengths, i.e. both legs are represented from the measurements taken from just one side.

Actor	**A**	**B**	**C**	**D**
General Information				
Age	28	32	25	41
Height (metres)	1.77	1.716	1.87	1.8
Weight (kgm/s^2)	80	97	73	68
Total Volume	0.091819	0.107767	0.076288	0.080325
Calculated Total Mass	80	97	73	68
Foot				
Length (metres)	0.26	0.272	0.28	0.275
Lower Circumference (metres)	0.27	0.28	0.27	0.26
Upper Circumference (metres)	0.20	0.20	0.19	0.18
Volume	0.001106	0.001157	0.001191	0.001191
Weight	0.963234	1.041013	1.139267	1.007898
Lower Leg				
Length (metres)	0.46	0.512	0.48	0.5
Lower Circumference (metres)	0.41	0.427	0.35	0.37
Upper Circumference (metres)	0.235	0.25	0.22	0.27
Volume	0.003901	0.004775	0.003156	0.004108
Weight	3.398534	4.297802	3.020312	3.477254
Upper Leg				
Length (metres)	0.32	0.257	0.41	0.34
Lower Circumference (metres)	0.63	0.615	0.54	0.57
Upper Circumference (metres)	0.47	0.435	0.4	0.42
Volume	0.007757	0.005692	0.007261	0.00668
Weight	6.758866	5.123422	6.94759	5.655181
Lower Torso				
Length (metres)	0.23	0.214	0.2	0.24
Lower Circumference (metres)	0.94	1.085	0.8	0.89

Upper Circumference (metres)	0.94	1.085	0.8	0.89
Volume	0.016172	0.020048	0.010186	0.015128
Weight	14.09061	18.04463	9.746871	12.80671
Lower Chest				
Length (metres)	0.31	0.39	0.35	0.32
Lower Circumference (metres)	0.94	1.085	0.8	0.89
Upper Circumference (metres)	1	1.15	0.92	0.91
Volume	0.023218	0.038768	0.020633	0.020627
Weight	20.22973	34.89449	19.7435	17.46222
Mid Circumference (metres)	0.93	1.06	0.83	0.84
Upper Chest				
Length (metres)	0.18	0.123	0.13	0.12
Lower Circumference (metres)	1	1.15	0.92	0.91
Upper Circumference (metres)	1	1.15	0.92	0.91
Volume	0.014324	0.012945	0.008756	0.007908
Weight	12.48012	11.65133	8.378654	6.694382
Upper Arm				
Length (metres)	0.19	0.21	0.31	0.29
Lower Circumference (metres)	0.47	0.495	0.31	0.36
Upper Circumference (metres)	0.29	0.315	0.28	0.25
Volume	0.002224	0.002786	0.002149	0.00217
Weight	1.937817	2.507811	2.056057	1.837073
Lower Arm				
Length (metres)	0.24	0.24	0.3	0.28
Lower Circumference (metres)	0.3	0.295	0.28	0.25
Upper Circumference (metres)	0.18	0.19	0.18	0.18
Volume	0.001123	0.001141	0.001283	0.001039
Weight	0.978442	1.026697	1.227497	0.879632
Hand				
Length (metres)	0.18	0.195	0.2	0.22
Lower Circumference (metres)	0.18	0.19	0.18	0.18
Upper Circumference (metres)	0.17	0.18	0.16	0.15
Volume	0.000414	0.000476	0.00046	0.000507
Weight	0.361092	0.428793	0.44064	0.428813
Neck				
Length (metres)	0.07	0.06	0.08	0.08
Lower Circumference (metres)	0.38	0.40	0.37	0.35
Upper Circumference (metres)	0.38	0.40	0.37	0.35
Volume	0.000872	0.000872	0.000872	0.000872
Weight	0.759346	0.784455	0.833967	0.737802
Head				
Length (metres)	0.19	0.16	0.22	0.2
Lower Circumference (metres)	0.48	0.48	0.47	0.47
Upper Circumference (metres)	0.57	0.60	0.58	0.58
Volume	0.004183	0.003082	0.004843	0.004403
Weight	3.644219	2.774006	4.634282	3.727185

Note: Lower circumference refers to the circumference nearest the limbs "origin"

Table D.2: Biomechanical information for 4 different actors, where the black number represent the manually measured fields and the red values give the calculated fields for both volume and limb masses



(gold rings indicate circumference measurement)

Figure D.1: Body measurement reference guide used to record the manual measurements of Table D.2